
# Listener Feedback #87

**Description:** Steve and Leo discuss the week's major security events and discuss questions and comments from listeners of previous episodes. They tie up loose ends, explore a wide range of topics that are too small to fill their own episode, clarify any confusion from previous installments, and present real world 'application notes' for any of the security technologies and issues we have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-238.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-238-lq.mp3

---

**Leo Laporte:** It's time for Security Now!, the show that covers everything you'd ever want to know about being safe online, about protecting your privacy and more. With us, as always, our expert, our guru of security, Mr. Steve Gibson of GRC.com. He's the guy who wrote the first antispyware program, discovered the first spyware and coined the term "spyware." And he's ever since been helping us keep safe online. Good morning, Steve.

**Steve Gibson:** Ever since.

**Leo:** Ever since. How long ago was that?

**Steve:** Maybe eight years ago?

**Leo:** Wow.

**Steve:** I think so. Shields - well, I don't know. I'd have to look back. Yeah, in the early 2000s, I think.

**Leo:** Holy cow.

**Steve:** Because ShieldsUP! is probably about 2001, I think, and here we are in '10. So it was after I got that all up and running, and I think it was, like, right around that same

time. So, yeah, it's been many years.

**Leo:** It's amazing, just amazing.

**Steve:** And the problems have not gone away.

**Leo:** Yeah, and the gags and laughs still keep on coming.

**Steve:** Oh, boy.

**Leo:** Yeah, they got worse. In fact Steve decided, I'm going to let somebody who has more time pursue this. You handed it off, I think was it to Ad-Aware or Spybot?

**Steve:** Yeah, it was the Ad-Aware guys.

**Leo:** Ad-Aware, yeah.

**Steve:** Yup, because they said, hey, we want to do this. And it's like, okay, please.

**Leo:** Please.

**Steve:** You're going to be chasing your tails around forever on this one.

**Leo:** You have much more interesting fish to fry these days.

**Steve:** Yeah. And I'm much more interested in sort of fundamental technology things, rather than playing a cat-and-mouse game with the bad guys. It's like, I mean, because there's just too many mice.

**Leo:** Yeah. Well, it's repetitive.

**Steve:** Yeah.

**Leo:** Yeah. And, I mean, you like to do new things all the time.

**Steve:** Exactly.

**Leo:** I admire that about you, actually. Well, we've got questions from our audience, as usual. We've got 10 questions good and true from all of you. We've also got a bunch of security news, I see. All right, I guess we should get the security news going here, Steve.

**Steve:** As always, we have some.

**Leo:** Oh, yeah.

**Steve:** It's Mozilla and Firefox's turn for updating this week. A bunch of stuff has come to light that they dealt with in a series of updates since we last spoke. There was - I thought it was interesting to sort of give people a sense for just how convoluted some of these problems are. There's a cross-site scripting hazard, which we've done a whole podcast on in the past, that was discovered, which uses SVG documents, which are - SVG is Scalable Vector Graphics, which is a line drawing-oriented means for showing sort of resizable, that is to say scalable, graphics on a web page, as opposed to, for example, GIF and JPG and...

**Leo:** Or Flash. I mean, it's used often instead of Flash, as well.

**Steve:** Yes, exactly. And so using an SVG document with a content type, we've talked about that, too. A content type is a header which is sent along with the return of this asset, whatever it is, an image or a web page or a Flash object or animation or whatever, where the server is declaring to the browser, this is the type of content of this thing. So it uses a content-type which is wrong. A security researcher reported that when this SVG document is served with a content type of application/octet-stream and embedded into another document using the HTML imbed tag with type of image/SVG plus XML, the content type provided by the server is ignored by Firefox, and the SVG document is processed normally.

So a website which allows arbitrary binary data to be uploaded to it, but which relies on the content type application/octet-stream to prevent script execution, would have such protection bypassed because of this, I mean, a tiny little flaw in one little corner of Firefox. And so that means that an attacker could upload an SVG document containing JavaScript, that is to say, not SVG, but like JavaScript instead, as a binary file to such a website, which would use the embed tag to present it to other users, as many Web 2.0 sites do. And that would - so that document would end up being able to run its JavaScript, which was by design prevented by the website. But this little mistake would allow it to get rendered as JavaScript rather than an SVG, even though that's what it's supposed to be.

So, I mean, that's an example of just how obscure so many of these problems are. But what we see is that were this not fixed, and it became known, there's, like, zero chance that it would not be exploited. I mean, you might argue that a Firefox problem is less common to see an exploit in, you know, out in the public than, for example, an IE problem, just due to the relative size of the install base. Firefox has about a quarter of the browser market right now, browser share, as opposed to IE that has pretty much most of the rest of it.

But still we know these kinds of things, even as obscure as they are, do end up getting picked up by the bad guys and used as a way in, especially in the kind of attacks that we've talked about recently that we're seeing more of, where they're not just scattershot attacks anymore. A company is targeted to be exploited, to be a victim, and research is done, like are the employees in that company using Firefox? And, if so, then it's not a matter of just spraying this on the Internet. It's like, oh, you know, what do we have in our Firefox bag of tricks that we can use as a way into this company's network? And, you know, this is the kind of approach being taken.

So Firefox 3.6 has been available for a couple months. That's been fixed. And then the 3.5 lineage needs to be updated to 3.5.8. And even though Firefox did say that they were discontinuing support for 3.0, and I had by that time abandoned it, although I'm a slow adopter of these things, I'm not moving to 3.6 yet...

**Leo:** Where are you now? 3.5?

**Steve:** I'm on 3.5.8.

**Leo:** Okay. That's good. That's all right.

**Steve:** [Laughing] I'm going to let that settle down a while. But if you're still using 3.0, it really - you ought to, it's time to move over. When I did try to adopt it, immediately after release, some of my Firefox add-ons complained. And all I had to do was wait a few months, and then when I later tried, everybody was happy. So and then also Thunderbird and SeaMonkey, other code from the same code base, the Mozilla folks, also needs to be updated, Thunderbird to 3.0.2 and SeaMonkey to 2.0.3. So I just wanted to let everyone know, it's time to bring that stuff up to speed. And that's one - what I've just described in some detail is one of a number of fixes, I think there were five or six, that are cleaned up there. So you definitely want to do that.

We talked also in the last couple weeks, maybe not, if not last week, then the week before, about a problem that had been found in Adobe's Download Manager where unfortunately, if you had used Download Manager to update some Adobe stuff, which of course is happening with metronomic regularity these days, the ActiveX control that is the Download Manager would stay in the browser until it was closed. During that time it was exploitable to be used by malicious third parties to download anything else that they might want into your computer. The good news is that's been fixed. So you don't have to do anything about it. There's nothing to go and get and download from Adobe's site because, as long as you shut down your browser, that will then flush the old buggy version of Download Manager away. And when you next are downloading something to update Adobe products, which probably won't be long from now, you'll get the new version of Download Manager, where they've closed this hole.

**Leo:** Right.

**Steve:** Now, Microsoft every six months publishes what they call their Intelligence Report, which is full of really interesting data, Leo. And I want to give our listeners the link that I discovered this current edition through, which is a SANS.org link, because it is a sponsored link, and I want to give SANS credit for bringing it to my attention. So it is -

and I'd like you to put it in right now and grab this, if you would.

**Leo:** Oh, yeah.

**Steve:** It redirects to Microsoft. And when I was looking at this, thinking what link am I going to provide, well, a Microsoft link is like one of those nightmares from hell. I mean, I can't even tell you what that is. So it's www.sans.org/info/55704.

**Leo:** Of course it is.

**Steve:** So you put that into your browser...

**Leo:** Okay.

**Steve:** ...www.sans.org/info/55704, which will give them credit for providing this, and bounce us over to...

**Leo:** Oh, but this is the Microsoft page.

**Steve:** Yes.

**Leo:** I got it, I got it.

**Steve:** Yeah, bounces to Microsoft. Now, there's two versions, either a PDF or XPS format - I imagine PDF is what we want - and two sizes. Grab the smaller one, which is the second offered thing, the summary, or I think they call it the "key findings." That's a 19-page PDF summary of really interesting stuff.

**Leo:** It's the third link, though, it's 1.7 megabytes.

**Steve:** Right. And, now, the big one, for those who are more interested or who want more details, is I think it's like 10-something megs?

**Leo:** Yeah, 32 megs, I think.

**Steve:** Oh, 32. That's 232 pages.

**Leo:** Oh, yeah, no, it's 10 megs. The XPS is 32.

**Steve:** Right, right.

**Leo:** 232 pages, wow.

**Steve:** So it's a serious report. Although it contains the same pretty pictures as the key findings.

**Leo:** Well, that's good.

**Steve:** And, for example, I mean, this thing is just full of cool stuff, like pictures of the world, colored with synthetic coloring to show instances of botnets and infections and so forth. What I thought was really interesting was on page 7. If you scroll down to page 7 there's I think a figure 10. There's a stacked bar chart showing the evolution of the types of sites where phishing is used. And started in May of '09, and through June and July, you see this explosion on social networking sites.

**Leo:** They were almost nothing in March '09. And now it's by far, it's like 80 percent.

**Steve:** Just taken over.

**Leo:** Yeah.

**Steve:** Yeah, it's just taking over. So, I mean, we could spend a whole episode just talking about this very cool report. I don't think we need to because everyone can just grab it and browse through it. It's really nice diagrams, bar charts, percentage charts, showing just basically what's going on in security with all kinds of different breakdowns of information.

**Leo:** Oh, this is really neat, yeah. And revealing. I mean, I think that those social network sites are probably mostly Facebook and Twitter, both of which have been prone to phishing, crazy phishing scams.

**Steve:** Yeah.

**Leo:** Even lately, so...

**Steve:** Well, and they're getting - they're probably getting better and tightening up. But anyway, I wanted to turn our listeners onto this report because it's very neat.

**Leo:** Yeah.

**Steve:** And Microsoft has confirmed, speaking of Microsoft, a new zero-day exploit which they're not happy about, but they'll certainly get onto fixing. The idea is that VBScript on a web page which is opened by IE, which of course IE is the only thing that's going to run VBScript. Everybody else has JavaScript. And VBScript was of course Microsoft's attempt to overthrow JavaScript, and thank goodness that didn't work very well. But it's still supported because Microsoft will support everything forever, or at least decades.

So VBScript can be used to open a message box which is invoking Windows Help. And it turns out that the Help file can contain macros which will execute if you open the Help file. So if the system has outgoing access to remote Samba shares, SMB, Windows file and printer sharing, and personal systems generally do - typically corporate firewalls will block outbound Windows file and printer sharing, but most of our, like, normal home and small office systems will block incoming Windows file and printer sharing, as will even personal firewalls; but they'll generally allow outgoing filesharing. So it turns out that a malicious site could host a malicious Help file and then put VBScript on a page which, if you go there, will pop up a message box. Now, you do have to press F1, the Windows Help key, in order to sort of complete this process. But there's lots of social engineering mechanisms for allowing this to get done.

**Leo:** Yeah, just press the Help key now.

**Steve:** Yes, and we'll take you further on your journey.

**Leo:** Yes, yes.

**Steve:** And so what would happen is your system would then reach out to lord knows where, somewhere where, you know, following a URL, and bring in a Help file containing Windows Help macros, which would then execute at that point on their own, and of course you've lost control of your computer. So I did, you know, Microsoft will get this fixed, I'm sure, with whatever patch cycle. But I wanted to alert all of our listeners that, if they are browsing around and some dialogue box pops up, the last thing you want to do is press F1 at that point. You don't want to say, oh, look, read the text and press F1.

Just remember this, that this is out there now. Microsoft got caught off guard. They've confirmed it. But it's a problem until they fix it. Or Firefox users that are running a normal Firefox without any kind of an IE plug-in capability will not be supporting VBScript. So presumably, I don't remember now in IE if you're able to turn off VBScript support. I know...

**Leo:** Oh, I'm sure you must be.

**Steve:** I'm, well, clearly we can turn off JavaScript. But I'm not sure whether - maybe it's just scripting. I think in the configuration boxes in IE they refer to it as scripting. So you don't have independent control. I was just thinking, nobody needs VBScript. I mean, like, nobody. So you could turn that off. And if you could turn that off and leave JavaScript on, then that would be a painless solution for this. But I don't think we have that level of granularity.

**Leo:** No.

**Steve:** And there's one interesting new little module has been added to Metasploit that - Metasploit is this framework which is - at best it's quite controversial, I would say. It's a malware exploitation framework which is a sort of a test bed and a hosting bed for all the different kinds of things that we talk about here. Generally, by the time our listeners are hearing about it, someone has taken a vulnerability which is then understood, and written some code.

And so what's controversial about it is, it would be one thing if the hackers had to implement this stuff themselves. If they did, there would be much less of it in the world. Unfortunately, the Metasploit framework creates a foundation that allows other people to write this once and then for it to be used almost with pushbutton ease by anyone who wants to. So a new module got added that did something very clever that I hadn't seen before, and I wanted to just sort of share it with our listeners.

We've talked about how in, for example, an open WiFi scenario, for example, in a hotspot, where you're just able to walk into a caf and connect to the Internet, we've talked about all the various dangers there. And in the past the approach that we've taken for getting control of that network has generally been playing games with ARP packets, Address Resolution Protocol, where the idea would be that we would pretend to be the gateway and knit ourselves into a conversation with various other laptops that are accessible thanks to the fact that we're all sharing the same local Ethernet wireless domain, by, like, beating out the gateway, or playing various timing games. I mean, it works. But it's, you know, it's on the fringe.

Turns out some clever hackers have come up with a much simpler solution. They exhaust the DHCP server's resources. The DHCP server is what we're all running in our little routers, where the computer is configured to obtain IP address automatically. We've talked about this before in the podcast, where as the computer wants to get on the network, it sends out a blind broadcast saying, hey, I'm just arrived here. I need network connection specifics. Is there a DHCP server that can provide that?

So in response to that broadcast, the DHCP server, which is running, it's one of the things running in the router, will respond with an IP address, with the IPs of DNS servers that we've talked about also extensively very recently, and what other information, you know, like NTP, Network Time Protocol servers, whatever the DHCP server wants to provide, it's able to hand then to the client. Well, we know that our little home routers have a limited number of addresses they can give out. And oftentimes that's something configurable. In the web page interface it'll say first IP address, last IP address. And the first one might be 192.168.0.2, and the last one 192.168.0.50 or something. There's a range.

Well, it turns out that when the server has given all of the IPs that it has, it simply no longer responds. It doesn't - there's no, like, error message, or I don't have any more or anything. It just doesn't respond. So it turns out that it's now become substantially easier to be a bad guy on a network where you have a bunch of machines because you can simply have one computer continually broadcast requests for IPs. And the DHCP server will dutifully peel them off of its list until it has no more. And at that point a legitimate machine attempting to join the network will send a broadcast which will go unanswered unless the malicious computer answers it, which it's as able to do as the access point because - and now there's no race. There's no competition. There's no, oh, shoot, the other one got me, you know, beat me to it last time, I'll have to try it again.

It's completely leisurely.

So you simply send out requests for IPs, collect all the IPs available yourself so that nobody else can have any. Then, when anyone else attempts to get one, the DHCP server will not respond. And so you give it your information - you as DNS server, you as gateway, you as anything. And so you have just very casually, leisurely knit yourself into the network in order to then pursue whatever attacks you want to. Not good. Not good.

**Leo:** Very clever, very clever, yeah.

**Steve:** Very clever. And I had a very short little note to share about SpinRite. A listener, Richard Frisch in Weston, Connecticut said - his subject was "Gibson, Laporte, and SpinRite: A Great Combination."

**Leo:** Aw.

**Steve:** And he said, "Steve and Leo, I've used/owned SpinRite for many, many years. I believe I purchased the first version a long time ago, and most if not all subsequent versions. I own and use the current one. It has fixed hard drives numerous times over the years. I could tell you stories about what it's done for me, my family, and friends. But the simple truth is more compelling. It just works. I'm a fan of you, your work, and Leo's TWiT.tv network."

**Leo:** That's neat.

**Steve:** "Thank you for being you."

**Leo:** That is really neat.

**Steve:** So thank you, Richard, I really appreciate the note.

**Leo:** Yeah. Steve, I have in my hands, they've been sealed on Funk & Wagnalls' porch, 10 questions. Are you ready to answer those 10 questions and put yourself in the security hall of fame?

**Steve:** Yeah, these are good. Nice variety of questions, and good feedback from our listeners.

**Leo:** Excellent, as always. Let's start with Warwick, Rhode Island, my old stomping grounds. This is Robert Sylvester, who wants really to be sure. He says: Steve, I thank you for looking at Steganos LockNote. A lot of listeners had asked us to take a look at that. I wouldn't trust it if you hadn't looked at it. But could you got a step further and post some hash signatures of the download you examined for the TNO

knockout? P.S., I own SpinRite and use it all the time. What is he asking for? What does he want?

**Steve:** Well, this really brings up a good question, or problem. He's saying, okay, this is an open source utility, acknowledging that I took advantage of the fact that it was open source in order to examine the source code myself to figure out what the design was of the product because Steganos themselves said nothing about it. I mean, frankly, if Steganos had said, "For anyone who understands crypto, here's exactly what we did and how it works," I would have taken their word for it. I would have said, okay, good. I mean, I read through that, I see what they say they're doing, they look like they're a legitimate, real security company. They've shown me they know what they're doing. I got the same, I got an exactly equivalent impression or set of information the hard way, which was because this LockNote is open source, I was able to read the source and reverse-engineer the same information from it, which I then shared with our listeners.

Now, that's different, well, there's two problems. I didn't look at it closely enough to be able to assert that there isn't something else going on. That is, I mean, I looked at the source. But it's difficult for me as a programmer to explain to a nonprogrammer how, I mean, like next to impossible it would be for me to, even looking at the source, to know that there isn't something I missed. I mean, it's just - that's an effort that is like orders of magnitude greater than reading the source to figure out what the programmer apparently intended.

So I couldn't trust something that I didn't myself write from scratch because there's just - there's so much more going on, so many ways some little tiny mistake, I mean, if there was some deliberate mistake made in the source, I wouldn't see it, you know, a one that should be a zero that allocates an extra word which can be used as a trapdoor where, if you put FFFF in it, then the hash always returns this value. Or, I mean, it's so possible to deliberately hide something in plain sight, which is, frankly, one of the reasons I don't do my stuff open source, because it's just, it's too possible to hide something. So for me the value of them making it open source was that it served as documentation of their intent. But it could never, for me, serve as a sort of a freestanding, trustable record from which I would then move forward.

I downloaded the executable from their site, as the authors. And there's necessarily some implicit trust in their intention to do me no harm. That is, that the executable does match the source that I saw. It certainly looks like it does. It behaves like it does. I have no reason to believe it doesn't. But, sadly, the level of work required to obtain this kind of assurance is phenomenally stratospheric. And that's one of the problems we have with computers today. I mean, I don't know what the solution is. But I'm not going to hide from the problem just because we don't have a solution.

I mean, this is one of the problems, is that software, the way we write it today, the way we craft these solutions, is so complicated that even having the source, if it was malicious source, it could masquerade as the greatest thing since sliced bread, and people could stare at it all day long and not see that there was a mistake in it.

So I don't have hash signatures for what I looked at because I just took what I looked at as documentation that they didn't provide on their site of the technology that this uses, which convinces me that they went to, I mean, they went overboard in making this thing secure. I don't doubt that it is. But I don't know that it is. And I can't know that it is. I mean, and this is exactly what we talk about every week when we talk about, whoops, brilliant people who had lots of experience designed TLS, Transport Level Security, from

SSL, and they made a mistake in renegotiation, which is just now in the process of being fixed a decade later.

**Leo:** Right, right.

**Steve:** So, I mean, this stuff is - it's a problem that it's so hard. But I think at least appreciating and understanding the nature of what we're expecting is important. And all I could do is say I liked that they provided the source because now I understand how this thing works deeply and could say, and I did, they really did a beautiful job.

**Leo:** That's great.

**Steve:** Was it perfect? I can't say that it was perfect.

**Leo:** Well, you know, RedStapler in our chatroom posted a link to the Underhanded C Contest, which is Underhanded.xcott.com. And this is a perfect example. They have assignments, the fifth contest is now open, to write code that is, on the surface, on the face of it, in every respect, completely secure and reliable. And the contest is, what can you get away with? So this challenge is to write a luggage-sorting program. And they give you all the parameters as if it was a real assignment, a real request for code. And they give you all the details. And then they say, okay, but here's the underhanded part. Your program must inexplicably misroute a piece of luggage if the right kind of free text comment is provided by the check-in clerk.

**Steve:** Yup. A perfect backdoor.

**Leo:** So it's, yeah, it's just - but anybody reviewing the code should be completely satisfied that it's safe.

**Steve:** Would never see that, yes.

**Leo:** I don't know how you'd do that, but apparently there are ways to do this.

**Steve:** Oh, I know immediately. For example, say that you would somewhere make a little hash of the comments, which you said you were using for this purpose. But you would be checking the hash against a constant somewhere else, and that would open the trap door and cause a misroute. And, I mean, so that's one of the problems. See, for example, people are going to ask me, when I get CryptoLink happening, hey, we want to see the source. I'm going to say, sorry. I'm going to document the protocol. I'm going to document exactly what I intended. But the source is not available because it's not useful to anyone. I mean, it'll be in assembly language, so you could disassemble the code if you wanted to.

But, I mean, I'm going to be as completely open as I can be, but I'm not going to have my source taken and mutated and turned into something else. I mean, because I will

write it, I mean, every single byte of it, I'll be able to make a representation about, to the best of my ability and knowledge, this is what I created. And there is, I mean, I'm putting my reputation on it, that I've made no mistakes and there's certainly no deliberate backdoors. But beyond that, I mean, there just isn't a way to be absolutely sure.

**Leo:** So forget the hash. Steve's promising nothing. You're asking too much. Paul Scribbans in Cumbria, UK wonders how to play with machine code. He says: Steve and Leo, I've been a listener since Episode 1. I think the work you do is excellent. SpinRite owner, as well. My father tried to get me into machine code when I was about 12 years old in 1982 after we purchased our first computer, a ZX81.

Remember the Sinclair? They also bought a book, "How to Program in Machine Code." It never really took off for me, but through the proceeding years I've always wanted to give it another go, so your episode on the subject really interested me.

I do have one request, though. Would you do a bit on which assembler package or software to use? I'm currently using Windows 7 and plan on buying a MacBook Pro soon, as well. But I have no idea what software to download to enable me to play with assembler based on your teachings. Keep up the good work. Scrib.

**Steve:** There have been a number of people, listeners who have said, hey, you've got me interested. Where do I start? And the problem is there really isn't a, that I'm aware of, a starting-from-ground-zero tutorial.

**Leo:** There's a good book. Wait a minute, let me see if I can find it because I have it, I think, on my shelf. There's a number of very good books on this.

**Steve:** Well, there's a site.

**Leo:** Oh, good, all right.

**Steve:** There's a site for Windows called MASM32.com. And this is relatively high-level stuff. I mean, it's, again, if you want to play with assembly code in Windows, I know of no better resource. There's lots of links, lots of samples. They have a complete SDK, as it's called, a Software Development Kit, that provides all the pieces that you need, linkers and help files and header files and include files that provide the definitions and all sorts of stuff. None of this existed back when I was doing assembler in 16-bit code, but it has come together now for the 32-bit world. So MASM32.com for people who are interested in beginning to poke around. I mean, again, it's not what I would create if I were going to create a from-the-beginning tutorial. But I haven't had a chance to do that yet.

**Leo:** I'm going to try to find this book. It's a thick book, I think. It comes with MASM in the back. And it's all about assembly language. And it - yeah. And there used to be a wonderful assembler on the Macintosh that Apple made. I don't know if they still do it anymore. But I do believe they offer free, you know, they have all the free developer tools, and I believe they offer...

**Steve:** The whole Xcode system...

**Leo:** Xcode; right.

**Steve:** ...is on the disk.

**Leo:** And it's probably got an assembler. I'm sure it has an assembler.

**Steve:** Oh, absolutely it does.

**Leo:** Yeah. So you're set on the Mac, as well. It's the education that's hard.

**Steve:** Well, I haven't ever talked yet about my long-term plan legacy project. I will tell everyone about that when we're through talking about computers.

**Leo:** Now you have intrigued me, Mr. Gibson. Question 3 from Eric Stearns in Denver, Colorado. He wants to know how a computer starts up from the very beginning: Steve and Leo, I've enjoyed the discussion so far on the basics of how a computer works. After the first two podcasts I feel I largely understand the basics of how a computer works once it is operating. But one point that's never been clear to me is how a computer starts to work. So we have this dumb box of rocks that's powered off. Now we provide it with a flow of electrons. Well, maybe not a flow, but at least a voltage that can provide a flow. But how exactly does this lead to the ability to do something useful like executing some very basic instructions to eventually be able to run a program? How does it start?

**Steve:** Yeah, it was a neat idea, I mean, a neat question, which we never talked about.

**Leo:** No.

**Steve:** And this is something which has evolved over time. The very early machines, the mini computers, my favorite the PDP-8 or the 11, those early machines which used core memory, because the memory itself was magnetic and could and would persist across a power-off and power-on, it was often the case that the computer still had the program in it when you turned it back on. I mean, it's magnetic in the same sense that a hard drive is magnetic. And of course we know that the intention of hard drives, they don't always succeed, but the intention is that they will retain their data, even when they've been powered off.

So mini computers would typically - you'd turn it on, and it would come up just sort of sitting there saying, okay, I'm beginning to burn up power and making whirring sounds. What do you want me to do? And so the operator would put the starting address of the program, like timesharing basic or something, if you had a bunch of teletypes in a classroom of kids, you'd put the starting address in, load that into the program counter and press Run, and it would just pick up sort of where it left off.

Then some machines of that era, specifically, like, doing process control work or real-time stuff, there was something called Power Fail Restart, where there was typically an option that you could get such that, if Power Fail Restart was installed, then once the computer's power levels came up and stabilized, this option would simply jam a starting address into the program counter and sort of electronically do the equivalent of pressing the start button. So it would just - it would just start itself up.

And in fact you might have a power fail occur while it was running, in which case this power fail sensing auto restart, when it would see the power beginning to fail, it would interrupt what the computer was doing. And we're going to be talking about interrupts here before long because that's a huge issue, and it's the problem that Toyota is having at the moment. And it would interrupt what the computer was doing and save the state of the computer, that is, where it was, what was in the accumulator, and the carry bit, so that later when power was restored the state of the computer could be restored to exactly where it was. So these things would be stored in core, and then the computer would stop itself before the power levels had dropped to a level that it was no longer reliable.

And then similarly, when the power came back on and what was stored in memory would tell this power fail option that the computer had been running when power was interrupted, and so read from core memory these things like the original content of the accumulator, the original content of the carry flag at the time of this interruption, put those back, and then resume execution from literally the instruction immediately following the last one that had been executed before the power failed. And so it would be just as if nothing happened.

So now we move forward decades to our current machines. And

our current machines all have some code in them separate from RAM, that is, they have ROM, and that's traditionally called a BIOS, which is - BIOS is an acronym, Basic I/O System. And the startup code for the machine is in the BIOS. So to directly answer Eric's question about contemporary machines, how our computers that we're using now start up, it's as simple as the chip, the actual processor chip is designed so that when it comes out of reset, when reset is over - and, now, reset is the state the machine is in when it's powered up, or if your computer still has a reset button on the front, that's a physical hardware reset button that typically pulls a wire down, pulls the voltage on a wire down that's going into the chip that just stops everything.

And essentially, when you remove your finger from the button or when power levels have come up and stabilized, the processor is hardwired in its hardware to start executing code at a certain location. We've talked about how the program counter steps through locations in memory, reading instructions one at a time. So it's simply a matter of the chip always starting from, it might be zero, it might be FFFF, you know, like all ones. It'll be some predefined hardware location. And at that location is a little bit of ROM, a little bit of Read-Only Memory. That is, the rest of the machine's memory, RAM, will just be random. It'll be not necessarily zeroes.

It's interesting, too, when you power up RAM, it generally comes up in some noisy state, but not all necessarily zero. One of the first things that the BIOS normally does is go and clear out the RAM in order to start it. And it also begins refreshing the RAM in order to keep its contents alive. So there's housekeeping being done. But literally it's just a matter of the processor always going to a predefined hardware, defined in its hardware, starting location, where there'll be a little bit of memory, a little bit of ROM that will be the first instruction that executes, and the second, and the third, and the fourth, and the rest is Windows or Mac or whatever.

**Leo:** [Laughing] And the rest is Windows.

**Steve:** And the rest is Windows.

**Leo:** Question 4 from PDP-10 Programmer - hmm, I like that - regarding - is the PDP-8 different than the PDP-10? Or is it pretty much...

**Steve:** Oh, yeah. They had different - the PDP-10 was a 36-bit, really long word machine, a beautiful, beautiful computer. But much bigger. I mean, it was like one of those raised-floor, forced-air cooling sort of machines, a beautiful machine.

**Leo:** You wouldn't have it blinking behind you, with the blinking lights.

**Steve:** No. Or if you did, you wouldn't be able to hear me talking.

**Leo:** PDP-10 programmer writes: Steve and Leo, with respect to the Lower Marion School District (LMSD) spying case, I heard you mention on the podcast that kids in the district had become so suspicious that they started putting post-its over the camera. I also read on some Mac-related web forums posts by the IT guy at LMSD regarding this. He said if Macs came to the IT with post-it notes over the camera, the IT guy should just put a pinhole in the post-it, but leave it in place so the student thinks the camera is still covered. With info coming out like this, seems they've got these guys dead to rights. They were spying on the students. Boy, that is shocking.

**Steve:** Yeah. We have another interesting question a little bit later on because this news really stirred up our listeners.

**Leo:** Rightly so.

**Steve:** And lots of people had some questions, but more about the technology. This is more about the ethics of it. I just - I wanted to put this in the podcast because, I mean, assuming that it's true, it really does seem wrong. You can argue that, well, that the camera technology was there to be used by the district to recover stolen and lost laptops. The problem of course was that there's evidence that is conclusive that non-stolen, non-lost laptops were having their cameras turned on. And whoever was doing the looking was feeding information back to the assistant principal, who was then confronting the kids who were registered to this laptop about their behavior at home.

So given that that's the case, the notion that the IT people were being told to poke a little hole in the post-it note so that the camera would be reenabled - I'm not really sure how well that would work, by the way. I mean, it sounds a little apocryphal to me. But given that that was what they were doing, that seems extra annoying and slimy. If you were to defend the whole practice, that is, if notification had been provided, if students and family knew that this was a feature of the laptop, certainly they would have reason to believe that no one was going to be spying on them unless the laptop was lost or

stolen. But at the same time, then I guess you could argue, and you could imagine the school district's attorney arguing, that, well, this is a security feature which we need as part of our laptop loaning program. So cutting a hole in a post-it note is necessary and justified in order to keep the security feature functioning.

**Leo:** Oh, please.

**Steve:** I know. I'm not defending it, I'm just saying this is, you know, you can imagine what the LMSD attorney would be doing. But still...

**Leo:** Uh, Your Honor - it's just pretty hard to explain.

**Steve:** It's bad.

**Leo:** Yeah, it's not good. All right, get ready for a long one about assembly language, once again. Jeff in Washington, DC wonders about assembly language, CryptoLink, and Intel Macintoshes; and, as long as we're throwing everything in, how the computer works series. Steve, first I'd like to thank you and Leo for a terrific podcast. I thoroughly enjoy listening each and every week. The information covered and so elegantly presented is truly a gift to the world of computing.

He says he's found the "How a Computer Works" series very informative. While reviewing the "Machine Language" module - oh, they're modules now, not episodes - I discovered I had a few questions. One, a general question. I'm wondering how well assembly language lends itself to multiple platforms and OSes - Linux, Mac, OS X, and Windows - as well as various architectures, 32-bit versus 64-bit. Does assembler compare with other languages like C, C++, et cetera, that have odd nuances when deploying on different OSes? That is, does it require heavy modification, depending on the OS in which it's used? C code on a Unix system doesn't really mirror a Windows implementation of the same C code. Does a change from a 32-bit to 64-bit architecture generally require major changes to the assembly source code? Why don't you answer - you want to answer this one first, and then I'll give you the next part?

**Steve:** Yeah, I think I'll probably end up covering his specific example as part of this.

**Leo:** Okay, well, I'll give you his specific example, then. Knowing that you write the majority of your programs, including the upcoming CryptoLink, in assembler, and most of your applications are ultimately delivered in an EXE file format, I'm wondering if your ASM-developed applications easily lend themselves to other non-Windows platforms. For example, will CryptoLink be available on the aforementioned various platforms - Mac, Linux, 64- and 32-bit Windows? I really hope that will be the case. If not, will it run well under an emulator like Wine, like your DNS Benchmark does? Keep up the phenomenal work. Jeff.

**Steve:** So we could shorten this by sort of asking, what's the relative portability of assembly code compared to higher level code? And the answer is they're pretty much

equally portable and not. That is, C code is portable from one processor architecture to another. That is to say, it was designed so that the same code could be compiled to different machine language where the compiler would be doing the translation between the C code and the specific hardware architecture, how many registers the chip has and so forth.

But C code is not portable across operating system families, that is, for example, Linux, Mac, Windows, because the operating systems provide radically different sets of services. That is, the operating systems - and we're going to talk about in detail in the future what is an operating system as we continue moving up the abstraction hierarchy from diodes and resistors where we began all the way to a working system. But essentially operating systems provide an abstraction to the software running in or on the operating system of the outside world. It's the operating system provides an abstraction for the I/O and for memory and for storage and for the passage of time.

Sort of the application itself in a modern operating system does not actually connect or contact the hardware. Rather, the operating system publishes services which the program takes advantage of. For example, the program says, hi there, operating system. I need a block of memory for my own purposes. Can I have one, please? And the OS looks at its pool of available memory, which is shared among all the applications, and has a chunk that's free, and provides a descriptor back to the program saying, oh, yeah, here's a chunk of memory. Let me know when you're done with it because we want to put it back into the common pool once you're through.

And so that's the case regardless of what language you're programming in. That is, for example, I'm talking to Windows, "I" meaning Steve who writes in assembly language. My assembly language code is using the same services that someone writing in C uses. So I'm using the same operating system services as someone in C. But if we took either me and my assembly language or a program in C over to a Mac, while you could recompile the C program under Mac's assembler to create machine language that would technically run on the Mac, that program would be lost. It would be trying to - it would be expecting Windows-oriented services when none are available because the Mac has an entirely different view, it presents a completely different abstraction of the outside world to the programs that run on it.

So relative to CryptoLink, for example, I'm already aware that I will be under immediate pressure to make this thing run on platforms other than Windows. And this has been discussed already in the GRC newsgroups. Because I really do want to serve as large a market and offer this solution to as many people as possible, my plans are already to carefully modularize my code so that the bulk of CryptoLink will be portable to different platforms, which they will all have to be Intel chipsets because this is largely going to be assembly language.

But I could still deliberately, for example, keep the code that deals with the LAN adapter, that deals with the kernel driver. I'll have a very different one on a Mac and a very different on a Unix or Linux system. But if I'm careful, rather than just mixing all of that code together in a big stew, if I'm knowing that I'm going to be moving this thing other places, then it behooves me to deliberately keep some modules to this so that, for example, when I do want to port it to the Mac, most of my code was carefully written so it doesn't care where it is. But then it then asks a network module to do the network magic on that particular platform. So there would be some things that are OS-specific. But they would be well encapsulated from the beginning.

So it's possible even for assembly language to be as portable as C, as long as you're not changing processor platforms. C allows you to jump from one processor, you know,

PowerPC to Intel to any other type of microprocessor relatively easily because you're abstracting the architecture of the hardware in C, whereas you're absolutely not doing that in assembly language. The language is the language of the hardware, exactly as we've been discussing.

**Leo:** Yeah, that was the point of C was to create a non-processor-specific portable language. Back in the days when you didn't have GUIs, you didn't have really a lot of interfacing with the operating system.

**Steve:** Yeah, and Unix was written on a PDP-11.

**Leo:** Right.

**Steve:** The PDP-11 was the first Unix machine.

**Leo:** So they, I mean, they provided libraries for low-level file access, things like that, so you didn't have to see anything that - but you can do that with assembler, too. As you said, encapsulate it.

**Steve:** Right.

**Leo:** Just separate it out.

**Steve:** Just plan ahead.

**Leo:** Yeah. So that's a revelation to me. I had no idea that you were planning portability because you've never done that before. You've only written for Windows.

**Steve:** And I'm already getting heat...

**Leo:** Or DOS.

**Steve:** ...for it about SpinRite. So I'm not going to make that same mistake.

**Leo:** Yeah, wow, that's very interesting. Well, SpinRite relies on, what, N13? A BIOS call; right? So you can't really do that on a Mac.

**Steve:** Yeah. The Mac has the EFI BIOS. And so it just, you know, it would be neat at some point for me to move SpinRite over. I know that there are a lot of people who would like it. It's just right now it's down on the queue a little ways, yeah.

**Leo:** But CryptoLink, because it is not relying on the OS so much, or the BIOS so much, you could do - that's portable, you can make that portable. It's the UI that wouldn't be portable.

**Steve:** Well, I mean, frankly, SpinRite could talk to the Intel hardware just as easily. It's just because of its history it doesn't. But were I to do it today, I would be talking to the hardware directly.

**Leo:** Sure.

**Steve:** And get a lot more power and flexibility.

**Leo:** Oh, interesting, yeah. Next question comes from Bangkok, Thailand. I'm going to butcher this name. I think it's Teerawat Issariyakul.

**Steve:** Very well done, Leo.

**Leo:** Yeah, but we don't know if it's right. But it's...

**Steve:** And we're glad that Elaine has access to this file that we're reading also, so that she can transcribe his name correctly.

**Leo:** Yes. This is a data leakage question. Steve and Leo, I've been a long-time listener to Security Now!, proud owner of SpinRite, although I haven't had to use it so far. A couple of episodes ago you answered a question about whether we should be worried about unencrypted data in RAM. The conclusion was it's unlikely to be a problem since the data in RAM disappears almost immediately after we turn off the computer.

Well, how about this? You know from time to time the data in RAM will be written into a swap space on a hard drive, pagefile.sys for instance. Since that data is not encrypted, I figure the data in the swap space is unencrypted, too. Am I misunderstanding something? If not, should we watch out for applications that store sensitive information in RAM? Love the show. Keep up the good work. You should also add hibernate files, which are RAM dumps.

**Steve:** Yeah, and first of all, this is a great question. And he's completely right. The pagefile.sys, as Windows calls it, and he gives that example in his note here, is generally a substantial percentage of the size of physical memory in a Windows machine. And I think that's, what, is it one and a half times the size?

**Leo:** Well, you know, that's the rule of thumb. But if Microsoft does it, all sorts of weird things with it, resizes it and...

**Steve:** Yeah, it does. It does, yeah, it does vary. And so it is the question previously that this questioner was referring to was someone was asking whether RAM itself should be kept encrypted and only decrypted, like, on the way to the processor. And we decided, well, technically you could do that. But the threat surface is so small for anyone getting access to the RAM prior to the point where it degrades down to noise, that it just doesn't really seem like it'd be worth the overhead of, I mean, substantial overhead of performing encryption and decryption on the fly as you're writing to and from the RAM chips.

However, it is the case that the swap file is used by forensic examiners like the FBI when they grab machines. They absolutely look through the page file because they know that it's a record of what has been in RAM. And it's not the case, though, he suggests, if not, should we watch out for applications that store sensitive information in RAM? Well, watch out for applications that use electric power. I mean, or computers that do. There isn't anywhere for sensitive information to be other than in RAM. It's like when we were talking about decrypting DVDs and HD and Blu-ray and all that, we were explaining how it's not possible, I mean, it's literally, theoretically even, not possible to protect that encrypted data from somebody who wants it because the machine itself has to decrypt the DVD or Blu-ray or HD disk in order to play it. It has to exist in the clear.

Similarly, sensitive information has to be in plaintext format. I mean, if you're writing a document, and it's there in Word, I mean, even if the information is sensitive, it's there. It's living on your screen, and it's in RAM. So it is the case that there are security tools which will wipe the page file as your computer is shutting down, exactly for this reason.

**Leo:** Oh.

**Steve:** They will wipe the page file as part of the shutdown because it's understood that otherwise that file is sensitive. Now, we've talked about various types of hard disk encryption, tools like TrueCrypt, which is our favorite because it's open source and well supported, and it's evolving nicely. And Leo, you mentioned hibernation. That was missing from TrueCrypt initially and has now been included, meaning that it's sort of extra tricky to encrypt the hibernation file, which is to say the state of the machine, when we talk about machine state, as we were a minute ago, contents or registers and RAM, we need to, in order to, like, just freeze the computer, you need to save the state in some sort of nonvolatile form so that you can later reverse that process and restore it.

So it's tricky to save the state encrypted because it means that you need to have an authenticated, validated decryption running before the operating system runs in order to restore encrypted data from hibernation. So they did solve the problem in the case of TrueCrypt. With TrueCrypt it's possible to encrypt a sort of a pseudo drive or encrypt a file or a directory. And they make a point of saying that make sure you understand that doing that, those sort of smaller encryption modes, does not encrypt your page file, that it's only if you use the so-called "whole system" encryption, where you're encrypting the primary partition that the operating system is running on. And they also make a point of saying make sure that the page file is living on that same drive and not off on a different drive somewhere because, again, then it would not be covered by the encryption umbrella that whole system encryption provides.

But in the case of TrueCrypt, if you do use whole system encryption, and the page file is located as it normally is by default on that system partition, then even though you are writing RAM to the hard drive, it's passing through the TrueCrypt encryptor on the way to the page file and being decrypted in the reverse direction as it comes back out. So with

something like that, where they've made a point of protecting the swap file, you're safe. But otherwise it's like the number one place that the forensic guys go to see what - just to see if they can find something. And often they do.

**Leo:** That and slack space is another great place to find...

**Steve:** Yes.

**Leo:** Although if you're encrypting your drive you don't have to worry about slack space.

**Steve:** Exactly.

**Leo:** Actually I guess if you're encrypting your drive you wouldn't have to worry about the page file, if you had whole drive encryption.

**Steve:** That's exactly right.

**Leo:** Yeah, yeah, they'd have to decrypt the drive. Chuck in Tampa wonders and worries about his Lenovo S10 camera and security. I guess we're all starting to worry about cameras on our laptops nowadays. He says: Being an S10 owner, as well as owning other laptops with integrated cameras, some have disable switch and buttons. I bet you more do soon. In the S10 case, Function Escape seems to disable the camera, most likely the USB connection at the BIOS level. You get the device unplugged from Windows, and it's no longer in the device manager. And sites like TestMyCam.com in the Flash menu can't see the device anymore.

And seeing as how the function key can't be emulated, he says - and that's interesting, I didn't know that - I haven't seen a program capable of doing so, and they have to release a BIOS update to allow you to swap the control and function keys. Is this not secure enough without having to put something in front of the camera? Providing the BIOS has not been infected with something that could override this, is this feature providing adequate security? Although an activity light and maybe a thumb slide lens cap over the camera - ooh, that would be nice. Keep up the good work. Chuck. What do you think?

**Steve:** Well, Chuck is representative of a large body of listeners who wrote in asking about these things. I had to choose which question to put in the Q&A. And I decided not to put the one in where the guy listens to us without his clothes on because I thought, well, I wasn't aware that our content was that stimulating, but one never knows. If we knew for sure that the light which is sometimes next to the camera was directly connected to the power for the camera, then we could assume that they were inseparable, and if the light was on, the camera...

**Leo:** You couldn't turn on the camera without turning on the light; right.

**Steve:** Correct. But we don't know that in the case of any given laptop. Certainly it's possible that the camera power is on all the time, and the light is sort of a courtesy light, which is turned on by the software driver when it wants to, but that it'd be equally possible for someone to circumvent that.

In responding to another piece of email about this, I mentioned, I said to - somebody was wondering if the light was always associated with the camera. And I said, well, even if it was, how long does it take to catch some incriminating pictures or something that you would rather not have get out. You'd be furtively looking at the light every time you walked past the open laptop, worried that the camera might be on. Which really seems like the wrong approach. What I would like to see, the only thing that would satisfy me would be a physical lens, a physical lens cover of some sort.

I have a very high-power handheld laser, and it's so high power that, well, for example, it's able to pop a balloon. I mean, it puts out a seriously strong, coherent beam of light. And at that power level the government requires a number of safety features. It has to have a physical lock and key, and it has to have a delay between the time you press the button to turn it on, and it emits radiation. And it has to have a physical shutter which blocks the opening completely. Otherwise it's illegal at that power level. So whoever put those regulations in place understands that, even though you've got a keyed lock and a time delay before this thing turns on, there is no substitute for a physical shutter, which is physically blocking, in the case of the laser, this light getting out.

And I would argue as a technologist, nothing could satisfy me other than a physical cover over the lens because if this thing is controllable by software, and anything like this today is, I mean, if the camera had a physical switch, electrical switch up next to it, and if we knew what the schematic was, and that it actually did disconnect the power, then I'm satisfied. But we don't have schematics of our laptops. We don't get that level of detail anymore on our computers. So I'm 100 percent behind a post-it note, and make sure there's no pinholes in it, until manufacturers start giving us a little slide lens cover. And I'll bet that begins to appear.

**Leo:** I bet you're right. Oh, yeah.

**Steve:** That'll be a very nice feature.

**Leo:** Universal says in our chatroom that the lights on the camera are using GPIO, which means they would be software controllable. They're not physically hardware controlled.

**Steve:** Yup, doesn't surprise me at all.

**Leo:** If he's right, yeah, then it means a bad guy could easily keep it from turning on.

**Steve:** Well, and for example, if there was ever some software that had a feature of, for example, blinking the light in order to give you some level of, like, okay, we're about to start the connection, or doing something like that where the light is being controlled separately from the camera, well, game over. Now you know that it's under software

control.

**Leo:** Right. And of course let's not forget the mic. You could tape over the camera, but people could still hear what you're doing.

**Steve:** Yeah. And the two together, that could really be a problem.

**Leo:** Question 8, Paul Welch in Gold Coast, Australia says "Security Now! Taught Me Well." My short story, guys: My credit card company rang me yesterday on my mobile, but they blocked their number. Happens all the time. "Hello, Mr. Welch, I'm an operator from your credit card calling. May I have your four-digit passcode so I can validate you?" My question was, "But how can I validate you?" Short pause. "But I'm from the credit card company." I replied, "Well, so you say. But how do I know that? You want me to validate myself by disclosing my secret information, but I can't validate you? You could be anyone." So I asked for a number and called them back and got the information, as I knew who rang, thus validating them. You taught us well, Steve. Thanks. That's great.

**Steve:** Isn't that good?

**Leo:** It's a great story, yeah.

**Steve:** Yup, just a nice little perfect example of thinking correctly. You know, someone calls you on the phone, you can't see who they are, and suddenly the first thing they say is give us your secret password. Uh, no.

**Leo:** Yeah. I tell my wife that all the time. I say, if they claim to be from the bank or anywhere else, just call them back. Just say I'm going to need to call you back, I need a number.

**Steve:** And you know, I'll bet you it's not the first time they will have heard that. They'll understand. They're not going to throw a fit. It's not like you're trying to stick your knuckle on your fingerprint sensor.

**Leo:** They should do what I always do when somebody does something that enhances my security. When the bank, for instance, they're always apologetic when they call to verify a credit card charge. And I say, no, don't be apologetic. I'm thrilled. Keep it up. Keep up the good work. I love that.

**Steve:** Yes.

**Leo:** I want you to be doing this. It is not an inconvenience. Well, it is after about the 18th time. But I still want you to do that.

**Steve:** Yeah, the card I have which I cannot use for buying gas, that really does annoy me because it's like...

**Leo:** Because every time you use it they think you stole...

**Steve:** Yeah, because there's one other thing, apparently what they tell me is that's what bad guys do is they get a card, and they go to a gas station because it's an unattended transaction. And they're right next to their car for their quick getaway if they're not satisfied. So it's like, okay, fine, I'll just use a different card.

**Leo:** But you'd think they'd realize that you want to still use it to buy gas.

**Steve:** Yeah.

**Leo:** No. Dvorak says the biggest red flag is you go out and you buy a pair of Nikes, and then you go and fill two or three tanks at the same time. He says you will definitely get your credit card yanked if you do that. And actually I guess that makes sense. Question 9 from S.L. Garwood in North Carolina, USA wonders: Has authentication - or, I'm sorry, Authenticator, I guess that's a product - been cracked? I see that someone has a man-in-the-middle attack against the Blizzard Authenticator and is using it to grab data. Since Authenticator is the same as the PayPal football, would this attack work against any authenticator? Steve, help.

**Steve:** Okay. Here's the problem. As always in security, it's necessary to have a very clear definition about what it is that we're expecting our security system to do. I mean, I can't state that often enough or strongly enough. And I've said it before, but the PayPal football that we've talked about, a so-called one-time password kind of architecture, is designed to protect us from a replay, but not a first play. That is, a replay attack where if we used the same username and password every time, somebody could record that and later impersonate us by using the same username and password.

By using something like the PayPal football or the Blizzard Authenticator, which is exactly the same thing, it looks identical physically, and this is an authentication device that Blizzard introduced in order to enhance authentication of their members and players, just as PayPal introduced it to enhance the authentication of people using the PayPal service.

In the case of PayPal, you are absolutely on an SSL connection so that the data flowing across the wire is protected by the secure socket layer encryption. And that's where the one-time password data flows to prevent a man-in-the-middle attack. But absent the prevention of a man-in-the-middle attack, a one-time password doesn't give you anything. What you've done, if you've got someone in the middle who is able to intercept your correct username and password, even if it's only good once, they only need it once, that once. And so they turn around and impersonate you from their man-in-the-middle position, do whatever nefarious things they want to, and get away with it.

So given that this is the case - I just read this email for the first time as I was putting this together an hour ago, and so I didn't have any chance to go into any deep research into what's going on with the Blizzard Authenticator, but I didn't need to in order to understand that Garwood here is saying that there's a man-in-the-middle attack against

the Authenticator, it's like, yes, of course there could be because...

**Leo:** Yeah, just get in the middle.

**Steve:** ...that's not what it protects you from. No one-time password system can protect you from the man in the middle. So that's why it's really necessary to understand that its job is different from protecting you from any possible attack. It will protect you from one particular type of exploitation. But as always, security requires a complete ecosystem which is closed in order - with every part doing its job. And so it's like it sounds like there's a bit of a design flaw. Somebody, again, I don't know whose fault it is, how it was designed, what the system does. But it's definitely the case that something like the PayPal football, a one-time password, if it can be grabbed that one time, that's enough.

**Leo:** Yeah. Yeah, that makes sense. I really like how you say you've got to understand what the purpose of this is and what it can and cannot do. It's not an all-purpose, use this and nothing ever bad will ever happen to you again. That's not quite that good.

**Steve:** Right.

**Leo:** Question 10, our last question, and one that will be of great interest to those of us who use LastPass. Brent Longborough in - oh, boy, it's Welsh - Abersychan, Abergavenny, Wales, UK, wonders about LastPass: Hey, Steve and Leo. First let me say thank you very much for the podcast. I haven't missed one since #1. I just got my attention drawn to LastPass.com and wonder what you think of it. On the face of it, it seems like a good idea. But as I Trust No One, with the possible exception of Steve Gibson, I doubt. On the other hand, they appear to provide a full one-time password facility and have YubiKey support, so they can't be 100 percent evil. I wonder even whether this might be worth a Gibsonian investigation, perhaps with a dedicated podcast. Signed, Brent.

**Steve:** And it's going to get one.

**Leo:** You're kidding. Oh, good.

**Steve:** I was chatting with our friend Stina of Yubico.

**Leo:** She must know them, right, yeah.

**Steve:** Well, as a matter of fact, they're putting together a deal. I'm not sure what the nature of it is. But she's in town, that is, on the continent for the RSA conference, which is going on right now. And so she's going to come down to Southern California. We're going to do some coffee next week and sort of catch me up on all of the newest dealings in Yubico land. And one of the things she mentioned in passing was that, if I didn't know about LastPass, I ought to take a look.

**Leo:** Okay, yeah.

**Steve:** Well, I've heard you mention it. I've heard Paul speak favorably of it. So I have contacted the founders and the developers and opened a dialogue, telling them that I sort of wanted to find out who I could talk to, not some PR schmo, but the guy who actually made the bits all line up in the right order, so that I had a contact that I could ask some high-level technical questions of when I was doing a study of it. And they knew us and were glad for my interest. And I got email addresses and names of the founders. And so it's definitely on my to-do list. As soon as we get through with our existing stuff, it's one of the first things I have planned to do is do a thorough analysis of LastPass and explain it to myself, to you, and all of our listeners.

**Leo:** Well, the only thing that I thought was kind of interesting is that they - you can see here on the map, they are from Vienna, Virginia, which is just down the road...

**Steve:** From the spooky place.

**Leo:** ...from the spooky place. And I thought, well, I wonder if these guys, you know, because okay, here's my conspiracy theory. If I were, say, a three-letter government agency, with some interest in finding out what people are up to, I would make a password program that was so phenomenally useful and affordable and effective that everybody would use it. And I would just make sure that I had the only backdoor.

**Steve:** If you ever...

**Leo:** So would you ask them that?

**Steve:** If you ever hear of me or learn of me discontinuing CryptoLink for no reason, you'll know that for whatever reason I felt no longer able to offer something whose security I could put my reputation behind. Because I'll kill it before I would compromise it.

**Leo:** Yeah, of course you would, of course you would. And I'm just saying that, if I were the NSA, I would write my own password protection program. I'd make sure that it was the best in the world. I'd give it away and make sure everybody used it.

**Steve:** Yeah, the White House, you may have heard, just earlier this week, and I haven't had a chance to look at it, but actually I did hear it was pretty tame, the head of our cybersecurity for the U.S. currently, along with the administration's intention to be open and transparent, posted a bunch of information about the nation's, the U.S. nation's cybersecurity...

**Leo:** Infrastructure, yeah, they opened it.

**Steve:** Infrastructure, footprint and so forth.

**Leo:** Yeah.

**Steve:** Except they did, they are holding out any offensive information.

**Leo:** Right. Yeah, they don't want to talk about cyberwarfare particularly.

**Steve:** Well, they'll talk about cyber defense, but not cyber offense. That is, not what we do to hit back, assuming that it's back and not first. So it's like, okay, well, I'm not that interested then. And they kept the good bits out.

**Leo:** Well, but I'm kind of glad they did. Aren't you?

**Steve:** But the LastPass guys seem like good people. And I want to understand and explain what it is that the system does and what the architecture is. But exactly like I was saying for our first question, when the guy was saying, hey, you know, what about the encrypted notepad app, are you putting your reputation behind it? It's like, no. I'll put my reputation behind something that I'm 100 percent responsible for and wrote from scratch. But no one can put their reputation behind anything else, no one who's responsible.

**Leo:** And I can't say that it's safe. But I can say it is a great program in terms of functionality and usability. And I use it, because it's cross-platform, I use it everywhere.

**Steve:** And apparently it either does or will have Yubico support.

**Leo:** It currently does have YubiKey support. It's built in, yeah.

**Steve:** Very cool.

**Leo:** I mean, that gives you - but see, again...

**Steve:** I know.

**Leo:** If I were a three-letter agency, I'd make darn sure it did all of those things,

you know.

**Steve:** And does its operation require that it's connecting to the mothership?

**Leo:** Yes.

**Steve:** Yeah, see, that's always going to be a caveat for me. I mean, I'm deliberately designing CryptoLink so that, for convenience, you could use a rendezvous server, for example, for NAT penetration. But I and the people who I imagine my product would appeal to will say, wait a minute, I want TNO, even when Gibson is the third person I don't need to trust. And my point is, good. I don't want to be responsible for that, either. I'd much rather design something that needs no third party.

**Leo:** I think the password store is local and encrypted and on your system. I think it's unlocked by - I don't know. Because I notice I have to log in, but I'm not sure what - this is where we need you.

**Steve:** And this is what I will figure out.

**Leo:** This is where we need you.

**Steve:** I will lay out the architecture and understand exactly what it is and what they're doing. And with no reason to imagine at all that they're doing anything wrong. It's just...

**Leo:** No, no, no.

**Steve:** No.

**Leo:** Well, if you look at all the awards they have from PC Magazine, PC World, ZDNet...

**Steve:** But that all means nothing, of course.

**Leo:** If it doesn't have the Gibson Seal of Approval, I don't care.

**Steve:** Doesn't mean anything to me, no.

**Leo:** That's what we love about you. All right, good. Well, that'll be fun. Will that be next week?

**Steve:** No.

**Leo:** No. Sometime.

**Steve:** We've got to keep talking about computers for a while.

**Leo:** Oh.

**Steve:** People are really enjoying this journey through how this stuff works. And I love the foundation that we're creating. And besides, I've got to - this is not going to be something I just do in an hour. I mean, this is - if we get a comprehensive analysis from me, it's going to take a while.

**Leo:** Yes, of course. Well, good. I'll look forward to that. Next week more on the fundamentals of computing.

**Steve:** The stack, I think, is next week.

**Leo:** Oh, yay.

**Steve:** Really, really wonderful piece of conceptual computing technology.

**Leo:** Yeah, yeah.

**Steve:** Very handy.

**Leo:** Steve Gibson's the guy in charge of the Gibson Research Corporation. That's why his website is GRC.com. You can go there and get SpinRite, of course, the world's best hard drive recovery and maintenance utility. It's just a must-have for anybody with a hard drive. But he also has great free stuff you can get there like his Perfect Paper Passwords thing, and his ShieldsUP!, and oh, there's just a ton of stuff there.

If you have questions you'd like to get in the next Q&A, go to GRC.com/feedback. If you'd like the show notes, go to GRC.com/securitynow. There are 16KB versions of this show that Steve puts together on his own. He's the hardest working man in show business. Also transcripts from Elaine and all the show notes, as well. That's GRC.com. Steve, thank you so much.

**Steve:** Always a pleasure. We'll talk to you next week for another great episode. And till then.

**Leo:** Bye bye.