SECURITY NOW!

Transcript of Episode #236

Listener Feedback #86

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-236.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-236-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 236 for February 18, 2010: Q&A #86.

It's time for Security Now!, the show that covers all things secure and insecure, mostly insecure and how to make them secure, your privacy and all that. Steve Gibson is here. He is the guru at GRC.com, the Gibson Research Corporation, the author of SpinRite, the world's best hard drive maintenance and recovery utility and many great free security utilities. And this is Episode 236, so that makes it our fifth year of security.

Steve Gibson: Into our fifth year.

Leo: Wow.

Steve: Yeah, and no sign of running out of stuff to talk about. We've got - it wasn't a super active week, but we've got the usual suspects lined up, a little bit of errata, and some great questions from our listeners.

Leo: It's a Q&A.

Steve: Yup.

Leo: Well, let's see, here. Let me just look - I guess let's start with the errata and the news, and then we can come to the Q&A.

Steve: Yeah. Last week we talked about Microsoft's second Tuesday of the month standard February, in this case, security update. A disturbing number of people began to get the notorious and infamous Blue Screen of Death, as it's called, the kernel crash essentially, after installing Microsoft's monthly set of patches. It was a big one in February, unlike January that was pretty modest. And further analysis revealed that one of the patches in particular, MS10-015, which fixed a problem that we had spoken of a few weeks before, which was this local privilege elevation vulnerability, where someone who had local logon privileges would essentially leverage some old code in Windows, the NT Virtual Machine Manager that does the DOS box stuff, in order to elevate themselves to an admin user with full system privileges. So Microsoft fixed that, but for some users the result was a Blue Screen of Death that Microsoft couldn't figure out.

Well, Symantec figured out what was going on. It turns out that there was an interaction between that fix, which moved things around in a couple of the kernel system modules, an interaction between that and a very bad rootkit trojan backdoor. So this thing was detecting - don't you just hate it when the rootkit trojan backdoors...

Leo: Interacts with your software?

Steve: ...get in the way of your security updates?

Leo: That's really interesting. But, you know, I remember when Service Pack 2 came out. And, you know, some people had no problems; some people had problems. And I really always just kind of suspected that the real problem was that, if a system wasn't clean, that you were going to have more problems with these updates.

Steve: Well, I mean, the whole concept of this incremental moving modules forward, with all kinds of device drivers and other things, I mean, it may very well be that there are other problems in addition to this particular rootkit that is Tidserv, which opens a backdoor, opens ports essentially on your machine, that allows for remote control of your computer; uses rootkit technology to hide itself, so it's not easily detectable by software. And it had hardcoded the offsets of specific code in the kernel which changed under this particular patch, which caused it then to make your machine crash. So it was sort of a backhanded rootkit detector that was triggered by this update. Microsoft has suspended offering the update as of last weekend. They said, okay, we're going to take this out of our update batch because we don't want to be BSODing people's machines, even though it looks like it's a rootkit detector. So I thought that was a little odd bit of news.

Once again we have a critical, out-of-cycle patch from Adobe. It turns out that it's been found that Flash, running in so many browsers, can be used to spy on people who have Flash.

Leo: Wow.

Steve: You could go to a malicious web page that would bring some Flash script to your browser, run in the Adobe Flash interpreter, which is essentially what it is. And that malicious code has the ability to look at the pages and other web pages and web browser windows that you may have open, read them and then, like, get usernames and passwords and banking data, anything it wants to, scrape the other windows that you may have open in your web browser, and send those somewhere. So, and this is as of the 15th of February. So our listeners ought to go - you can just go to get.adobe.com/flashplayer in order to make sure you're running the most recent current version of Flash. And Adobe believed once they were going to do quarterly updates. And they're not even surviving monthly updates at this point because they've got so many problems.

One other little bit of news caught my eye, which is that on March 1st a data protection law, which is the first of its kind in the country, is going into effect in Massachusetts. It was originally scheduled to take effect on January 1st of '09, but there was a huge amount of pushback from Massachusetts-based businesses that thought that this thing was a little too strong and overreaching. It's being heralded by the security community as a really good thing, something that they hope more states will enact, and maybe we'll even get something at the federal level at some point.

Basically, it says that businesses doing - businesses located anywhere doing business with customers in Massachusetts must encrypt any of the business data, any personal private data of Massachusetts residents on portable storage devices. And it's like, well, yeah. Sounds like a really no-brainer good thing to do. And also specifies that any data that is being transacted must be encrypted in flight. So again, it's like, yeah, use SSL to interact with these customers. So the law, which was going to happen on January 1st of '09, has been modified a couple times. Now it's moving forward and takes effect on March 1st.

So anyway, that's just a good thing. There was some language where companies doing business with Massachusetts residents had to make representations about what third parties, that is, like the companies' third-party affiliates might be doing with any data that was being shared. And that's where most of the problem with this was coming from because these companies were saying, wait a minute, we can't really make affirmative representations about what our partner companies might be doing. And it's like, well, why not? Are you turning this data over and not safeguarding it? So they were concerned there. And so that actually got softened somewhat so that they can - it's like a best effort verbiage now instead of making stronger assertions.

And then in my favorite - this is, like, too bizarre to be true, but it is. There's a very prevalent fake AV, antivirus software, known as Live PC Care, which for some time has been bothering users, popping up on their screens, telling them that they're going to run an antivirus scan and so forth.

Leo: Yep. I got a call this weekend on it.

Steve: Well, the company has gone one step further.

Leo: Oh, boy.

Steve: They've added live, real-time support chat and...

[Laughter]

Leo: C'mon, chat with us. Let's talk.

Steve: And it's legitimate. I mean, it's legitimately real. There's a yellow button on the screen that comes up, and "If you'd like to chat interactively with one of our technicians, please press this button." And it uses a free live chat system called LiveZilla, pops up a window. And from the analysis that's been done, it appears that there are real people at the other end of this, answering people's questions about...

Leo: Sure. They pay for a call center.

Steve: ...about a completely fake AV scan and basically convincing people to pay their money in order to get software that does nothing.

Leo: And for all we know, those people don't even know that they're part of a scam.

Steve: That's a good point. Hadn't occurred to me. They may not be aware of it.

Leo: Yeah.

Steve: Unbelievable. So last little bit of - I have two pieces of errata. We also talked last week about two malicious plug-ins that had somehow crept into Mozilla's database of Firefox plug-ins. And I remember you asked me, there was one from a reputable company that I actually like, the Sothink company, that do the Flash decompiler. They had something called the Web Video Downloader. And neither of us could really figure out, well, if it was malicious, how did it - at what point did it creep in? Turns out it never did. It was a false positive on the virus scan.

Leo: Oh. Oh, good. That's a relief.

Steve: So I just wanted to clear - I wanted to clear that up, yes. They were using a compressor some time ago called an "armadillo" compressor was the name of the compression technology. And because some malware uses the same compressor, there was some AV software which was false-positiving on that. And I've had that happen with my own code. That can happen from time to time. And Leo, do you have a web browser - I know this is a stupid question. Do you have a web browser in front of you?

Leo: Yeah.

Steve: Google "Best DNS Benchmark."

Leo: Best DNS Benchmark.

Steve: Best DNS Benchmark. The first link...

Leo: Googling it right now. Yes, "Probably the best consumer DNS Benchmark tool in the world," Steve Gibson.

Steve: Now...

Leo: Unfortunately, the lead is "Steve Gibson is an old man." Well, isn't that nice, you little punk, Michael Tan. Screw you, Michael Tan.

Steve: I've got to get - and later on he called me a kook, but in a good way.

Leo: That's funny. "Even when he was young, in 1989, I used his software." When he was young.

Steve: Isn't that great? I just love that...

Leo: That's only 20 years ago. C'mon.

Steve: Steve Gibson is an old man.

Leo: Oh, I can't wait till Michael Tan is over 20. Then he'll...

Steve: Actually it's a really nice little article. I was - I'm working again on getting the documentation finished for the benchmark. And I took a break Monday around noon, I thought, I wonder what comes up? Because I'd noticed how many...

Leo: That's so funny.

Steve: ...are being downloaded. And...

Leo: Oh, it went up fast; right?

Steve: Oh, yeah. I mean, tens of thousands of copies are out and being used and recommended. So I just - I Googled "Best DNS Benchmark," wondering what Google would have to say. And there was that first link. And I was caught off guard by "Steve Gibson is an old man."

Leo: Steve, by the way, he says that - I think the highest praise, despite this old man crap, he says, "He's not really a programmer, he's a craftsman."

Steve: Yeah.

Leo: And I think that that's apt.

Steve: He understands me. I have to say I really - I appreciated what he had to say, so.

Leo: Yeah.

Steve: And then I got an interesting piece of feedback on SpinRite that also refers to several of the show sponsors from Derek, who wrote, said "SpinRite and TWiT.tv Saves My Mum." It said, "More than just a SpinRite story, this one gives thanks to a few of the TWiT.tv sponsors. My mum is now 70 and lives some distance away from me, a full-day trip. I went to see my mum at Christmas, and almost as soon as I walked in she said, 'The computer won't boot. It's doing a blue screen thing.' And yes, it was a 'Windows can't boot, kernel DLL is missing or damaged' error.

"My first move, of course, was SpinRite. And as expected, it had things working again within an hour. I then discussed replacing the hard drive. Mum decided to leave it until it died and then replace it. At this point I installed Carbonite. After all, we are now waiting for the hard drive to fail. I went home. The computer was backed up and running well.

"And then, about three weeks later, the drive died. That resulted in another overnight trip. BIOS would only see a 0MB disk, and the drive would not spin up at all. RIP. I replaced the drive and reinstalled Windows, then did a restore of data using Carbonite. A few days after I got home, Mum had questions about missing applications such as Skype," because I guess somehow that hadn't been restored. "So I used GoToAssist.com and, using the remote tools, I installed the webcam drivers and Skype, which she was missing.

"I now remote into Mum's computer as needed and help with any questions, make sure updates have been done, and such like. A big thanks to SpinRite, which fixed the drive the first time; Carbonite, easy backup even my Mum can use; GoToAssist, remote access without needing NAT settings changed. And, last of all, thanks to TWiT for the best podcasts on the Internet."

Leo: Aw. Isn't that sweet.

Steve: "Keep up the good work."

Leo: Oh, that's nice.

Steve: Thank you, Derek, and I'm glad we could help with your Mum.

Leo: Yes. Question 1 from Sean McLeary in Brampton, Ontario, Canada. He worries that packet loss could be a threat to security: Steve, I've been your Security Now! listener since Episode 60. Of course I had to listen to the rest from before, too. So now he's a hundred percent. I do have a question that may interest you. Are packet losses all right when your router has a secure connection and is using WPA or WPA2 on its WiFi? Or are they bad? Can they give an intruder access to your network without knowing? Sometimes when I do ping tests I see lots of lost packets. What does that mean? Should I be worried? Where are those packets going? Can they use those against me?

Steve: Well, there have been some exploits historically which essentially used interference in WiFi transactions to induce either the client or the base station to generate packets at a much greater rate than they normally otherwise would in order to acquire a much larger set of samples to be used for cracking. Now, the good news is WPA and WPA2 are not susceptible to that kind of attack. The older WEP encryption was.

And so one of the things there was - originally people said, oh, well, we know there are some weaknesses in WEP encryption; but it would take hours, weeks, years - well, not hours, but, I mean, like much longer time in order for enough data to be collected. Of course the bad guys are clever, and they figured out how to induce endpoints to generate a much higher rate of the kind of packets they needed in order to analyze to crack WEP encryption, ultimately bringing the entire time to crack it down to about a minute. So in the case of wireless, that's not a problem.

Now, when you do ping tests, though, if for example you're pinging Google.com or Microsoft.com or something, what you're seeing is something which is a natural function of the way the 'Net works. The concept is, and we've talked about this in different contexts pretty much for four and a half years of the podcast, is that when we - we think in terms of connections, in terms of connecting to something and exchanging data: connecting to a server, getting a web page; connecting to an email server and retrieving mail. This is a so-called "virtual" connection, meaning that it's sort of an agreement between the two endpoints that they are connected. And what that means is that individual packets are sent in little bursts back and forth. And it's done so quickly that we sort of see it as a continuous flow of information.

However, the packets themselves are freestanding. They stand alone, each one aimed at its destination. They may get there. They may get there out of sequence, or they may get lost along the way. Routers have a sort of a best-effort approach. They try to send packets from one input to one output. But you could have a lot of inputs all feeding to one output and receiving more packets in total coming in from different sources than you're able to squeeze into the output. And there are buffers in routers to sort of allow them to briefly store up packets, hoping that the output will be able to accommodate the packets that are being routed.

But it is the case that routers have by design the right, essentially, to just drop the packets that they're not able to forward onto the next hop across the Internet. So packet loss normally just represents some congestion, some point on the Internet between where the packet originates and where it's going, where there's just some brief inability for all the bandwidth to fit through a certain point of congestion. So it's really not something to worry about. And as long as you're using WPA or WPA2 encryption, there's certainly no concern about what you're seeing in terms of connectivity troubles and security. With WEP that could be a problem. With WPA it's not.

Leo: Good to know. Dropping the packets. Tom in zipcode 33441 - I don't even know where that is, but I'm sure I could Google it - wonders about defeating keyloggers on a public computer. He writes: Steve, is there any way to reliably thwart hardware and software keyloggers and screen scrapers on a public computer? The only solutions I've discovered are the following: IronKey with its built-in password manager; a combination of LastPass with YubiKey. Are there any other ways to do it? Thanks. Love the show. Happy SpinRite owner Tom from 33441. Each day I eagerly hope that Steve will start working full-time on CryptoLink. Each day my hopes are dashed. When? When? Sigh.

Steve: Okay, well, first, it would be nice to be able to say that there's a way to reliably defeat keyloggers on public computers. And I've got to say that there isn't. I can't think of probably anything more frightening than using a public computer, that is, like a computer in a library or in an Internet caf that is being used by lots of people, I can't think of anything more frightening than to use such a machine for critical, sensitive work. They're just - I don't think there's any way you could argue that anything you could do as you approach that machine could make it safe. There could be a keystroke logger in the keyboard. Now, okay, so we've used cut-and-paste in order to avoid typing the credentials in, but the browser's still going to get it.

There's all kinds of ways of intercepting even SSL data at the machine. You can't intercept it once it leaves the machine. But there's many layers in the operating system where the data is available before it gets down to the TCP/IP stack and actually leaves. So there are some programs which allow developers to see the contents of SSL connections outside of the browser. I've got one which I use when I'm developing stuff over SSL because it's inconvenient not to be able to see those SSL connections. So in the same way that good software could do that, so could malicious software. So I would like to completely disabuse anyone from the idea that using a public computer can be safe.

Leo: I can't. Just can't.

Steve: You can't. Do anything you can to avoid using such a machine. The best thing to do is bring your own along because, I mean, we know that end-users may even have problems with their own machine. But at least you've got some sense for the history of it. If you're the only one using it, you know how safe you've been, you know what security provisions you've got. If you could bring your own and plug it into a public location, that's far safer than walking up to some screen and keyboard about which you know nothing. I just - there just isn't anything you can do that could make that safe.

Leo: That's really good to know. I mean, I think to make that unequivocal statement is exactly right.

Steve: There's nothing you can do to make it safe.

Leo: There's no point.

Steve: Yeah, just rearrange your life so you don't have to do that.

Leo: Right. Which makes me sad for people who don't own computers, and many don't. Because what are you going to do?

Steve: Well, I would say that I understand. And certainly library - banks of accessible machines that are affordable in a library makes great sense for surfing the web and browsing and doing things. But don't do your banking on that machine. There's where you've got to go up to the teller window, if you don't have a way to do it online in a safe way. Just you can't use a public machine safely.

Leo: Ray Herrera in Oakland, California - did you answer the CryptoLink question?

Steve: Oh, I forgot the CryptoLink. No, I didn't. Thank you. It is my pending big project. I'm back working on the DNS Benchmark documentation. That gets followed by the DNS Spoofability documentation, which then is followed by the GRC Third-Party Cookie documentation. Then I've got to do a little work on Security Now! because that Security Now! page at GRC is getting ridiculously long. All 236 episodes are on one page, so I need to spend some time on that. Then I work on CryptoLink. So I'm just doing a bunch of housekeeping. And I'm salivating myself at the thought of getting onto a really nice, meaty development project. I can't wait. So it's definitely going to happen.

Leo: One thing you can count on. Steve loves writing software.

Steve: Oh.

Leo: It's his highest art. I mean, when you're an artist, you do everything else as a means to doing the one thing you care about most, which is your art.

Steve: How I got to be an old man, Leo.

Leo: Yeah [laughing]. A kooky old man, I think.

Steve: A kooky old man.

Leo: Ray Herrera, Oakland, California, with the question: "Do you still use Jungle Disk?" Steve, I've been putting off going with Jungle Disk since I first listened to Security Now! Episode 123. Then, after a recent power supply failure, I've been looking to actually start using Jungle Disk or something similar like Mozy or Carbonite. My question is, do you still use Jungle Disk today, even after they were bought out by Rackspace and have moved to a subscription-based model? Sincerely, Ray Herrera.

Steve: Well, the answer is yes. But I'm also very glad that I got onboard in the beginning because I did recently notice, I think they went to Jungle Disk v3, and I saw that while they are grandfathering all of their original customers in as free forever, and I really appreciate that, no new customers get that deal. So any of our listeners who did jump on Jungle Disk and signed up for it, well, basically got it for free back when they did - or I guess maybe it was 1995. I think there was a one-time purchase, and then you had it forever. And now apparently it's a subscription model. I did notice that they had changed that, and I was sorry to see that happen.

I should say, though, that I still love it. I mean, I love it because I understand it, because it is simple, and because I absolutely know that it is performing very strong encryption at my machine before that data leaves. I receive a monthly bill from Amazon for 14 cents, I think is about what it costs me. And just, in fact, it was Monday when I was working on the DNS Benchmark, or no, it was yesterday, Tuesday, because we're recording this on Wednesday, I just - my finger slipped, and I deleted a file from my Benchmark directory. And I had it in several other places. But I thought, oh, I'm sure that's up on Amazon. And sure enough, I opened my "J" drive, "J" for Jungle Disk, and grabbed the file and dragged it back into the DNS Benchmark directory. It was a file of little colored icons that I was using for the documentation on the web page. And so I use it all the time. I still think it's a great solution.

Now, the good news is there are other interfaces to the Amazon S3 offering. And I haven't looked at any of the others, so I can't vouch for them. I really like Jungle Disk because I understand the way it works. And I guess they made a business decision about selling off to Rackspace and going to a subscription model, which is - I think it's unfortunate, but if that works for you, I can vouch for the fact that the technology is solid.

Leo: Yeah, I still - I'm also grandfathered, and I still do use it. You know, the thing about Carbonite, Mozy less so - I have to say, if you want to - try Mozy first, and try restoring from Mozy before you commit to Mozy. The software is not elegant. But the thing about Carbonite, and this is true of Mozy, too, is that they are very easy to use, very easy to set up. And you know what it's going to cost. There's no surprises. Jungle Disk isn't expensive. But if you're backing up gigabytes, it rapidly can become expensive because you pay for both bandwidth and storage at S3.

Steve: Correct.

Leo: So it's cheap for you and me because we're not backing up that much.

Steve: Right, I'm just using it, literally, to, like, store data files.

Leo: Current, like current documents and that kind of thing.

Steve: Right.

Leo: And I think it's good for that. It also requires a much more sophisticated user. On the other hand, if you want more control, I think it's a very good choice for somebody who wants more control. I haven't seen what the new Jungle Disk is like.

Do they require you use Rackspace instead of S3?

Steve: No. You have a choice between the two. It still uses S3, so you can choose between the two. And they go through - they have a nice page where they show the differing economic models. And I remember that they were different, and that I was still happy that I was at Amazon, although I don't remember the exact details. So anyone who's curious can go take a look at the Jungle Disk site. And it looks very nice. It's getting additional features. I'm not Mr. Big Feature King, so I just - I want - I'd rather have things that are feature spare and just work in a lightweight, robust fashion. And at least that's how I'm using it. So I'm happy with it.

Leo: Yeah. I use them both, to be honest. It's nice to put Carbonite on the - and they have a new Carbonite Pro product which is for small business. That's what we're going to use here because it's multiple machines, but one dashboard. And what I really liked using Jungle Disk for, and this is a good solution, if you have a NAS that is backing up everything, which our NASes are supposed to be doing - I no longer do the IT, so I don't know what it's currently doing. But if the NAS is backing up everything, and then that NAS is running Jungle Disk, which you can do, and backs it up to S3, then that's kind of the best of both worlds. You have a completely automated system that you shouldn't have to maintain or pay attention to. And I think you have fairly high certainty that you've got good backups locally and offsite. And I, you know, I'm so paranoid, like you I'm sure, I'm always backing up locally on a USB key. And I have all of my documents are on all of my machines.

Steve: I think that the users who have been at computers long enough have taken enough hits from working all day and then, like with not saving a document, and then having the computer lock up on them, it's like, oh, god, you know, that we learn our lesson. And I'm hitting Ctrl-S all the time. And I don't believe that my laptop is ever going to boot again when I shut it down. So any of the work that I've done I'll drop it over onto the "J" drive on my laptop, and off it goes through Jungle Disk to Amazon. Or I have got a little encrypted USB drive on my keychain, and I'll make copies there. I mean, stuff that I really care about I have in multiple locations because you just have to.

Leo: Mm-hmm. Moving on - I think that's a good discussion. We should talk about these things more because I think that really simple stuff that you and I just assume, well, everybody knows is also the stuff that's sometimes the most useful for our audience.

Steve: Yeah.

Leo: You back up; right? You know how to back up. I don't have to tell you that. Patrick McAuley in Guelph, Ontario, Canada wonders about a cool new feature of the Opera web browser called Operate Unite: Steve, I was wondering if you could comment on something I just read on Lifehacker. Their How-To Geek - which is, by the way, a great column - wrote a piece about the best way to share large files with a few friends. It involves using a feature of the Opera web browser called Opera Unite that apparently sets up a web server inside your browser. Oh, dear.

Steve: I know. Just shoot me now.

Leo: I don't have to read much more. But okay, let's keep going. It enables standard HTTP protocol to allow your friends - it's putting a web server on your system.

Steve: I know.

Leo: Using not just Opera, but any browser to access files directly on your computer. What could be wrong with that? They explain that Unite - you know, I played with Unite, and I - anyway, they explain that Unite automatically hooks into your router using UPnP...

Steve: It just keeps getting better.

Leo: ...to automatically open port 8840, but it can also use the Unite proxy server to dynamically - when you're behind a more restrictive firewall, though it will obviously be slower. They say the files are password protected. But I wondered what you thought of this from a security perspective. And then he gives the URL. If you Google "Lifehacker" and "Opera Unite" I'm sure you will find it. What do you think? What do you think?

Steve: [Groaning] Well...

Leo: [Groan] is right.

Steve: Yeah. I mean, if you want to give access to your machine, I think just remove your router and turn off your personal firewall. It'll be fine...

Leo: There you go. It makes it easy.

Steve: It'll be fine for about a minute. No, this is just the worst idea I've ever heard of. I mean, with all due respect to the Lifehacker blog and those guys, I mean, this just sounds like trouble waiting to happen. There will be, predictably, if this were to take off, scanners looking for connections on 8840, which is the default port, which people won't change. And people will say, oh, well, you know, I don't want Aunt Mary to have to use a password. Let's just - I'll just leave that blank because that'll be easier for her.

Leo: [Laughing]

Steve: Apparently a part of Opera Unite is it ties in with their own dynamic DNS server so that you're able to give your instance of a web browser a nice, memorable URL name. So this just - I loved the posting because this is, I mean, just all the way through it we learn about things that we've spent the last four and a half years warning our listeners about. For example, the fact that this thing uses Universal Plug & Play to connect to the router and open a port without you having to do anything so that unsolicited traffic is able to come in through port 8840 and get to your machine. We hope like crazy that there's no buffer overflows, overruns, mistakes of any sort in the web browser, I mean, sorry, the web server that Opera Unite is running in your machine because, if there is, it's game over. Anyway, this is just unfortunately not the way I would recommend people share files.

Leo: Yeah.

Steve: I did notice on the Lifehacker page, someone had posted in the comments that Dropbox was a workable alternative. And...

Leo: We've used both Dropbox and Pogoplug. What do you think of those solutions?

Steve: I think they're great. And, I mean, we're using Pogoplug now for you to get the audio to me, and it seems like a very nice solution. I like the idea of someone doing this whose entire business model is about it, rather than adding it as a button to an existing browser. I don't know, that just scares me. It takes a lot of responsibility to run a web server. It's not an easy thing to do, as we see with people's websites being hacked all over the place. And this thing has all kinds of other features and add-ons and plug-ins. And I just shake my head and think, well, nothing could make me do it. So I would caution our users about it, too.

Leo: Question 5 from New Zealand. Colin Perry wonders about reverse-engineering assembly language. This kind of, I guess, ties to what we talked about in the last couple of episodes, the machine language story.

Steve: Yeah.

Leo: Hi, Steve and Leo. In your opinion, would it be easier to reverse-engineer or hack a program written in assembly language, the machine code, the one-to-one correspondence code that we've been talking about, as opposed to a program written in a higher level language like, say, C, or a scripting language like Perl? I ask this as the resulting machine code is directly related to what the author was thinking, as compared to a compiler's interpretation of what the author was thinking. That's a good point. When you look at source, when you disassemble - well, first of all, what is disassembly?

Steve: Well, so I thought this was a great question. And we've had a ton of really great feedback from our "how a computer works" episodes. Our listeners are really enjoying them. So this of course stems from that. When you write in assembly language, exactly as Colin suggests, you're putting down - and as we talked about last week - you're putting down the individual step-by-step instructions that you want the computer to execute. And so when you analyze the result, the actual EXE, the executable code, it is exactly what the author wrote.

In fact, it's funny, there have been discussions in our newsgroups about people saying, hey, Steve, why don't you do stuff in open source? And some people who have replied before I saw the posting or had a chance to reply have said, well, Steve writes everything in assembly language. So if you just disassemble it, that is the source. It's the same as what Steve wrote. Of course, it's absent meaningful labels and comments and things, and my source code is extremely legible compared to what you get when you disassemble something.

But what a compiler does, a so-called higher level language is you're writing in sort of an abstract language, a language that is not directly related to the way the computer underneath performs the work. The beauty of that is that you can have the same code that you write in a high-level language will run on entirely different types of computers with different architectures internally and different machine languages, where the compiler compiles that universal, like a language like C, for example, down into the specific machine language for different machines. The result of that, though, is if you look at the code which a compiler produces, you can immediately tell, I can immediately tell as someone who writes this kind of code, that no human did this. This was...

Leo: Yeah. Compilers do all sorts of weird optimizations and weird stuff that...

Steve: Well, yeah. And in fact one of the things that keeps me coming back to or staying at assembly language is there's this argument about, oh, modern optimizing compilers are so good now that there's really no difference between the code they produce and what people write. It's like, oh, give me a break. I've looked at this stuff, and it is just godawful. It's just horrendous.

So because of that, and as Colin's question aims at, it is much - it's truly much more difficult to understand what a compiler's code is trying to do than assembly language which directly translates into machine language to do the same job. I would say it's like maybe at least twice as large, maybe three times as large. Meaning that there's instructions that you see that just sort of are just - look like spaghetti. It's just not at all clear what's going on. So it's definitely the case that reverse-engineering assembly-written code I think is dramatically easier than automated produced code. And for one thing, it's, like, much smaller. We know how small my applications are compared to things other people write. So if you've going to reverse-engineer something that's a megabyte, you've just got a lot more physical work to do than reverse-engineering something that's a few tens of K bytes.

Leo: I've talked to hackers about that. And most of them say, especially those who do a lot, you know, if you do a lot of disassembling, you're doing mostly high-level languages. Nobody writes in assembler anymore. And they say it's just as, you know, they're so used to it, they can immediately recognize patterns. It's all about pattern recognition.

Steve: Yup.

Leo: So they immediately recognize patterns. Oh, yeah, that's the loop. Oh, yeah, I see what they're, you know. And so they know, especially as they learn the different compilers, oh, I recognize that bunch of code.

Steve: Sure.

Leo: And really that's what you're doing, isn't it. When you look at code, disassembled code, you're looking at patterns. The thing that you're lacking is the symbol table. So without the names of the variables, you really have to kind of - it's very interest- it's fun to do. It's a great puzzle.

Steve: It really is, yes.

Leo: And if you want to learn to program or understand how computers work, it's a good exercise once you become more advanced. But of course, now, I've never looked at things like Smalltalk and these interpreted programs and scripting languages. That must be very difficult, to look at the tokens. I don't know if you could do that.

Steve: Yeah. Well, you can. I've done so. And it just requires sort of taking a deep breath and not being in a hurry and making lots of notes. And after a while you begin to see patterns. And, I mean, again, it's like a big pattern-recognition test. And it all makes sense. It's just a matter of giving it time and sitting there and working out what's going on. It's really fun.

Leo: Yeah. It is. I'll tell you, there's - maybe it's a certain - if you have a certain mind. But there's something really fun about computers and learning how they work, learning how to program them, learning how to take it apart. It's just - it's like untangling a ball of string. Some people, it will make them insane, and they go [sound]. And some people just go, ooh, this is fascinating. I could do this all day. And you know which one you are.

Let's see here. This is - we're going to go to Swansea, Wales, in the UK. Phil Coleman asks: Is there really such a thing as a private search engine? Steve, I've come across a search engine called Start Page. The European name is IXquick. And it says we don't store the user's IP address; we don't install cookies. We destroy search data after 48 hours. It allows one to search via secure proxy. You could go there right now: eu.startpage.com. In fact, you can use HTTPS with it. It has won the European Union's Privacy Seal. The EU is certainly very concerned about privacy. Can there really be anything this good? They have to make their money somehow to pay for their infrastructure. I've asked them what their business model is and will forward their reply if I ever get one. While you recommend Google for their filtering of malware, the privacy conscious among us are uneasy at how much of the data we generate while searching is warehoused and mined and sold. I love the show. It's essential listening. I'm a network administrator for an interactive museum.

Steve: Well, I think this if course is a great issue and concern. My feeling is that a company like Start Page could run a very similar business model to Google without storing data over the long term. As I understand it, what Google, the brilliant thing that Google did relative to search-based advertising is they just recognized that, if they could aggregate a bunch of advertisers who covered a large area of different aspects of the Internet, that when someone entered a phrase searching for something, they could return all of their results, but then also, in the case of Google, down the right-hand side show a bunch of search context relevant ads. Well, that doesn't require any kind of state or IP address or cookies. You're just - that's a standalone operation. You search for this, Google gives you your results and provides you with some commercial links that may be of interest to you.

So I could easily see where the Start Page group that are big on highlighting the fact that they're taking privacy so strongly could have an offering that works in very much the same way and not be retaining data. Certainly there's the opportunity, given the kind of databases that a search company could build and then monetize somehow, to do more than that. But it's not the case, I think, that a company needs to. And we continue to hope that Google will continue honoring their motto of doing no evil.

Leo: Yeah, do the right thing. The problem with any company like this is you could trust Google now, but they're collecting all this data. And who knows what the next management team will want to do.

Steve: Yup.

Leo: That's the thing that concerns me. It's the same thing with the government. You might say, well, the government's been benign, because we can trust the U.S. government. But you give them that power, and then what if somebody comes along that decides to abuse it?

Steve: Yeah.

Leo: So you've got to keep controls on these companies, even if you trust them right now.

Steve: Yup.

Leo: Let's see, here. Dan White in Winchester, VA wonders about incrementing the program counter. Oh, good. We got some programming stuff here. Just listened to the last episode on machine language, thoroughly enjoyed it, he says. It brings back memories of when I programmed in Z80 machine language for a computer my dad and I built based on the S-100 bus, if you remember that.

Steve: Oh, yeah.

Leo: Oh, yeah. I'm looking forward to your discussion of indirection next week and wherever you go after that. My question relates to - I want you to do recursion, too.

Steve: Yeah, we're going to.

Leo: Oh, good, okay. That's the one I'm still wrapping my head around.

Steve: Yup, because we need to talk about stacks.

Leo: Right.

Steve: And so the plan is to talk about adding various types of complexity onto the very simple model that we built last week.

Leo: Yeah. My question relates to something you just glossed over in your jump from the previous discussion of simple gates and flip-flops - which was excellent, by the way - to this discussion of machine language. You spoke of the program counter to allow the program to step through instructions. But doesn't that require more than just simple gates? Seems like it would involve an adding function, a timer, and a looping mechanism to continually add one to the counter. But that seems to require more complex functions of a program which depend on the program counter. So would you then need a program to create a program? How do you get this chicken-and-egg thing started? Is the program counter all done in hardware? Did I miss something in your previous discussion, or is this something you plan to address in future episodes? Thanks for Security Now! and for SpinRite. No great stories, just the normal maintenance of my drives. Dan.

Steve: Well, I thought that was a neat question. I did sort of just talk about the program counter incrementing and never talked about how that's done. It's not worth a whole episode, so I thought this made a great question for a Q&A. Counting in binary is a process that is interesting and sort of fun to work out on the back of a napkin, or on the front of a napkin, for that matter.

It turns out that a binary counter has a very simple logic to it. If you have a string of bits, say individual bit cells, and say that it's initially all set to zero, well, to turn it to a one we invert the lowest order bit. And so now we've got all zeroes and then a one. To increment again, we invert that first, the rightmost bit again. But when we invert the bit, and the bit goes from a one to a zero, we invert the next bit to the left. And what's really cool is that simple logic. You just - you invert a bit. And when you invert it, and it goes from one to zero, you invert the bit to the left that is the most, the next most significant bit. If you apply that, that counts. So you start off with all zeroes.

Leo: "Counts" in the sense of counts one, two, three, four; not "counts" as in is significant. It's counting.

Steve: Yes.

Leo: It's adding, yeah.

Steve: It's essentially, it's adding one every time. So we start off with all zeroes. We invert the least significant bit. That goes to a one. And then we invert it again. Well, that bit goes to zero, which kicks the next one, causes it to invert. So that goes - now you have a one zero, which is the number two in binary. Now we invert the least significant bit again, so now we have a one one. Now, when we do it again - and a one one is a three - now we invert the least significant bit. So that one goes to zero, which kicks the next one over. It's a one. It goes to zero. Which kicks the next one over, forming a one. So now you have one zero zero, which is binary four.

So the logic in our machine, this program counter is a register of flip-flops that I talked about before. And there's some logic you can put on a flip-flop such that you're able to cause it to toggle. It flips. If it's on, it flips off. If it's off, it flips on. And so just by wiring those sort of in series, you get a counter. And that allows our machine to address successive incrementing memory locations in sequence.

And we also talked last week about altering the instruction flow, that is, this notion of skipping an instruction if the result of an operation was zero or had some particular characteristics. Well, skipping an instruction merely requires incrementing twice, rather than once. So you just send two pulses in, in the event that you want to skip over an instruction, and it adds two to the program counter instead of adding one. So it's a very elegant, very simple solution. And it works.

Leo: It's amazing. I just love the - elegant's the word. In fact, that's one of the - I mentioned art. That's why programming is an art. Because it's not, if it's done right, it's not a struggle. It falls into place in a way that's elegant. And you know immediately that's the correct solution because of the elegance of the solution.

Steve: Yes. I think that's - that really says it well. I've seen that when I don't really grasp the problem that I'm trying to solve, but I just sort of start writing code because I'm in a hurry or I'm anxious or impatient, I can sometimes code myself into a corner. I get myself tangled up where I think, oh, wait, I forgot about this. And so I add some code to fix that. It's like, oh, wait, I forgot about that, and add some code over here. And then before long you end up with just this big mess.

And in fact one of my very best friends once said something to me. This is, oh, I am an old man. This is about 30 years ago maybe. More than that, actually. He said that sufficiently complex problems need to be coded three times because the act - and you have to solve it all the way three times. Because his observation had been that when he - and this was, like, a big system, like a CAD/CAM PC board routing problem. It's like, you know, you start off, and you think you know how to solve it. So you start programming. And the act of reducing this abstract problem to individual instructions to reach a solution, that act teaches you so much about the - more than you knew in the beginning about the problem you're trying to solve, that when you're done you really need to scrap it. You just need to throw it away and start again.

And when you're starting the second time you know, you understand the problem you're trying to solve so much more clearly than you did the first time, even though you thought you understood it then. Now you do it the second time, and again you still haven't got it. You got it better. But now you're solving it sort of almost at a meta level because you really do have a grasp of it, having solved it once before already.

And then his point was the third time's the charm. I mean, and here's the problem. The real world never lets us do this. The real world has managers and marketing schedules and timelines and commitments and all this. And so it's so difficult to ever have an environment where - except as you said, Leo, as an artist, where fundamentally you don't have a commercial goal. You have an artistic sort of aim. And there you can say, okay, wait a minute, I'm just going to throw this away because I don't like it, and I'm going to start again.

Leo: You have to do that.

Steve: Yeah.

Leo: It's part of the process.

Steve: Yeah.

Leo: I've got to do, I keep wanting to do - and we've got people like Randal Schwartz and you who are just topnotch, best in breed programmers. And I just would love to do a programming show. It's such an abstract subject that I don't know how we would do it. I mean, I guess it's no more abstract than what you and I talk about every week. But I would like to do a show about programming as art. And there are people like Paul Graham. Paul Graham's fascinating on this. He wrote a book, in fact, called I think "Hackers and Painters," that compares fine artists and great programmers. It's just a fascinating subject. Anyway, maybe someday.

Moving on, Question 8, listener Curtis Clark can't get anyone to listen about security, dang it. Steve and Leo, I really need your help with this one. I constantly listen to your show, and I'm constantly reading up on all the things you guys talk about every week. So many times I find myself seeing a friend or family member doing something that would make you guys cringe, like four-letter passwords. I cringe every time my bank says give us a four-letter PIN. They don't even say letters, four-number PIN. It's like, c'mon.

Steve: Yeah.

Leo: Or opening random attachments in email. Let's see what this is. If it weren't for me - oh, hey, somebody has a movie of me I can watch. If it weren't for me keeping up with your show, I would be reinstalling Windows every three days for these guys. No matter how many times I tell them "Don't open that" or "Why is your password the dog's name?" they just don't seem to believe that it can do that much harm. How do I convince them they really need to be careful and that you just can't browse the web like your computer is invincible? Thanks for all you guys do. Curtis.

Steve: You know, this is a lament that we hear from time to time, and this is not the first time I've even chosen it to share on Security Now!.

Leo: That's right, we talked about this before, yeah.

Steve: Yeah. There's no magic bullet. All I could tell Curtis is that he's probably having an effect, even though it's not as profound as he would like. Certainly all of our listeners who are taking time out of their lives to listen to this podcast and think about these things clearly understand what's going on. We also know from statistics that a huge body of users are in fact getting themselves hurt through their actions. I saw a statistic just the other day that said, from Microsoft's Security Essentials scanner, that one out of every three machines Microsoft is finding has malware on it.

Leo: Oh, that's so depressing.

Steve: One out of every three. So, sure. You could look around and imagine that the people that you know aren't among that one third of the users. But that must mean that there are some others close by who are. So I just think being an evangelist is the best we can do. So, Curtis, I would just say keep trying to get people to understand that security matters.

Leo: Yeah.

Steve: That's all you can do is just sort of, just, you know, I mean, don't be a big pain about it. Don't overdo it, or they'll just - people will tune you out and say, oh, that's all you ever talk about is these problems. But just sort of nudging them up in password length would be a good thing to do.

Leo: You know, the other day our chatroom, somebody in our chatroom was saying, Leo, I can hack your iTunes account because your security question isn't secure because the answer to it is on your Wikipedia page. I said, "Be my guest." Because I don't answer the security questions with the right answer.

Steve: Yup.

Leo: I mix them up. So if you ask me what street I lived on when I was growing up, I might tell you the name of the dog I had when I was growing up instead. And, see, a simple little thing, it's not a hard thing to do or remember.

Steve: Right.

Leo: But if you answer the question with something that is publicly available or easily guessable, what city were you born in...

Steve: I was just going to say, there's no way I'm going to write that into a form. I have...

Leo: It's crazy.

Steve: Exactly.

Leo: Especially somebody like you or me, who people can find and stuff.

Steve: Yeah.

Leo: Last question, from Anders Wold Eldhuset in Norway, actually two questions about our recent episodes. Steve, I've been listening to Security Now! for some time, and I've particularly enjoyed the two recent episodes on how computers work, as well as earlier episodes on programming and engineering. Question 1, when working with machine language - and we should say you do not work with machine language. You work with assembly language; right?

Steve: True, but they're the same.

Leo: It's a one-to-one correspondence. But instead of writing 1010001110000, you're typing MOV.

Steve: Yup.

Leo: You know, three comma...

Steve: There's one more one on the end of that one.

Leo: Sorry, I left that out. Were you counting? When working with machine language, how do you - so typically when the programmer says I program, they say assembly language, which it renders down to machine language one to one, but it's not machine language - how do you deal with things that aren't easily expressed as numbers by humans? For example, how would you store a string or send instructions to a windowing system? Oh, this is good. I'll let you answer that, and we'll do question two, unless you think they should go together.

Steve: No.

Leo: Yeah, let's do that first.

Steve: Yeah. That's a great question. When you are at the machine level, waiting for the user to answer a question, would you like to proceed, and there's a question mark on the screen, there's a keyboard in front of the user. And every one of the keys they press will send a different pattern of ones and zeroes, a different byte, essentially, to the computer. So the computer knows what the so-called ASCII, American Standard Code for Information Interchange, ASCII, what the ASCII pattern of bits is, which is a uniform standard, for each of the keys. So the keys are actually byte-size, that is to say, eight-bit numbers, with each key corresponding to a different number.

So say that this question is supposed to have a Y or an N, and those are the only two acceptable results. And the user might use capital or lowercase, we're not sure which. So what the computer would do is wait for the user to press a key and then receive the byte, which is just an eight-bit number, essentially. And we know that there's four acceptable numbers: the number that represents the capital "Y" character, a different number for lowercase "y," another number for capital "N," and another number for lowercase "n." So what the computer would do is compare the number that was received with, one by one, with each of these four possible valid responses.

And the way compare instructions work at machine language is they normally subtract one from the other. And only if they're the same will the answer be zero. So you'll do a compare instruction, comparing what was received versus the pattern for capital "Y." And then you'll check to see if the result is zero. If so, that means that they were equal, so you've figured out the guy was responding affirmatively. Then you'll check to see if it's - but if not, if they weren't equal, then you go to the next step, which is see if it's equal to the lowercase "y" number. And if so, again, you know that he's responded affirmatively. And so you'll take a branch off to that code in your program. If neither capital "Y" nor lowercase "y," well, you want to check to see whether they've responded negatively, with capital and/or lowercase "n." So again you perform two comparisons. And if none of those have happened, if they hit Q or something else, then you'll end up with no equality matches in your four tests, and probably want to say back to the user, please respond with Y or N to the question.

So there's a very simple approach or a simple example of how, even though we're dealing with bits and little globs of bits, bytes, the human sees the computer as, oh, look, it knows about Y and N. It knows yes and no or something else. When in fact inside the computer is just saying, okay, or the programmer knew what the patterns were for those keys and checked to see whether the patterns matched or not. And for me the charm of computing is that what's really happening down at the machine level, as we discussed last week, is no more complex than that. It's just do things compare, are they bigger, they smaller, equal? If so, jump here.

So you get to build this fantastic, flexible, powerful system that we're all using, we're sitting in front of. I mean, we're listening to audio coming, you know, which is represented as bytes of numbers flowing, and it's turned back into sound, I mean, and being processed. But fundamentally, at the lowest level, we're just dealing with abstract patterns and bits. And from that we get to build something just spectacular. To me that's just magic.

Leo: You can, I mean, if you want to represent a string, most assemblers will allow you to write a string, a quoted string, and stick each letter in a place; right? I mean, that's not a - it's not like you're looking at it, and you have to go, you know, do it in hex or something.

Steve: True. There are - the assembler - exactly, Leo. The assembler does some things for you to - for example, you don't have to have a chart of A through Z...

Leo: Right, in ASCII...

Steve: ...and be looking them up and writing them all down. You're able to do - it'll understand, like, quotes, and that you have a series of characters. And it understands what the ASCII codes are and will do them. So, for example, I would, when I'm writing, I would say cmp al, which is the short name for the lower half of the A accumulator, so it's a byte size. I would say cmp al,'Y', so it's saying compare what's in the al register to the ASCII value of Y. And then I would say jne, jump if not equal, or je, for jump equal, and then to somewhere else. So it's definitely the case that, you know, I'm very comfortable writing that way. But what I like is that what I write is exactly what the computer does. And I enjoy writing, so I don't mind that I have to write a lot of stuff to get not that much done. I just like it that I just can't see it being any better than, I mean, it's exactly what I intend is what the computer's doing.

Leo: Well, that's easy for you to say [laughing]. A lot of people think that they are saying exactly what they mean, and the computer says, uh, really? You wanted to do that? Okay. And it's not what they thought they were saying. You're just a good programmer. And experienced. Question 2...

Steve: Well, I'm old.

Leo: You're old and goofy.

Steve: Yeah, yeah.

Leo: Or kooky.

Steve: Kooky.

Leo: Question 2 from Anders, do you use anything but ASM yourself, for instance Perl on your web server or Tcl/Tk for your GUIs or something like that? Why or why not?

Steve: That's a great question that I get asked, of course, often. I love writing in assembler. I'm very comfortable with it. But it's certainly the case that I can't do it everywhere. For example, GRC's news server was based on INN, which is sort of the standard Internet NNTP protocol newsgroup server. It's open source. It's in C. And what GRC is running is a much-modified version of that. Our newsgroups have all kinds of cool features which I've added, like authenticated deletion of posts so that the people who post to the newsgroup server, into the GRC newsgroups, are able to simply delete their own posts, but nobody else can. Well, that's a problem in general for newsgroups on Usenet is that deleting other people's posts, there was no authentication mechanism for that. But I added that to the newsgroup server.

And we have all kinds of other features. The people who post there don't want the postings to be sucked out by someone else. For example, once briefly they were appearing on Google's groups. And consistently the people who hang out in the GRC newsgroups just want them to stay there. They don't want them to leak out. So there's another feature where, as postings are being pulled from the news server, the news server tags on the IP of the other end of the connection, that is, the first client reading the post.

Well, the beauty of that is that we were able, for example in the case of them leaking out to Google, able to look at the headers of the GRC articles which Google had, and we could see the IP that was causing the leak. And then I was able to block that IP, which turned out to be some corporation somewhere decided they wanted to start cloning all of our newsgroups on their own server. And we said, eh, we'd rather you didn't do that. So those are all things, since this code is written in C, I modified the C code, I mean, in classic open source fashion. I took advantage of the fact that it was an open source server and went in and made all kinds of changes in C, which, you know, I also...

Leo: So you're comfortable with C.

Steve: Oh, absolutely.

Leo: C is a beautiful language, especially...

Steve: I really like it.

Leo: Yeah, if you - it's the next best thing to assembler, really. It's just gorgeous.

Steve: Yeah, I really - there's a lot of things I like about C. And then the wacky language of my choice is Perl, which is completely loony tunes. I mean, it's what I use for any sort of one-off things I'm doing. I have processed all of GRC's web pages through Perl scripts when I've wanted to make wholesale changes throughout the whole site. Or it's great for parsing logs, where I'm just sort of doing ad hoc stuff. I don't want to - it doesn't make sense to write a whole bunch of assembly language for something where I just want to do some quick pattern match and search or something. So there are places where I'll certainly, like, use Perl just because I'm just banging out something really quick.

But all of the code that I write that's serious, the eCommerce system for GRC, all of the stuff that runs GRC's server and all the apps that I write, I just write in assembly because, for me, it's still what I prefer. But, so, yeah, there is some variation from that. But largely assembler unless there's some reason not to.

Leo: So you're not anti other languages. There's just one you prefer.

Steve: Yeah.

Leo: I find most programmers are like that, that they - because, you know, Randal says this. He says if you don't program three hours a day in Perl you're going to lose your fluency. You need to use it constantly to maintain your fluency. I don't know if assembler's like that. But I think that there's a certain amount of fluency gained by doing it day in, day out. Maybe you've done it long enough now that you would never lose it.

Steve: When I've not - there have been times when I have felt rusty. But I understand what Randal means about Perl. Perl is just so bizarre.

Leo: Yeah, it's kind of - yeah.

Steve: For me, I don't lose assembler because for me there's less to lose. There just isn't much...

Leo: Yeah, that's different, you're right. There's not a lot of syntax in...

Steve: Exactly. I was going to say semantics.

Leo: Right.

Steve: There isn't that much semantics in assembler. But Perl is just insanely, like, wild...

Leo: Arcane, yeah.

Steve: ...with the things it will do. And so you really do, I mean, I spend a lot of time debugging Perl. I spend almost no time debugging my assembly language because that I really understand. I've got that nailed. And so very often I'm just, when I'm in a debugger, it's really the exception in assembler. But I use - I'm single-stepping through Perl all the time, going okay, what just happened here? I just, you know, this did something I don't understand. And that's the point, is the reason I love assembler is it never does something I don't understand because there's nothing there. I mean, it's just moving data from one register...

Leo: That makes sense, yeah.

Steve: Yeah.

Leo: Yeah. Well, so my point was that people who are working in higher level languages, where the syntax is tough to remember, tend to pick one and work in it all the time. They might dabble in others. And then there's dilettantes like me. I collect languages. I love learning languages and playing with them, writing small pieces of code. But every time I come back to a language, I have to almost relearn the thing from scratch, just because I've forgotten everything. But I started with Basic, like a lot of people. Did a lot of C programming, which I love. Did assembler, 68000 assembler. Wrote several fairly large programs in assembler and loved it. Learned Forth, which was really weird and great.

Steve: Ah, Forth is a remarkable language.

Leo: I love Forth.

Steve: It's a write-only language.

Leo: Well, it can look like pure English. But you make up your own vocabulary. So in that sense it is write-only because you have to - it's your vocabulary.

Steve: Well, and when I say that, my experience with Forth has been that it is possible to write it, I mean, obviously you're able to write it. But reading it is like, okay, wait a minute. Maybe I just am not fluent enough in Forth. But it's...

Leo: The theory behind it - also Forth is very weird because it's stack based, I mean, it's got this little interpreter running in it. It uses RPN, not, you know, it's postfix, not infix. So it does a lot of things very differently. So there's some weirdness to that. It's used, it was originally written by Charles Moore. And I interviewed him, by the way, a few years ago at Tech TV. It was really a pleasure.

Steve: He's an astronomer.

Leo: He's an astronomer, right.

Steve: Yup.

Leo: He wrote it to control telescopes. So it's used a lot and embedded because it's very compact. But you can write, if you set up your vocabulary, you can write in almost plain English, postfix English.

Steve: Well, not only is it compact, but the core interpreter that you need is very small.

Leo: Right.

Steve: So you're able to bring up a Forth interpreter on a new chip family...

Leo: In just a few K.

Steve: ...very easily, yeah.

Leo: You could do it in hardware.

Steve: Yup.

Leo: So anyway, and then Perl, and Python, and Ruby, and C++. And for me it's fun to look at how languages are implemented. Somewhere I have a document that a guy wrote, "10 Things to Do When You're Learning a New Language," like 10 programs to write, like Towers of Hanoi, that kind of thing.

Steve: Right.

Leo: And his position was, and I think he was right, I've preserved it, that if you use these as your exercises, once you've finished it, you will be fairly fluent in a language. So somewhere I'll find that, and I'll put that up somewhere. But it's a great hobby, as well as for somebody who uses it professionally, like you do. But I just always loved it as a hobby.

Steve: Well, and we're surrounded by it. All of our listeners who are using computers, I mean, all of this was written by people in one language or another.

Leo: I talked to a high school class a couple of weeks ago, an engineering class, and I said, "Learn to program." It's simple, just learn to program.

Steve: Yeah.

Leo: There is no better time than right now to know how to program because, if you have a good idea, you can implement it virtually for free on Amazon's EC2 or on Google, the Google App Engine. They provide bandwidth for a basic site. And you can create a proof of concept. You can really create a business for nothing, if you know how to program.

Steve: And I think that understanding some basics of programming really helps using a computer. I think, if you understand something about the way computers work, you're able to guess about how to get things done when you're a user of a computer. You just sort of, you know, you have a mindset that helps you sort of grok what it is that the programmers were probably thinking.

Leo: Well, but that's the point also is that some people will never - if you took geometry and couldn't do it, couldn't get it, or if you took chemistry and it was like, this is too hard, I mean, there are some people who just - it's not right for.

Steve: Just out of reach, yeah.

Leo: But if you're the kind of person who can get that kind of thing, or logic, you're good at maths, then programming should be fairly easy. And it is well worth pursuing.

Steve: Yeah.

Leo: Oh, it's so much fun. And it's not going away.

Steve: Nope. We'll be talking about it next week, as a matter of fact.

Leo: Oh, what is next week?

Steve: Next week we're going to build on our foundation, which we laid last week, of a very simple but completely functional machine language. We're going to now talk about the real world extensions to that, the things that have been done since, which took that basic platform and made it much more usable.

Leo: Hmm, that's intriguing. Next week on Security Now!. You'll find Steve at GRC.com, the Gibson Research Corporation. That's where SpinRite lives, the world's best hard drive recovery and maintenance utility, a must-have if you've got a hard drive, but also all those free programs he writes. If you have questions for our next Q&A session two weeks hence, go to GRC.com/feedback. Show notes, 16KB versions of the show, transcriptions, the works, available at GRC.com/securitynow. Just browse around. GRC is really a playground for geeks. And Steve, we'll see you next week.

Steve: Thanks, Leo. Talk to you then.