SECURITY NOW!

Transcript of Episode #40

Listener Feedback Q&A #7

Description: Steve and Leo discuss questions asked by listeners of their 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 they have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-040.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-040-lq.mp3

Leo Laporte: Bandwidth for Security Now! is provided by AOL Radio at AOL.com/podcasting.

This is Security Now! with Steve Gibson, Episode 40 for May 18, 2006: Listener Questions and Answers #7. Security Now! is brought to you by Astaro, makers of the Astaro Security Gateway, on the web at www.astaro.com.

I look forward to Thursdays every week because I know I’m going to learn so much about security. Ladies and gentlemen, Steve Gibson, security guru.

Steve Gibson: Hey, Leo, good to be back with you.

Leo: It’s good to have you. This is Episode 40, so it means it’s one of our Mod 4 episodes. We’re going to do questions and answers. But there’s some big – a lot of big security news, and I know you wanted to cover some of that.

Steve: Yeah, a couple things. For one thing, Apple announced 31 vulnerabilities this week. Yeah. With, you know, the OS X version 10.4.6 and previous. And those, you know, there are local and remote code execution problems, information disclosure, denial of service, and local privilege escalation problems. I mean, basically just a whole collection of the typical kinds of problems that, you know, exactly like what we were talking about last week and the week before with the – this is the fundamental problem of writing secure software. And as we all know, Apple has for a long time enjoyed a history of much fewer problems than Windows has been exposed to. In fact, you’ve probably seen these commercials that they’re running now, Leo, you know, the guy who’s standing in for Apple and one who represents the PC.

Leo: Yeah.

Steve: And how the PC gets viruses so quickly, but there aren't any for the Mac.

Leo: I have some problems with those ads. But they're ads. They're marketing, you know.

Steve: Yeah, and they're getting the message across. What I'm noticing is that these are coming out just as it's beginning to be no longer true.

Leo: Right, right. Well, it's interesting because Apple never wanted to play that virus card. And I think that they – one of their resistances was they didn't want to antagonize virus authors. It was the best kept secret in the computing world was there's no viruses, and don't tell anybody.

Steve: Well, and historically, as we have said before, people who write viruses write them for the computers they own. And Macs have, again, you know, years ago not been in the mainstream. They've been, you know, in schools to some degree. But what kids had to play with were PCs, and so that's what they wrote viruses for. Increasingly now, of course, Macs are affordable, and they're an increasing target.

Leo: You watch, though. I don't think we're going to see the same degree of virus problems on the Mac platform, just because many of the things that make viruses possible that we've talked about on Windows, things like ActiveX, ActiveScripting, the fact that every user is really running as admin, those things aren't as prevalent or even possible on the Mac side.

Steve: And population size. I mean, a virus author who wants a well-propagated virus...

Leo: True.

Steve: ...will still write it for Windows, just due to the population size.

Leo: Right. I mean, nowadays, the reason people write viruses is to co-opt computers.

Steve: Right.

Leo: And you're going to go where you can get the most computers.

Steve: Exactly. Well, another...

Leo: So that's one issue. Yes?

Steve: Yeah. Another really biggie is that a flaw has been found in the RealVNC remote authentication. RealVNC is a very popular open source remote desktop system that allows people to, you know, basically connect to a remote machine and then get access to its desktop.

It’s super well used. This is a really bad vulnerability because it defeats, completely defeats the authentication, meaning that anyone who can find a RealVNC server, or that is to say service port, exposed, until this patch is applied and RealVNC is updated, can connect to it and basically log in immediately.

Leo: Ooh, boy.

Steve: Now, I got a kick out of the what to do in the meantime. People were saying, well, you know, close any ports that you have exposed to the Internet. And this comes back to something that I’ve said before, and I want to keep reminding people, which is never leave default ports alone. People who, for example, are running web servers, you pretty much have to have the server running on port 80 because that’s where everyone’s browser is going to expect to see it. Now, if you were running your own web server, for example, from home, just for remote access for you, for example, if you wanted to access your documents or something remotely, then by all means move that web server to some funky port of your own imagination. The only time you really need to leave ports in their default is if you need people who don’t know what port you’ve chosen to access your service remotely.

So in the case of RealVNC, someone running a RealVNC server at home, so that, for example, they’re able to access their home desktop remotely, by all means change the default port to some random number between 1024 and less than 65535 because what happens is, when a vulnerability like this is discovered, the bad guys start scanning the Internet for connections on the default RealVNC port. When they get one, knowing what the vulnerability is, they can log on to your desktop, and you’re theirs.

Leo: There’s another way you could secure it. We use RealVNC for the radio show for call screening. But what we do is we establish a VPN link first and then open the RealVNC. So it’s within a VPN tunnel, which until VPN is – which could be, but until it’s compromised, that’s safe. Right?

Steve: Well, as long as the RealVNC server itself is not exposed...

Leo: No.

Steve: ...that is, the RealVNC port is not exposed to the Internet...

Leo: It’s internal. It’s internal.

Steve: Then you’re absolutely right, that’s a great solution.

Leo: Yeah. So in other words, I have to get on the Local Area Network by using VPN...

Steve: Right.

Leo: ...before I can get access to the VNC server.

Steve: Right.

Leo: It's not exposed. It's not naked to the world.

Steve: That's a great way to fly. So for anyone listening to this who is using RealVNC, immediately update your RealVNC to the latest version that you can get from their website. They've fixed the problem, but we want to get the message out. And while you're doing that, if you are not running it on just some non-default random port that you've chosen, make that move now. Write the port down or use something memorable, but that's not leaving it at the standard service port. Because you always want to run services on non-default ports when you can. This is a perfect example of why that's a really good lesson.

Leo: Of course it's not fully secure to do that because some of the sniffing software will just attempt to make a connection across all ports. But it's a lot slower to do that. And so I would imagine most people just stick with the default ports. And there are enough open ports that they can just use those.

Steve: Right.

Leo: But if they really wanted to attack you, they could test every port for a VNC connection.

Steve: Well, yes. For example, another approach is, instead of scanning the Internet, if somebody knew your IP, for example, from it being posted in a bulletin board, blog, or in email that was made public, or in a newsgroup forum where the email was available...

Leo: On IRC, on BitTorrent, I mean, people know your IP address in many cases.

Steve: Exactly. It's not something that it's easy to keep quiet. If someone knew your IP, and they knew somewhere on that IP, at some port on that IP, there was a RealVNC server, you know, for example, you'd bragged about it in the past or talked about it or something...

Leo: They could find it.

Steve: They could, exactly. Then, you know, the fact that they know what IP it's on narrows their search dramatically.

Leo: Right, right. Ah. If you used Hamachi to establish the connection and then had RealVNC running locally...

Steve: That's a great application for Hamachi, and you'd be completely safe.

Leo: Good, all right.

Steve: Another interesting tidbit, though, I just wanted to mention, is there's sort of a boiling controversy over the purpose and the future of the Whois database that ICANN is trying to decide what to do with.

Leo: Now, this is of great interest to me. I'm sure it is to you, Steve. But when I have a website, and I do have quite a few, my address, my phone number, those are made public in the Whois database.

Steve: Well, for example, one of our listeners created or grabbed the domain SecurityNow.info.

Leo: Oh, dear.

Steve: Well, I know. SecurityNow.info brings up a little page that redirects people to my SecurityNow.htm page.

Leo: Well, that was nice of him.

Steve: Very nice of him. But I didn't want to sort of formally endorse that because he has control of the domain and I don't. So I tried, I used the Whois database to try to find him. I got an email address. I don't remember now, it was maybe at GoDaddy or something, and sent him a note saying, hey, I notice you registered SecurityNow.info. Would you be willing to transfer that to me? I'd be happy to pay your expenses, blah blah blah. I never got any reply. Chances are it went into some bit bucket somewhere, and he never saw it.

Leo: So he's listening now. Yes, we're looking for you.

Steve: That would be – I'd really love to have the domain and be happy to, again, cover his expenses. But, again, I can't endorse it because we don't have control of it. So, you know, so that's a problem. But, for example, both sides of this is interesting because, on one hand, people, for example, political dissidents or bloggers may want the option of retaining some anonymity. And of course the anonymity is something that the Internet to varying degrees has always offered people.

Leo: You know, television show hosts might want some anonymity. Don't forget that.

Steve: Sure, sure. And so on one hand you can say that the Whois database needs to not necessarily identify someone to the degree that would potentially compromise their own personal safety. The flipside is that, in the post-Katrina disaster relief stuff, we know that many fake "donate to Katrina victims" websites were created, and the Red Cross was very upset with those sites. And they were able to use sort of the soft and fuzzy, doesn't necessarily have to be accurateness of the Whois database to hide themselves, and also to, like, put non-verifiable obscure records in. So ICANN, and in general the Internet community, are really at odds right now over what to do about the problem.

Leo: There is a simple solution, by the way, I think, is a third party involved. And in fact that's what I do with GoDaddy. There's a company that offers its service, for a fee, at

GoDaddy. I think it’s Domains By Proxy. And everything – they hold the information. And everything in my – if you go and do a Whois on TWiT.tv, you’ll see Domains By Proxy. And then I can go through Domains By Proxy, and presumably a police agency or ICANN could go to Domains By Proxy and say, hey, look, you know, this is a spurious site, we need that information.

Steve: Right.

Leo: I think that would be the way to do it, and I hope that they find a solution. Have they come up with anything, or...

Steve: No, it’s really interesting. I spent some time the other day reading through the, I mean, just the incredible bureaucracy of this kind of process with, you know, opening the forum to the public, allowing people to give their pros and cons. And, I mean, as with any complex issue, it isn’t clear that there’s a cut-and-dry solution. And in fact the notion of running through a third party, as you suggest, is discussed there. And, again, there’s like the pros and cons of needing to do that. So...

Leo: Well, it’s government. And whenever – I mean, it’s a quasi-governmental organization. And whenever government gets involved – I was briefly on the technology committee for my small town here. And it was so frustrating because the laws of California – and they’re good laws that require, you know, the public to be informed about everything you do, things like the Brown Act – made it very difficult. We couldn’t have, for instance, a private message board or even a mailing list among the group members because the Brown Act prohibits that. So it’s both good and bad. And this is – that’s the way things are. But I do think the third party ultimately will be the way to go. I’m happy with it. Nobody can get my information, but it is available to, you know, somebody if they absolutely needed it.

Steve: If they can demonstrate a real need.

Leo: Yeah, exactly. You know, they’d go to court or whatever.

Steve: Yeah, I think, I guess, the issue for me is that I can see validity to both sides of the argument. So I think you’re right, Leo, that...

Leo: Oh, there is, yeah.

Steve: ...that having some intermediary who acts as a buffer for the information probably is the way it’s going to wind up being handled.

Leo: And you and I are on both sides of it, really, because on the one hand I don’t want somebody to have my phone number, my home address, or even my office address. But at the same time I don’t want anybody impersonating me on the ‘Net.

Steve: Exactly.

Leo: So of course we can see both sides of the equation, yeah.

Steve: Exactly. And then one last thing. A listener named Weaver wrote from the U.K. He loved this notion that I talked about a couple weeks ago of browsing the 'Net with basically, in my case, with IE locked down tight, and then specifically whitelisting sites where I wanted scripting and more features to be available so that – in fact, the way he put it, he said: "I loved your idea of interviewing a website and then deciding if it was worthy of trust and then giving it trust." He wanted to make sure that, for people who are Firefox users, that they knew about a plug-in called NoScript. It's just – the URL is www.noscript.net. And it's apparently a highly regarded plug-in. I did a little browsing around, and it's got five out of five ratings, and it's been recommended by lots of people. The idea being that, similarly, JavaScript and Java are prevented in Firefox until you decide that you want to or need to enable those for a site that you trust. And just a single click on a little block icon down in the status bar will enable, then basically whitelist the site. And this thing works in Mozilla and Firefox.

Leo: Important to make the distinction, though, between the kind of scripting that can be in Internet Explorer with ActiveScripting and ActiveX and what's available in Firefox. I mean, JavaScript, as far as I know, doesn't have anywhere near the risks inherent in ActiveScripting.

Steve: Some people did comment, in fact I think it was on your TWiT board, Leo, that, you know, plug-in code in Firefox is as vulnerable as ActiveX code is.

Leo: Absolutely, sure. Because it's running on your system.

Steve: Exactly. It's client-side code that you're inherently trusting if you use those plug-ins.

Leo: Although most plug-ins are written in JavaScript. They're written in XUL, which is basically JavaScript. And so the plug-ins, they're user interface. But they don't have, again, they often don't have that kind of capability.

Steve: And, I mean, they don't have that level of power.

Leo: Right. I mean, they could. And that's the point. You should always be careful when you install any kind of plug-in.

Steve: And I guess that is to say the bad ones would.

Leo: Right. One other point I wanted to bring up, we got an email from a fellow who said, you know, Steve's method of locking down IE is great as long as the bad guy hasn't modified your hosts file. But as soon as somebody modifies the hosts file, he can change where that apparent script is coming from and breach the security.

Steve: It's a very, very good point.

Leo: So if somebody has access to your system or has gotten a Trojan on your system, it could modify your hosts file, and then it'd be all over.

Steve: But flipside...

Leo: By that time it's too late.

Steve: Yeah, I was going to say, the flipside of that is, if they're already in your system to the level that would allow them to modify your hosts file, then you're pretty much hosed anyway.

Leo: And a lot of spyware does that, by the way.

Steve: Yeah. We have not talked about the hosts file, and we're definitely going to because it's ubiquitous. It's an interesting sort of not well known, unless you've really dealt with this stuff before, sort of way station for DNS lookup that is powerful and vulnerable, both at the same time.

Leo: And people can use it to protect themselves. And I think...

Steve: Yeah.

Leo: ...I would love to have you talk about that sometime.

Steve: We'll do it.

Leo: We'll save that for another one. Right now, though, we've got dozen questions from listeners, a dozen out of the 5,000 you collected.

Steve: Actually I did the math. I have two folders, one for people who submit things anonymously and one where I could reply to them because they've given me an email address. The sum of the email in those two folders is 6,010 messages at the moment.

Leo: Which is why you're not getting a personal reply from Mr. Gibson. Or from me, either.

Steve: Well, I do when I can.

Leo: Yeah. As do I. First, from Jason in Atlanta, Georgia, he actually has two questions. So he's – double your money, Jason. He says: I have Microsoft Small Business Server 2003 at our office, and I'm curious if the SSL certificate that our server created itself is as secure as if I'd bought one from VeriSign. That's a good question.

Steve: It's a great question. And the answer is yes. The SSL certificates, which any contemporary system will generate, are absolutely as secure as what you buy from VeriSign.

Leo: However...

Steve: The only difference is that VeriSign is standing behind the identity that that certificate represents. Which really, if you've made one yourself for your own users to access your server, that's really not an issue. You just have to make sure you don't lose control of it.

Leo: Would it be seen by browsers as trusted, though, since it's not – they can't relate that certificate, as you've talked about before, back to those handful of trusted third parties?

Steve: Well, that's correct. So the idea would be that, if you're using a certificate that your own small business server has generated, then everyone who needs to trust that would also have to import into their browser or into their system the matching key for the server so that it's able – so your browser is able to verify the certificate. Because, as we know, web browsers come pre-built with a huge array of certificate – certificates is what I'm try to say – with a huge array of existing certificates that allow them to automatically check the security of anyone who's trying to connect.

Leo: Right, right. He also says: If I'm at a secure site, and I go to a web page on that site that has another page embedded in it, like as in a frame or an iframe, would the information be encrypted between my machine and that embedded page in the secure page?

Steve: Ah, that was a good question, so I wanted...

Leo: That's a great question.

Steve: Yeah, I wanted to include that. The answer is no. The level of encryption is at the level of the connection, that is, the asset or the resource which is being pulled from a remote server. So if you had a frame, and the frame loaded a non-secure URL, your browser would make a non-secure connection, so that the security on the outer frame, or the outer page, does not necessarily imply anything about the security of content inside.

Leo: That's why I get that error message from my browser sometimes that says "Part of this page is encrypted, and part of it is not encrypted"? Is that what that's talking about?

Steve: That's just what I was going to mention, exactly. For example, in IE and other browsers, sometimes it's called "mixed content," the idea being that not all the assets on the page are secure. Some – and it might just be images, for example, that are going to a third-party server that is not a secure connection. And so it's saying, hey, you know, you should know that some stuff on this page is not secure. So it may be that your browser will alert you, but it's not necessarily the case unless you're sure that it's going to be a secure asset.

Leo: I think some browsers have a modified key icon, or lock icon, when you're on a mixed site like that. It's funny, I don't remember for sure, but I think I've seen that.

Two people asked questions relating to President Bush's proposal for tamperproof ID cards. Steve Gilliam of Pinehurst, North Carolina, wrote: President Bush recently said he wants to give tamperproof digital fingerprint IDs to legal guest workers. How could public and private key encryption be used to generate these IDs? And after it goes into effect, how would counterfeiters try to defeat it? That's one of the problems right now is that illegals often in fact have papers that look real because counterfeiters have gotten very good at forging birth certificates, Social Security numbers, and that kind of thing.

Steve: Sure. Go ahead and read Ben's question, too?

Leo: Ben also asked: The President suggested we needed a document that includes biometric info to make a tamperproof ID card so that it'd check your fingerprint or your iris or whatever for every legal foreign worker. Do you think this can be done to a reliable degree? He says: I tend to think this would require a federal database so the information on the ID card can be validated and probably so the card could be used as a kind of token. I would imagine that a fingerprint could be used like a key to unlock the encrypted info. But I wonder about the likelihood of a standard electronic fingerprint reading the same thumb each way each time. I know when you come into the U.S. from a number of nations, you do, in fact, get fingerprinted. In some cases they also do iris recognition. I mean, this kind of system does exist.

Steve: Well, I liked this question, or both of these questions, because it plays perfectly into the foundation in crypto that we've laid before. Now, biometrics, of course, is sort of valuable but scary, depending upon how it's employed. But our crypto side gives us enough of an understanding of how something like this could work. Imagine that you had an ID card, and on the card was a picture of the person who had applied for the card and some printed textual information, you know, name, address, whatever textual ID that they want to have. Then on a, for example, on a mag stripe like we have on our credit cards, on a mag stripe the same information, that is, a JPG of the person's face and the same information printed on the front of the card would be on the stripe. All that we have to do is take that information and hash it, as we've seen before, that is, create a digital signature of that information. Then somewhere there would be a master government public key which would – well, public and private key. The people generating these cards would use the private key to encrypt that digital signature. And that's basically all we need to do.

What that means, then, is that anybody carrying the card who wanted to have its authenticity verified could present it to anyone's reader anywhere. The reader would read the mag stripe, put the information up, that is, basically redisplay the photo and the textual information, so that the operator could compare it to the person standing there. So in this sense, you know, the biometrics is a photo. And, of course, people are pretty good about comparing pictures to people's faces and deciding, oh, that was you two years ago kind of thing.

But the key is, because this information had been digitally signed by the government, all of the readers would only need to have the government's matching public key. So any reader could verify that this information, the information on the mag stripe, not necessarily the visible information on the front of the card, but the information on the mag stripe had not been changed by a single bit. And there's nothing, then, in terms of intellectual property, that could be stolen from these readers. I mean, it could just be freeware that you load into a PC that has a little add-on mag stripe reader. Anybody could then verify that the digitally signed information on the mag stripe was originally signed by a government agency.

Leo: So it's really just like me using my PGP key to sign my email. You can verify that I sent you that email, that that was my key and that the email has been untampered with, just by going to the PGP site and matching the keys.

Steve: Exactly. It's an absolutely analogous application.

Leo: Yeah, yeah.

Steve: And crypto solves the problem for us.

Leo: It's really amazing. Public key crypto. And that's what's so great about public key.

Steve: Right.

Leo: Ryan from Ontario, Canada writes: I know you've never had an episode specifically about key loggers. I'm sure we'll rectify that at some point. But can a key logger be fooled if you use a program like RoboForm – which, by the way, is a great program. I recommend it. It fills out forms for you.

Steve: Right.

Leo: To fill in your passwords. What are the ways to fool key loggers if – a big if – a key logger manages to get on my system? I know software firewalls can warn you if a key logger phones home. But viruses and spyware maybe disable your software firewall. Or, frankly, there are hardware keystroke loggers that you'd never see.

Steve: Right. Yeah, I wanted to address this because we actually did discuss key logging stuff early in our podcasts when we were talking about passwords and entering passwords. And many people came up with clever ideas for defeating key loggers, like...

Leo: Oh, yeah, I do remember that, yeah.

Steve: ...typing a couple characters, then clicking the mouse into, like, a notepad...

Leo: Cutting and pasting.

Steve: ...type in a few more, cutting and pasting, scrambling things up, I mean, there's all kinds of things you can do if you were concerned about that. So it certainly is the case. First of all, RoboForm, which is down at the API of Windows, is not necessarily going to be proof from a key logger because, in order to fill in these forms, most of these utilities – and in order to be like a universal form filler, many of them literally simulate the keyboard, typing into the field, so that the same hooks into the system which allow a key logger to see keystrokes will similarly see something like RoboForm basically typing that information in for you.

Leo: And I guess it really depends on the key logger, too, because some are lower level than others, and some are higher level. I mean...

Steve: True.

Leo: ...if a keystroke logger's just looking for scan codes, well, it probably would miss it. But if it's working at the API level it would probably see it, so...

Steve: So actually, some of the things that people suggested, some of the clever ideas in the brainstorming back at the beginning of our podcast series, I thought were very workable, where nothing that was just looking at the keyboard would be able to tell where you were typing these things. So change your keyboard focus a few times, putting characters in different places, so that the end result of the string you want to have correct is correct, but nothing passively watching would be able to figure out what you had done.

Leo: Unless it's smart enough to be watching those fields and looking at the input in those fields. I mean, in other words, it depends on the keystroke logger.

Steve: Yeah. Bottom line is you really don't want to get anything bad in your system.

Leo: No. And it's possible to put a keystroke logger on your keyboard that memorizes stuff. If you use a public terminal...

Steve: Public access terminal, yes.

Leo: ...very likely – or not very likely, but very possible that all your keystrokes are being logged. But RoboForm would avoid it in that case. Of course, it would probably leave enough traces behind that it wouldn't make any difference.

Brian of Everett, Washington is concerned about MD5 hashes. In one of your cryptography podcasts you talked about the MD5 cryptographic hash as a way to verify downloaded files. However, I just read on the Internet the MD5 algorithm has recently been found to be unreliable, and hashes generated with it can be cracked. I think we talked a little bit about this.

Steve: Yup.

Leo: He says: Is it true? A different Internet site talking about MD5 said you could salt an MD5 hash to prevent it from being broken. What exactly does "salting a hash" mean? Sounds like something they did in the Navy. But does this fix the problem, or is the MD5 algorithm truly unreliable?

Steve: Okay, there are a couple things. MD5 generates a 128-bit hash, which is beginning to be not long enough. So there are now 256-bit hashes and 512-bit hashes which are arguably way overkill. But, you know, with computers being as fast as they are and storage being as big as it is, there just isn't really a reason not to use a larger hash to prevent hash collisions, which is what adding more bit length to the hash does. So on one side it's worth mentioning that this 128-bit is a little short.

But what the sites are talking about are something we've discussed in the context of hashing before, which is brute forcing a hash. If you use an unsalted MD5, what sites are doing are they're precomputing for many common passwords and pass phrases, that is, up to a certain length. You start getting too many of them as the length grows. They precompute the hash. So you end up with a huge table of hashes, and the password that, if you hashed it, would result in that hash.

So what salting does, salting adds some extra fluff, some extra stuff to the input in order to completely change the output. So, for example, if you put in the word "dog" into an MD5 hash, you're always going to get the same 128 bits out, no matter how many times you put the word "dog" in. So in a precomputation dictionary attack on MD5, the hash for "dog" would probably be there. If somebody tried it, then they'd be able to figure out what your password was because you'd chosen a really bad password. I mean, there's no way that "dog" is a good password to have.

On the other hand, if you just added, for example, a minus sign or an underscore or some sort of what's called "salt" to the hash, then that would completely change the output so that you'd get, I mean, with any good hashing algorithm, a completely different set of 128 bits. So adding any other debris to the algorithm will completely change the result and render it basically uncrackable to this kind of brute force attack.

Leo: But you can't just add it to the hash. It has to be – the coding program has to know about this underscore. You can't just make up something and put it in the hash.

Steve: Well, the idea would be, for example, UNIX systems use a salted MD5. They've got some junk that is unique to the system which they always add when anyone types in a password.

Leo: Ah.

Steve: So that nobody who sees the resulting hash is able to, you know, basically have a universal crack for the system.

Leo: Ah, very clever.

Steve: Yeah, that's very nice.

Leo: Very clever. Carl Schweitzer of Hilbert, Wisconsin managed to untangle last week's discussion of the stack and buffer overruns. Actually a surprisingly large number of people wrote to us thanking us for it...

Steve: Yeah.

Leo: ...and saying they got it. And that's great. I mean, that was one of the more complicated things we've discussed in the podcast.

Steve: I think I used...

Leo: You did an admirable job, but...

Steve: I think I used the term "aggressively technical" as...

Leo: But he did have a question. He said: When it comes to the stack, why did it have to write up? Wouldn't it be more secure to write down? Or is there something more complicated going on with that?

Steve: Okay, now, I didn't quite get his question.

Leo: He's talking about why does it go up in memory instead of down in memory. But I don't think it makes any difference, does it?

Steve: Well, actually there's something brilliant about this.

Leo: Oh.

Steve: Which I really liked. First of all, he says, why does it write up, meaning why does your – when you're overflowing the buffer, you're going from lower memory to higher memory.

Leo: Right.

Steve: And that's writing up the stack, and that's overwriting the stuff that's higher in the stack, which is critical information from previous programs or previous subroutines. Because as we remember from last week, the stack grows downwards. Well, his point was, if buffers overflowed the other direction, then they wouldn't be overwriting the previous information, they'd just be writing – they'd be overwriting stuff that wasn't allocated yet on the stack. Well, he's completely correct. Now, of course...

Leo: So why don't they do it that way?

Steve: Exactly. I mean, they should. I mean, I love the question. It's brilliant.

Leo: Wait a minute. Wait a minute. Now, come on. You're turning years of computer science on its head. Literally.

Steve: Well, okay, now. First of all, it's not easy from a practical standpoint to fill buffers downward. That is, you know, just everything about the way we think has a buffer being filled from lower memory to higher memory.

Leo: But even if you did that, you would still overwrite – you could overwrite in the negative numbers.

Steve: Well, but hold on. But if the stack instead started at the bottom and grew upwards...

Leo: Yeah.

Steve: ...and there's no reason it can't, if the stack started at the bottom and grew upwards...

Leo: So you're saying if it started at memory location zero...

Steve: Or, well, zero, not everybody can start at zero.

Leo: Not really zero.

Steve: Yeah.

Leo: Let's say whatever zero is, some arbitrary zero.

Steve: Yes. And then allocating memory on the stack you do by moving your stack pointer up to, like, a certain amount. Now, what's happened is you've reserved all of that memory below that point for your own use. Anybody else who wants some, they move the pointer up, and now they have – they've reserved that region that they just moved the pointer across for their own purpose. And the beauty of this is, the person who is currently using the stack, so-called is like on the top of the stack. And if stacks grew from the bottom up instead of from the top down, then buffer overruns would overrun out into unused stack space, not already used stack space. It's brilliant. I mean, it would really work. And it would solve the problem.

Leo: Somehow I think there's more to it than that, but all right. If it were that simple, wouldn't somebody have done this?

Steve: No. Because, again – no, I mean, I...

Leo: All of this was set up long before this ever became a problem.

Steve: Exactly. That's exactly it, Leo, is that it's a little more elegant from an architectural standpoint to have, for reasons that are sort of complicated and deal with the way pointers are handled, to have the stack growing down from the top of memory is architecturally simpler. But it was just done that way basically as an arbitrary choice. Someone said, shall we have it come down from the top or up from the bottom? And it's a little simpler to have it come down from the top. But in terms of the problems that it creates, the idea that overrunning the allocation, if you're coming down from the top, you're inherently overwriting the information of someone who's already allocated information on the stack before you.

Leo: Right. Somebody else's stack frame.

Steve: Somebody else's stack, exactly. But literally, if you allocate instead from the bottom up, you don't overrun. It's wonderful.

Leo: Okay. I know we're going to hear from Randal Schwartz, or one of our other programmers is going to say something. But I can't think of what it could be. Maybe you're right, and maybe this is a way to design a whole new architecture.

Garrett of Rochester, Minnesota is wondering about Skype security. That's what we use, of course, to do the show. He says: How could I properly configure Skype for the best security? I'm sitting behind a NAT router with Universal Plug and Play turned off and all the security settings on high. Anything else I need to do for a Skype connection?

Steve: No.

Leo: Yeah, we don't – I don't do anything particular.

Steve: Exactly. Now, the one...

Leo: You're going to give away our secret.

Steve: Well, the one gotcha is that it can be useful to allow incoming connections to Skype in situations where both people who want to connect a conversation are behind NAT routers. But in this case, Garrett was asking about SkypeOut connections, where he's using Skype to connect to the Skype servers in order to use Skype sort of as his telephone system.

Leo: But that's – isn't that – that's how Skype avoids NAT issues is we're always going through a server to make the connection; right?

Steve: Well, no. Between two users there is a server to rendezvous the two users.

Leo: Yeah, so we don't have to worry about the incoming connection because I've already established a connection with a Skype server. Right? Right.

Steve: Exactly. So Garrett is completely secure. He's behind a NAT router which is giving him the inherent stateful firewall action that NAT does. He's got Universal Plug and Play turned off.

Leo: Good.

Steve: So nothing nasty is able to reconfigure his router behind his back to open ports. And he's calling out, he's connecting out to the Skype servers for his SkypeOut service. And so nothing is able to connect back in.

Leo: But I wouldn't have an issue, I mean, and I don't have an issue, if you call me on Skype. Right? I mean, once I've logged into Skype, anybody can call in, as well.

Steve: Correct.

Leo: So, again, because it has a third-party server. I thought you were going to reveal our secret, which I don't want anybody to know, about how we get these Skype calls to sound so good, which is that we use a dedicated Skype port. You told me to do this, and I have to say it has improved the quality of the Skype calls. Why?

Steve: Well, because I've configured my Skype so that I'm able to – I have basically, of course, a non-default random port where incoming Skype connections are able to reach me.

Leo: You can do that in the Skype preferences. It's in the advanced preferences. It's easy to...

Steve: Yes. And the reason – I have a NAT-hostile network configuration. As you can imagine, Leo, my security here is pretty strong. And you and I, when we were connecting, were not able to get a direct connection because my NAT router wouldn't allow it. So that our dialogue was going through a so-called "supernode." It was being relayed by a third party. And when you were telling me that you and I were having such great success with Skype but you were having some trouble with other uses of Skype when you were talking to people, that's when I suggested, well, if you did the same thing, if you allowed a fixed port to come all the way in from the Internet to your Skype, then basically, you know, if you – then it's like you're only behind a single NAT router. That is, the person you're connecting to is only behind a single NAT router. They are always able to initiate a connection through to you, so a supernode is never used, and you get this kind of Skype quality every time.

Leo: The supernodes are a clever hack that the Skype folks did so that you don't have router problems, NAT router problems. But you can avoid it by using a dedicated port. Now, I do that unilaterally, right, I don't have to tell them what port I'm using. I just choose a dedicated port in my connection options.

Steve: Exactly. And suddenly you're just a higher quality Skype connector location.

Leo: Do I have to open that port in my router or anything like that?

Steve: Yes, you have – yes, yes, yes. You do need to have that port mapped through so that somebody – the idea is that, if both people at each end of a Skype conversation are behind NAT routers, then they may not be able to negotiate to connect to each other directly on the fly, for complex reasons of the way NAT works. But if either of the people will create a statically mapped port through their NAT router, then the other person can always send a packet directly to that port, and the whole Skype protocol manages making sure that everyone knows what the port is.

Leo: Heh heh heh.

Steve: It works.

Leo: Now you know our secret. It really does work. It makes a huge difference. Not for everybody, but it does on those people who have – and do you map just TCP, or is it UDP? What do you have to map?

Steve: It's just UDP.

Leo: Just UDP. Okay. Now I've given away our secret. We have no advantage whatsoever anymore. That's all right. Except our great good looks.

Kim White of Secunda, South Africa wonders about securing mobile laptops: I have a question pertaining to security of notebooks. Could you guys do an episode on securing of notebooks so that when, not if, it gets stolen, the data should be safe? I've seen some ideas that use USB keys and so forth that are used as tokens, but these only protect from log-in. What about from boot? If the drive is locked from boot then there's almost no way that the data can be stolen. My concern is that most of my laptop users have IPs from large corporations here in South Africa. Not IP addresses, intellectual property I should say.

Steve: Right.

Leo: It's worth hundreds of millions of dollars in the open market. He's concerned, and this happens all the time, in fact, it happened very famously to the British Secret Service whose laptops were stolen and secrets were □escaped.

Steve: Well, of course, you know, in the news from time to time you hear about exactly this kind of problem. A laptop, you know, is left unattended briefly at an airport, and it disappears. Some corporation has a ton of valuable data on their laptop. Okay. There are a couple things going on. Some IBM ThinkPads actually work with the drive's serial number to lock the laptop drive so that it will not boot unless a password is given at boot time to unlock the drive. That's very good security, but it has caused problems because it is such good security that, if that password is ever lost, you're hosed. I mean, there is no way, short of returning the drive to the manufacturer, to unlock the drive. And the problem, of course, is that this is not generally available on most laptops. So a solution that is laptop specific may not be a good one, either.

Leo: It's often said that if somebody has physical access to a machine, eventually they can get the data.

Steve: That's not true.

Leo: No?

Steve: No. Because – and this is going to be the topic of next week's Security Now! podcast.

We're going to talk about TrueCrypt.

Leo: Yeah, we love TrueCrypt.

Steve: It is a fantastic open source encryption solution. Now, what Kim's company would have to do would be, I mean, and this is not just trivial to do, but the idea would be that most laptops are configured with everything sitting in one single huge C partition, you know, from a Windows drive designation standpoint. What you need to do is do some preparation for this, the idea being that you create another drive where all of the intellectual property resides. And you just have the discipline to keep everything that you need to have safe in this second drive. And then a program like TrueCrypt, which we'll be talking about next week, is able to provide unbreakable, I mean, truly state-of-the-art, you know, past the end of the time of the universe-style encryption to prevent anyone from accessing that inadvertently. And it can be configured so that, if the drive is turned off, if it goes into standby, if it's unattended for a certain length of time, that you must reenter a password in order to access that content. And it works for laptops; it works, well, you know, for PC, desktops, anything. TrueCrypt is a fantastic solution, which is why we're going to be covering it in detail next week.

Leo: Generally people set TrueCrypt so that it automatically unlocks once you log in. So you might want to put on the BIOS passwords and the log-in passwords and all of that stuff, just to, you know, keep it a little more difficult to log in to the computer.

Steve: Well, yeah. Certainly, if what you're suggesting is that they have TrueCrypt configured so that you don't need to enter a password...

Leo: Which for convenience is good.

Steve: Well, you've basically defeated the whole purpose.

Leo: Unless, well, yeah, I understand. But if you turn on the – you haven't really because as long as you trust Windows's log-in, XP's log-in to be secure...

Steve: Right.

Leo: Because if they remove the hard drive, they're not going to be able to log in without your password.

Steve: Yeah, unfortunately...

Leo: By the way, Macs have the similar – built into the operating system a similar strong encryption called FileVault. Does the same thing as TrueCrypt does. But it also is set to, when you log in, unlock. And in that case you want to modify your firmware so that nobody can change the master password.

Steve: Right.

Leo: But a good system. We like TrueCrypt. You can even put it on a USB key.

Ben, writing anonymously from his Gmail account, asks: The question is probably really dumb. Maybe that's why he wanted to be anonymous. But I have looked everyplace for somewhere to download PGP, and every site I try leads me to the same place, PGP.com. And he doesn't want PGP Desktop 9.0, which isn't free. Can you please send me a link to a site where you can download a free PGP program?

Steve: Well, and Leo, we know what that is.

Leo: Yes.

Steve: And you talk about it, and it's what you use.

Leo: That's what I use, which is the open source version of PGP. The story is long and tangled. I mean, Phil Zimmerman originally made PGP open source. But the assets of PGP and intellectual property was sold to a corporation, which still, if you search at PGP.com long enough, offers a free version. But it's a long, hard search, and not worth it when you can just go to GNU Privacy Guard, gnupg.org, and download an open source free version of PGP. And that's what I use on the Mac and what I – everybody should just use that.

Steve: Right. Well, when I saw this question from Ben, I just went to Google and put in "GNU PGP"...

Leo: Right.

Steve: ...and took me right there.

Leo: Yeah.

Steve: So that's how anybody can find it.

Leo: They call it GNU Privacy Guard to avoid the...

Steve: Copyright, trademark.

Leo: ...probably a copyright issue with PGP.

Steve: Yeah.

Leo: But so gnupg.org will do the job. And it's great. I mean, it works on all machines. It uses PGP-style keys. In fact, I'm reusing PGP keys that I made. It's completely interoperable.

Steve: Well, and PGP is now an RFC standard. There's an RFC associated with it.

Leo: Good, yeah.

Steve: So it's been well defined.

Leo: So you don't have to go to the PGP Corporation to get PGP.

Steve: To get real PGP.

Leo: Yes, it's real PGP even without it.

Jason in Madison, Wisconsin received a nastygram from his corporate IT department. Oh, I hate those. He writes: I just received a nastygram essentially stating that I must remove Skype from my laptop because it is P2P, or peer-to-peer software, that lets viruses in. Their words, not mine. What are the security risks posed by Skype? I realize my company may have a bandwidth issue regarding the supernode feature in Skype. But is there something else Skype users should be aware of? That's a great question.

Steve: Well, it is. And I wanted to first say that, if this is a corporate laptop, and it's their property, then you want to make sure this doesn't get escalated beyond a nastygram.

Leo: Because they can do anything they want.

Steve: Yes. Certainly you could argue that this is their computer you're using, I mean, if that's the case, and that they have a right to specify what software is used on it and not and so forth.

Leo: Well, even if you bring your computer into the corporate network, they can tell you what programs you may and may not use.

Steve: Sure.

Leo: I mean, that's their right. It's their resource.

Steve: Exactly. So he brings up the issue of supernode, which we were just talking about, and about the bandwidth being consumed, and it sounds like his corporate IT department somehow discovered that he was using Skype on his laptop. All of that sounds like he does not have a firewall running on his laptop.

Leo: Because you can't be supernode if you have a router or a firewall.

Steve: Exactly. You wouldn't – supernodes are random Skype users who are not behind a NAT router or a firewall, who suddenly discover that they're being used as a relay point for other people that Skype cannot otherwise connect between.

Leo: And we should point out that that should be a small number of people, getting smaller all the time. I mean, who's on the Internet without a firewall or a router? Come on.

Steve: Well, and it is an annoyance that Skype provides no user interface option for...

Leo: You can't turn it off.

Steve: Exactly, for disallowing supernode usage of your own client. So I would tell Jason that, if he wants to continue using Skype, and I would imagine he's behind Windows XP, or he's using Windows XP, that if he turns on the Service Pack 2 firewall that ought to be built in, or uses any third-party firewall so that inbound connections are not allowed, he should be able to use Skype without any trouble, and there's no way his IT department could ever determine that that's what he was doing. But again, I'm not suggesting that he goes against the wishes of his corporate IT people. But, you know, their argument that Skype is peer-to-peer and therefore is – and for that reason should not be used, I think that doesn't hold much water. But I can certainly understand them not wanting to get into all this.

Leo: And it's their right not to. But I think they're somehow conflating it with Napster or Morpheus or, you know, one of those peer-to-peer – it's not peer-to-peer that way.

Steve: Right.

Leo: Yeah. Although I think there is some kind of tenuous connection with Kazaa, I seem to remember. But that's not the peer-to-peer, that's...

Steve: Well, in fact, it was the Skype – it was the Kazaa guys who figured out NAT traversal...

Leo: Right.

Steve: ...who then started Skype and used that NAT traversal technology from their peer-to-peer experience in order to create a system that just works behind NAT routers so easily.

Leo: But it's not sharing music. You're not allowing virus – I don't think there's any way, I mean, it's not allowing viruses in, is it? No, of course not.

Steve: No.

Leo: No. Not unless, you know, somebody sends you a file, and you accept it.

Steve: Well, in fact, at the beginning of this podcast, Leo, I sent you a PDF. And we used Skype in order to do a file transfer because it worked just beautifully.

Leo: And you can do that in chat clients and instant messenger clients and so forth.

Steve: Right.

Leo: In fact, the truth is, if they allow any instant messenger at all, it's no different from Skype. I mean, AIM, Yahoo!, all of these offer voice over the Internet in very much the same way as Skype does.

Steve: Right.

Leo: Russell of Salt Lake City asks: We all know that chat clients...

LEO & STEVE: Speak of the devil.

Leo: Chat clients like MSN, AIM, and Yahoo! Messenger are about as insecure as it gets as far as a channel of communication goes. My question is regarding Google Chat. I'm not talking about the standalone Google Chat client. But when you log into your Gmail account – this is something new they just added – using HTTPS://gmail.google.com, in that web interface you're able to conduct chats. So my question is, I'm inside the secure server, HTTPS. I'm encrypted and secure. Am I – or is it using a different port? I mean, is my chat secure?

Steve: Right. Well, this is really interesting, and I thought this was a great question. You have security guaranteed by HTTPS, that is, by an SSL connection, which we know is really good security, from your browser to Google. The problem is, everything that you are chatting with then goes through Google's servers. And who's to say that next year Google's not going to announce, hey, by the way, you can search for everything you ever chatted about.

Leo: Which they might.

Steve: Because we've been sucking it all in and keeping it forever. You know? And...

Leo: It's a possibility.

Steve: ...Google tends to do that. So, you know, with our government becoming increasingly concerned about, you know, terrorist activities within our borders and all of this NSA eavesdropping, and, you know, now all of our phone records being sucked up and analyzed, I would be concerned using any third-party server to be the relay point for my chats. There is a very good, very secure chat client called BitWise IM.

Leo: Oh, yeah, we like that.

Steve: Yes. In fact, you and I, Leo, use it because it also offers VoIP services, although the quality still wasn't up to – oh, in fact it uses the Speex codec, which is why we gave it a try. But it still wasn't up to what the GIPS, the Global IP Sound technology that Skype is using is able to give us. So we backed away from it for our podcast recordings. But BitWise IM, if you just go to Google and put in "BitWise IM" for instant messenger, I know the author, I've interacted with him, in fact I've worked with him to improve his NAT traversal technology so that it works a lot better than it did before. It really – this guy's a neat guy. It's cross-platform Mac, Linux, and Windows. And it just looked, you know, if you're looking for something that isn't a huge bulls-eye, that isn't the same sort of target that MSN and AIM and Yahoo! and Google are, where you want to do point-to-point chat and not run through a third-party server – oh, and everything is encrypted. So it's very strongly encrypted chat. It's what I would recommend.

Leo: There are corporate versions available of AIM and MSN that provide encryption, as well. But this is just, you know, you just download it, it's free, and it's easy to use.

Steve: Yup.

Leo: It's great. David from Bossier City, Louisiana discovered Security Now! two weeks ago. Welcome, David. And he just – he says he just finished listening to all 39 episodes. Whoa. Okay. He must be little exhausted right now. He had a question...

Steve: I love that.

Leo: Yeah, I think that's great. He had a question about Linux, open source, and buffer overruns. He says: Unless I'm misunderstanding something, I would think there would be more exploits from buffer overruns with Linux, since it's open source and a programmer could see all the problems in the code easier than Windows – in other words, could find these holes. Or does that make it less likely to have problems with things like buffer overruns because of the number of eyes looking at the code in the open source community? I ask because I would think this may be one of the great concepts that comes from the open source community and wondered your thoughts. This is actually a common area of discussion among open source folks.

Steve: I would say that it is the big, unresolved issue. That's why I thought this was a great question that David had posed. And, I mean, it's a religious battle in the same way that, you know, Mac versus Windows is sort of a religious thing. It's, you know...

Leo: Closed versus open source.

Steve: Exactly, closed versus open, what is more secure? You made a comment that I didn't react to last week when we were talking about buffer overruns, which was that many of these are discovered by people just pounding on the code. And in fact the guys at EI down in Aliso Viejo, they've got a roomful of machines that just throw junk at software. And when one of those machines crashes, it's because they've just found some new buffer overrun.

Leo: So they basically have a million monkeys in a room, trying to crack it. But that's closed source software.

Steve: Yeah, and the EI guys are finding stuff all the time...

Leo: Right.

Steve: ...in exactly this fashion.

Leo: Whereas you could look at a source listing of a program, if you were really an accomplished programmer or accomplished hacker, and perhaps find buffer overflows. I guess, you know, it comes down to the choice of security through obscurity, that is, you're secure because it's kind of hidden, or true security, which is that it's open, it's exposed. And, you know, people, yes, will find it; but people will also fix it right away. I bet on open source, to be honest with you, but that is a religious battle, you're right.

Steve: My sense is that, from a programmer's standpoint, if there were backdoors deliberately programmed into code, those would be found immediately...

Leo: Right.

Steve: ...in an open source environment.

Leo: Which is why PGP was open – I wouldn't trust any encryption program that wasn't open source because you need to know what they've put in there.

Steve: And so that's the advantage of open source. But the nature of buffer overruns is that it wasn't clear to the programmer who wrote it. And when you look at the source code, there's this inherent tendency to adopt that same mindset and to sort of follow along with the code. It's just so hard to see the mistakes that somebody has made, even when you're looking at it. So I don't think that open source would be more secure because people would tend to find other people's mistakes. People tend to just read along and, you know, get through the job that they're doing. It's more the case that you find these things sort of just by pounding on the code, exactly as you had said last week, Leo.

Leo: And there's certainly, I mean, the truth is there have been plenty of buffer overflow exploits in open source software. Certainly no programmer is immune from it.

Steve: That's a very good point.

Leo: And, you know, here's the question. I mean, if you write a module or a part of a major open source program, is anybody going to rewrite it or check it for security? Maybe, maybe not. I don't know.

Steve: Well, and that's – actually that's a very good point, too. It's cool that this source is open. But many people wonder if anyone else ever looks at the source code again.

Leo: Right. I mean, in theory they do. I mean, there's somebody with commit access to an open source project who's supposed to review all this stuff. But it's, you know, I mean, yeah, I think there's no guarantee it's been highly reviewed.

Steve: Well, yeah. A perfect example is my use of a news server. GRC runs a very active, wonderful set of newsgroups relating to security and privacy. Anyone who's got a newsreader, and in fact everybody who is using a contemporary browser probably has a newsreader built in, you can just go – you can aim your newsreader at news.grc.com, and we've got a news server with a bunch of fantastic newsgroups there. Well, I've got all kinds of extra features that, over the years, I've built into our news server. Like we have secure authentication, and people who post notices are able to delete theirs, but nobody else can delete theirs. And I've taken the open source standard INN news server, and thanks to it being open source, I've gone in and made a whole bunch of custom enhancements for, you know, basically turning it into a hybrid, adding a bunch of GRC stuff, which I would never have been able to do if it were just some news server I had bought in executable code fashion and then inherently been unable to modify. So it's a boon for a programmer. But it's not clear to me, from a standpoint of tons of people going over the source code, if problems are going to be found that way.

Leo: We'll win you over yet, Steve. You're starting to sound like an open source kind of guy, I have to say. You know, and it also depends on how low level the code is. I think if it's a PHP, you know, message board system, a lot more people are looking at the code than if it's an Assembly language kernel module. So...

Steve: Well, and that's a perfect example, too, of something that is very vulnerable. A PHP-based message board system is inherently, you know, it's like sticking its face out there on the 'Net...

Leo: Come and get me.

Steve: ...looking for someone to find a way to exploit it. And of course we know that there's just PHP exploits by the hundreds a month.

Leo: Right, right. Our last question. Paul Winters, nestled in the Colorado Rockies, asks: I'd love to hear your comments, Steve, on recent legislative proposals to have ISPs maintain logs of all activity. What good would this be if a user is using VPN or encryption of any kind?

Steve: He is completely correct. I mean, it is disturbing when we see legislation – I mean, and the ISPs are as disturbed as their users.

Leo: Oh, they don't want to do this.

Steve: No, it's a huge burden. The idea being that they've got to maintain logs for law enforcement to use in order to track down illicit activity. I mean, it's a good thing if it works. It's hugely burdensome for ISPs. And then you end, you know, with people feeling like their

ISPs are spying on them, much as we do now with, you know, all of our phone records going to the government for processing and analysis. And so the question was, does running all this traffic through a VPN solve the problem? And the answer is yes. There is, again, contemporary VPN technology cannot be broken. And all an ISP would have is, I mean, literally random bits of data. Because as we know from our security podcasts, once you have true security, the data that results from good encryption is indistinguishable from random noise.

Leo: Yeah.

Steve: And so that's all an ISP would be able to see.

Leo: Yeah. I mean, I have to say I'm using encryption more and more. Although, you know, in Britain the House of Commons, I guess, just proposed a bill – I'm not sure where the bill came from. The government...

Steve: To legislate against encryption?

Leo: No, to make it illegal – actually, no, I take it back. This has always been the law, but they've never enforced it, and now they're talking about enforcing it, that you must reveal your password or decrypt if requested to by law enforcement. And if you don't, you just go straight to jail.

Steve: Wow.

Leo: And I think that that's going to be a concomitant of any kind of move like this. If they're starting to log all the information, and encryption is widely available, well, you've got to pass a law saying that you have to hand over the keys.

Steve: So that's interesting. So the idea being that, if they capture a bunch of this random noise we're talking about...

Leo: They can go to the guy.

Steve: And they say to you, you must tell us...

Leo: Decrypt.

Steve: Yeah, exactly. Basically we are legally compelling you to decrypt this data so that we can see it. You have to or suffer the consequences.

Leo: And go to jail anyway.

Steve: Wow.

Leo: Whether you've done anything wrong or not. So, and I think that every – I have a feeling that every major industrial nation will pass laws like this because that's the only way to deal with encryption and crime, especially as encryption becomes more well known and commonly available.

Steve: Yeah. Well, I mean, here we've been talking about the NSA literally eavesdropping on the content of calls where they have reason to suspect that there's something bad going on. And it's like, well, okay, all you have to do is use VoIP and encrypt it because we know no one, as far as we know, not even the NSA is able to decrypt deeply encrypted, you know, state-of-the-art encryption. On the flipside, if this legislation passes, then all they do is collect the data and compel you to decrypt it for them.

Leo: That's right. Wow.

Steve: I mean, it makes sense; but, you know, it's not easy to feel so smug now about encryption.

Leo: Yeah. In fact, there's even some move that the U.K. may actually, ahead of time, ask businesses to hand over the cryptographic keys, just in case. Just in case.

Steve: Hmm.

Leo: I tell you. I don't want to get you upset. I know how you feel about these things.

Steve: Where can we move to, Leo?

Leo: Nowhere, that's the problem. Now, that's why – one of the things we like about TrueCrypt is its plausible deniability. It doesn't even look like encrypted data. It just looks like noise.

Steve: Yup.

Leo: But if you've got noise – I guess you could say, if you were on your Internet service provider and sending something that sounded like noise, you could say, oh, what are you talking about, that must have just been a burst of noise.

Steve: Prove that it's not.

Leo: Encryption? What encryption?

Steve: Yeah. Yeah, exactly. I mean, and in fact it is not possible to prove that encrypted data is not random noise.

Leo: Right. So maybe that's our defense.

Steve: There we go. We solved the problem.

Leo: I hope. Good luck. Do we have a plan for next week, or are we going to wing it?

Steve: Well, if you're not going to make me prove that a stack is much better...

Leo: I guarantee you, I'm going to hear from Randal Schwartz, and he's going to explain why there's no other way to do it. But anyway, we'll see. Maybe, you know, he could very well send me an email saying, you're right, why didn't I think of that? But other than that, do you have any other ideas?

Steve: Next week is TrueCrypt.

Leo: Oh, we'll do TrueCrypt. Great.

Steve: Yup.

Leo: Can't wait. That'll be a lot of fun. We do thank you for joining us. Thanking our friends, of course, at Astaro Corporation for making this possible. We couldn't do it without them, and they've just been so generous, our great sponsor. If you are interested in security, you should know the name Astaro. If you're interested in open source, you should know the name Astaro. They make great open source security solutions like the Astaro Security Gateway for your business, or even, if you're a home user, for you, absolutely free. You download it and put it on a PC, and you've got all sorts of great security. You can pay a little bit more and even get antispam and antivirus and spyware filtering, too.

Steve: Well, and thanks to the fact that it is using, you know, lean and mean open source OS and solutions, you can take that old cranky machine from the garage that, you know, will no longer play games anymore, and turn it into a really nice Internet firewall appliance.

Leo: Download your free copy today at Astaro.com. And of course it's not exactly a sponsor, but we do give a lot of credit to SpinRite because that is Steve's day job. And without it, well, we'd have to find a way to pay him or something.

Steve: It keeps the lights on, and it helps people all the time.

Leo: Yeah, it's wonderful. If you want to read some testimonials for how people have used SpinRite to recover their data, to save their drives, to save their lives, go to SpinRite.info. Go to GRC.com/securitynow.htm. I guess you can go to SecurityNow.info for a while, anyway, to get the show notes, 16KB versions of the show for the bandwidth impaired, and full transcripts, thanks to Elaine, of each and every show. And of course, Steve's notes about the topics we talk about. It's also a good place to read the security newsgroups that he has going there at GRC.com, and even submit your questions for future editions of Security Now!. GRC.com, that’s the home of SpinRite and Security Now!. Thanks to our friends at AOL Radio for providing the bandwidth for this podcast, AOL.com/podcasting.