SECURITY NOW!

Transcript of Episode #475

Shocked by the Shell

Description: After covering a very busy and interesting past week of security and privacy news, Father Robert and Steve explain, examine, and dig down deep into the many fascinating details of the worst-ever, two-decade old, latent and pervasive Internet bug known as "Shellshock."

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-475.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-475-lq.mp3

SHOW TEASE: Apple lies about preventing spying, the FBI whines about us stopping spying, and Kevin Mitnick sells tools for spying. Also BASH leaves us crying in our beer. Security Now! is next.

FATHER ROBERT BALLECER: This is Security Now!, Episode 475, recorded October 1st, 2014: Shocked by the Shell.

It's time for Security Now!, the show that covers your privacy and security online. I'm Father Robert Ballecer, the Digital Jesuit, in for Leo Laporte, who's currently roaming through Europe. Here with me is the purveyor of truth, knowledge, and justice online, it's Mr. Steve Gibson. Steve. It's good to see you again, my friend.

Steve Gibson: Hey, Padre. It's great to be with you again. We've been anticipating this for some length of time, ever since Leo said that he was taking what he now regards as a too-brief vacation. He wishes, now that he was doing it, that he was going to be there longer. I've been hearing him grumble about it, you know, "I'm going to be here for TWiT on Sunday and then back for TWiT on the following Sunday." So he's missing no TWiTs. He has been prerecording his Tech Guy show on the weekends in order to be able to have that happen.

Yeah, so you and I get to do this today. And it's like an amazing opportunity. I did send Leo email, I guess like the middle of last week to apprise him of something that we'll be talking about at the top of the show, which is the result of Mozilla briefly switching their Mozilla.org certs to SHA-256 and then realizing the error of their ways and switching back. And so in the show there's a lot of week-to-week continuity. And so I realize that Leo is going to miss a lot of the conversation that is happening this week. And when that's happened before, I've found myself saying, "Oh, wait, yeah, you don't know about that." And so I was a little preemptive this time.

So we couldn't have a more fun topic because I started getting tweets Wednesday of last week when the news of Shellshock broke, people saying, wait, you just did Security

Now!. We need another one right now. And I said, eh. You know, and actually it's been good that it's been a week because there's really been nothing individuals had to do. Largely this was a function, it had the greatest impact against Internet-connected servers of various sorts. And it's taken a week for sort of things to settle down and for us to know better where we stand. So I think the timing is probably just about right.

FR. ROBERT: I actually think it's perfect because one of the things that we see every time a new vulnerability comes up is every network is jumping in to be the first to try to explain it, to try to explain how bad it is. It's really been kind of jumbled this last week. It's not even a week since the exploit was first released to the world, and so many people have gotten it wrong. There have been many patches that have failed because they don't quite understand the scope of the problem. And so now that we've had a couple of days to look back, I think it's easier to look at it and say, okay, so this is the issue, not this. This is the issue, not that. And thankfully we have you. So, I mean, I'm actually glad that this didn't break on Tuesday because, if it was Tuesday, it would have been a rush job, and then we would have had to correct errata.

Steve: Well, and it is a tricky technical problem. There was one person's communication posited that maybe this wasn't actually a bug. And they demonstrated some useful application of processing a command after the function. And we'll rewind this and start at the beginning here in a second.

But the point, but what I was put in mind of was something very controversial from the early days of this podcast, the so-called Windows Metafile bug, where in looking at the code that was in Windows at the time, it looked to me to be deliberate. That is, not that it was a bug, but that at the time, back in a day before there were bad guys - and there actually was such a day, and I think that's where this same BASH problem originated, or some of the practices that ended up being able to exploit this characteristic of the BASH shell.

Back in those days, what Microsoft did with the Windows metafile was they had, basically, they had an interpreter which would interpret the contents of the metafile. But there was one command that said jump into the metafile, meaning take it out of interpretation and into native code execution. And I could easily see, like, the original designer of the Windows metafile, saying, you know, it might be handy if we weren't stuck just using the interpreter, if a metafile could also contain native code.

Now, again, not something you would ever do in this day and age. But back in the Windows NT days, or even earlier than that when the original metafile format was put together, it wasn't that unreasonable. And we sort of had that same thing happening with Shellshock where now when we look at it, everyone's just, [gasp] oh, my god. But it's really, it sort of harkens from a time when it wasn't that crazy to do what is now common practice. And now we look at it, though, with our contemporary understanding of the way exploits are being found and how what used to be innocent is now considered a horrible vulnerability, it's like, oh, yeah, that's really not so good.

FR. ROBERT: Well, I mean, when you consider the fact that the Bourne-Again Shell was released back in 1989, so, I mean, we're talking about a utility that was created over two decades ago.

Steve: Before bad guys.

FR. ROBERT: Before bad guys. And before we really had a concept of what the Internet would be, that you would have your server facing the rest of the world. That wasn't even in the programmer's mind. And so it makes perfect sense that they would try to make

BASH as functional as possible. And I think, I'm right with you, I don't think this is a bug. I think this was something that someone included in the original implementation of BASH as something that if, you know what, if we were connected to networks that we didn't trust, it would be very useful.

Steve: Yup.

FR. ROBERT: And unfortunately we've continued using the same tool, and it has vulnerabilities. I mean, can you imagine if we were still using Microsoft DOS from 1989 as our primary way to communicate with the outside world? You wouldn't do it. And there would be people crying foul, saying, oh, there are so many bugs, so many exploits, so many ways for someone to destroy your system. And it's, well, yeah, of course, because that's not what it was made for.

Steve: Right, right.

FR. ROBERT: So but the question now becomes, even if this is not a bug, even if this is intentional, why are we still using it?

Steve: Well, let's talk about that. We've got some news to cover, also. We'll talk briefly about Apple's OS X update, which they put up yesterday, but not with a great deal of emergency. Also some news has come to light about the iOS 8 MAC address randomization that turns out to be almost next to worthless. We were excited about it when we heard about it. It seemed like a cool thing that they had done to iOS 8. Turns out it's, eh, they've really overblown what they're doing.

Also I wanted to chat with you a little bit about what's been in the news this week relative to the head of the FBI complaining now about the fact that corporations have adjusted, as we knew they would, we talked about this a year ago, the Snowden revelations and the sense we've had that there's been some stretching of the U.S. Constitution. I do want to talk about Mozilla's experience with SHA-256. And a friend of ours, I don't know if you ever had the chance to meet Kevin Mitnick, but Leo and I both know him pretty well. And it turns out he's opened what he calls the Absolute Zero-Day Exploit Exchange, so buying and selling zero-day exploits. So we've got that to cover, and then we're going to do lots of deep - a deep dive into Shellshock.

FR. ROBERT: So in other news, no real news this week, then.

Steve: Yeah, it was a sleepy week.

FR. ROBERT: Any of those stories - iOS 8, Mitnick opening up the Zero-Day Corner Store, or Shellshock - could be the main story any other week. The fact that we've got them all in the same week, something's weird happening, Steve. I don't know what's going on.

Steve: Perfect time to have you.

FR. ROBERT: Perfect storm. Where do you want to start?

Steve: Well, at the top of the notes. I always try to put a picture now in the show notes because I post them online. I tweet the URL, although it never changes except the number increments. But someone sent me this, I think it was yesterday, and I just loved it. It's a picture of an airport security door keypad. And the trick is, can we guess the number? And I just love this because, for those who can't see, it's a three-by-four keypad, the number-style pad with a star and a, I don't know what that is in the lower right. Normally it's a pound sign. But the point is that it teaches us a lesson about the nature of security because, on the first day this was installed, it looked perfect.

Now there are only four buttons that almost have the paint rubbed off of them. And we know why. We know it's because those are the ones that are being pushed. And they happen to be 3, 4, 5, and 6. So not only do we know absolutely which buttons contain the code, but I'd even wager that this door opens with 3456 because, again, yeah, you know, in the beginning I'm sure it, you know, no one would be able to guess that it would be 3456. They want to make it easy to remember, and so not some random combination of numbers. And when the pad was first installed, you'd have no clue. Now you look at this, and clearly the 3, the 4, the 5, and the 6 are the most often-pressed buttons there is. Or there are. So it's clearly what the code is. Anyway, I just got a kick out of that.

FR. ROBERT: Although you could combine that picture with a little bit of wear analysis. JammerB, if you go back to the picture, I remember someone was trying to tell me, similar to this, if you look at the way the doorjamb is shaped, someone would probably be opening it with their right hand, which means they're probably going to rest against the side of the keypad. Looking at where the wear is actually happening on the keys, so on the lower right-hand corner of the four, on the lower left-hand corner of the six, and top of the five, and then dead center on the three, it could be 4563 because that's where your fingers would naturally go if you were resting your hand on the side of the...

Steve: Yeah?

FR. ROBERT: ...of the jamb.

Steve: Good point. Good point.

FR. ROBERT: But, yeah, I mean, the fact that you can, I mean, even if you had to do complete guesses, you go from a 12-key keypad to a four-key keypad.

Steve: Yes.

FR. ROBERT: It's just now I have to look through airports to see how many of those doors look like that.

Steve: Anyway, I loved it because it's just a classic failure of security, something that starts out seeming like a good idea. You know, sort of like a fence that just sort of decays and falls apart. And it's like, even the cows are looking at it, thinking, you know, if I really wanted to go anywhere, this fence would no longer keep me in.

FR. ROBERT: But fences only keep honest people honest; right? So keypads aren't really going to keep you out if you really want to get in there.

Steve: And lazy cows in.

FR. ROBERT: And lazy cows in.

Steve: Okay. So support.apple.com/downloads will take all of our Mac listeners to Apple's current support page. And the first item in the upper left at the time of this recording, support.apple.com/downloads, is Apple's apparently optional offer to patch OS X, OS 10, sorry, for the Shellshock bug. So I wanted to put that at the top of the show. This requires no reboot. It's a tiny download. It's like 3.5MB or something. So it is easy to get. You install it the way you install any Mac app. It just says, okay, I'm going. You have to give it your password, of course. And then it fixes itself, and it says, "I'm done." So it's not a crucial emergency because it's not going to be an exposure that most Mac users have.

But now, essentially, one way to look at this, and we'll come back around to this for the main topic of the show during our second half. But a way to look at the whole thing is that there is now, probably somewhere in every UNIX system, a component that could be invoked in a way that could give the invoker more control over the system than its owner wants. So that component should be removed. It's just not good to have it there. It's the sort of thing where you don't want to set up intrusion detection systems on your front lines to hopefully block any incoming known exploit because this thing is going to be with us for a long time.

And it is, already it is - there's just been a flood of Internet traffic, you know, trying to exploit this across the entire Internet. There have been white hat hackers who've been scanning for it in order to get some sense for it. But a flood of exploits against random IP addresses. Everybody is picking this up who is looking at their incoming traffic. So sooner or later, I imagine maybe with the next major update, Apple will include this, just fixing this. But for our listeners who are more security conscious, because it's small, quick, easy to do, requires no reboot, when you've got OS X running, just go to support.apple.com/downloads and fix this. Remove this problem from the systems that you're responsible for.

Again, not probably an emergency. It doesn't look like the typically configured Mac has a vulnerability. The only way I was ever concerned was there were some reports, and they were confirmed, of a malicious DHCP server being able to use the DHCP client in a Mac to invoke the shell and cause a problem. And so that would have affected people. And DHCP is the Dynamic Host Configuration Protocol. It's the thing that the system uses when it comes up or when the network connects to obtain an IP address from the network. And again, the nature of the way some of these have been implemented did invoke the shell. And so if you were on a malicious network, that is, a network with a malicious DHCP server, which sort of hasn't been a problem in the past because DHCP wasn't vulnerable, again, this is a perfect example of the many different subtle ways that this could still get you in the future. So it's worth removing it.

FR. ROBERT: I actually read a report from Incapsula. They've been a guest on one of my other shows, This Week in Enterprise Tech. And they were saying how just on the domains that they protect, so that's about, what, 4,115 domains, they've seen since the attack, and this was up to Monday, since the attack was released, about 217,000 attempts to exploit BASH. And their guestimation is that there's probably been about a billion attempts so far. I believe they're going to come on this Friday on a special edition of TWiET to show us what a stream of BASH exploit attempts actually looks like. That actually should be some pretty decent TV.

Steve: Cool. I have that also at the end of our notes, in today's show notes, is from FireEye. They did an analysis. And I grabbed samples of the strings that they're seeing. And we'll be talking about that here in about an hour.

So we were excited when we learned that MAC addresses were going to be randomized by iOS 8 devices when they were not connected to a WiFi network. Which seemed like a fabulous idea. Before iOS 8, your phone or your pads that have WiFi turned on, and as you're roaming around, you're in stores and malls and so forth, it's always scanning, probing for networks that it may know. And traditionally it's been giving out its MAC address. And the MAC address is supposed to be a globally unique, 48-bit identifier, globally unique because it has to be unique on a single Ethernet network. And since you never know which devices may also be on the network with you, and it has to be unique or you'll have a MAC address collision, and those of us who've ever played with Ethernet networks extensively know that that's not a good thing to have, so they make them all unique.

Well, what this meant was that, essentially, you were a beacon as you walked among stores and in malls and movie theaters and so forth, basically broadcasting a token that represented you. So the idea that Apple was going to respond to this and make them random until you actually connected, first of all, that's a very clever hack. So the moment you hear about it it's like, hey, that's very nice. Turns out, no. Based on some analysis performed by Nick Arnott at iMore.com, who took a much closer look at iOS 8's MAC randomization, it is almost never active. That is, only when the phone is in a deep sleep state for whatever reason is this in effect. And it's like when the phone awakens out of deep sleep, even with the screen still dark, to receive push email. Then it's not randomizing the MAC. Anytime anything rouses it in any way, it returns to its fixed MAC.

Now, I don't know why because the hack that they originally described seems entirely feasible. So I hope they get some pressure to, like, make this better. Maybe they couldn't do it for architectural reasons. Maybe there was a glitch. Maybe they were - someone was lazy, and they didn't care enough to do it as well as they could have. I haven't looked at it closely enough to understand why.

So my upset, I mean, it's like, [frustrated sound]. My upset is that they told us something which is not true. And that's the one thing I don't want. It's like, it's fine if you're going to offer something that's weak, but better than nothing. But don't advertise it as your MAC address is always random until you're connected. That's not factually true. And Apple's on this big campaign now about them not collecting data at all. And exactly, there's the page. And privacy is built in, and rah rah. And if you scroll that page down, I think to the second to the last item on the page, you'll see their statement.

They said: "When you're out running errands with your phone in your pocket, WiFi hotspots have the ability to track your movements and behavior by scanning your WiFi MAC address. A MAC address is a string of characters that uniquely identifies your device on a network. With iOS 8, we've introduced an innovative feature designed to protect your privacy by randomizing your device's MAC address when the device is passively scanning for WiFi networks." Not true. "Because your MAC address now changes when you're not connected to a network" - not always true - "it can't be used to persistently track you." Not true. "This is in line with Apple's industry-leading effort to do away with persistent identifiers and is unique to iOS devices."

So my problem is that's a lie. That's a complete mischaracterization of a feature that they're touting. So they've got to fix this. Either tell us the truth, or make it true. It would be great if they could make it true. Again, I'm only annoyed that, based on Nick's reporting, this is almost never true. It's, I mean, you could make it true, but you might as well just turn off WiFi. I mean, because our phones and devices are always coming out of a deep slumber to look around and see what's going on, to receive push connections, to receive text messages, to do everything that they're doing while they're operating in the background. All of those events drop this out of MAC address randomization.

So maybe there's an architectural limitation. If so, they should have just kept quiet because it's useless the way it is now. And I'm, obviously, you can tell, I'm really annoyed because, like, just don't lie. Lying is not what you want to do if you want to convince us that, for example, you can't listen in on our iMessage sessions. You've said that, too. And I don't believe that. So...

FR. ROBERT: You know, the only time that this would actually work is if your smartphone wasn't acting like a smartphone. I mean, the whole idea behind carrying your phone with you is that it's going to be able to notify you when something important is happening. You're getting an email. You're getting an iMessage. Well, any of those events is automatically going to put the phone in a mode where this isn't going to be applicable.

Steve: Right.

FR. ROBERT: Even then, what it sounds like to me is, when I heard about this, I thought, oh, cool feature. Actually, I'd really like to see that. I thought Apple got it right. But this sounds as if this was just a PR thing. Some PR person was talking to an engineer, saying, well, what are some of the new features that you're working on? And they heard "privacy," they said, great, we'll sell that. And the problem with doing this, and yes, you've said this much more passionately than I have, is that if you try to sell people on security features that turn out not to be security features, then the security features you actually have, I have no reason to trust you that they actually work.

Steve: Exactly. Exactly. And so, I mean, they've sold something that they've said, basically, you can no longer be tracked with your MAC address. That's not true. That is, they have said you cannot be tracked until you connect to a network. Not true. Period. Now, it's also been noted that what the MAC is doing is also transmitting a bunch of SSIDs, that is, there's a fixed list of SSIDs that it's got in its beacon. And that's not subject to change, even when the MAC address is randomized. So there's still persistent sticky stuff which can be used, but not globally the way the device's MAC address can be. I mean, that really, the MAC address is a nasty global tracking item. And we were led to believe they had fixed that. They didn't fix it, and they told us they did.

FR. ROBERT: Right. And I think that's the biggest problem. Because people should turn it off. If you really don't want to be tracked, your phone can't be emanating anything. I'm not just talking about the MAC address.

Steve: That's a very good point.

FR. ROBERT: You've got to put it into airplane mode. Everything shuts off.

Steve: Yes. Cell tower connectivity gives you, I mean, in all the movies now, that's how they're localizing people is checking their cell towers, which ones they're connected to.

FR. ROBERT: [Sighing]

Steve: Yeah, really.

FR. ROBERT: When you start sacrificing the integrity of your security solutions in order to make some PR hay, I worry. And this is not just Apple. I'm sure Google's doing the same thing. Google has been also doing this big push on, well, our new phones, they can't be tracked, and they're NSA-proof. Anytime I hear PR slogans like that, I just - I think either your engineers aren't being honest with you, or your PR people are lying. It's one of the two.

Steve: Well, yeah. And in fact this comes back to the famous comments about iMessage. Tim made them on what's his name's show on PBS. I can't think of the talk show host.

FR. ROBERT: Charlie Rose?

Steve: Yes, on Charlie Rose. He made a big point of saying we cannot intercept iMessages. Except we know that they manage the keys. That is, when we send a message out, they provide the other party's public key, which we use to encrypt messages for the party. So it's true that they cannot decrypt those messages. But since the key management is transparent, nothing prevents them from putting another public key into the key bag for themselves, in which case they will be able to decrypt the message. So saying that they don't have the keys is different from saying we're unable to give you a key which we could use. They clearly have the ability. So again, it's like, well, yeah.

FR. ROBERT: Yeah. A smart person once convinced me that the only way to have true privacy is if you own all the keys. Nobody else has a copy to anything else. And if you lose them, they're gone. If Apple is saying, well, we back up your iMessages in case you have to recover them, it almost implies, "and there's a way to recover your password," and therefore that means there's a way to recover your key. Ergo, it's not secure. Ergo, if they get a warrant, and the NSA really wants to look at your iMessages, they're going to be able to. Again, just, you know, be honest with us. Tell us that this is secure as long as we don't get a warrant from the NSA, someone tapping us on the shoulder saying, "This is a legal warrant. You need to turn over your data."

Steve: Yeah, or that your MAC address is randomized is long as your phone is off.

FR. ROBERT: Right, right. Although the people in the chatroom, they're good. They recalled that case. Do you remember - this might actually have been before the iPhone was released. There was an FBI operation where they were able to listen in on some Mafia folk in New York, even though a BlackBerry was off, because they were able to turn the BlackBerry back on without it looking like it was on, and they were able to listen in to the microphone. And so now that's conspiracy theory. It's wait a minute. So unless I can take the battery out of my phone...

Steve: Exactly.

FR. ROBERT: ...I can't ever assume that it's off.

Steve: And, gee, none of these Apple devices have removable batteries.

FR. ROBERT: Although I hear that if you bend the 6 Plus, it no longer - it can't listen in anymore. So maybe that's actually a security feature they couldn't tell you about.

Steve: Okay. So we have to talk about, because this made a bunch of news this week, and I know you'll enjoy talking about it with me, that James Comey explained that he was "very concerned" about new Apple and Google privacy features. He said that he was concerned that the two companies are, quote, "marketing something expressly to allow people to place themselves above the law." And I heard in the news podcast just before this one, they were talking about this, and noting, I mean, basically that this was a reaction which we predicted a year ago in the wake of the Snowden revelations, I guess almost a year ago, that we have the technology to absolutely lock this stuff down. And it'll take a while for it to get deployed. But once it has, it's game over for the intelligence agencies. This is not an insolvable problem if we choose to solve the problem.

And the argument is, as we've been covering all the Snowden stuff, these companies were begging to be allowed to disclose what they'd been compelled to do because, without being able to do that, consumers were fearing the worst. And they were not allowed even to tell the truth, even to, like, say, give a number of requests that they had honored and so forth, as if a number was actually a national security issue. I mean, basically the federal government has, I think by any measure, pushed their position way too far. And so now they're complaining when the technology companies have reacted, giving their customers what their customers want. And which unfortunately for law enforcement is all too readily possible.

FR. ROBERT: I'm going to play minor devil's advocate here.

Steve: Okay, good.

FR. ROBERT: I don't believe [indiscernible]. I believe in privacy. I believe that you own your data. I believe that, if you take the steps and are willing to put the resources into locking down your information, then it should be locked down.

Take a step back, and let's go before the Snowden revelations. You could honestly argue in a forum, without some major lash-out, that the government had the right in extreme cases to be able to read communications, with a warrant. I think most of us would have actually said, okay, yeah, that makes sense. In fact, he mentions it when he's testifying. He's saying, look, there should never be a closet that's locked completely from us. If we need it in order to recover, say, a kidnap victim or stop a terrorist activity, then most people would say yeah.

The problem is, and you pointed this out, is the government has so far overstepped that we're no longer willing to even give them that little leeway. We're saying, look, we were willing to give you the extreme circumstances clause. But you treated it as if everything was an extreme circumstance.

Steve: Yes.

FR. ROBERT: So again, it kind of harkens back to the Apple situation. Why would I trust you? Why would I believe you? If you're going to cry wolf every single time and then tell me it's for my own good, I'm just going to stop trusting you. I'm going to lock everything away.

Steve: Well, yes. And I completely agree. If there was a facility which under a warrant allowed this, I mean, that's the way our system has functioned, like from its founding. And it's the issue of the judge and the warrant, which has been explicitly bypassed. All of this, well, that's why it's called "warrantless wiretapping." And the argument is like, well, tap it now because we need the information now, we don't have time to get a warrant, and similar things. I mean, we've been hearing about basically these constitutional protections just being ignored. I mean, and the fact, for example, that simply by calling someone a terrorist, they can be locked up without - and all of their civil rights suspended.

I mean, again, I understand the law enforcement argument for the instance that you can create that demonstrates the necessity. But now the country is subjecting its entire population to that argument, en masse. And that wasn't the way the system was designed. And of course the problem is, and in fact there was a story that I ran across, and I don't have it in my notes today, it just flew by like two days ago, saying that the Chinese government was found leveraging and abusing some ability that Google had to decrypt or monitor its networks somewhere. And so this was a demonstrated example of what is the concern. And that is, if there's a backdoor, it won't only be opened under warrant by law enforcement. It's inherently able to be opened by others. And, you know, that's a concern, too.

FR. ROBERT: And that's the plot of every hacking movie ever, that we always put in backdoors when we program a system.

Steve: Right.

FR. ROBERT: Let me ask you this. If we were to move to a perfect security system in which everyone had their own keys, and they were locked into their own devices, and the government couldn't access them in a technologically feasible way, you could say, and you'd be protected under your Fifth Amendment rights, "I don't remember my password." They can't persecute you and prosecute you if you say you don't remember your password. But if you, say, locked down all your devices with your fingerprint...

Steve: Yup.

FR. ROBERT: ...they could compel you to touch the fingerprint reader. Yes?

Steve: Yeah.

FR. ROBERT: And that's perfectly within the law.

Steve: Yes. And in the recent court cases that we've seen, it's been, if it's something you know, you cannot be compelled. That's considered testimony against your own interests. And as you said, the Fifth Amendment of the Constitution protects us from divulging that. But if it is something physical, a physical key or a fingerprint, then you can be compelled to supply that.

FR. ROBERT: The first time I heard about this, do you remember those little sensors that were available for PCs, the U.are.U sensors?

Steve: Yes.

FR. ROBERT: It was the first time you had a low-cost, relatively accurate sensor that you could plug in via USB and get encryption for pretty much everything. I was really big into that until we had a Jesuit who was a lawyer in the community, and he said, you know, that actually wouldn't protect you from search and seizure. That's a physical item. They could compel you to put your finger on that reader. If you're actually concerned about having your information snooped on by law enforcement agencies, that's not the best way to do it.

Steve: Yeah.

FR. ROBERT: And surprise, you know, 15 years later it's actually still relevant.

Steve: Okay. So this was really interesting. We talked a couple weeks ago about what I consider to be Google's heavy-handed mistake in announcing that, starting next month, this is October 1st, so in November, they're going to accelerate the deprecation of what they consider to be unsafe security certificates, web server identity certificates signed by SHA-1. They're saying that the SHA-1 signature is no longer strong enough to protect us. And Microsoft has already said they're going to stop honoring them starting in 2017. That's the hard deadline that the industry has accepted. We received that information from Microsoft back in November of 2013, so almost a year ago. And nobody complained. That gave us all of '14, '15, and '16, three years, to know that SHA-1 would be no longer supported by Microsoft, that Microsoft is a large enough presence that that effectively means no one else can use them, effectively.

So Google decided to just be preemptive, to march this way forward, starting next month. And in a series of releases of Chrome, starting with 39, then 40, then 41, they're going to incrementally alarm their Chrome users about the certificates that sites are using now, that have certificates signed using the SHA-1 hash, even though they are signing all of their properties, all of the Google servers are now signed with SHA-1. They're not changing that. There's no indication that they are. And in fact now we know you can't really.

I loved this little bit that I saw on Mozilla.org under Bugzilla. Two guys in a dialogue, chatting about Mozilla's attempt to switch their servers over to SHA-256 today. Someone named Jake Maul first posted: "SHA-2 certs on www.mozilla.org cost us around 145,000 Firefox downloads per week." Chris - I have a typo in my notes, it's Chris More - said: "Yes, please don't change SSL certs on www.mozilla.org without checking with #www or #webprod" - I guess two people that are known - "as we killed one million downloads recently by switching to SHA-2. A lot of the world is still running old browsers and come to our website to get Firefox." Jake replies: "Let's re-issue the cert for www.mozilla.org as SHA-1, expiring 2015-12-31. That gives us the maximum amount of time to support old users without breaking Chrome or IE. At that time, we may just have to start trying to detect WinXP users (ideally just pre-SP3, but that seems like it would be hard to detect) and force them to a non-SSL connection."

And he writes: "Sucks, but better than giving them an error and making it impossible for them to download Firefox. Firefox is a great fix for them because it ships with its own SSL stack and thus avoids the underlying OS limitation." And Chris finishes, saying: "Let's keep SHA-1 on www.mozilla.org until we find a better solution. Switching to SHA-2 will kill 5% of our downloads, and that has a direct impact on ongoing Firefox usage unless we have a better solution to deal with legacy browsers. Let's start the discussion now in a separate bug on how to handle legacy browsers before December 2015." And in fact...

FR. ROBERT: But Steve, wait a minute. I mean, I understand why they want to do this. This was a bit of a panic on their part. But doesn't this just push the problem down the road? I mean, aren't we going to run into this in December 2015 again? Because as far as handling the problem, it's basically you have to upgrade your browser.

Steve: Okay. So XP pre-Service Pack 3 has no knowledge of SHA-256. So, and it turns out, believe it or not, I mean, these are the users, apparently, 5% of the people visiting Mozilla.org have a pre-Service Pack 3 version of Windows XP. Windows XP and Chrome on Windows and IE on Windows all use the OS native cryptographic stack. So IE and Chrome on Windows pre-Service Pack 3 cannot connect to any web server signed with SHA-256.

FR. ROBERT: Right. Which is why Firefox is perfect for them, because it has its own stack. But...

Steve: Precisely.

FR. ROBERT: I mean, if you look at Windows usage numbers, especially for XP, they're not going down nearly as fast as we thought they might. Even though it's been deprecated, it's end of life, Windows XP is still running strong.

Steve: Right.

FR. ROBERT: So it gets me back to that point, which is, well, at the end of 2015, the people who are using XP now are probably still going to be using XP. If they haven't switched, it means they're not interested in switching.

Steve: Right.

FR. ROBERT: So you're going to run into this again; right?

Steve: That's - yes. But it's more than two years from now. We have all of, I mean, it's plenty of time. So, I mean, even the other certificate authorities, or actually all the certificate authorities, were really upset. The CloudFlare people were really upset because this meant that they had to do a mass rekeying with very little notice. The argument was, look, basically Google just dropped the bomb and said, everybody must change, or we're going to start penalizing sites that don't.

Now what we see is sites themselves will be penalized because the world isn't ready to change. And so my position is, yes, it is only kicking the can down the road, but it's another two years of sites experimenting like this and beginning to come up with solutions to push people up to SHA-256. Especially when Google themselves are not switching. They can't switch because they don't want the lack of connectivity that the Mozilla experiment demonstrates you get today.

FR. ROBERT: Yeah. So it's a kick down the road, but it's a really long kick. And maybe in two years all those XP machines will die.

Steve: Well, I guess my feeling is I won't feel the sympathy for them in two years that I do today. They will, if nothing else, have had plenty of notice. This just isn't sufficient notice.

FR. ROBERT: Right, right. I can see that. I can see that. So it's all about timing. I mean, you could just wait for this problem to organically go away, rather than forcing a major change that's going to have a great impact on the people who are actually supporting your solution.

Steve: Yeah. And, I mean, and I can see Chrome being proactive, maybe during 2016. For example, we've already established a hard deadline, thanks to Microsoft, at the beginning of 2017. So it would be beautiful during 2016, for the year leading up to that, for Chrome to start warning people and thus warning the sites through their visitors that this is looming. But Chrome is doing that in 2014. They're doing it this year. They're doing it next month, they're starting this.

FR. ROBERT: They're overachievers.

Steve: Yeah.

FR. ROBERT: Steve. Kevin Mitnick.

Steve: Yeah. I guess the world considers him, or at least in the past, one of the most notorious hackers. He was jailed, I think, for about four and a half years. He spent many months in solitary confinement. There were a lot of people who were upset with the way Kevin was treated. There was like an industry of Free Kevin T-shirts and mugs and things. And Leo and I both know him. We've met him. He's really a nice guy. I mean, he's like - he's really a nice guy. And so I don't quite understand what's happened here. I guess it might just be that he wants to make some money. But he announced that he is going to start selling zero-day exploits to, quote, "discerning government and corporate buyers."

These will sell for no less than $100,000 each. And although Kevin's company, MitnickSecurity.com, employs its own hackers, apparently to dredge up some of these, mostly he's serving as an exchange, a zero-day exploit reseller, purchasing exploits from those who discover them, and reselling them to his customers. I don't know. I'm just sort of at a loss. It just seems, you know, he was regarded as - I think he was wrongfully regarded as a black hat. Then he sort of was wearing a white hat for a long time. And I'm afraid now it's sort of gone to dark gray because, I don't know, this seems to sort of endorse the notion of zero-day exploits. I mean, it gives entities a means of acquiring them.

There are different things you can do. You can sign up to be on a notification list so that, I guess, if you were Microsoft, and they acquire a zero-day exploit for Windows, Microsoft will have some notification means. The problem is that that dramatically reduces its resale value. So it's against Mitnick Security's interest to notify Microsoft because then Microsoft can patch it. And the only reason to buy a zero-day exploit is either to preemptively patch a problem - and a $100,000 is a lot of money to pay for that. Seems to me the market is clearly in exploits which are going to be used maliciously or for surveillance. Somebody who has deep pockets is going to be paying $100,000 for zero-day exploits. And they're going to want the largest window of exploitability possible. Which means there's got to be something in the agreement saying that this will be kept absolutely secret until something.

So anyway, it's an interesting site, if anyone's interested. He calls it The Absolute Zero-Day Exploit Exchange at MitnickSecurity.com. And there's a lot more to read about it, for anyone who's interested. I just thought it was, like, you know, I mean, interesting and maybe a little troubling.

FR. ROBERT: Can I pay for the zero days with bitcoin? That's the old - let's get serious here for a second. Let's talk a little bit about how the security researcher field has traditionally worked. And I know plenty of security researchers. In fact, one of the ones I've been following recently, have you met a man by the name of Dan Geer?

Steve: His name I know.

FR. ROBERT: Yeah, he was the keynote speaker at Black Hat this year.

Steve: Okay. Yeah, that's why I know the name.

FR. ROBERT: I'll talk about some of his theories a little bit later because that actually ties into Mitnick. But if you're a security researcher, especially if you're sponsor of security researchers, so you work for one of the main research companies, typically you look for these zero days. You look for these bugs. You publish a report. So you show exactly how it's used. You show exploit code.

Steve: Demonstrate it.

FR. ROBERT: You demonstrate it. You offer it to the company that's vulnerable so that they can patch it. And then you make your money off of selling your consultancy to companies, to say, look, are you sure you're patched? We can do all of your vulnerability testing. We've developed a software that will allow us to look through all your systems and tell you where you are and are not vulnerable. And typically, if your security research company is able to find these zero-day exploits or these bugs in a timely fashion, then you make a decent amount of money. That's a security researcher.

Steve: Yup.

FR. ROBERT: What Kevin Mitnick is doing is arms supplier.

Steve: [Laughing]

FR. ROBERT: Because there is no disclosure. There is none of this old world, I'm researching so that I can make the world a better place. It's, no, I'm researching so I get paid.

Steve: Yes.

FR. ROBERT: I mean, and you touched on this. No one's going to pay $100,000 except a company trying to exploit another company, or a government trying to exploit another government, or its own people. And, you know, it really - you know him. I don't know him except by reputation. But I've always seen Kevin Mitnick as sort of like he's the antihero, you know, he's the guy who fought the government, he got away with it, and smirk-smirk, he's still doing what he's doing.

Steve: Yeah.

FR. ROBERT: This just - it feels so wrong.

Steve: Yeah, it's sad. I mean, this is not the Kevin that I remember. So again, it's like I said, I don't know what happened. Wow. And I love your analogy. I think arms supplier is absolutely apropos because we're no longer in a world where the Internet is optional, and where the notion of cyberwarfare is an abstraction. I mean, it is surprisingly real. I'm lagging behind a little bit. I'm a little bit more still old school, where it's like, oh, come on, you know, really? But yeah, really.

FR. ROBERT: At Black Hat 2014, Dan Geer made a statement that sent chuckles and then some nervous chuckles through the audience. And that was he thought that the U.S. federal government should actually buy up zero-day exploits and then release them. That should be part of their job. Part of their job should be to keep the infrastructure of communications safe. And if that meant buying up zero-day exploits from hackers who would otherwise exploit them, basically give their money on the front end, rather than on the back end, then they should make that part of the responsibilities. It's interesting that, you know, because I know Kevin was there, I'm wondering if he heard this and said, okay, that's not a bad idea. How about this? We'll sell it to you. And if you're a good government, you'll give it to your people.

Steve: You know, one of the oddities of the software industry that has existed from day one is the license agreement that says "We're a company selling a license to use our software. You're going to pay us money to do so. But we're not going to give you any warranty of any sort over its merchantability or fitness for use or the fact that it has or has not any bugs." So, I mean, and car companies would love to have that deal. So would drug manufacturers and all kinds of other commercial enterprises that are producing property and intellectual property.

Somehow the software business has always had this hold-us-harmless clause in their license. And I guess they're able to do it because of the power that they inherently have.

Microsoft was in a position to say, okay, so don't use DOS. Good luck. Don't use Windows. Good luck. You had no choice except to relinquish all right. But it's weird that here we are in a situation where - and this comes from what you were saying, that a company, a commercial company with an incredible amount of cash, like Microsoft, is producing a series of operating systems and programs - Internet Explorer, Office, Outlook Express, oh my lord, that is completely vulnerable for years, creating all kinds of damage, and then people are suggesting, well, you know, maybe the federal government should further inoculate them by purchasing the bugs in their code and selling the solutions back to them.

And it's just like, what? What has happened? How does any of this make any sense? It's just so upside-down. But that's the deal that software's always had, is just, you know, a zero liability situation. I mean, nobody else has it.

FR. ROBERT: That's actually another point that Dan Geer made in his talk, which is he wanted accountability for security. Specifically, he said, look, we understand that developers are going to have bugs. Bugs are part of programming. But he believed in a world with total transparency. He wanted zero transparent failure. So when you have a software solution that fails, it is your responsibility to make sure that people know about it as soon as possible. And that's shining the light on zero day. Zero days don't exist if people aren't afraid to tell people that, yeah, we failed. Eh, kind of strange.

Now, did you see the little shot that Kevin Mitnick took? If you look at the Wired story, there was a security researcher - do you know Christopher Soghoian?

Steve: Oh, yeah, of course.

FR. ROBERT: Of course, yeah. He's a big, big voice against the use of zero-day exploits.

Steve: Yes, yes.

FR. ROBERT: And he fired off a kind of a snarky tweet. And he said, "Well, look, if you're going to sell zero-day, which I don't recommend, exploits using a convicted felon as a broker, well, that seems unnecessarily risky." And Mitnick shot back, in typical Mitnick form, "My clients may use them to monitor your activities. How do you like them apples, Chris?" So, and again, that kind of breaks my vision of what Mitnick is.

Steve: Yeah.

FR. ROBERT: I can see him like this kind of a person. And I'm wondering, did you nail it? He just wants to get paid. Like I've been doing this game for the longest time, it's time for me to actually be comfortable.

Steve: Yup, cash in. And I think he's wanting to leverage his celebrity. He certainly has celebrity. Everyone knows the name Kevin Mitnick. So he just figures, hey, you know, I'm going to get paid.

FR. ROBERT: Yeah.

Steve: He does have a book coming out, not quite published, called "The Art of Invisibility." And it was described, I don't know what copywriter wrote this, but it says the world's most famous hacker reveals the secrets on how citizens and consumers can stay connected and on the grid safely and securely by using cloaking and countermeasures in today's era of Big Brother and big data." And it's like, yes, unless they buy any of the author's zero days and use them [indiscernible].

FR. ROBERT: I think we already actually talked about how you stay secure. Turn off your phone, don't ever get on the Internet. That's about it. That's the only way you stay secure. Oh, and by the way, don't get in a car that goes anywhere near a security camera, a traffic cam, or a license plate cam. And also don't call anyone on the phone, and don't send any mail. You probably shouldn't leave your house.

Steve: And don't use the transponders used to run...

FR. ROBERT: FasTrak.

Steve: FasTrak, exactly, the FasTrak transponders.

FR. ROBERT: Why, by the way, you have no choice anymore. You know that. In the Bay Area you can't go across a bridge without that. They're not taking tolls on the Golden Gate, and they're going to repeat that down the chain here.

Steve: Wow. So in Miscellany I have just one quick little thing. I ran across what I thought was just a charmingly clever idea. We talked years ago about the problems of webcams on laptops, and now often on desktop machines, that is, of the camera staring at you. And my advice, my first advice was just take a small piece of the sticky top of a Post-it note and stick it over the lens. That is, you just don't want this thing staring at you. There are too many past and known ways for that to be used to eavesdrop. In fact, we covered the famous case of an elementary school that was loaning these laptops out to their students, and the security firm that they were using were turning the webcams on in the students' bedrooms in the evening and spying on these elementary schoolchildren.

FR. ROBERT: Also known as "child pornography."

Steve: Really, really...

FR. ROBERT: I remember [indiscernible], yeah.

Steve: Really chilling. So, and then one problem with that advice was that, if the system used the webcam to measure room illumination, then you would be losing the automatic brightness control. So then the second-generation solution was use a piece of the frosted Scotch tape. That would completely frost the image so nothing would come through, yet allow the light through in order to still give you brightness control. Somebody came up with just a simple, elegant, I just loved it, solution. I tweeted it. It's on Kickstarter for only $5 for anybody who can't do their own.

I actually have a fascination with magnets, so I've got just a huge library of magnets. But you're showing it now on the screen. It's called Nope, N-O-P-E. And it's Nope Live Free on Kickstarter. So imagine two very thin disk magnets that are stuck to each other. And they naturally just sort of roll around each other. I'm sorry, they're stuck edge to edge, not flat, but edge to edge. One of them is sticky, has adhesive on the back. You stick that just to the side of the webcam hole. And then the other one is able to sort of roll around, but easily roll and just sort of stay put, they sort of naturally stay put, over the webcam hole. I just - it's so clean. It's very elegant. If you're a person who never uses your webcam, then, yeah, sticky tape probably makes sense. But if you're like a frequent - say whatever it is you're doing is like several times a day you're using Skype, then it'd be nicer to have essentially a cover.

We've talked about covers that you can slide open and closed. This one you just sort of roll the other magnet off to the side, opening the shutter, essentially, or bring it closed again. Anyway, just wanted to make sure our listeners knew. It's very simple, very clean. I mean, if you find a source of magnets, you could just do your own. But anyway, I just really - I thought it was great. So thank you, Mike Cunningham, for tweeting that and for bringing it to my attention. And the Kickstarter is doing really well because it's just, you know it's capturing people's imagination, saying, hey, I think that's kind of a cool idea.

FR. ROBERT: Yeah, it's simple and it's elegant. It actually looks kind of nice. But Steve, I need something to cover over the microphone.

Steve: [Laughing]

FR. ROBERT: I'm actually more concerned about what they'll hear, rather than what they'll see, because my laptop's always closed.

Steve: And actually that's a very good point. Oh, and I should note that a couple people did wonder whether the magnets were thick enough to interfere with a laptop cover's closure. That is, often laptops do have a gap between the lid and the bottom. But some of them don't. Some of them come down, right down tight. So if you were using it on a laptop, because there is some nonzero thickness to these little magnets, you'd want to make sure that it's still going to be compatible with you being able to close your laptop. Otherwise it's not so convenient.

FR. ROBERT: Not so convenient at all. But wait, we've got something big coming up because we do have to cover Shellshock, yes? Now, Steve, the main event. Everyone has been talking about Shellshock. How much should they be afraid of this? Is this bigger than Heartbleed? Is this actually going to affect them? Is this just something that system administrators have to worry about? Tell us all about it.

Steve: Okay. Okay. So let's sort of get a foundation of understanding the problem. UNIX has, that is, to give some underpinnings for those who aren't UNIX developers and hackers and users and so forth, and understand how this happened. UNIX has always taken a sort of a toolkit approach involving the invocation of multiple small separate apps. That's just sort of the way the UNIX culture of development works is you have a bunch of simple, capable tools, and you script them together. You build up an ad hoc solution to solve a problem.

There are sort of two main ways of communicating between these separate pieces. One is to pass things on the command line, either as arguments or so-called "piping," where you pipe the output of one of these into the input of the next in sort of a chain, and each one processes it in some fashion and then passes it on to the next. Another convenient means of passing arguments is to use the so-called "environment." The environment is the wrapper around the executable. And among other things, the environment can have name value pair variables. Meaning that you have a name equals a value, and really extreme flexibility over what the values are and what the names are.

People who've been around for a long time will remember back in the DOS days the path variable that tells the operating system which set of directories to look for executable files in. There's a large set of standard environment variables which the operating system and applications refer to for their own operation. But they can also be created on the fly. And we can think of it sort of like a named scratchpad where data can be placed, and then another program can be invoked, and it looks in the environment for the variables, for the parameters and the arguments.

Well, the BASH shell, which as you mentioned earlier in the show is decades old, it processes the environment when it starts up. It's become very powerful over time. And even back then it was scriptable. You could write small programs in it. You could use it to invoke other programs. So it could sort of be the hub. One of the things that it could do is you could define small functions, that is, executable functions, in an environment variable. And when BASH started up, it would read in the environment, sort of incorporating it into itself, and in the process define those functions.

So in this sort of toolkit glue approach, one of the things that was done very often in the earlier days was, as we began to have data submitted by users, submitted by web users to web servers, the so-called CGI was the typical way that - it stands for Common Gateway Interface - was the typical way that a user submission, that is, which is always a query, remember, users query the server, the user's query would invoke some code running on the server which would receive the data that the user sent. And whatever that program emitted, whatever it generated, would be the response that would go back to the webpage.

Well, just because of the design of UNIX, the various parameters, the headers in the user's query were put in the environment because in the environment was a convenient way for the CGI script to reference things like the host header and the excepts header and the various query headers which are used by the operating system, or, I'm sorry, the headers that are submitted by the browser and then used by the CGI side to process the query.

So this is the way things sat for several decades, until several weeks ago somebody noticed that when the shell, the Bourne Shell, which technically stands for Bourne-Again - there was a Bourne Shell, B-O-U-R-N-E. This is the Bourne-Again Shell, B-A-S-H, thus BASH. When it is invoked, and it sucks in the environment and processes the values of the environment variables that it finds there, it turns out that after it processes a function definition, it continues looking at the balance of the value of the environment valuable. And if that's a command, it will execute it.

Now, as we talked about at the top of the show, that's not necessarily, like, a horrible thing. You could imagine back 20 years ago when the Internet wasn't, hadn't happened, that the designer of this could have thought, hey, you know, I'd like to be able to define a function and then invoke the function. So you can actually do that, or could do that. It's not clear today whether that still has survived. You could, at the beginning of the environment variables value, have a function definition, you know, open parens, closed parens, open curly brace, contents of the function, close curly brace. That's the function definition. But then afterwards you could refer to the function that you just defined.

So you could argue that this never was a bug, that this was a feature that the original implementer of this extension to the shell thought, hey, this would be a handy hack. And it no doubt is. And it's entirely possible, just due to the nature of everything else going on, that this is sort of a relatively obscure feature of the shell that wasn't being used very often, that nobody just noticed until a few weeks ago that, over time, as the Internet did grow and mature, applications using the Internet continued to also use the shell. That is, maybe began using the shell, and then it was handy.

So, I mean, the shell is always there. On a given system you can count on it. It's the default shell for UNIX and Linux systems generally. And so because it's there, it's handy, you can count on it, UNIX developers just used it. I mean, it became one of the tools that Internet-connected things took advantage of to do some of their processing. And so the nature of what became thought of as a bug, certainly an exploit - although it may have been an exploit by design, innocently, several decades ago - was the observation that, if anything that a user outside of - a remote user connected to a server, if anything they did could allow their actions to create values in the environment, and then either in that same event or subsequently they could get the Bourne Shell to be invoked, they could use the fact that even a null function, that is to say, open parens, closed parens, open curly brace, semicolon, which is a null statement, closed curly brace. That's a null function. But then the balance will be handled like a command.

And if they could write that variable and something that they subsequently did invoked the Bourne Shell, they were essentially - it's hard for me to even say it. They were essentially executing a command of their own design, remotely. With the privileges of the shell, which in many instances are root or much higher than restricted privileges. And so...

FR. ROBERT: You know, Steve, I have no trouble believing that this was a feature. I mean, it does sound useful.

Steve: Yes.

FR. ROBERT: I put myself back in 1989, 1990. Here's the problem I'm having, which is I know a lot of UNIX guys. And they all tell me they never knew about this. So if it was a feature, it was an undocumented feature, and then a lost feature. Whoever put it in there just never told anybody else about it.

Steve: Yeah.

FR. ROBERT: You know UNIX way better than I do. Do you know anyone who even hinted that this was possible with BASH?

Steve: No. And subsequently to this coming to light, I've seen people demonstrate the utility of it. I think the only way to know, and unfortunately it's probably lost now, maybe there are, like, older copies of the source. Certainly source from before a few weeks ago, before the mad patching, so, yeah, obviously there's going to be source around, would be to examine the source and see whether this looks like a mistake, or whether it looks deliberate. Because this sort of thing, I would think that the source code would tell you whether this was a feature that the programmer, like there's a comment that says, you know, upon encountering the closing curly brace, continue parsing the line for anything else that we need to do, with the function we just invoked. That would be a comment that the source would have if this were deliberate. So I have not looked at the source to answer the question. And I haven't read anyone anywhere either way saying yay or nay.

FR. ROBERT: Wow. Steve, we are going to get back to BASH and the vulnerability that's shellshocking the world. But before that, let's hear some good news. I want to hear some SpinRite testimonials.

Steve: Well, just one. We've been talking about SpinRite and RAID arrays for the last few weeks. And in fact our talking about it has incented other people who've used SpinRite to successful recover from RAID problems. We talked about one, I think it was last week, where the array fell off the back of the moving truck. The good news is it wasn't moving at the time, but the person was moving. In this case, this is sort of a different one.

Bob Guarda in Ottawa, Ontario, Canada, he said: "Hey, Steve and Leo." And of course in this case Padre. He said, "Just want to say hello, and that I am a longtime listener of Security Now!, and I'm also a proud owner of SpinRite. As we all know, we 'computer guys' do the support for family and friends, and I'm no exception to this rule. Here is a story for SpinRite - a success story, that is.

"A friend of mine has a Netgear ReadyNAS network NAS box. When he purchased it, I set it up for him with four drives and RAID 5 configuration. At the time I told him that this is a very good configuration because, should one drive fail, the information is still available. The odds of two drives failing, well, that's super high. This would help in keeping all of his pictures and videos safe.

"To make it even more safe, I set up the built-in backup utility of the Netgear NAS to do a weekly backup to an external drive connected to a USB port. One morning, he gets an email from the device saying that Drive 1 had failed, and that another drive was about to fail. The NAS box had turned itself off for protection. He calls me up in a panic, and I tell him no problem, the worst scenario is that he has 'lost' [in quotes] one week of work. He then admits that the backup had not worked in months. He kept forgetting to tell me. Yikes.

"He contacted Netgear for help and, after a valiant effort from both their Level 2 and Level 3 support at Netgear, the [in quotes] 'solution' was to get a data recovery company to save his data. It appeared that Drive 1, 2, and 3 were all dead. When he inquired at a local data recovery establishment, cost was $3000 starting, no guarantees. Yikes," he writes. "Then I recommended" - oh, he says, "Then I remembered that SpinRite does not care what the file format is on the drive. Be it NTFS, FAT32, MAC OS, Linux, or RAID, it does its magic.

"So I said to my friend, 'You have nothing to lose since you already lost it.' He agreed, and let me attempt to save his data. So I did Drive 1, 3, and 4 at SpinRite Level 2. Drive 2 was giving me the click of death, and the computer would not recognize the drive. I figured, well, it's supposed to be RAID 5. If I saved the other drives, the Netgear box should rebuild. I replaced Drive 2 with a new drive and placed the freshly SpinRited," he says, parens, "(a new word?), drives into the NAS enclosure. I powered up the unit and, voila, the RAID started to rebuild.

"After the RAID finished rebuilding, all of his work files and, most importantly, the videos and pictures of his two kids were available. Thank you, Steve, for making such a great product. And keep up the great work you do with Leo with Security Now!, one of the best sources of security information there is out there. Thanks. Bob, in Ottawa, Ontario, Canada." And so, yeah, even having a RAID, as we have been talking about, isn't often enough protection. If more than one drive dies, in this case three died. Or I guess, no, all of them died. SpinRite was able to fix all of the that were still even willing to get online, and that was enough to rebuild the RAID and get this guy's data back.

FR. ROBERT: I didn't know it worked for RAIDs. I actually have, we have a little Netgear ReadyNAS over there, an NV+, that I've got to give it a go because its drives died. And I just - I kind of threw it away. I'm like, ah, I didn't want to deal with it. I want to see if SpinRite will get it back because I completely forgot that it doesn't look at the OS. It doesn't care what's on the drive. It's actually just looking to see if it can read sectors. It's going sector by sector and saying, I can't read this, move it someplace else.

Steve: Yup.

FR. ROBERT: That's fantastic.

Steve: And it does recovery enough that you can get things back in order to recover your data. So, whoo.

FR. ROBERT: Although I have to say, he was saying his friend received an email from the RAID array essentially saying, yeah, I'm dead. That's not what happened. I know the ReadyNAS system pretty well. It would have been sending him constant emails saying, hey, look, the error count is going up. You really should probably do something about this. And then when...

Steve: It does sound like his friend was being a little negligent all along.

FR. ROBERT: Right. Which just goes to show you, we could have all the tech in place to make sure that this stuff is redundant. But if we're lazy about it, it doesn't matter how redundant it is. It's going to fail.

Steve: And that's why I'm able to have blinky lights behind me and a couple employees to answer customers' questions. It's what's made my life possible, is people almost always wait too late. Few people, you know, the problem is they turned the drive on day before yesterday, it was fine. They turned their computer on yesterday, it was fine. They turn it on today, dead. And, I mean, so really preventive maintenance has been demonstrated to be fully useful on hard drives. But people generally don't do it. They wait until it's too late. And I get to be the hero, which I'm really glad for. And as I've said before, I don't mind if someone helps a friend out when they're in need. There are plenty of people with hard drives in the world, and we'll be fine. But I just like to have our listeners protected.

FR. ROBERT: And I have to say a lot of people will balk at, oh, $89 for something I'll use only once? No, no. It's $89 for something you're going to use on your worst day, at least up to that point.

Steve: Well, and actually for the rest of their lives. I mean, SpinRite is about 25 years old now. We honor upgrades even from Version 1.0. Everybody knows that as soon as I get finished with SQRL I'm going back to SpinRite 6.1 and then probably moving on to 7. So the purchase you make will - you can use it on all the machines you own, now and for the future. I'm going to keep it alive.

FR. ROBERT: I think I have a Version 3. I may need to update that. It's been a couple of years. Oh, and by the way...

Steve: Okay.

FR. ROBERT: ...if you ever use it on someone's computer that has gone completely dead, they will think you're magic, and they will wonder what this wonderful tool is that you've used.

Steve: That's right. You get to be the hero and the guru, and then I'm your hero.

FR. ROBERT: All right. So we've had our dose of good news. Thank you. After all the vulnerabilities, after the BASH bunch, we got a little dose of happiness. So drop us back into the sadness.

Steve: Okay. So what I found really interesting was the industry's understanding of this, the graph was instantaneous. Invariably, we were going to see the press comparing this to Heartbleed because Heartbleed is relatively recent. It's the last most recent big bad thing that happened. And so people were saying, oh, is this better, or worse, or how does this compare to Heartbleed? And the answer is it doesn't compare at all.

The way to think about Heartbleed is that there was some concern that, if a hacker tried really, really, really hard, really fast for a long time, they might end up capturing a buffer that contained some sensitive keying information from a server. After the initial announcement of the discovery of this potential buffer overflow, there were security researchers who said the region of memory that you capture is unlikely to contain anything. So CloudFlare, among others, they put up a couple test servers and said, "See whether you can crack this." The answer was yes. Hackers were able to obtain keys after a lot of hard work. They got lucky.

And in fact there was another exploit, I'm forgetting which one it was, but a few weeks we talked about one that was big and bad and was believed to have been another exploit of Heartbleed against that organization, a machine at that organization. As I remember, it was a router. It wasn't a main server, it was something in their network, like a VPN server or something. But something was exposed to Heartbleed, and keys were extracted. Oh, yeah, they were able to get VPN authentication keys and then get onto their network and then perform the intrusion that way. So, okay. So that's the nature of that.

This is completely different. This is, if you find a server that is vulnerable, and it's trivial to find them, you basically - you can send them pings with your own IP address, and it will ping you. And when you get pinged, you know, oh, I just found a - I just executed a ping command on a remote web server. I can execute any command I want. Consequently, this thing got a 10 out of 10 on severity, a 10 out of 10 on impact, a 10 out of 10 on exploitability. The access vector was remote network, the worst it could be. The complexity was low, the worst it could be. The authentication, none required. I mean, across the board. This was red. I mean, this was unbelievably bad.

And whereas when Heartbleed first was announced, you know, everyone sort of said, oh, that doesn't sound good. We know buffer overflows are not good. But we're not sure you're going to get anything of any value. Well, and it took a while for, you know, two people to demonstrate, oh, we tried for a week, and a hundred thousand billion gazillion queries as fast as we could make them, and we captured some keys. This is, you know, I basically sent out some probes, and 3,000 servers replied. And that was only in a small piece of the Internet that I chose to check. And I could have executed any command with relatively high privilege on any of those servers. I mean, this is night and day amazingly bad.

And my very favorite hack is one that the guys at FireEye found. They looked at their blogs and posted a number of the so-called techniques that they saw. My favorite is a no-malware reverse shell. You send a query to a web server, which is just GET /cgi-bin/. So you're just - you're making a query at the root of the cgi-bin directory. Oftentimes something is there to pick up the query and run. You give it a specially formed - the query gets a specially formed user agent string, meaning that the cgi interface will take the user-agent header in this fake query of yours, stick it as an environment variable called "user-agent," and then the value. And then what you give it is a command.

It's not common knowledge, which is to say many people are unaware of the fact that BASH actually has built-in commands for sending and receiving network traffic. You need no other software to be installed on the system. BASH itself can send and receive network traffic. This simple command causes it to open essentially a reverse shell to the IP you provide, on the port you provide. So you're an attacker. You open a terminal session, listening on a port. You send this command to a remote machine, causing its bash shell to connect to you on this port in interactive, "-i" means interactive mode, and you now have an interactive session with the root shell in this remote machine. And it works.

FR. ROBERT: And not only does it work, it's really easy to make it work.

Steve: Yes.

FR. ROBERT: There are a lot of hacks that require timing, luck, and more than a little bit of skill. This is, as you said, if you send out pings, you can find a bunch of vulnerable machines. Throw the code at all the machines and see which ones respond and start up an interactive session, and you're in. You're essentially sitting at the keyboard.

Steve: You have the dream, yes, you have the dream of every hacker on the planet, which is an interactive command line inside the heart of a server, a remotely located server that you just got access to. So you could do anything you want. You poke around. You start sending the log files and the passwords file and, I mean, it's just - it's unbelievable.

FR. ROBERT: Yeah. This is not Heartbleed, where, okay, I might be able to capture some of the communications. Maybe I'll even get a credential that will let me in. This is, no, I'm not waiting for you to do something stupid, I could just take over.

Steve: This is not a backdoor. This is a front door.

FR. ROBERT: This is the front door [laughing]. The thing that gets me about this story is, as you said, it's got legs. This, if I were still doing anything questionable, this would be the entry point for an advanced persistent threat.

Steve: Yes. So now...

FR. ROBERT: I wouldn't use it right away. I would just make sure that I'm deep enough in the system, I've got enough privileges, that I'll be able to exploit something later on when I want to. That's what I would - I would be doing a rush to get into every system before it gets patched.

Steve: So, yes. Now we know the nature of the problem. Here's the reason this thing has legs, is that it has breadth and depth. The problem is that there is a vast ecosystem of stuff on the Internet. Of course, famously, the Internet of Things. Now, the IoT devices tend to be much leaner than big-iron servers. So, and BASH, because it has all these capabilities, is a big shell. There are much smaller shells which tend to be more often used in consumer routers and, you know, unfortunately, and light bulbs and switches and things. But there's all sorts of other stuff.

For example, we know that webcams use the BASH shell to essentially make their little web server. They've got little brain - I don't want to say "brain dead," but I started to say it. Simple, simple-minded. Simple-minded web servers just written in BASH script. So webcams that you can connect to can probably be taken over. I'm not sure what you could do with them, but you don't want your webcam taken over. The point is that the shell is universally omnipresent. And there are all kinds of things globally sitting on the Internet connected. And it has been the habit of developers who are just putting together a quick hack for one thing or another to use the shell and to use some of these powerful features without understanding. And the problem is a lot of this stuff gets baked into products. It gets baked into firmware. Or that was 10 years ago, and the developer who knows that's what he did has gone on to greener pastures. He's moved on to other companies.

Yet this stuff is sitting there on ports, waiting for connections open to exploit. Not just port 80. Not just 23. Not just, you know, the common ones. But appliance ports and random function ports. The problem is this is just scattered throughout the infrastructure, the deep infrastructure of the Internet. And that's why we're going to be encountering this issue, I predict, for years to come. It's going to take a long time for this to drain out of the Internet. I mean, and to some degree it never will.

FR. ROBERT: There's going to be an Internet support group for IT administrators who are - right now they're trying to think of every box they have sitting in a server, every forgotten piece of equipment that they haven't patched, they haven't updated in the longest time because they never thought it was important. Because I guarantee you, if I go through my inventory, I can't remember every box I might have that has this vulnerability.

Steve: Yeah.

FR. ROBERT: I mean, imagine how far back this tool goes. That's how much time there has been to forget that you're using this on a box that's now going to become a vector for infection of your network.

Steve: Yeah. And in fact, the intrusion detection systems immediately updated their patterns, trying to look for this open parens, closed paren, open curly brace, not much inside, close curly brace pattern, and catch it at the border. The problem is there are many ways of obscuring that, and other ways to get this in. And so it's in general, I mean, it's better to have that than not. By all means, have that. But you just - you can't count on it. And there are doubtless devices that are not behind IDSes.

FR. ROBERT: I know. Actually, I was just thinking about that. I have a few IDSes running in Brian Chee's colo right now that I'm pretty sure have this BASH vulnerability. Now, that would be ironic. Someone just owned that IDS. Ah, Steve, is there any good news? Is there some good news about this? I mean, of course people have been applying patches, and the patches have worked like gangbusters; right?

Steve: Yes. Well, okay. So the very first patch came out and didn't solve the problem. Then - this is one of the neatest things about open source. What I love about open source, I think more than anything else, is what happened this past week. Yes, it's unfortunate that this thing sat for two decades. And no matter how many eyes were looking at it, nobody was looking at this particular aspect to see the problem. So we could argue that open source - and, I mean, this is not the first time we've seen a failure of open source. I mean, the Heartbleed vulnerability itself was introduced two years ago by a well-meaning programmer making a change where he didn't do a range check and should have. So even the fact that it was - the fact that it was open didn't protect that from happening.

What I love about open source was the pile-on effect because everybody on the 'Net simultaneously had access to the source of BASH. In mass multitasking, everybody attacked it at once. And so within hours this thing was remediated and immediately fixed. Now, the initial fix had a problem that it could be worked around. And then people began bashing on BASH, essentially, really pounding on it hard, using fuzzers in order to find other things that would get through, and they did. There were several other remote code execution vulnerabilities found. So essentially an aspect of the Internet that had never received adequate focus suddenly got massive attention and, in a series of vulnerability patches, got tightened up very quickly.

And so, I mean, this, for example, as opposed to announcing, telling Microsoft about a vulnerability, and six months later nothing has apparently happened, despite the fact that it's there, and anybody else has been able to do it. That's the beauty, I think, of the environment, the open source culture that we have today is that there's a huge number of really smart guys. And, boy, you show them something like this, and you really get their attention. It's like, oh. Because everyone's got their own self-interest in mind. They're like, oh, my god, this affects me. And so, you know, I want to understand it.

FR. ROBERT: As a Windows person, a person who spends most of his day in Microsoft Windows, let me reach out to my fellow Windows users and say, look, don't be that guy. I know there's a few of you who love the fact that this seems to be hitting the open source community. But as Steve says, just don't be smug. Don't be a jerk.

Steve: Oh, Windows users have no...

FR. ROBERT: Exactly.

Steve: Our pride has been smashed out of us a long time ago.

FR. ROBERT: Security Now! spends 51 days of the year on Windows and one week of the year - well, actually I'm messed up with weeks and days. You know what, I think you've just BASHed my head.

Steve: The UNIX problems are the exciting ones, I think. They tend to be, you know, breathtaking problems. Whereas the Windows ones are, okay, you know, we found a problem with a font parser over here, and so we patched it. It's like, oh, okay.

FR. ROBERT: We're kind of desensitized to Windows vulnerabilities. We just know they're going to be there.

Steve: Yeah, but, boy, I'll tell you, the open source UNIX things, ooh, those are a lot of fun.

FR. ROBERT: Steve Gibson is at GRC.com. That's the place where you'll find SpinRite, the world's greatest maintenance and repair tool. You'll also find 16Kb versions of this episode, transcripts, and of course some great information about security, everything from SQRL to BASH to whatever Mr. Gibson has been working on. You'll also find an active forum for community members to discuss things like, well, what is the best way to hide in the sand now that all these exploits have come out in the open. If you have a question, you can always submit them at GRC.com/feedback. And maybe your question will be used on a future episode of Security Now!.

You can find all the versions of Security Now! at our show page at TWiT.tv/sn for Security Now!, and wherever fine podcasts are aggregated. You could also use our apps or watch us live. We gather every Tuesday - well, normally a Tuesday, today it's a Wednesday - 1:00 p.m. Pacific time, 4:00 p.m. Eastern, 2000 UTC at live.twit.tv. I am Father Robert Ballecer in for Leo Laporte. Steve Gibson, thank you so very much for fueling our nightmares. And until next time...

Steve: Okay, Padre, thanks.