SECURITY NOW!

Transcript of Episode #451

TrueCrypt & Heartbleed Part 2

Description: Not surprisingly, the previous week consisted of nearly a single story: Heartbleed. It was only "nearly," though, because we also received the results from the first phase of the TrueCrypt audit. So this week Leo and I discuss these two topics in detail.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-451.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-451-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to talk about two things. Actually a lot of security news. But an update on Heartbleed, Heartbleed Part 2, and the audit for the TrueCrypt app, which we've longtime strongly recommended, is finally in, and there are some surprising issues. Stay tuned. Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 451, recorded April 15, 2014: TrueCrypt & Heartbleed Part 2.

It's time for Security Now!, the show where we cover your security and privacy with this guy right here, the big-handed Steve Gibson, Explainer in Chief. You know, I don't know if you realize, but your hand, because of where the camera is, your hand looks bigger than your head. Hi, Steve.

Steve Gibson: Hello, Leo.

Leo: Welcome back.

Steve: I feel like we were just here.

Leo: We were.

Steve: We ought to let people know that you did have me back on for a Part 2 of the continuing series of How Did This Happen?

Leo: You told some great stories about being kicked out of the Boy Scouts, about getting into the Artificial Intelligence Labs at Stanford as a teenager, things that just really, I've known you for years and had no idea. So it's a great, must-listen-to Triangulation from yesterday. And I thank you for doing that. And of course the reason we did it is because we didn't get very far in your life story the first time. And we got less far, I think we went backwards...

Steve: We went backwards and covered some, yes, previous bits, which, you know, bore some mentioning.

Leo: But by the time we're doing with this series, which may be 10 or 11 episodes, you won't have to write a biography. This will be your biography, The Story of Steve.

Steve: Well, and we also spend a lot of time not talking about me, but talking about just industry stuff. And from the feedback that I saw in Twitter, people really enjoy just you and me talking about other stuff.

Leo: Well, and of course Heartbleed was one of those things. And I think it'll be a bit of a topic again today.

Steve: Well, yeah. It's funny because, when I wrote the description in the email that I sent you and Elaine, which Elaine puts into the transcript, I basically said, well, two things happened this week. Almost nothing other than Heartbleed, but that "almost" was taken up by TrueCrypt, whose first audit results came back.

So we're going to talk about the Phase I Audit Complete of TrueCrypt with some detail, some interesting results, and what we learned from it. And there is a lot to learn, really about some of the thing about the open source movement that you and I are going to talk about, both in the context of TrueCrypt and then again with OpenSSL and Heartbleed. And of course we've got to talk about everything that happened since we essentially were on the leading edge of the breaking news one week ago. And, oh, there's an interesting timeline that one reporter put together that was published in the Sydney Morning Herald, of all places, showing how long before the world knew it was actually known by, like, people at Google.

Leo: Ooh, that's not good.

Steve: Turns out it was back in March.

Leo: Wow.

Steve: Yeah.

Leo: Well, that's a month.

Steve: Oh, I mean, it wasn't, oh, no, no, it wasn't by any means many - it wasn't a long time. But there was - the wheels were turning behind the scenes with people promising each other not to mention it until this process could be managed. So we have all of that follow-up to talk about. And I had intended initially to do some Q&A. But I just don't think we're going to have time. So we're going to put the Q&A to next week, where we'll do a catch-up. There was a ton of email in the mailbag. And so certainly some pressure to get some questions from our listeners answered, which we'll do next week.

And then the week after I currently have slated to - although, again, that's always subject to what the world brings us - to talk about what I call the "revocation revelation," which is something I learned about how little, well, essentially how broken one part of the whole certificate authority system is, which has really not been getting much attention. Everybody knows about it behind the scenes. Unfortunately, no one's really caring too much about it because users don't know. And so I've actually created another resource at GRC, and I could talk about it next week in advance of the following week's podcast. Or people can just look under Services on the Main Menu, and you'll see something about certificate revocation. And if you're curious, go take a look there. You'll find that I've been busy in this last week. So lots happening today. And we've got the next few weeks all planned out.

Leo: Well, this is good. And we'll get to a Q&A eventually. I don't think that's going to be a big problem.

Steve: We always do.

Leo: All right, Steven. Let's get into the meat of the matter, as they say.

Steve: Okay. So, first TrueCrypt. I know this is on the radar of our listeners because I got a tweet flood as this began to happen.

Leo: Well, you talked about the fact, you promoted that they were trying to raise money to do this audit.

Steve: Yes. And in fact the site IsTrueCryptAuditedYet.com has been tracking this for some number of months. And Matthew Green, the professor of - he's a research professor focusing on cryptology at Johns Hopkins. He's been one of the people organizing this. They've created a site called OpenCryptoAudit.com, which is sort of the - it's the umbrella site for all of sort of what we're beginning to understand we need to do that couldn't be more highlighted than what we've just gone through in this past week with the OpenSSL Heartbleed problem because here was open source code that had been sitting there for a couple years. And, yeah, it would have sure been nice to have, if not an audit of the entire package, which apparently is almost beyond auditing for reasons we'll be discussing later, just someone really looking at any changes being made because it's been noted that a change was apparently made, apparently on New Year's Eve. And it was given an eyeball by another developer who said, yeah, that looks fine. And it got committed into the project and became part of OpenSSL. We'll be talking about that more in a second.

So where we are with TrueCrypt is that Phase 1's results are available. I've got a link in the show notes. And I've seen people saying, well, where are these show notes you talk about? Well, first of all, I just tweeted them. I always tweet them every week. So if you're not on Twitter, well, you can still look at Twitter.com/SGgrc, and you will find my feed and my link to the show notes, which is what Leo's looking at, I'm looking at, and they're available. They will also be on the Security Now! page as soon as the transcripts and audio is available. We pull all that together, so that's there every week. So I am putting links in the notes. And obviously you can find these things if you just google around. But I make them as easy as I can for you.

So there is a, I think it's a 32-page PDF from a group called iSecurity, iSEC. They called it their "Final Open Crypto Audit Project TrueCrypt Security Assessment." And I am very impressed with the work they did. However, no one should believe that this is an audit of TrueCrypt. What it was is a very circumscribed audit of only two pieces of TrueCrypt, specifically the bootloading process and the Windows kernel driver, and nothing else. So there's much more to be done. And this highlights one of the problems with having an independent group come in and audit code. And that is, it's really hard. I mean, it is hard and time-consuming to do this really well.

So what these guys did, the reason I'm so impressed is what they did, it looks like from all the evidence they did really well. But given a couple months to do this, the size in terms of the whole project of what they did was still very small. Which is to say, for this to be useful, it has to be in-depth. But in-depth analysis of somebody else's code, where you're looking, where you're scrutinizing it at a level which the coders themselves didn't scrutinize, where, for example, these guys note that some integers are treated as signed in some places and unsigned in others. Well, it works. But they're noting, you know, that's really not cool because there could be side effects from that when the high bit ever - if the high bit were ever to get set, that makes it look like a negative value. And then your equality checks are not going to do the right thing. And it's probably the case that these particular values are small enumerations. They're just never going to have the high bit set in the integer. But still, it just sort of demonstrates, you know, not the kind of integrity you would like to have in a piece of software that we're now depending upon to the degree we are.

But Matt Green, looking at the audit results, said - he was quoted saying, "The results don't panic me. I think the code quality is not as high as it should be; but, on the other hand, nothing terrible is in there." So that's reassuring. Now, again, that's of what they've looked at so far, which is a small piece. It's just the bootloader and the Windows kernel driver. So to give you some sense for this, this 32-page PDF has a summary of findings. And they wrote: "During this engagement, the iSEC team identified 11 issues in the assessed areas," meaning just those two, bootloading and the kernel driver. And I should mention that those were looked at specifically because the question was, if there were some backdoors, if there was something really bad, the kind of thing that would panic Matt, that's probably where they would be. Which is not to say there couldn't also be, for example, they specifically disclaimed that they didn't look at their random number generator. And we know how crucial that is.

So no one has looked at that yet. This hasn't looked at that yet. This is specifically to say, if a backdoor existed, we want to make sure it's not in the bootloader or the Windows kernel driver. So they can now very definitively say nothing malicious was found as a result of a very careful analysis of all of that code, yet none of the rest of it, which is way the majority. So iSEC says, of these 11 issues, four issues were medium severity - and I'll give you some examples of those in a second - and another four low severity, with an additional three issues having informational severity, meaning just sort of defense in depth, were there enough layers of security, so just sort of talking about the architectural side.

iSEC wrote: "Overall, the source code for both the bootloader and the Windows kernel driver did not meet expected standards for secure code." And let me - I'm going to pause again and say, now, remember that, unfortunately, what's happened is something that just sort of started getting coded by some guys who thought it would be useful, way before BitLocker came along, and Microsoft gave us their implementation, it's like let's do a whole-drive encryption thing for Windows. So they just sort of sat down, started writing some code. I mean, if they had known then what they know now, their original coding style may have been different. On the other hand, it may have never gotten off the ground if all of this was imposed on them from day one.

So, and it's also been noted, I was reading a lengthy blog by Dan Kaminsky, who responded, and we'll talk about this later, about the Heartbleed problem, I mean, what a catastrophe OpenSSL, everyone agrees, is in terms of its code. It's just - it's a horrible mess because it just sort of grew organically. And unfortunately, now the whole world uses it. Similarly, now we all wish TrueCrypt were something more than it is. But it is what it is. No one has to pay for it. It's free. But as some have said, you get what you pay for.

So iSEC says it did not meet expected standards for secure code. "This includes issues such as lack of comments, use of insecure or deprecated functions," for example, like transferring blocks of memory from one buffer to the next and not using the newer transfer features which protect you from overruns, for example. Which is not to say there are any, but that, if you were writing code now, and you really, really focused on every aspect of security - and again, this is hard and time-consuming, and maybe they wouldn't have bothered if we were to impose all this on them from the beginning. But we don't have that in this code. Inconsistent variable types and so forth.

Continuing with iSEC, they said: "In contrast to the TrueCrypt source code, the online documentation available at TrueCrypt.org/docs does a very good job at both describing TrueCrypt functionality and educating users on how to use TrueCrypt correctly." And actually it turns out that some of the advice in the docs helps to mitigate some of the potential problems that were found; although, again, none of these would put anyone, should put anyone off from using it at all. I mean, certainly having it is way better than not. "This includes recommendations to enable full-disk encryption that protects the system disk, to help guard against swap, paging, and hibernation-based data leaks," which we've talked about in the past when we've talked about TrueCrypt.

iSEC continues: "The team found a potential weakness in the volume header integrity checks. Currently, integrity is provided using a string" - which is four characters, TRUE, all uppercase, T-R-U-E - "and two CRC32s. The current version of TrueCrypt utilizes XTS as the block cipher mode of operation." Now, we've talked a lot about block cipher modes. CBC, for example, is the one we often talk about. And they mention how this doesn't have authentication. And iSEC writes: "...which lacks protection against modification; however, it is insufficiently malleable to be reliably attacked. The integrity protection can be bypassed, but XTS prevents a reliable attack, so it does not currently appear to be an issue. Nonetheless, it's not clear why a cryptographic hash or an HMAC was not used instead." And you'd have to agree with them. It's sort of odd to use a pair of CRC32s. But that's what they did.

Finally, writes iSEC: "iSEC found no evidence of backdoors or otherwise intentionally malicious code in the assessed areas. The vulnerabilities described later in this document all appear to be unintentional, introduced as a result of bugs rather than malice." And again, even so, those vulnerabilities were not severe. Then they also did a recommendations summary. So they said, under Recommendations, "Outside of the specific short- and long-term recommendations detailed in Section 3 of this document, iSEC Partners also makes the following high-level recommendations." And again, I'm sharing this because this helps to kind of paint the picture, to give you a sense for what TrueCrypt is, not as the user installing and running it, but as, like, someone looking at the source, I mean, the whole source environment.

So they said: "Update the Windows build environment." Okay. Is everybody sitting down? "The current required Windows build environment depends on outdated build tools and software packages that are hard to get from trustworthy sources. For example, following the reproducible build instructions requires access to VC++ v1.52, released in 1993."

Leo: What? That's 21 years ago.

Steve: 21 years ago. I happen to have a copy, but I'm not sure how many people do. So that gives you a sense for it.

Leo: But couldn't you use a more up-to-date compiler, it would work the same, or...

Steve: It would break.

Leo: It would break. Interesting.

Steve: Oh, yeah. Wait, well, we'll get to it later. But because the compilers produce all kinds of warnings about the code, there's like a whole list of pragmas to disable those warnings in the TrueCrypt build in order to get it to build. So this gives you the sense...

Leo: That's just probably because most of the focus is on the GCC open source version. And whoever wrote the Windows build instructions and builder moved on. That's the problem with open source.

Steve: Well, and they say, yes, they say: "...in addition to various Windows ports of GNU tools downloadable from wherever they can be found," you have to have this VC++ v1.52. "Using antiquated and unsupported build tools," these guys write, "introduces multiple risks including unsigned tools that could be maliciously modified, unknown or unpatched security vulnerabilities in the tools themselves, and weaker or missing implementations of modern protection mechanisms such as DEP and ASLR. Once the build environment has been updated, the team...." So you're right, Leo. I mean, they're saying, if we were to recommend something, it would be, among other things, fix the build environment because it's creaky and old. And they say: "For the purpose of auditing, TrueCrypt should release instructions on how to create reproducible builds." And that's not something that's really ever been a concern. It's like, hey, it works. As far as we know, it's secure. Enjoy.

And then their second recommendation was improve the code quality. "Due to lax quality standards, TrueCrypt source is difficult to review and maintain. This will make future bugs harder to find and correct. It also makes the learning curve steeper for those who wish to join the TrueCrypt project." Which is a good point, that it can be a little off-putting not to have, like, it all done in a uniform, really nicely well-documented and put-together mode.

And so then they said, just so that they were clear: "The assessment explicitly excluded an analysis of the volume parsing as it relates to file containers; rescue disk code paths activated with the disk does not contain the private key," that is, you know, the recovery from an emergency analysis; and any cryptographic analysis which, by the way, is what's coming next, including the random number generator analysis, the algorithm implementation, security tokens, keyfile derivations, hidden containers, Linux and Mac components - this was only the Windows components - and all other components not explicitly included, meaning everything else.

So, let's see. I'm not sure how much more time I want to spend on this. I've got extensive notes here in the show notes, if anyone is curious to dig in.

Leo: You could do the TLDR, just bottom line.

Steve: Yeah, okay. So, yeah, bottom line is that this is, I mean, in my opinion, it is entirely understandable that this is what TrueCrypt is today.

Leo: Yeah. I bet you see this with a great many, if not all open source projects. That's how it works.

Steve: That's exactly the case, Leo. It's a function of them generally having a long history. There's developer churn where oftentimes the people doing it now, working on it, aren't the people who founded it, and the founders have wandered off onto other things. Their lives change. They've graduated from college. They have a family to feed. Who knows? But so there's that. There's over time there's different people bringing their own coding styles. We've talked about ridiculous things, well, which many people feel religious about, like where do you put your curly braces in your C code.

Leo: Oh, yeah. You want to start a war, just post your thoughts on that, yeah.

Steve: Right. So, I mean, but there were little things. For example, just to give you the third thing that I highlight here, there's this function, burn(). "Burn() is used," writes iSEC, "to clear sensitive data throughout most of the TrueCrypt Windows kernel driver." And what that is, in Windows, burn() wraps the secure zero memory function, which securely overwrites memory with zeroes, to remove any cryptographic keys or other data, anything sensitive, before you release the memory back to the system. And they write: "This is guaranteed to securely erase memory and will not be optimized out. However, in a handful of places, memset() is used to clear potentially sensitive data." Memset() is the C equivalent where you're able to just say, you know, set this block of this length to this value, which is typically null or zero. And it does a nonsecure zero. And the point is that calls to memset() run the risk of being optimized out by the compiler.

Because what'll happen is...

Leo: That's an interesting bug, yeah, yeah.

Steve: Isn't that? Isn't that interesting? So what happens is the compiler comes along and goes, do do do do do do do.

Leo: We don't really need this.

Steve: Yes, exactly. It's doing this very sophisticated flow analysis, and it notices that some moron has written some stuff to a buffer and then freed it. And it's like, well, if you're going to free it, why bother...

Leo: You mustn't need it. Yeah, why write to it? Right.

Steve: Well, yeah, this must have been some old code left over from somewhere, so let's, oh, I'm an optimizing compiler. I realize this can never have any effect at all on the operation of the program, so get rid of it.

Leo: That's really - I hadn't really thought of that. But, yeah, optimization can often, I would bet, eliminate security precautions because they don't affect the operation of the program.

Steve: And so, again, I'm very impressed with what iSEC did. They have then, after explaining this, they have an exploit scenario: "A user has a system with a TrueCrypt-encrypted partition on it, in which they save sensitive information. An attacker creates a low memory situation on the user's machine, forcing key information that should have been securely wiped to be paged out to the unencrypted system disk. The attacker later gains access to the disk and extracts the key from the paging file." Now, don't everyone freak out because, first of all...

Leo: You have to have access to the disk, yeah.

Steve: You have to have access to the disk. You have to not be encrypting your system disk, which everyone should be doing and probably does now that TrueCrypt offers it. And this is just theoretical. That is, they didn't actually demonstrate that sensitive data was left behind in one of these memset() instances and that even in a low memory situation you could actually swap it out. They're just saying this is the danger of not securely wiping memory.

So then they said, for their short-term solution: "Alter the above code to call burn() instead of memset()." Okay, that's easy enough.

Leo: Or I bet there are compiler switches saying don't optimize that stuff out. You wouldn't even have to rewrite the code; right?

Steve: Yes. You could certainly just say leave memset() alone and trust us that we did this for a reason. And then they said: "Audit the code for other instances of memset() calls that should be replaced with calls to burn() to prevent potential information leakage." Now, what, for example, I'm doing with SQRL is I'm deliberately bundling everything that is sensitive in one place so that, rather than, unfortunately, for example, in TrueCrypt is strewn all over the place - because I'm thinking about this with security as the only thing I'm thinking about, really, everything is allocated in one place, and that is, it is locked in physical memory using actually the virtual memory system to lock it so that it absolutely can never leave physical memory. And because every single time I add something to that structure, I'm, wait, is this sensitive? Is this sensitive? I just - that's where I put it. I put it in there so it's always aggregated safely in one location so you can treat it correctly.

And they did say in their other advice, they said, under suppression of compiler warnings: "Microsoft compilers used to build TrueCrypt will warn against some of the issues mentioned above. However, in both the bootloader and Windows kernel driver, some of these warnings have been suppressed." And I skipped those when they were talking about above there, like things like, oh, wait a minute, you used this integer in a signed fashion here and an unsigned fashion elsewhere. That's absolutely something the compiler will raise a flag about. Unfortunately, rather than fixing it, the coders suppressed the warning. So they say: "Some are suppressed with #pragma in the code, while others are suppressed in the build scripts. This results in the code compiling without warnings, even though it contains issues that should be corrected."

And then they give an example of four different pragmas being used to disable warnings that would otherwise be generated. And of course for something arguably mission-critical, like TrueCrypt, you want to look at those warnings. It's like, wait a minute. Did it find something that is important? So we can assume the authors carefully disabled these. But as an auditor coming in with an inherently adversarial stance, which an auditor has to have in a mode like this, it's like, well, it'd be better not to have these. And so no one's going to argue.

So what all this means is that complex security software is so difficult to write and verify that, as we have seen time and again, it becomes porous. I mean, it's a matter of, like, how much pressure do you put on it, and some water leaks out? So I think we're beginning to recognize with this, but certainly more poignantly with the OpenSSL problem, is that we the people who use this code, and a subset of us who write this code, don't have the resources that are required. I mean, this was a very close look at a very small portion of TrueCrypt. And lots more still to be audited. Yet it took, no doubt, a substantial investment of a couple guys' time really looking at this closely. And that means it's expensive. Someone has to pay for this.

And the thing that's a little unsettling is that, whereas we the people don't have the resources to deeply verify open source software, the NSA does. I mean, it's got reportedly a thousand people looking carefully at open source code. And there have been some claims, unsubstantiated, that the NSA may not always be acting in what we would consider the best interests of the industry, but rather they're using these thousand people to gain intelligence for themselves and for their own purposes.

So Matthew Green said, he tweeted: "The second phase is now to perform a detailed crypto review and make sure there's no bug in the encryption." So that's what they're going to do as Round 2 of this effort.

Leo: And maybe the quick bottom line, Bruce Schneier on his blog said, "Quick summary, I'm still using it."

Steve: Yeah.

Leo: Yeah. So if it's good enough for Schneier, and good enough for you - do you use TrueCrypt at all?

Steve: Yes. I do have it on a laptop because I carry the laptop around.

Leo: Outside the house, yeah.

Steve: Yes, exactly. If there's anything where there's exposure. Well, I mean, it just makes so much sense. I would have been stunned, and still will be, if anything deliberate is found. Again, in a multi-developer mode, if there were an interest in putting a backdoor in - and we know you-know-who might have had some interest. And, I mean, the world's getting a little creepy because we're sort of seeing influence, and we've been discussing this in the last months, sort of the notion of influence to affect the crypto that we're all using, maybe through employees implanted or solicited to help with the U.S. national security interest. Who knows? But, I mean, as hard as it is to make something absolutely secure and bulletproof, it is so easy to sneak a mistake in. And so the problem is, as we know, the weakest link problem, something this complex, a small change could be introduced that looks just fine, but does create some leverage.

Oh, and in fact, one thing I skipped, I want to mention, is that one of these issues, I think it was with the handling of an integer problem, in their exploitation scenario they mentioned that somebody could alter the boot code in order to use the overflow in a theoretical attack to capture someone's login. And but standing back from it, it's like, well, wait a minute. We know that the boot time is the vulnerability, that is, the boot code comes up and asks you for your password. So, yes, if you allowed bad guys to play with your TrueCrypt-encrypted hard drive out of your control, then they gave it back to you and said "Login," or actually if they snuck in and did it in the middle of the night without you knowing, then you gave your password to TrueCrypt, it's certainly conceivable that that modification would capture the password. Or who knows what it would do? I mean, it could just change it on the fly to something that was known to them. So there's still the vulnerability to the system getting going from the start, which is worth noting.

Leo: Steve Gibson, Leo Laporte. We're talking security. And on we go with our wrap-up.

Steve: So a comic, we have to start with a comic because the fabulous...

Leo: Is it xkcd?

Steve: Yes.

Leo: Yes. Love it.

Steve: He just summed up Heartbleed in such a wonderfully techie, perfect fashion. So this was xkcd.com/1354. So if anyone didn't encounter it over the course of the last week, it's just great because last week we did a full deep dive into what heartbeat and Heartbleed is in TLS. And basically this cartoon could have saved you an hour of listening to me.

Leo: Well, it explains how Heartbleed works. I mean, it doesn't say anything about mitigation. But it does explain exactly how you're able to get 64K out of server memory.

Steve: Yeah. It's just - it's wonderfully whimsical and a simple graphic explanation. So xkcd.com/1354.

Leo: He's really smart. He really - he gets this.

Steve: Clearly. And my all-time favorite is that circuit diagram. Oh, my god. I always - I keep encountering it, and I just stare at for a while more and look at the spider web and the blob of solder and, I mean, it's just a classic cartoon. Clearly somebody who understands schematics did that. So, yeah, that never gets old.

So Robin Seggelmann said he accidentally - oh, yup, there it is, Leo. Oh, my god. It's just - oh, and that resistor, that resistor network, oh, god. There's a flux capacitor from "Back to the Future." Oh, no. The more you understand about engineering, it's just too wonderful. Oh, I love that little nest down in the lower - in the far left, the lower left. Oh, god, yeah, too fun. What number is that one? Or is it just - it's always linked.

Leo: xkcd.com/730.

Steve: Oh, that one is inspired.

Leo: Good one, yup. He's publishing a book, by the way, and I look forward to seeing that.

Steve: Oh, good. That would be great. So Robin Seggelmann, who was the individual who coded what turned out to be, as Schneier initially said, on a scale of 1 to 10, this is an 11. Many people said the worst Internet security event to happen in the history of the Internet. It's like, eh, okay. Well, it turned out not to be as bad, I think, as it could have been, certainly. Anyway, so he said, quote...

Leo: This is the guy who introduced the bug, Robin Seggelmann, yeah.

Steve: Yes. He said: "I was working on a research project at the University of Mnster using...." Or is it Mnster? I don't...

Leo: Mnster, Mnster, yeah, Mnster.

Steve: "...Mnster, using the OpenSSL encryption library and releasing bug fixes and new features that were developed as part of my work on the OpenSSL project." So he was at university, contributing. "The various changes were checked by a member of the OpenSSL development team and then incorporated into the official code." Which tells us that's how it works. "In connection with one extension, the TLS/DTLS Heartbeat extension, I failed to check that one particular variable, a unit of length, contained a realistic value. This is what caused the bug, called 'Heartbleed' after the extension. Unfortunately, the OpenSSL developer who reviewed the code also did not notice that a mistake had been made when carrying out the check. As a result, the faulty code was incorporated into the development version, which was later officially released."

And of course, as we know, from v1.0.1 through 1.0.1f, everything had that code, which had this unchecked buffer length claim. Basically the user said give me this much. And even though you hadn't given it that much for it to return to you, it returned it anyway, which often caused it just to give you up to 64K of stuff, whatever it happened to have in that area of memory.

So what was interesting is I found a timeline of the discovery and dissemination which a reporter, Ben Grubb, did, reporting for the Sydney Morning Herald. There's a link, because I'm not going to go through the entire timeline, because it just goes on and on and on. But the beginning of it is interesting because we now have evidence that Neel Mehta of Google Security discovered Heartbleed no later than March 21. So Google was aware of it several weeks before its official unveiling.

Then, later in the morning, and we have a timestamp on this one, at 10:23 a.m. - and all of these are Google time, essentially, Pacific time, Bodo Moeller and Adam Langley - Adam is, of course, famous for - we are talking about him often. He is deep into security of Chrome and the whole Google Chrome and Chromium project. They commit a patch - oh, that's how we know the time, because of the timestamp. They commit a patch for the flaw. This is according to the timestamp on the patch file Google created and later sent to OpenSSL, which OpenSSL forwarded to Red Hat and others. The patch is then progressively applied to Google services and servers across the globe. So Google, essentially, Google discovered this. And I was also originally unclear about how this Codenomicon deal happened. Turns out...

Leo: It's two researchers. But Codenomicon was in Mountain View. Right?

Steve: No, he's actually Finnish.

Leo: Yeah, but he was in the states.

Steve: No, well, I don't know where he was physically located. But it was an independent discovery.

Leo: Oh, it was independent, okay.

Steve: Yeah. So that's what was - and in fact it was due to the dual independent discovery that suddenly people started getting worried that, if people - like, lightning can strike once, and it's like, well, okay, what are the chances of it striking again in the same place? Well, it struck twice in the same place. So suddenly they were - that sort of amped up the concern that this needed to get remediated. So 10 days went by after the first indication we have of Google knowing of it on March 21st.

On the 31st, the last day of March, someone tells - and we don't know really who, but we just know when because we know what - content distribution network CloudFlare about Heartbleed, and CloudFlare patches against it and then later boasts on its blog about how they were able to protect their clients before many others were protected. So there was sort of a pact of secrecy among a small group which begins to fragment as leaks spring. And of course this is much easier to reassemble retrospectively than it is at the time.

Then on April Fool's Day, which was the Tuesday a week before we were first reporting on Heartbleed, which only happened the night before the April 8th podcast, which was last week's podcast, so a week before that, Google Security notifies OpenSSL about the flaw it's found in OpenSSL, which then becomes known as Heartbleed. Mark Cox at OpenSSL says the following on social network Google+. He says: "Original plan was to push a fix that week, but it was postponed until April 9th to give time for proper processes." And of course even that April 9th date didn't stick because we found out about it on the 7th.

Then on the 2nd, which would have been the day, well, obviously after April Fools'. In fact, I would have been worried about April Fools' announcement, for all the reasons we talked about. Then on April 2nd a Finnish IT security testing firm, Codenomicon, separately discovered the same bug that Neel found at Google in, of course, the same thing, the heartbeat, the OpenSSL bug. Friday on April 4th, content distribution network Akamai patches its servers. They initially said - and it's interesting, too, because, again, they know that they've done things that they can't disclose. They initially say OpenSSL told them about the bug, but the OpenSSL core team denies this in an email interview which was conducted by this reporter. Then Akamai updates its blog after the denial, and Akamai's blog then says that an individual in the OpenSSL community told them, not part of the core team.

Then rumors, turns out there are rumors on Friday, April 4th, within the open source community, about there being a bug in OpenSSL, but nobody has any details, so it ends up just being ignored. It's like, well, if you can't tell me anything more, okay. So it's just a rumor. And then finally, on April 5th, and this is something that sort of puzzled people was how it was that this Heartbleed sprung into existence with a cool logo, as several noted. So it was on Saturday, April 5th, that the Codenomicon group purchased the Heartbleed.com domain name and apparently began working on a cool dripping-blood logo for the whole thing. And if anyone's curious, this goes on to talk about the timeline after we've all found out about it, which is less interesting to me.

So to me, this is sort of a snapshot into how the actual world deals with something that they're hugely concerned about. I mean, initially Google believes it's the only organization that knows about this. They very quietly take care of their own network because, again, we talked about it last week, the whole cat-and-mouse problem of when we announce this, it's going to take time for people to fix their servers. And in that window from announce time to fix time there's a real heightened vulnerability. And in fact Bruce said on your TWiT podcast on Sunday, Leo, one of the many things that was interesting that Bruce commented on, he said a very careful look at logs has not revealed widespread scanning behavior.

So looking at - there are organizations that just capture everything, not just like protocol level or like the logs on an HTTP server that only shows specific requests, "get" and "post" requests to server, but actual raw traffic logs. And they're big because they have everything in them. But that's also why they're valuable is that they allow you then to retrospectively go back and look for things that you didn't know at the time were important, but you later learn are important, and then you want to know, whoa, was this going on before?

So it really - and the problem, of course, is that there is no place you can tap the entire Internet, if you're not the NSA. You're only tapping specific blocks, in the best case, like maybe all of a large class network of IPs. So you're not going to see targeted attacks against specific sites. Those are always going to be invisible to anyone other than somebody monitoring the traffic on that one, to that one IP or cluster of IPs, depending upon how large the site is. But you will see scanning activity. And Bruce said there was no sign of that. Yet, did he say within minutes or hours? I mean, it was like almost...

Leo: He said almost instantly, basically.

Steve: Yes. I think it was in minutes of the announcement widespread scanning traffic appeared.

Leo: Yeah.

Steve: So, and that was one of the problems, was this was not difficult...

Leo: It was really easy to weaponize, he said.

Steve: Yes. Yes, exactly. So we know that Google knew of it prior to release on March 21st. CloudFlare knew about it on March 31st. OpenSSL found out about it on April Fools' Day. Codenomicon independently discovered it the day after, on April 2nd. Oh, and they informed the National Cyber Security Centre in Finland, their own local cyber group, who found out about it the next day. And Akamai on the 4th, and apparently Facebook was also informed quietly, though no date was given.

So that's just the way this happens. Something is really big, and everybody's trying to do the right thing, and the right thing probably means successively disclosing to organizations with whom you have less close ties because you have less confidence in their ability and willingness to truly keep this quiet while remediation occurs behind the scenes in order to minimize the damage that's going to be done because everybody - the concern was, oh, my lord, we - their claim was they did get keys. And we'll talk about whether you could actually exfiltrate keys in a second because there were several people claiming that it didn't seem feasible until it was again several times proven to be feasible.

So, and I think that timeline that I just laid out is just the way it's going to be. You're going to have the major players who know each other in the security end talking to each other. And they're going to say, look, we found something bad you've got to fix. You've got to fix your networks. There's a small change that fixes this problem. Here it is. Get the old one off your servers. And the idea is you - so it's a secret that's hard to keep because it's significant. And so that says you're going to tell the fewest largest groups first to get the greatest number of servers fixed, that is truly doing the best service for the greatest number of end users because you're fixing the largest services first. And then you just sort of work your way down the hierarchy.

This pyramid explodes in terms of the size. It's not a square-edged pyramid. It's a spike that's highly spiked, and it drops off quickly in terms of the size at which networks shrink and the number of networks explodes as these are decreasingly influential. But I just think this is the reality of it. You're going to have an inner sanctum of people who are notified, and everyone will find out sort of hopefully quickly enough. And of course then what is incumbent, as we have seen, on the administrators of those who only learn at the same time that the attackers do, is that they act fast.

And so one of the - even though I think in retrospect this turned out not to be as bad as it could have been, the fact that Bruce thought it was an 11, and that people were breathless, that gave the security people the impetus to make the change. And in fact that's why it didn't turn out to be as bad. And so, Leo, it turns out you were exactly right when you compared this to Y2K. It's not that Y2K wouldn't have caused a problem. It's that it really did get fixed in time for it not to be a problem. And similarly, it was the relatively quick changing of certificates by all the people that really mattered because it looked like it was going to be a huge problem. Individually, for every single one of them, in a microcosm, it was a huge disaster. It was like, oh, my god, you mean people can steal our key? Well, we have to prevent that, obviously.

Leo: Andy in Sweden points out it does seem awfully odd that, after two years, it was discovered virtually simultaneously.

Steve: Isn't that interesting, yes.

Leo: That seems odd to me, but...

Steve: And I thought about that, too, since I learned of it. And I just sort of think there are sort of things in the air, like all these people are following the same mailing lists. And somebody'll say something. And maybe something was said on some mailing list that both of these guys subscribed to. And it wasn't about this, but it was about sort of something maybe in something else, and something sort of simultaneously led them both to look in the same area. So it wouldn't have even had - it wouldn't have been explicit. It's just, if you give hazy information to enough people, several people will wind up in the same place. And so I can sort of see how, you know, there's a just sort of network effect, that there's enough commonality within a community. Because these were both security guys, focused on security stuff. They're reading the same lists. And so nobody said, oh, go look at this line of OpenSSL. But they may have said something that just sort of led these two guys to be curious about the same thing.

Leo: Makes sense. Makes sense.

Steve: Now, what's really interesting to me is, given the number of servers that we knew were vulnerable, that there hasn't been a certificate revocation, what Netcraft calls a "tsunami." I have a link here. Probably if you just Google "Heartbleed certificate revocation tsunami yet to arrive," I would imagine - that's in the URL - you'll probably get taken to it. And there's a nice chart, I've got it here in the show notes, for anyone who has the show notes - and Leo, you can scroll down below, it's in the next page of the PDF - where they chart the rate at which their, you know, they're a network monitoring company located over in the U.K., Netcraft is. And great stuff. They've had a lot of neat pages about all of this as they've been tracking it.

So they said: "Activity on certificate revocation lists peaked at a rate of 3,900 revocations per hour on the day the Heartbleed bug was announced," that is, April 7th. On a typical Monday, they say, we would expect to see a total of about - now, this is really interesting. On a typical Monday they see 22,000 to 30,000 SSL certificates being revoked over the course of one day. So it's like, yow, 22,000 to 30,000 SSL certificates a day being revoked.

So what they were seeing on Heartbleed Monday, on April 7th, was 3,900 revocations per hour, so definitely more than average. On the day that Heartbleed bug was announced to the public, there were 29,000 revocations. So actually not above the high end of what they typically see. On the next day, Tuesday, 33,000 certificates were revoked, followed by 32,000 on Wednesday. Oh, and Mondays do tend to be higher because revocations don't occur on the weekend, I guess, just because...

Leo: Nobody's there.

Steve: No one's around, either at the certificate authorities or the companies. They're all home drinking their iced tea or whatever, not busy revoking things, or not issuing new certificates and revoking old ones. Because remember, not all revocations are due to theft. Most are, well, many, actually not most, because Netcraft also has a neat chart, a pie chart showing the percentage of revocations, or of revocation reasons where reasons are known. We'll be talking about that a lot more in two weeks. And they said on Tuesday 33,000, followed by 32,000 on Wednesday, so even beginning to drop. And then they say, "These were both above average, suggesting that around 5,000 certificates were revoked in direct response to the Heartbleed bug." But wait a minute, 5,000? I mean, that's...

Leo: That's tiny.

Steve: ...nothing. Then Netcraft reminds us about the formal certificate authority policy. I mean, this is not optional. And that is, by agreement, by contract as a certificate authority, you must revoke certificates within 24 hours if there is evidence of a key compromise: "A private key is said to be compromised if its value has been disclosed, or if there exists a practical technique by which an unauthorized person may discover its value. Arguably, all certificates on sites vulnerable to the Heartbleed bug" - and we believe that's about 500,000 - "should be revoked by now, as such a technique was successfully carried out by the researchers behind Heartbleed.com."

And subsequently it's been demonstrated several times by hackers who took up the CloudFlare challenge we'll talk about in a second and succeeded. So it's a little puzzling that we're not seeing revocations, published revocations by certificate authorities. It's, again, it's curious. One wonders whether people remembered to revoke what they were replacing, or whether they just replaced them, and they left their previous keys no longer in use, but still valid, which would be a little creepy, if that's the case.

Leo: They also point out that, because browsers do such a crappy job of - and we talked about this last week.

Steve: Oh, baby, and that's our topic in two weeks. I've tuned myself up into an expert on this topic.

Leo: Browsers don't notify you as often as they should of a revoked certificate.

Steve: Correct. Click this bit.ly link next, Leo, before I mention it, because the site is very slow, and I would rather not mention it until you're able to bring the page up.

Leo: Go ahead.

Steve: So that you can show it. So the question is how do we reliably - everyone's been wanting to know - reliably detect key reissue and possibly revocation? Now, many sites are attempting to do this. LastPass added a service to their offering where they would show you when the keys had been changed. I'd heard reports that unfortunately it wasn't reliable. And what I'm guessing, and I never had a chance to check with Joe at LastPass and see, was that they were just going by the issued date in the certificate. The point being that sometimes reissuing a certificate that uses a different public key - because of course you're going to rekey for safety, so you're going to have a different private key. But you're not really repurchasing a certificate. So that doesn't change the expiration date. It may very well not change the issued date. And I've tried to find anywhere where it was specified that a CA had to change the issued date if they reissued a certificate sort of under the same purchase agreement as the original one. I couldn't find anything definitive. And I did see some indications that issued date is not always changed. So that's not definitive.

What is definitive, naturally, is the certificate itself, which absolutely will have a different hash, that is, a different serial number, essentially. So we've talked about years ago, I think it was back in September of 2011 [SN-319], we talked about the whole certificate authority hierarchy problems and alternatives. And we've talked about, for example, Certificate Patrol, remember, which is an add-on for Firefox that I finally stopped using because Google was just issuing themselves certificates constantly, and it was just driving me crazy because Certificate Patrol kept telling me their certificates changed. Well, so there's one tool that is useful, but unfortunately not retrospectively. There's something else called the Perspectives Project which started as, I think, as an MIT research tool. It's continued to survive. And I created a bit.ly link for everyone: bit.ly/sn-

451.

Leo: Oh, you're right, it is slow. I mean, it took a long time for this to come up.

Steve: Well, and you should get some charts. They're scalable vector graphics charts, so they'll pop in right in that empty area there, yeah, Leo.

Leo: That's slow.

Steve: Now, here's what's cool about that. The idea was how about a different approach than using the certificate authority hierarchy? How about if we - and they call it "Perspectives" because, they said, how about if we had servers scattered around the Internet, all looking at the security certificates being issued by all the websites, and I'm not sure what that means. I mean, like, the popular ones certainly. Ah, there, you've got the graph finally.

Leo: Finally.

Steve: And isn't that cool. Now what we're seeing is a number of different servers that all detected the change in the certificate serial number between six and seven days ago that LastPass made. And that bit.ly link, sn-451, unfortunately, to go to this page, this notary page that queries the servers, you need to give it a nonblank quantity. So I just gave it LastPass.com because everybody knows LastPass. But you can, up above, you can put any website you want to in, which is in their database. And this page queries the servers that are running constantly, continually retrieving certificates from secure websites and logging the hash of the certificate. So you don't depend upon issued date. What you get is an absolutely definitive instance of them changing their certificate.

So I wanted to let our listeners know there is a very good way to know when a certificate is changed retrospectively. There's both a 10-day history and a - is it a 100- or 200-day history? I think it's a 200-day history of change. In fact, if you use that on GRC, I haven't had to change my certificate because I was using Windows, which didn't have this vulnerability because it wasn't using OpenSSL. And so nothing will show on the 10-day example. But you will see when I rekeyed for expiration a few months ago, when I got my new DigiCert certs and talked about how great they are, the certs and DigiCert.

So anyway, I wanted to let everyone know there is a way to definitively find out. You have to be a little patient. And as soon as I get a moment, I want to write a fast front end for the Perspectives Project. This site is really slow. And unfortunately, for the last week, because the word did get out, it's been, I mean, unbearably slow. It works. You just have to be very patient, you know, start the query, go away, and you'll get results. But GRC should have a snappy front end because it turns out the servers themselves are there. They're able to respond and give results immediately. There also is a plugin for Firefox. Firefox users can use this Perspectives plugin, it's now at v4.something, and it gives you instant results. So if you don't want to use this painfully slow website, and you are using Firefox, you can install the Perspectives plugin, and then go to a site, and then get the Perspectives readout on what that site - it gives you the same cool chart of like a timeline of when the certificate changed from all the various servers.

So in the process of the last week, one of the other things we had was a number of people saying, you know, we're not so sure there's really any danger at all. We know that the researchers claimed to have attacked themselves and obtained their own keys and passwords and other vulnerable things. We tried it, and we didn't get anything except noise. So we're going to call you on it. Robert Graham, who's well known in the security industry, he was the guy who did the BlackICE firewall years and years ago, back in the early personal firewall days. He extensively blogged about how he didn't think - and there were, like, arguments to support this about the way malloc() works and the way memory is allocated and all these reasons why it isn't the case that anything useful is going to be there. So for a while, those of us who were following this moment to moment were like, oh, okay.

So to their credit, CloudFlare, who as we mentioned before were among the very first to rekey their network when they got the word, they said, okay, we're going to create a challenge. It's at www.cloudflarechallenge.com/heartbleed. And actually they created that domain. They've since killed...

Leo: Oh, [indiscernible].

Steve: Yes.

Leo: Okay, go ahead.

Steve: They've revoked it. But what's interesting is - so first of all, they created a challenge. It was answered by two hackers who successfully recovered the private SSL security key for their - I think they're using nginx - for their nginx server and basically won the challenge. One of them then installed the CloudFlare private key on his own server.

Leo: That's one way to prove it.

Steve: And then said, put this IP address into your hosts file. Because you'll remember last week I explained that it was one thing to get the key, but you also had to convince the user's browser, and thus probably their computer or DNS, that the fraudulent server was at the stolen domain. So you need some sort of attack on DNS. Either a man in the middle would allow you to poison DNS on the fly; or, as I mentioned last week, modifying the hosts file. So this guy said put CloudflareChallenge.com and make it map to my IP in your browser, and it works. You could go www.cloudflarechallenge.com, and up comes the page. It's got the perfect padlock, everybody's happy, and you are not at CloudFlare.

Leo: Wow.

Steve: You are at this guy's site. So it was very cool.

Leo: That's one of proving it, huh.

Steve: Yeah. And we absolutely verified that in fact it's not easy. In fact, these guys had to - they wrote a script that pounded on the server. The idea was that, you know, memory is churning. And as all of the servers' different users come and go and log in and log out and are being served images and pages and everything, so there's a huge churn. So what they needed to do was they needed to create a high frequency of snapshots of 64K and essentially exfiltrate a stream, a constant stream of noise from the remote server and then analyze that noise for the fingerprint of a key. And after gobs of memory and many, many hours, they caught one, in two different cases. So yes, it works. Yes, it's a pain. It may not work on every server. And in fact Robert may well have been right that, for example, that on Apache, due to the way Apache allocates memory, it can't work; but on nginx maybe it can. I mean, so these are things, details that maybe people will work out over time. Or maybe we'll all just decide that it doesn't matter because we've all updated our certs.

Leo: All right. So Bloomberg says - and, boy, this just doesn't make any sense to me that they have two sources who confirm that this was planted by the NSA.

Steve: So as I - no, no, no. That the NSA knew.

Leo: Oh, that's different. You're right. Not planted, because we know who planted it.

Steve: Right.

Leo: That they knew about it early on, maybe day of, and have been exploiting it ever since.

Steve: So they wrote, Bloomberg - and this was very controversial. I just rolled my eyes. They said: "The U.S. National Security Agency knew" - they're stating it as a fact - "knew for at least two years about a flaw in the way that many websites send sensitive information, now dubbed the Heartbleed bug, and regularly used it to gather critical intelligence, two people familiar with the matter said." Now, I guess journalistically, if you have something doubly sourced, is that, like, the requirement?

Leo: Well, it was in "All the President's Men." It's just every publication has their own standard. But two, if they were two independent sources, that's a pretty good confirmation that it's true.

Steve: Well, okay. And so...

Leo: We don't know if these two sources were independent, by the way.

Steve: Given what I just told you about how you must exfiltrate data in order to get anything valuable, it's like, "used it regularly to gather critical intelligence."

Leo: It's not a useful tool.

Steve: It's a noise spray, is what it is. And it's like, okay. Yeah, two guys sucked out for like a day, trying to find something of value. The thing that really did upset people was the fact that servers wouldn't log this. Remember, it left no trace in the log. Well, if someone was looking at traffic from an IP, there was certainly a spike in traffic to the IP of the guys that were exfiltrating all this noise from the servers. But the problem is, as I mentioned, logging is on the other side of the connection. Once you get the connection, and a query is made, that's when an HTTP server will log the query. This is pre-query. This is you establish the TCP connection. Then you bring up the TLS handshake, and it's right then, while you're sort of holding, that you're able to try to pull all this data. So it's like, okay.

Well, first of all, these people refuse to go on the record. They refuse to be named. We don't know who they are. Could have been Mutt and Jeff, and they both told Bloomberg, oh, yeah, it's been going on for a long - for years and years, yeah. So I don't know. Did they know? We don't know. It certainly is the case that this is the kind of thing they would love to have known about. Maybe we just succeeded, maybe the best consequence is that we just succeeded in snipping a conduit of information from the NSA.

Now, the EFF, with their strong recognized bias, because it's worth mentioning, we have to remember who is saying this, said there does, well, I'm saying there does seem to be some evidence of previous, as in November 2013, use of Heartbleed. So they wrote, and to their credit, they first dismissed some probably false positive reports there because there were some early false positive reports. But then another bulk data log turned up. They said: "The second log seems much more troubling. We have spoken to Ars Technica's second source, Terrence Koeman, who reports finding some inbound packets immediately following the setup and termination of a normal handshake" - meaning setup and completion of a normal handshake - "containing another Client Hello message, followed by the TCP payload bytes," and then it's ones I've gotten used to seeing in the last week: 18 03 02 00 03 01 40 00. That's sort of its - that's the signature of Heartbleed, "...in ingress packet logs from November 2013." So for whatever reason Terrence was logging raw incoming traffic to wherever.

EFF continues, saying: "These bytes are a TLS Heartbeat with contradictory length fields, and are the same as those in the widely circulated proof-of-concept exploit. Koeman's logs had been stored on magnetic tape in a vault." Now, yeah, because they're big. I mean, if you're going to capture every byte regardless, then it's going to be a lot of data, and he's not the NSA, so it's not in a big server farm in Utah. "...[M]agnetic tape in a vault. The source IP addresses for the attack were 193.104.110.12 and 193.104.110.20. Interestingly, those two IP addresses appear to be part of a larger botnet that has been systematically attempting to record most or all of the conversations on Freenode and a number of other IRC networks. This is an activity that makes a little more sense for intelligence agencies than for commercial or [what EFF calls] 'lifestyle malware developers.'" So it's like, oh, that's interesting. Those are not random bytes. You're not going to get that by mistake. So we do, I think, have strong evidence that something late last year was poking. Although, again, a poke is not valuable except to say - except as a vulnerability test. So that was a vulnerability test against those servers, as opposed to any exfiltration of useful data because that takes...

Leo: Yeah, because once is not enough. I mean...

Steve: Exactly. That's going to take megabytes or gigabytes in order to just happen to come up with something useful. And I've got some other stuff here, but...

Leo: And I'm looking at it, and...

Steve: I'm looking at it, it's like, eh, we've sort of - we've covered it. And I think we've done our job here. Dan Kaminsky, I mentioned. Remember Dan, of course, from many, many...

Leo: Pwn2Own, yeah.

Steve: Exactly, Pwn2Own and also the famous DNS low probability spoofing warning from years ago. He waxed poetic about sort of what does this mean, what does the OpenSSL vulnerability mean to the Internet? And then it got so long that he wrote his own abstract at the beginning of his waxing. And he said Heartbleed - I'm going to share just the abstract with you. He said: "Heartbleed wasn't fun. It represents us moving from 'attacks could happen' to 'attacks have happened,' and that's not necessarily a good thing. The larger takeaway actually is 'This wouldn't have happened if we didn't add Ping.'" He's calling the heartbeat a ping.

He says the larger takeaway, I'm sorry, "The larger takeaway actually is NOT 'This wouldn't have happened if we didn't add Ping.' The takeaway is 'We can't even add Ping. How the heck are we going to fix everything else?' The answer is that we need to take Matthew Green's advice, start getting serious about figuring out what software has become critical infrastructure to the global economy and dedicating genuine resources to supporting that code. It took" - he writes three, but the number is actually two - "years to find Heartbleed. We have to move towards a model of No More Accidental Finds." Meaning being proactive. And being proactive means generating the funds and create an organization that identifies essentially hobbyist-written code.

Leo: Right. That's the issue, I think, yeah.

Steve: Which, as a consequence of its success and sort of organic popularity on the Internet, has gone mainstream, and everybody is using it, and no one has ever audited it. And so I think that closes the loop. That's really, I mean, this was a beautiful wakeup call for, like, oh, my god, as Schneier said, 11 out of 10 in terms of alarming. But what do we learn from this? It's that a guy in a university extended a protocol that the world, 66%, even though it turns out only 17.5% actually had this enabled or updated, two thirds of the world is using OpenSSL.

And we haven't talked about the fact that Android, the Android people have in their hands v4.1.1, has this vulnerability, and that there's really no mechanism for them updating the firmware in their phone. And that it's not just a server-side vulnerability, it's also a client-side vulnerability because the server, or whomever you connect to, is as able to ask you for a heartbeat back as you are to ask them. So it does affect clients, as well. And of course Cisco and Juniper both reported that their routers were affected. Those routers have OpenSSL in their large kernels. So I really think that's the takeaway.

And I had one last little bit, which is not about Heartbleed, but is another example, a perfect example of why security is hard. Because Google has just patched an Android icon permissions attack. Believe it or not, your icons can attack you.

Leo: [Laughing]

Steve: A group called FireEye found malware that could change other icons on the Android home screen, sending victims to phishing sites. InfoWorld reported this story. It was the security research firm FireEye. And it turns out that the malware is abusing a set of permissions known as com.android.launcher.permissions.READ_SETTINGS and all of that .WRITE_SETTINGS. Those two permissions have always been classified as "normal," which is a designation given to application permissions thought to have no malicious exploitability. Android users aren't warned about granting those permissions when they install an application because no one ever saw any reason why you couldn't let applications have those permissions. But using these normal permissions, a malicious app can replace legitimate Android home screen icons with fake ones that point to phishing apps or phishing websites.

FireEye developed a proof-of-concept attack using Google's Nexus 7 tablet running Android v4.2.2 to demonstrate that icons could be modified to send people to another website. During their tests, they briefly uploaded the application to Google's Play Store, only to demonstrate that it could get there. And then they removed it quickly, during which time nobody else had downloaded it. And they did that because Google's Play Store does check applications for security issues, but did not prevent this from appearing in the store. Then they tested it on the Nexus 7 running CyanogenMod...

Leo: CyanogenMod.

Steve: CyanogenMod, as well as a Samsung Galaxy S4 running Android 4.3, an HTC One running 4.4.2, and which are all that they had. Every one that they tested was vulnerable. All classified the read settings and write settings permission as normal. So I don't know, what is the whole patching story with Android and phones? Because a whole ton of phones now have this. I guess now what'll happen is Google will change it.

Leo: No, that's the problem. Nobody's going to change it. So the only one that's really vulnerable is 4.1.1. And unfortunately it's up to the carriers to push the update. So Google has gone well beyond 4.1.1.

Steve: Yeah. Wait, no, I'm sorry. I jumped from two different things.

Leo: Oh, I'm sorry.

Steve: Yeah, no, it's my...

Leo: I thought you were still talking about Heartbleed.

Steve: Right. That's my fault.

Leo: Okay.

Steve: So, yeah, 4.1.1 and vulnerability. Now it's worth saying that, again, what this means is that, if you connected to a server with your phone, that server could exfiltrate 64K. But we don't know that it's - that there's anything valuable.

Leo: We don't know what they'd get, yeah.

Steve: And we know statistically it's very unlikely that something is valuable. So it's like, eh. It's not clear to me that that represents a huge problem.

Leo: Right.

Steve: Yes. But these launcher permissions, the malicious icons...

Leo: Oh, I'm sorry. We're back to the icons. I'm sorry. I - it was my fault. You did say icons. I wasn't...

Steve: Yeah, so that's the one that has effect - it's always been this way. And so what I'm wondering is, so I guess what Google can do is they can reclassify the permissions and change their store filter so that apps are going to have to request this.

Leo: Google Play is updated by Google.

Steve: Right.

Leo: So this is not an issue at all.

Steve: Right.

Leo: Right? Because - yeah.

Steve: So they will prevent applications from getting in and...

Leo: So I'm rereading how this attack works. I mean, so they don't vet apps really that closely. But by the way, Apple, which claims to, lets malware in. In fact, this just happened the other day.

Steve: And we're talked about it. I mean, unfortunately it's...

Leo: You can't.

Steve: Yeah. And as you said, I heard you quote that 10,000 apps a day go through Apple to the App Store. And they do the best job they can. They'll take them down when they find out about them. But unfortunately they have to be reactive.

Leo: It sounds like the vendors are in fact the people who need to do this particular update.

Steve: Yeah.

Leo: So a Google device will be updated. Any Nexus device will be updated immediately. But if you have a Galaxy S4 or S5, it has to go through Samsung.

Steve: Yeah.

Leo: And then after Samsung, I think it still has to go through the carriers.

Steve: Right.

Leo: So you've got - it's difficult. I don't know.

Steve: So I have a media note for our listeners. When we first mentioned "Orphan Black" quite a while ago, a production of BBC America, or I guess BBC and it's now on BBC America, huge, we got a great response. For people who don't remember, this was this amazing show where one actress plays a large variety of roles as clones. And I don't know how you make TV like this where there's three of her in the room. And I mean, it is really well done. Anyway, we got a huge bunch of really positive feedback from listeners, so I wanted to give everyone a heads-up that Season Two starts this coming Saturday on BBC America, and that there will be, for anyone who didn't catch the first season, they're doing a marathon of the entire Season One back to back preceding that.

So you might check BBC America if you have access to it. Again, I recommend it without reservation. It's just very engaging, really interesting. SFGate commented that Season Two is delving a little more into the ethics side of cloning. You've got various forces back and forth, and an interesting story is woven around this. And SFGate said: "As good as it was last year, it's off to an even better start in its sophomore year." So very recommended.

I don't have any SQRL news because this entire week I have been saturated by Heartbleed and by my own work over on GRC. There's a lot more that I have to talk about which we will get to in two weeks when we talk about the revocation revelation, that is, we've referred to it a couple times, the poor job that browsers do of checking, even noticing that certificates have been revoked. The problem is certificates live for two or three years, and so there could be a multiyear window during which a certificate that has escaped could be abused if the browser that was using it didn't know that it was revoked. So I now am really up to speed on revocation. There's a number of pages over on GRC that are not yet public. They will be shortly. And we'll talk about it in two weeks. But I hope to finish that literally this afternoon, and then it's back to SQRL, which is where I want to get to.

And I just wanted to give a thank-you to the people who purchase site licenses for SpinRite. It's completely optional, and it's just the honor system. But what happens at my end is I see four - I hear four yabba-dabba-dos, like one after the other, just as quickly as Fred is able to give them. And what that means is that someone bought four copies of SpinRite. What's also neat is when I hear three because the way the licensing works, if you have one copy, you are formally entitled to use it on all the machines that you personally and privately own, forever. And "forever" is probably not that far from accurate, or that is to say is pretty accurate because here I am on the cusp of doing a major revision to 6.0 which is now 10 years old, and that's going to be free for everyone. So that ought to take everybody for another 10 years. And so you buy it once, and you probably get to use it forever.

But what we ask companies to do, if they're going to use it on, for example, all the machines in a site, is to get a site license, which is simply holding/owning four copies. And I kind of cooked up that approach because it allows someone to buy one, like the IT department to figure it out, to test it, see if it works, if it makes sense, and then they can upgrade themselves to a site license by buying three more. Or they can just buy four right off the bat, which often happens. And then what's cool is that, in the future, when I do move us to 7.0, that will be a paid upgrade, and then what we ask is for people to upgrade whatever they have. If you've got one copy and you want 7.0, pay for one. If you've a site license, upgrade your site license, which means pay for four upgrades.

So the whole thing is easy from our end. We're not having to, like, credit people for the one copy they purchased in order to test it before the decided they want to get a license and so forth. And it just sort of makes the whole system easy to run. But again, what's so neat is when I hear four yabba-dabba-dos in a row. It means that a company decided they were going to run SpinRite throughout their whole organization, which is always great to hear.

Leo: Congratulations, Steve. That is great to hear. Steve Gibson is at GRC.com. That's where you'll find SpinRite, and you can make those yabba-dabba-dos happen. You can also subscribe to the show. Well, actually you can download the show there. He's got 16Kb audio for the bandwidth-impaired, plus full handwritten human-created transcriptions.

Steve: It's funny, last week Elaine sent me a note when she was sending the transcripts back. "Why didn't I hear the garbage truck last week, Steve, during the podcast?"

Leo: We're getting good at putting an ad on top of it. That's why. So GRC.com. That's also where you can go to ask questions. And if we ever do another feedback episode - maybe next week.

Steve: We're going to, yes.

Leo: Don't promise.

Steve: If the industry allows us to. You're right.

Leo: Exactly.

Steve: We can't promise.

Leo: GRC.com/feedback is the place. Don't email Steve. But you can tweet him. He's @SGgrc on Twitter, and he does read all the tweets, even though he follows no one. They were giving you a hard time this week about that.

Steve: I know.

Leo: But he does read all his @ replies. You can also get full high-quality audio and video from our site, TWiT.tv/sn for Security Now!. But best would be subscribe. That way you'll get it each and every week, and you can do that in any netcast aggregator, iTunes and all the others. Steve, we are done.

Steve: We are, yes. For one week. We'll be back probably with a Q&A. I know that there's a ton of questions that people had. We'll go through them. And any that are still unanswered as of the end of this podcast, we will tackle next week.

Leo: Schneier and Kaminsky willing, we will be able to do questions.

Steve: Yes.

Leo: Thanks, Steve. We'll catch you next time on Security Now!.

Steve: Thanks. Thanks, Leo.