SECURITY NOW!

Transcript of Episode #36

Listener Feedback Q&A #6

Description: Leo and Steve 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 they have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-036.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-036-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 36 for April 20, 2006: One Dozen Questions. Bandwidth for the TWiT Network is provided by AOL Radio at AOL.com/podcasting. Security Now! is brought to you by Astaro, makers of the Astaro Security Gateway, on the web at www.astaro.com.

Yes, it's time once again to plumb the depths of the dark side of the 'Net – security, viruses, spyware – with the man who invented the term "spyware," coined it or first discovered it, Mr. Steve Gibson. Hello, Steve.

Steve Gibson: Hey, Leo. Great to be back.

Leo: We've been having so much fun. And here we are in our 36th episode.

Steve: Wow.

Leo: I know. It's been more than half a year of talking about security issues. Sometimes timely, but a lot of times really kind of education. We've just completed a big series on crypto, which has been great.

Steve: Yup. And in fact next week we're going to talk about the applications, all of the ways those five basic building blocks of cryptography can be assembled in order to achieve the things that people really need to get done with security on the 'Net.

Leo: But as usual every fourth episode – our modulus 4 episodes, as we call them, since this is 36 – we answer your questions. And Steve's got a lot. We've got a dozen, an even dozen questions today. Actually really a baker's dozen because, like most good programmers, you start with question zero. And that's James from Auckland, New Zealand, who wrote to us with a real-world WPA cracking experience. WPA is the encryption technology you recommend for Wi-Fi base stations.

Steve: Yup.

Leo: In fact, you said it's the only safe way to protect your Wi-Fi. But James said: This morning – I won't do this in a Kiwi accent. I'll leave that to your imagination. This morning at a security seminar I attended there was a demonstration on cracking a WPA preshared key, which took me a bit by surprise. I remember you were talking about WPA and how it was virtually unbreakable, provided a strong key was used. What I didn't know was how easy it was, with the right knowledge, to brute-force a weak PSK – pre-shared key. For this demonstration the presenter used Linux and a few freeware tools like Kismet, Aireplay, Ethereal, and coWPAtty. I love that name. He showed us how it's possible to capture data from the four-way handshake at the start of a session by sending 12 packets masquerading as originating from the access point to a station that basically blips the connection for a fraction of a second, causing the station to reinitiate – and this is the key, the four-way handshake. The exchanged EAPOL packets were captured by Ethereal, analyzed in another program, where the preshared key – in this case a really dumb one, 12345678 – was found in about, get this, 60 seconds. The whole process took less than five minutes. I just thought that was interesting, an extension and practical demonstration as to what you were talking about regarding WPA and the need for really actually good random passwords.

Steve: Well, you know, James posted this, and I was delighted to see it because it basically says everything we talked about when we were covering our Wi-Fi stuff, that is, even mentioning that an existing connection could be induced to reestablish, reauthenticate itself to the access point that would allow somebody who wasn't there at the beginning to capture the credentials; and then also how WPA, the best encryption you can get for Wi-Fi right now, how WPA is prone to an offline attack. And that's exactly what James described, where Ethereal was used to capture that transaction, and then offline the machine cranked on it. And because it was such a dumb password – I mean, they used 12345678 – it didn't take long to try 12345678 offline and see that the resulting credentials matched up. And that allowed them then to access, basically decrypt all of the conversation that they had been recording in the past and in the future.

Leo: And really 22345768 wouldn't have been that much harder. Besides being nonrandom, it's really short.

Steve: Yeah. And again, it's why I've got my passwords page at GRC.com. If people just go to GRC.com/passwords, my server generates really, really good WPA-qualified passwords. And, I mean, you can't type them in. You've just got to cut and copy and paste them. But, you know, nobody will ever get them.

Leo: They're, what, 64 characters long?

Steve: 63, actually.

Leo: 63. So that's as long as you can be. And if you have totally random 63 characters, brute-forcing that is next to impossible.

Steve: It's just not going to happen.

Leo: Nathan Clark of Allendale, Michigan, has been pondering public key cryptography and asks: After listening to Security Now! 34, Steve, couple of episodes ago, I don't understand why it's not possible for the private key to be inferred from the public key. I mean, why is it someone couldn't encrypt something with a public key, then try to decrypt it with every possible option? I realize it would take a long time, but is the reason it's impossible because it would take an unrealistic amount of time, or is it impossible for some other reason?

Steve: No...

Leo: That's a good question.

Steve: I really like that question. It is absolutely possible to do that. But these public and private keys are, like, 1,024 bits long. I mean, they are really, really long. And it's necessary, I mean, you know, we're used to as a society now talking about the existence of galaxies and, you know, how long the universe has existed. I mean, you know, we've sort of become inured to really, really large quantities. But 2 to the 1,024 bits is an insanely large number of possible combinations. And so yes, it would be possible, as Nathan suggests, to try every possible private key, making up private keys. But you just, you know, you would be dead before you made much headway at all.

Leo: This is pretty closely related to the previous question, then. Its randomness, of course, is important. But really length combined with randomness is what gives you an imperviable key.

Steve: Well, but I think we've got a question here today that talks a little bit about that. So we're on our way toward that.

Leo: Good. Doug from New York writes: Since both of you, myself included, use Skype – we're using Skype right now...

Steve: Yup.

Leo: ...after several failed experiments with other clients.

Steve: We keep coming back, Leo.

Leo: Skype seems to work the best. We've tried a number of different things, including – Ventrilo sounded really good, but the lag was killing us. Skype is just – it sounds pretty good, and there's just no lag. Anyway, he says, is there any truth to the fact that our bandwidth is being stolen, stolen by Skype, even when we're disconnected? I've read reports that this is not uncommon if you're not behind a NAT. What about when we're behind a NAT? Personally, I have not noticed any slowdown in my day-to-day activities. But would I notice anything unusual? He's talking about the fact that Skype is a peer-to-peer application.

Steve: Well, actually it's a little worse than that. Skype is peer-to-peer, so that, for example, you and I have UDP Internet traffic going directly between us. But actually that's because I have given Skype a fixed port that it can use to forward traffic through. Otherwise...

Leo: Oh, should I do that, too?

Steve: Well, no, because you've got a more well-behaved NAT router.

Leo: Oh.

Steve: I've got a misbehaving NAT router which technically is a little more secure, but it doesn't make a big difference one way or the other. But the only reason you and I are able to go direct is that I have established a fixed port for Skype to use. What happens for most users is – and this is an annoyance with Skype. If they're behind some older routers, for example, if I hadn't done this, then Skype will use someone else's copy that happens to be running and is logged into the Skype network as a forwarder. That is, they...

Leo: This is the so-called "supernode."

Steve: The supernode, yes. And there's no way to turn it off. There's no way to tell it in the options not to do this. It simply, if you've got a computer not behind a NAT router – well, of course, in that case you pretty much deserve what you're going to get – or also not behind a firewall, then, that is, if you've got a machine which is able to accept unsolicited connections into Skype, the Skype network determines that, and it will use your computer and your bandwidth to forward other people's conversations.

Leo: Does the whole conversation go through you, or just a handshake?

Steve: No, the entire conversation.

Leo: Oh.

Steve: And so the reason it doesn't work for you and me, Leo, is suddenly then we get quality degradation. And basically, instead of going point-to-point, we're going through that third party, they didn't give us permission, and the latency of our packets is increased also. So, you know, what I liked about Google Talk was that they used their own servers in the case that they were unable to do a direct peer-to-peer connection. Skype doesn't. They use unsuspecting users' computer.

Leo: But again, only if you don't have a firewall or you're not behind a router because they can't – that's the whole point. They're kind of fixing a problem with people behind routers and firewalls.

Steve: Right. So the idea would be that both of us would make an outbound connection to that third party, and that would allow our traffic in order to go back and forth instead of doing NAT traversal.

Leo: So when I configure – when I set people up for TWiT or some of the other podcasts, I should ask them if they're behind a router. And if they're not, I should say turn on your firewall?

Steve: You really, yeah, you ought to make sure they're behind a router. Now, one thing you could do, Leo, that hadn't occurred to me, you might not, you know, because you and I have been talking about Skype, and you were saying that it works well with us, but you've had some...

Leo: Not with everybody.

Steve: ...problems using it with everybody else, if you did establish static port-forwarding and assigned a port to Skype, much as I have – see, I've done it in order for you and me...

Leo: Right.

Steve: ...to have a direct connection. That would guarantee that you had a direct connection with anybody else who you were Skyping with.

Leo: So I should probably do – and it's something I can do locally that would fix everything generally.

Steve: For everybody else, yeah.

Leo: Is that in the advanced – where is that?

Steve: It's in there.

Leo: I'll have to find it. So I can – probably under "connection." Yeah, there it is. So by default it uses port 42868, it looks like, but I could change that.

Steve: You definitely don't want to use the default.

Leo: Right.

Steve: You never want to...

Leo: It says here use port 80 and 443 as alternatives for incoming connections.

Steve: That's something different.

Leo: That's for secure – that's...

Steve: Yeah. So for what it's worth, if we do have people who are using Skype who are finding that the quality is less than, for example, you and I get, Leo, they could establish static port-forwarding, but they're not going to want to leave Skype running all the time...

Leo: Right.

Steve: ...or they'll find that it ends up being a server for other people.

Leo: Oh. So if you use – okay. But that's only because they don't have a router. If you use static port-forwarding as you are, Steve, that doesn't mean you're getting used.

Steve: I don't know because I never leave it running. I start it...

Leo: You'll never find out.

Steve: Yup.

Leo: Well, that's good. I mean, it's not a security issue as long as Skype is secure. But it's something good to know.

Steve: Yeah.

Leo: And I think I'm going to change – so is there a port you recommend? I guess it doesn't really matter. I could use 428 something.

Steve: We can't say it on the air.

Leo: Right, we won't say what you're using.

Steve: The whole point is you just generate some random number greater than 1,024, less than 65535, pick it, and set up port-forwarding. And I'll bet you that really solves your Skype problem with other people.

Leo: And I don't have to tell you ahead of time.

Steve: I don't ever want to know.

Leo: And I don't want to know yours, either, so there.

Steve: I mean, I could sniff our traffic, and I could see which port it was going to.

Leo: Well, how do we then make a connection?

Steve: Same way. It just figures it all out.

Leo: It does – that goes through some server, the Skype central server.

Steve: Yes. Yeah, there is the, you know, we're both logged onto a server...

Leo: Yes.

Steve: ...in order for this thing to all work.

Leo: And the central server knows our unique weirdo ports, and it handles that.

Steve: Yup.

Leo: Very interesting. Jay Martin, Florence, Alabama, asks: I have a question about the time it takes to decrypt various bit encryptions. You give a certain amount of time for how long it would take to brute-force decrypt a password. My question is, would your estimation of time consider the actual successful password decryption is the last one tried? I mean, what if by chance the brute-force decryption utility chose the correct password in the first short period into the process? Which is why you don't use 1111111 as a password. I understand this is still real strong security, but should this not be considered? What is he asking? I'm not sure what he's asking.

Steve: Well, it's really interesting, and it's a very good point. It's that, you know, we're dealing with statistics. And the idea was, when we have a certain bit length, then we're going to be trying different – for example, using symmetric encryption. We have a 128-bit key. Well, one of those 128-bit combinations is the right key. Only one of them. So he was saying, what if you hit that right off the bat?

Leo: What if you guessed it?

Steve: What if you guessed right? And he's exactly right. You would crack it in a heartbeat.

Leo: But people don't try because they figure, well, chances are it's not going to be the first one or the second or the third or the hundred-thousandth or the millionth.

Steve: And again there are, even though we talk very glibly about 128 bits, if you raise 2 to the 128 power, that's how many possible combinations there are. The proper key is, in fact, hiding amid all of those. So it is there somewhere. But it's the fact that the chance of guessing it is so vanishingly small that everyone considers this to be safe.

Leo: People don't spend time looking for needles in haystacks, even though they might just happen to look right at it the minute...

Steve: Very small needle or very large haystack.

Leo: ...they look. I suppose, though, it means you probably wouldn't want 0000 – 128 zeroes would not be a good password. But you don't get to choose that.

Steve: I don't think anyone would start – I don't know, like, what algorithm you would choose. Okay, I just did the math here. We're talking 3.4 times 10 to the 38.

Leo: Wow.

Steve: I mean, 3,4 and then 37 zeroes. That's how many there are. So assuming that we've just chosen one at random, you know, good luck brute-forcing all of those.

Leo: Although, as you pointed out last week in the birthday thing, I mean, the chances are that it's not the last number you'd try. It's somewhere in the middle of there; right?

Steve: Right. You don't have to try them all. You would normally have to try half of them.

Leo: Right.

Steve: In order to get there.

Leo: John Whiting of Rotherham, Yorkshire – let's just say Yorkshire, U.K. – asks: I started to research digital IDs, ready to experiment with email signing and encryption between my friends and me. Trouble is...

Steve: See, Leo, you've got people doing this now.

Leo: Good, good.

Steve: Yeah.

Leo: Trouble is, you have to register and pay for most IDs. Hmm. Well, okay, I'll address that. Seems Komodo is free for one year. Can I just manufacture my own keys to share and use with friends and relatives? Ooh, this brings up a really – this brings up a whole issue of trust. That's fascinating. Would I use...

Steve: I put this in here for you, Leo.

Leo: Yeah. Would I use a key generator, like your perfect password generator? Are there are free, reliable digital ID services out there?

Steve: This one's yours.

Leo: Well, when you install programs like PGP, which is free – there's a free version. I recommend the GNU Privacy Guard, which is gnupg.org. That's absolutely free and open source. It generates a key. But he raises a very important issue, which is only you at this point have asserted that this is your key.

Steve: Right.

Leo: And, you know, I could create a key that said Steve Gibson and assert that that's your key. So the way that they make these keys reliable is through something called "signing," or "trust." It's the chain of trust. In fact, there are people who have, and from time to time you might go to one of these, PGP key signing parties, where they'll go, and you'll be there with your driver's license or your passport or some ID, and they'll sign your keys. And of course the more people who have signed your key to say, yes, I know this is John, and this is his key, the more trusted your key will be.

Steve: And of course in this mode it's called a "web of trust."

Leo: Web of trust, that's right, yeah.

Steve: Because we have a whole bunch of people. And the idea being that, you know, we build up a web of trust because everyone has proven their identity to each other in a large network, and so that, as a network, it carries some weight. As opposed to, for example, buying a credential from Komodo, for example, as he cites, where then you are going through some process to prove that you are who you are.

Leo: That's what they call "third-party signing," or "trusted third-party signing."

Steve: Exactly. And that's what we'll be talking about next week.

Leo: And that does cost you generally, although Thawte for a while, until Thawte was bought by VeriSign, offered free email certificates.

Steve: Yup.

Leo: And I imagine you can still get free email certificates somewhere. And that one is verified in the sense that they mail you the certificate, and you have to validate that, you know, you have to say, yes, I got this mail, I got it at this address. That at least validates that that address is yours, although, I mean, it's still conceivable that somebody could spoof it, do some sort of man-in-the-middle attack, intercept your email and say, yes, I got that at that address. So really, I mean, it's only slightly more trusted. If you get a key from the key servers, there'll be a degree of trust in there. And you can, by the way, if you correspond with somebody and you know it's them, you're pretty sure it's them, you can sign their key. You can go online and sign their key and say, yes, I want to validate that I know this is who it says it is. And so that's...

Steve: Right.

Leo: There's a trust database. You know, I think it's a really beautiful system, but that is a flaw, isn't it.

Steve: Yes. Somewhere, ultimately, no matter how far back you go, somewhere there is something you are relying on.

Leo: Right.

Steve: So, you know, even, for example, web certificates, when we get SSL certificates in order to establish secure web communications – and I have to do this every two years for, for example, GRC, in order to allow people to connect securely to my server – I'm going through some hoops. They check my Dun & Bradstreet, phone, fax, address, you know, I mean, there is some process that I go through to assert that I am Steve Gibson, Gibson Research Corporation, GRC.com. Then I get this certificate. But, you know, it's not an absolutely foolproof process. So at some level, something needs to be trusted.

Leo: Yeah. This is why I don't update my key or change my key frequently, because if somebody, you know, by now I've been using my key long enough, it's signed by a lot of people, so you know it's...

Steve: Right.

Leo: You know it's fairly trusted. Vincent Ragosta, writing from Pittsburgh, PA, wonders: Why are there nonroutable IP addresses, why, why, why, in the middle of my traceroute output? In fact, how could it get there? Does this indicate my ISP is performing NAT routing? If the ISP is performing NAT routing, how am I routing? How am I able to run servers that are accessible from the Internet? So I guess he's doing a traceroute, and he's seeing things like 10. or 192.168 addresses.

Steve: Right, right. And in fact I've seen that myself. If I've done tracerouting through my Cox cable modem connection, I'll see a few IPs that are public, and suddenly it, like, falls into this 10. space where it goes, you know, maybe five or six hops into 10., and then it comes back out into the public space. This is commonly done by large ISPs because they don't want to burn up or consume their public IP space. Those public IPs are the only way that they're able to offer those IPs. So if an ISP's got lots of network gear internally, they may use nonroutable IPs, you know, the 10. or 172 or 192.168 space, just in order to, like, identify their routers internally to each other, so they're able to move traffic among them. But once the traffic then crosses back out into the public space, of course, it returns to a well-known public IP. So this does not mean that the ISP is necessarily using NAT routing. It doesn't mean that they're not for sure. But what it really is is just a way for the ISP to be conserving their use of what is a relatively scarce resource, namely public IPs.

Leo: Yeah.

Steve: They don't want to burn them up on their own equipment. They want to make them available to their customers.

Leo: But how does it route if it's this private address?

Steve: Well, the idea is that the equipment inside has routing tables...

Leo: Oh, it knows.

Steve: Exactly. It knows, hey, we're sending this off to our Albany office or off to our San Francisco office. And, you know, that's a 10. something, and San Francisco is a 10. something else. So the internal routing tables know how to get the packets within their internal space back toward their destination, and at which point the packets then cross into the outside world.

Leo: Okay, that makes sense. He asks, how am I able to run servers? Well, that's not an issue because the traffic just – their local routers are doing the handoff.

Steve: Right. And in fact we may remember that four weeks ago we did answer a question similar to this. Some guy had his ISP delivering private IP space to him, 192.168.

Leo: Right, right.

Steve: And he wanted to run a router – I mean a server, excuse me – but he couldn't because he didn't have a public IP from his ISP. His ISP actually had him behind their own big NAT router. And I have seen this before. But that is not the case if an ISP is simply using private IP space internally, and many do.

Leo: Robin Robertson of Roanoke, Virginia, asks: I'm curious about LogMeIn.com. If I'm at a public Wi-Fi hotspot and use my laptop to access my home desktop via LogMeIn, which is a remote-access solution, then surf the Internet through my desktop, am I safe? I'm probably confused, but I think their take on UDP is the opposite of what I heard you discuss on Security Now!.

Steve: Okay. Well, LogMeIn is a little bit like GoToMyPC in that it establishes a secure connection to their servers, and then they will connect to something that you've got running at home. So essentially you're securely connecting to them, and your home system is also connecting to them. And so the LogMeIn server, very much like GoToMyPC, creates this bridge that allows you from, for example, a public Wi Fi spot to connect to your home desktop. So the question, am I secure, well, you're secure in your traffic out of the danger zone, which is the public Wi-Fi hotspot, or, for example, if you were using LogMeIn in a hotel to get to the LogMeIn servers. And that's really what you're trying to achieve with this. You're trying to get your traffic out of an area where it is in danger.

Now, if you then connect to your home system and use the Internet from there, well, then, your traffic is unencrypted as normal Internet traffic is, for example, you know, standard POP or IMAP mail and web surfing and so forth, unless you are using an SSL connection. So it's sort of a combination answer. The area of greatest danger, where you're in Wi-Fi or hotel, in any kind of a public setting, that's encrypted, and so you're safe. But in general, once you've left that encrypted tunnel, then your traffic is available just like everyone's is on the Internet.

Leo: So you're exactly as safe as you would be at home.

Steve: That says it in, what, five words, yes.

Leo: No more, no less. John in Australia believes that his SSL connections are not safe. Uh-oh. He says: I had the misfortune to be hijacked through crypto. My details were grabbed. And when I see an encrypted page, the hackers also see what I am seeing. Any suggestions on how they're doing it? I've tried using XP and Linux, and it still happens. Perplexed, since you say it can't be done. Now, I get variations of this question all the time on my radio show. And I think it's just people who are confused.

Steve: I put this in here because I, too, have seen the question. And I'm confused by the question. He says he's used both XP and Linux, and it still happens. Unfortunately, we don't know exactly what "it" is. He says that hackers are able to see what he sees. So...

Leo: I don't know how he would know that.

Steve: Exactly.

Leo: So what we would really ask is what are the symptoms of this.

Steve: Yeah. My first thought was that, if there were something evil installed on his machine...

Leo: Right.

Steve: ...then, for example, a keystroke sniffer or something running on the machine, then that would be a way of there being a problem. The fact that he says he's tried it with Windows and Linux, though, because he's got this cross-platform addition, that makes me more skeptical about something evil going on. The fact is, as far as anyone knows, SSL connections are themselves really secure.

Leo: If you're being spied upon, it's not because they're cracking into that connection.

Steve: Now, there is an exception, though, which is the other reason I put this question in here today, because when you connect to a remote server you are accepting the credentials of the server, which has been signed by someone who signed their certificate, like we were talking about briefly before, about VeriSign, for example, signing GRC's certificate. Some corporations do want to proxy and filter and literally basically decrypt and be able to read their employees' encrypted traffic. So what they do is they run a proxy server in their corporate environment, which has a certificate. And every browser in the company has been told to trust that certificate. So essentially the encrypted traffic is decrypted at the corporate border.

Leo: Ooh.

Steve: Yes.

Leo: That's sneaky.

Steve: It's decrypted at the corporate border and then, essentially, filtered, proxied, checked for spyware, or even for naughty content. I mean, they could do anything they want to once they've got your connection decrypted. Then it is reencrypted using a certificate that every corporate browser has been deliberately installed with in order for the browser not to complain. So employees may believe they have encrypted and secure SSL connections, when in fact it is being decrypted in, you know, en route, essentially, you know, by their corporate firewall or IT staff or proxy server for whatever corporate reasons.

Leo: Wow, is that sneaky. Oh, wow. You know, it's hard to tell what he's talking about. He may have been phished. And in that case, you know, you're not on the site you think you're on.

Steve: Right.

Leo: So that could be what's going on, too.

Steve: Right.

Leo: We can't really tell from the question what's going on. But the bottom line is SSL, real SSL, is secure.

Steve: Exactly. If the endpoints are authenticated – and, you know, I keep using the A word, authentication, authentication, authentication. That's the key for so much of what glues this together.

Leo: Mike Smith writes from nearby Los Angeles: Steve has indicated several times he doesn't have a virus scanner installed in his Windows-based computer. Why is that? Is the performance improvement that much significant over running with virus scanning on? Or is it just because you're a macho tech guy who can spot dangerous email attachments from a mile away? Or have virus scanners essentially become obsolete since the days of everybody sharing floppy disks? Besides email attachments and Outlook email scripts, how do most viruses get on PCs nowadays?

Steve: Well, I...

Leo: I should, by the way, say I don't run – in general I don't run an antivirus either. Or a firewall.

Steve: I know, Leo, neither you or I do. And I...

Leo: And most security experts I know don't.

Steve: I know. And so here we are telling everyone run antivirus, yet we don't do it ourselves. No force on earth could make me open an attachment I receive in email.

Leo: Period.

Steve: I mean, period. Really. It just doesn't happen. I just – I know who's writing me email. I know when I'm expecting something. You know, you and I will talk, Leo, and I'll, like, send you a DOC file, you know it came from me, you know...

Leo: You know, I did, I opened this Microsoft Word document that you just sent me. Boy, I trust you.

Steve: Yeah. And so, I mean, that's what it's about. But other stuff, I mean, I've never had a virus infection on my machine except once when I was doing some virus research, and I was sloppy, and it happened to me. But in that case I was running on, you know, on a red machine, on a deliberate viral research machine, not connected to anything else. So I was prepared for that happening. But I recommend people run antivirus if they want to run antivirus because certainly they're useful, they can catch bad things. And Lord knows, I mean, I'm seeing email that is full of this junk all the time. I just don't touch it.

Leo: Steve's a trained professional. Don't try this at home. By the way, there's another vector. You mentioned attachments, but there's another vector for viruses that, in fact, is very common. But you're not susceptible to that, either: HTML email.

Steve: Right.

Leo: So if you're using Outlook or Outlook Express, or Thunderbird, or any email program that does HTML, displays a web page in the preview pane, depending on how patched your Windows is and what exploits there are out there, that's also a vector of infection and has been in the past. But Steve rejects, actually rejects HTML email. I don't do that, but I just turn off the preview panes.

Steve: Well, and I also have my email reader. I'm still using Eudora. And one of the options is use the Windows viewer or use a text viewer. And I just use a text viewer because, again, I don't need web pages being emailed to me.

Leo: There's a third vector, which is less common, of virus infection, and that's worms, network worms. How do you avoid those?

Steve: Essentially I'm behind a very, very strong firewall. You know, you and I don't run personal firewalls, but we're both behind routers. I'm behind multiple layers of routers. And so there's just no exposed open ports.

Leo: Right. In fact, if you ran Windows built-in firewall, even that would be adequate to prevent Sasser and Blaster and Zotob and all these network worms.

Steve: Well, and that's why Windows XP after Service Pack 2 was released has really been doing a much better job. I mean, the exploits and the security problems we're seeing now with Windows are coding errors, buffer overflows that are, you know, seem to be in a never-ending stream. I mean, I'm not that unhappy that Vista has been delayed because it'll just be a whole new fiesta of security problems.

Leo: Let's get it right.

Steve: Just like XP was in the beginning.

Leo: And of course there's another way that you can get viruses, and that is via floppy. This is how we used to get, you know, the Michelangelo.

Steve: Sneakernet.

Leo: Yeah, where somebody would accidentally boot to a floppy. But those days are gone. I don't think you see those anymore. You know, what I would recommend, and I think that – I know you probably don't do this, Steve. We don't run – what we're saying – what I'm saying, anyway, is I don't run an email scanner in the background all the time because I do feel that that slows me down, and I know I'm not going to open an attachment. But I think it's prudent once a week or once every few weeks to scan with a good antivirus to make sure you haven't accidentally caught something.

Steve: Sure.

Leo: Do you ever do that?

Steve: I have done it in the past when my system has gone a little strange. I mean, that's...

Leo: Just in case.

Steve: That's the key for me is, if something seems weird, then it can cause me to just do a static scan of my system. The thing that is troubling is that now email has attachments saying "this email scanned by such-and-such," you know. It's like, okay, well, anything that was going to be a virus would just stick that on there.

Leo: It's completely meaningless.

Steve: Ugh.

Leo: Just don't open attachments. I mean, even if you have an antivirus, it's just a bad idea. I mean, what if it's a virus that just came out, and your antivirus hasn't got the signatures yet? And that's often the case.

Steve: Now, I have a friend who runs an embroidery shop that uses computerized embroidery. And they're constantly receiving pictures and patterns and data from people they don't know. Now, there's an example of a high-risk environment where it's reasonable to say, okay, we have to, for our business, open attachments from people we don't know. You can imagine, like, you know, any sort of a graphics service is going to have to be doing this, too. So there they want an isolated computer and do the best job they can of making sure that what they open isn't able to spread through the network. And as a matter of fact, these friends of mine are constantly rebuilding their machines because they're just overrun with viruses.

Leo: Wow. You know what I liken it to is the kind of public health recommendations you get. For instance, there's a general public health recommendation to not eat too much sodium, to reduce the sodium in your diet. Well, in fact, the truth is that some people are genetically predisposed to high blood pressure due to salts, and some aren't. Some people don't have to worry about it. But you're never going to hear the health experts say some people have to worry and some don't. It's not prudent to say that. It's much more prudent to say just avoid salt...

Steve: Wherever you can.

Leo: ...wherever you can. Because you don't know, I guess, if you're susceptible. So we're kind of – I would say we're like health professionals, Steve. Avoid salt, folks, even if you're not susceptible. Mike Smith – I'm sorry, that was Mike. This is Willem from the Netherlands. And I do say, by the way, that we have some great listeners in the Netherlands, in Sweden, in the U.K., all over the world, and we just love that. Willem says: I just listened to your podcasts about symmetric and asymmetric cryptography. I'm employed in the field of PKI, SSL, and end-user certificates. What does PKI stand for?

Steve: Public Key Infrastructure.

Leo: Okay. So most of the items were quite familiar. Note: I'm not an expert on theoretical crypto itself, but I'm more into the usage of PKI in corporate environments. The kind of things we talked about last week with the signing and things like that, applied crypto.

Steve: Yeah, exactly, applied cryptography.

Leo: There was a part where you talked about SSL, and that for every SSL session a new symmetric key exchange is being done. Five or six years ago I overheard something about the way Microsoft handles SSL connections. It seemed that Microsoft, well, IE in particular, does not renegotiate for every session, but caches the session key for a period of time. That way new sessions establish more quickly from your PC. On older hardware, of course, this would speed up your browser, but it might impose some security risks. Comments?

Steve: Well, that's interesting, and it's a really great question. It turns out that this so-called "SSL session caching" is part of the spec, and it's happening to all of us all the time, and it's not a problem. To back up a little bit, just to review, the way SSL – Secure Sockets Layer, now called TLS, Transport Layer Security – the way this operates is it uses a public key system to exchange a randomly generated so-called "session key." So when your computer, through your browser, wants to connect to a remote web server, they negotiate a random number securely using public key systems, like we talked about last week, and then this creates a so-called "session key," which they then use for their symmetric encryption, which can be done very quickly. The problem is that this happens every time an SSL connection is established, that is, the endpoints go through this whole process.

Well, as we talked about before, public key negotiation, the whole asymmetric key system, is a thousand times slower than the bulk symmetric key technology. So if connections are being set up on a server with, you know, hundreds of thousands of clients, this can be a real problem. It's been a problem, in fact, so much that there are hardware accelerators which commercial web systems can employ to move some of this brute-force public key technology into hardware just to accelerate it.

Leo: So it's really not to speed up the client computer, it's to speed up the server.

Steve: Well, yes. It's because, you know, both ends are doing a lot of work, but the real burden is on the server that is accepting web connections, you know, at tens of thousands per minute from tens of thousands of end users. But it still is a lot of work. So in the SSL spec they specifically have this idea of a session ID, which is different from the session key. And both ends will cache their keys.

So the idea is that, if I'm, for example, contacting Microsoft, and I establish a secure connection to them, we will go through the labor-intensive, time-intensive, public key system to establish a random secure session key. And we associate that with a session ID. That is known; the session key is kept private at each end. So then, if we drop the connection and then soon after reconnect, for example, I click on another link on their secure site, we don't go through building a whole new session key because we both have the one we obtained before in our own caches. So we say, hey, how about if we reestablish this session ID? But since no one else monitoring the connection would have known what that is because we each have it privately, each endpoint is able to say, oh, yeah, sure, let's keep using that one. And so, for example, the typical session key expiration is ten hours. Which may seem like a long time...

Leo: Yeah, no kidding.

Steve: But it's completely secret to each endpoint.

Leo: So there's no security risk at all.

Steve: There really is no security risk. I mean, these keys are long. They're 128 bits. And as we know, that's 3.4 times 10 to the 38 different combinations. So, I mean, there's plenty of combinations, and no one is going to crack that in a short period of time. And then the keys are always expired after that maximum length of time, and a new key is renegotiated.

Leo: Matthew in Normal, Illinois, is worried: I recently heard that the Mozilla Firefox browser plants spyware on your system. I'm a loyal user of Firefox and was wondering if there's any truth in this. I would prefer not to have to switch back to Internet Explorer, but I cannot continue to do this is if it plants spyware on my system.

Steve: Well...

Leo: No.

Steve: That was a bit of an urban myth that went around recently.

Leo: No, yeah. I think spread by Microsoft. I don't – that's ridiculous.

Steve: Well, actually there was – it was some sort of a weekly spyware newsletter that unfortunately had this as, like, a big, bold headline. Then the "Register," that outfit in the U.K., picked it up.

Leo: Known for their journalistic integrity.

Steve: Exactly. And anyway, it scared a lot of people. So I wanted to, because I saw this a number of times, I wanted to make sure people knew that this was not the case. It was just a complete mistaken, you know, basically completely false story.

Leo: You will get spyware checkers from time to time who'll report false positives. Microsoft's Defender says, for example, that Spybot was – or at least for a while said that Spybot was a piece of spyware. Of course, it's not. It's a very good anti-spyware program. And I think that that's just, you know, false positives.

Steve: Well, in fact, one of the reasons that anti-spyware programs often claim that each other are anti-spyware is not any kind of an anti-competitive thing, it's that sometimes they have to contain the patterns themselves that they're looking for...

Leo: So that's what it's seeing.

Steve: ...in their own code, exactly. So it'll see a pattern that it's looking for and say, wait a minute, this might be spyware, when in fact it is anti-spyware.

Leo: Now, Matthew might say, well, how can you be sure? This is how you can be sure. Firefox is what we call "open source." All of the source code, every bit of what Firefox is doing, is available for your download and perusal. You can look at it. You can see exactly what it's doing. There is no part of it that is not completely exposed and visible. Now, you may not be able to read the code, but there are plenty of people who can. And I assure you there are thousands of eyes looking at this. And if there were the slightest hint of spyware in there, you would know about it.

Steve: Yeah. It would never happen.

Leo: That's actually the beauty of an open source program. You can't really assert the same thing of a closed-source program.

Steve: Although that does speak to making sure you get your programs from an authoritative source. For example, if you downloaded your copy of Firefox Mozilla, or Mozilla Firefox, from, you know, some random website somewhere...

Leo: Oh, that's a good point.

Steve: ...you really then have no knowledge what it really is. You want to make sure you always return to the source.

Leo: That's where those MD5 hashes we were talking about last week come in handy. You can really validate that that's real.

Steve: Well, exactly. But again, only if you're validating against a real MD5. You could have a bad guy would create the bad MD5 and say, hey, here's my copy of Firefox.

Leo: See?

Steve: Download it and make sure.

Leo: Right.

Steve: Yeah.

Leo: In fact, because it's open source, it would be fairly easy for somebody who was a bad guy to take Mozilla and add spyware code to it.

Steve: As a matter of fact, Leo, many people through the years have asked me why I don't publish the source code for all of my freeware security utilities, and it's specifically for that reason. Even though somebody could certainly recreate the look and feel of my stuff – because it's really pretty simple, they could do it from scratch – I'm certainly not going to help anybody by giving them the source code and then, you know, have fraudulent copies of GRC stuff floating around.

Leo: That's a very good point. A very attentive listener, Dave Chismon in the U.K., asks: I have a slight confusion from the episode on asymmetric crypto. You mentioned that a vulnerability with public key crypto was the chosen text attack, that is, encrypting text with a public key in an attempt to work out the text. I'm confused. I thought you said the public key crypto was only used to encrypt the key for a symmetric crypto scheme. If that's the case, then the chosen text attack is surely as difficult as brute-forcing the symmetric key, if not harder, as you have to guess every possible key that encrypted. Well, that's a good point.

Steve: I loved his point. That's why I put it in here. He is absolutely right. So this is – and so just to restate what he said again, remember that what I stated was that asymmetric crypto had a problem. Since one of the keys was public, you could potentially use the public key to feed in your own plain text and see what the public key encrypted that to, and then compare that to what somebody's unknown key had encrypted it to. I'm sorry, I said that wrong. You were trying to guess, I'm sorry, you were trying to guess a private key. And you knew what the private key had encrypted something to, so you would be able to use what's known as this "chosen plain text attack" in order to feed your own stuff through. He makes the point that, as we know, we only use asymmetric cryptography to encrypt a symmetric key. So essentially, where we get confused here is I was talking, not about the dual use of asymmetric and symmetric crypto, which does solve this problem, exactly as David has suggested. I was sort of stepping back further, earlier in my discussion of just pure asymmetric cryptography, noting that that's a problem that asymmetric crypto has because one of the keys is known, whereas symmetric cryptography doesn't have this problem because the symmetric key is always kept private. I mean, that's the only way it makes any sense.

Leo: So really this is an explanation of why PGP and these other crypto techniques use both symmetric and asymmetric.

Steve: Well, actually they use both because asymmetric is a thousand times slower.

Leo: Oh, it's also speed, yeah, yeah, yeah, right.

Steve: So you just can't afford to use asymmetric crypto on your whole message. It's so much faster.

Leo: Even if you could, it would be open to this plain text cipher cracking method.

Steve: Yes, that's a very good point. And so the way that's resolved – and I've talked about this also in our asymmetric session – is you add some random stuff to what you're encrypting.

That way nobody else can encrypt the same thing because they'll only know, like, the part of what it might be that you are encrypting, not a bunch of other random bits. And that'll just completely throw them off the track.

Leo: But it's kind of unnecessary because it is symmetric key we're using; right? I mean...

Steve: Right.

Leo: So that even there, there's no – you cannot use this plain text crack on symmetric encryption.

Steve: Because you...

Leo: Or can you?

Steve: You can't because you don't ever have the symmetric key.

Leo: And that was David's point. You needed to have...

Steve: Right.

Leo: Yeah, okay. Our last question. Theodore Phillips from Chicago, Illinois, asks: I now feel like I kind of understand public key cryptography. Well, except for one thing. I thought it was also very hard to determine what numbers were prime. Where does RSA get these two giant prime numbers that it uses? How does it know if they're prime? How does it calculate them on the fly? I thought that was hard. Does it have a table with a lot of prime numbers? Does it go to a prime number warehouse on the web? While I know there are an infinite number of prime numbers, seems like you'd need a lot of them handy to make this work. If RSA only had a small pool, well, that would be a security issue.

Steve: Yup, a really good point. And a lot of people have wondered the same thing. They've wondered, okay, why couldn't you just, for example, pre-compute a whole bunch of prime numbers? And if you're relying on prime numbers being hard to factor, just try factoring with a whole bunch of prime numbers that you've figured out ahead of time. And the reason is there are too many prime numbers to do that. I mean, it turns out that prime...

Leo: But how does it find these base prime numbers to make this product of primes?

Steve: It actually does compute them.

Leo: It figures one out, okay.

Steve: Yes. And in fact, when I was – part of the process of setting up OpenVPN, that we will be returning to when we talk about GRC's OpenVPN Guide, part of the process is you build your own keys. And you can see it, I mean, I'm using a 3GHz P4 with a gig of RAM, I mean, a really fast machine. It sits here putting dots on the screen, dot dot dot dot dot dot...

Leo: That's what it's doing.

Steve: ...line after line after line, while it does all the math to come up with its own primes.

Leo: Right.

Steve: So it turns out that, I mean, it's hard to create primes, but "hard" in this case is a relative term. You know, it's harder than adding numbers together, but it's a problem that has been solved by the crypto guys so that machines are able to generate their own primes.

Leo: And you'd only have to, you know, you only have to do it once when you're creating the key. You don't do it over and over again.

Steve: Exactly. It's a one-time thing. Now, it's interesting, I also created what's called a "Diffie-Hellman key pair" as part of this, and there I wanted it to be – I wanted a 2,048-bit key pair. That's also part of what OpenVPN uses. I don't know how long it took.

Leo: You went to bed.

Steve: It was, like, a day, Leo.

Leo: Wow.

Steve: I mean, it cranked and cranked and cranked.

Leo: Wow.

Steve: So, as you tell it you want longer primes, you've got to be prepared to let that computer sit there and cook for a while for it to come up with, you know, its own unique set of prime numbers.

Leo: I wonder, I mean, you may remember in the early days of computing we would benchmark with the Eratosthenes Sieve...

Steve: The Sieve of Eratosthenes, yup.

Leo: Yeah, with the prime number generator. But, you know, I doubt – there must be other algorithms for generating primes.

Steve: Well, in fact the Wikipedia has a really great page on prime numbers. So if anyone's curious about this, go to Wikipedia; or just put "generate prime numbers" into Google, and one of the first links is the link to Wikipedia's page, where there's just a ton of really good information about prime numbers.

Leo: In fact, in general, Wikipedia's a very good source on crypto and all of these subjects. There are a lot of crypto experts who contribute to Wikipedia, and there's some great material.

Steve: Yeah. I'm very impressed with their pages.

Leo: Yeah. They've really done a good job. It's a very definitive and, I think, reliable encyclopedia, at least in that area, on crypto. That brings back some memories, the old Sieve of Eratosthenes. Wow.

Steve: And remember the way you do it is you take a big grid that is all full, and then you punch out every other one, then every third one, then every fourth one – well, actually every fourth would have already been punched out by every second.

Leo: It's really a brute-force method of finding primes. You eliminate non-primes.

Steve: Yup. And when you're all done, you end up with the ones that have not been punched out, or, you know, the ones that weren't evenly divisible by any of the numbers that came before.

Leo: Right. So I guess you'd make, what, a pool of, like, you know, 32,000 numbers and then eliminate them.

Steve: Yeah, but we're dealing – remember that when we're dealing with, like, 1,024-bit primes...

Leo: That's a big prime.

Steve: ...and this again, here, well, we know how big it is. We did the math earlier: 3.4 times 10 to the 38. So that, I mean, we're talking, okay, 38 digits, 38 decimal digits.

Leo: Wow.

Steve: And there are, I mean, there are that many primes. So these are really, really big prime numbers, and our technology today has the ability to generate primes that are that size and know that they are primes.

Leo: Eratosthenes would be proud. Proud, I tell you.

Steve: He'd have a sore finger from punching out all the little holes in his sieve.

Leo: Steve, we've had fun. This has been great. I love doing this. Every fourth episode we answer your questions. Keep them coming, and keep coming back because another Security Now! next week. We're going to continue with crypto. But this is applied crypto.

Steve: This is taking all the components that we've discussed in the last four weeks and putting them together in all kinds of neat ways.

Leo: For more information about our topic, for transcripts, and of course a 16KB version for the bandwidth impaired, visit GRC.com/securitynow.htm – also the home, of course, of Steve's wonderful SpinRite, the ultimate disk maintenance and recovery utility, pays Steve's bills. Visit SpinRite.info for more information.

Steve: Well, and more importantly, if you've got a computer which is hiccupping, really I would encourage people just to take a look at some of the testimonials...

Leo: Oh, yeah.

Steve: ...at SpinRite.info. That'll give people a sense for what SpinRite can do for them. And then, you know, if at any time in the future they're in trouble, consider SpinRite.

Leo: Call Steve, GRC.com. And that's it for this edition of Security Now!. We'll be back next week; I hope you will, too. For Steve Gibson, I'm Leo Laporte. We'll see you next Thursday for Security Now!.

Security Now is brought to you by Astaro, makers of the Astaro Security Gateway, on the web at www.astaro.com.