SECURITY NOW!

Transcript of Episode #317

TCP Part 1 - Getting Connected

Description: After catching up with a week of the amazing news of the security breach of the DigiNotar certificate authority, Steve and Leo continue their "How the Internet Works" series with the first of several episodes describing the operation of the Internet's most used protocol: TCP.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-317.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-317-lq.mp3

Leo Laporte: It's time for Security Now!, the show that protects you and your loved ones online. And here's the guy who does it, Mr. Steve Gibson of GRC.com. He's the guy who discovered the first spyware, coined the term "spyware," wrote the first antispyware program. Steve, we're in - welcome, first of all.

Steve Gibson: Hello, Leo. Great to be back with you again, as always.

Leo: Thank you. And as we've been talking about, we're in a kind of a different set here. My office is busy being soundproofed. I guess I screamed too much. They say I yell. I don't know. And...

Steve: Ah, you're just excitable.

Leo: That's what I think.

Steve: And you're drinking coffee. So that doesn't help, either.

Leo: That's part of the problem. Now, wait a minute, now, wait a minute, Mr. Sexti Venti Latte.

Steve: Six shots now.

Leo: Six shots. But basically, did you start at one, then two, then three, then four? I mean, are you ramping up? Are you building up a tolerance?

Steve: No. No. I just like the taste.

Leo: I drink it - it's like reading Playboy for the articles. I drink it for the taste.

Steve: That's right. I don't look at the pictures.

Leo: Ah. So they know, though, when you come in the door at Starbucks, they say, oh, here's the six-shot guy; right?

Steve: Oh, yeah. They fire up the machine. Actually they turn on the spare.

Leo: Bring - yeah, really, exactly. Bring in reinforcements. Here comes Gibson. So today we are going to start another part of what you've done so well, which is these kind of basic, educational, how-technology-works segments.

Steve: Well, yeah. How the Internet works, specifically. We've talked about the underlying packet-oriented operation of the 'Net. We then talked about, while you were on jury duty, about the ICMP and UDP protocols, which are built on top of that. Today we're going to start the beginning of discussion of the Internet's No. 1 most used, most important protocol, which is TCP, the Transmission Control Protocol. And it's much more complicated than the other ones. So this one is - I'm calling it "TCP Pt. 1 - Let's Get Connected."

We're just going to attempt to explain really clearly what TCP's goals are and the process of initiating a connection, and sort of talk about the most important aspects of that, and also what the hackers have done because it's interesting that every aspect of TCP has been attacked because it is pervasive. Since it's the glue that connects everything, essentially, it's been a big target for attack. So we'll look at how attackers have tried to leverage the way TCP works.

But we also have a week's worth of, first, amazing news on the whole DigiNotar nightmare that we began to talk about last week. You'll remember, Leo, that I said, you know, based on what I've seen and the source code of the Mozilla and the Chrome browsers, we haven't been told the whole story. We will be revisiting this. Well, we didn't waste any time because there's been a whole bunch of more information has come out. So we've got a bunch of other sort of interesting updates. We will catch everybody up on the DigiNotar news and then plow into TCP.

And I have to say, at the top of the show, where is Apple? Apple is completely absent from this. Everybody else has been scurrying around and fixing their browsers. And I fired up my Mac just this morning, just before the podcast, to make sure that Apple hadn't suddenly woken up. I mean, people are - PC World has an article wondering why Apple hasn't responded at all to this, which is now more than a week and a half old. It's crazy. But no word. Not a peep from Apple.

Leo: So, Steve, DigiNotar. I saw that they think it was an Iranian hacker involved.

Steve: Well, we're going to talk about that. It's been confirmed pretty much that it's the same guy who did the Comodo hack this last March.

Leo: He's claiming it, anyway; right?

Steve: Well, he's claiming it, and he's also offered some proof. The people that DigiNotar brought in to check over, like to do the analysis of their own systems, did find some fingerprint information left behind that was identical to what was used in the Comodo attack. And I've got two really interesting posts that I'm going to share with our listeners from this guy. I always like to start off by talking about security updates. And across the board there have been updates in direct response to this problem.

Leo: So Chrome is up to date. IE's up to date.

Steve: Yeah, Microsoft issued an out-of-cycle update and pushed out to all their supported platforms to remove DigiNotar's trust from Windows. Mozilla, we talked about 6.0.1 last week, which was the first phase of this. Now we have 6.0.2, which is the second phase. And we'll be talking in detail about what they did. But again, really, I mean, notably absent from the party is Apple.

Leo: Now, that's probably not a problem if you're using Safari, Apple's browser, on Windows because they use the Microsoft certs. So your IE update will have fixed that. But it's a big problem on the Mac.

Steve: Well, yeah. And, I mean, I would say, all ye who value security, abandon Safari. I mean, it's just - switch to Chrome or to Firefox or something. I mean, there have been blog postings where people are showing how to remove the DigiNotar trust. I have also seen other unconfirmed reports that doing so doesn't fix the problem. So the only advice I have for people for now is just don't, over on the Mac platform, just suspend your use of Safari until Apple responds. I mean, I just can't figure out what Apple is doing, why they're alone in not having responded to this. And we'll talk in a second about what "this" is. I have some other stuff I want to do first so that we can give DigiNotar enough time.

There was a sort of a strange DNS hijack which occurred which a lot of people tweeted me about to make sure I knew about it. The thing that was of concern was that people were saying that hundreds of websites had been defaced. I actually got email through my regular GRC channel saying that UPS.com had apparently been defaced and TheRegister.co.uk had been defaced. And so essentially UPS, Vodafone, TheRegister.co.uk, Acer.com, BetFair.com, NationalGeographic.com, and Telegraph.co.uk were defaced. And that's all.

There were reports that there were hundreds of websites, but it turns out that a site that was tracking this had a history of other defacements and hacks, and the press didn't properly understand what this third-party website was showing. And so they were saying, oh, this is hundreds, when in fact it was only that little handful. But what they all have in common is that they all used the same domain registrar, which was NetNames. Turkish attackers were able to hack into the DNS management panel of those NetName user accounts using an SQL injection, and they modified for those sites the DNS pointers. They took wherever their registered DNS was, and then essentially overwrote them, pointing them to ns1. and ns2. and then a name I can't even pronounce. I won't try. It's y-u-m-u-r-t-a-k-a-b-u-g-u dotcom.

Leo: Yumurtakabugu.

Steve: There you go. And those then pointed people to this defaced page. So The Register, I saw a relieved note that The Register posted saying, well, as far as we know, we haven't been hacked at all. It's that our DNS got redirected. So, I mean, for UPS to have theirs redirected was an embarrassment and a problem.

Leo: No kidding.

Steve: But as soon as this got fixed, all that had to happen was that they just reset their DNS. And after it was able to propagate out through the Internet, people would have been returned to the proper website. So this wasn't a penetration of those sites. It was actually a penetration of their registrar's that unfortunately had a security problem. And we'll be hearing a lot more about security problems on servers here in a second.

I got my attention drawn, thanks to another Twitter posting, from Drew, who posts as @dochouse7, he turned me onto a very nice-looking Finnish IT security banking trojan detector. It's at FITSec.com, FITSec.com/blog. And at the moment it's the top blog entry there. It's a very small and nice detector of the top five banking trojans: Zeus, SpyEye, Carberp, Gozi, and Patcher. And what I like about it is that it's small. It's less than a meg. It's 768K. And it only takes about less than a minute to run because it's not rummaging through your hard drive. It's just doing an in-memory check of the processes running in your system now to see whether the banking trojan, any of those five banking trojans might be present in your system, that is, present and running. So it's very simple and quick to run, and I recommend it for Windows users. So FITSec.com/blog, which I hadn't known about before. So thank you, Drew.

And finally, this just sort of popped up on my radar. This was the first time I had seen a US-CERT/NIST vulnerability with a CVE index number, which directly related to something that was arguably health threatening. Reading from this CVE statement, it said, "An unspecified vulnerability in Medtronic Paradigm wireless insulin pump models 512, 522, 712, and 722..."

Leo: What? Geez, Louise.

Steve: I know. Listen to this. An unspecified vulnerability in this wireless insulin pump "allows remote attackers to modify the delivery of an insulin bolus dose and cause a denial of service," which then it says "(adverse human health outcome) via unspecified vectors involving wireless communications and knowledge of the device's serial number, as demonstrated by Jerome Radcliffe at the Black Hat USA conference in August. Note: The vendor [Medtronic] has disputed the severity of the issues, saying, 'We believe the risk of deliberate, malicious, or unauthorized manipulation of medical devices is extremely low.'" And we want it to be, of course. "'We strongly believe it would be very difficult for a third party to wirelessly tamper with your insulin pump.'"

Leo: I hope so.

Steve: Yeah. "'You would be able to detect tones on the insulin pump that weren't intentionally programmed and could intervene accordingly.'"

Leo: You'd hear it being reprogrammed, in other words.

Steve: I don't really know what they mean. But let's - I just - I thought as I was reading through anything to see if there was anything new that I ought to bring to people's attention, I thought, okay, wait a minute. So now we have a hack of a wireless insulin delivery system by wireless attackers. Now, the good news is they have to apparently know the device's serial number.

Leo: And the range on these isn't great. I mean, you couldn't sit outside the hospital and target these guys.

Steve: Or across the planet, yes. So that's good news. But still, we would like to have our insulin pump security a little higher than that. Okay. So from where we left off last week, which was that we knew from looking at the source code of Mozilla with Firefox and Google with Chrome that a bunch of certificates had been created without apparently the knowledge of this DigiNotar certificate authority in the Netherlands. Now what we know from the report which has been issued by Fox IT, who is the independent auditing firm that DigiNotar hired to come in and analyze their breach, is that a number of DigiNotar's servers were compromised by intruders who obtained administrative rights through the interval between June 17th and July 22nd.

So first of all, that's six weeks, essentially, of time during which bad guy or guys had access to the ability to produce fraudulent certificates signed by an authority that all of our browsers at the time trusted completely. So that's really bad. DigiNotar detected this intrusion on July 19th and said nothing to no one. Which is really the problem. And I will get back to that in a minute because in the Mozilla blog, the Mozilla security blog, this is one of the reasons they have issued a second update to Firefox is they're not pulling any punches any longer with DigiNotar. They did more of a surgical removal of trust, which I described in detail last week. Now it's just gone.

They're not - they want nothing to do with these folks any longer because of the way DigiNotar handled the breach. It's one thing to get breached. It can happen. But it's an entirely different thing if somebody who is trusted doesn't immediately belly up to the bar and take responsibility for what has happened. And these guys stayed silent. What we now know is that - are you sitting down? 534 fraudulent certificates were issued. It was just a party. And somewhere, if you scroll down the file I've given you, Leo, you'll find a CSV link there under GOVCERT.NL, you might want to open it up and find some choice ones while I continue to talk to our listeners.

Leo: Okay. Will do.

Steve: So you can mention those. So, okay. So the first known mention of any activity using these certificates was what we talked about last week, back in August, with a Google forum posting where an Iranian user reported being warned by the Google Chrome browser that there was something wrong with his certificate. Well, that was because it was revoked. And that was a certificate created on July 10th. So there was a revocation of the certificate by DigiNotar, who began to realize that something was wrong, yet even then they said nothing. Investigating further, Fox IT found that the offending certificate was revoked on August 29th. But by that point it had affected, it had directly been used in 300,000 unique IP addresses of users, 99 percent of which were in Iran.

Leo: But look at these: AOL.com, Comodo.com, Google.com, LogMeIn.com, Microsoft.com, Mozilla.org, Skype.com, Thawte.com, I mean, these are all - and the number of certificates, I mean, Equifax Root CA, 40 certificates.

Steve: It's a Who's Who of the Internet. And so these guys had weeks to synthesize fraudulent certificates, all signed by somebody we all trusted. And in two weeks we're going to talk about, we're going to revisit the issue of this whole hierarchical certificate authority trust model because the problems that we're seeing are finally beginning to come to the attention of the industry.

Leo: But this is my question. I see all these certificates - WordPress.com, Twitter.com. But so they've got a certificate. That means they could pose as this site.

Steve: Correct.

Leo: And how difficult - what would they do? How would they do that man-in-the-middle attack? How would they grab your Internet connection?

Steve: So all you need, the only requirement for leveraging these certificates is that you somehow are able to get into the traffic flow.

Leo: I click a link, in other words. Yes.

Steve: Well, you as the user, if you were in Iran, and you attempted to go over an SSL connection to Google, you are passing through Iranian-controlled ISPs. That's where the certificate is installed.

Leo: Ah.

Steve: So there's a proxy setup which sees that you are trying to get to Google.com, and it connects to your browser instead of just allowing the traffic to go through to Google.com. Your browser then receives a certificate from apparently Google.com signed by DigiNotar. And if, for example, you were running Certificate Patrol - that I am a fan of, and I've talked about now for many months. If you were running Certificate Patrol, it would pop up at that point and say, wait a minute, I last saw Google's certificate signed by VeriSign. Now I'm getting a certificate for Google signed by DigiNotar. Oh, and VeriSign's certificate still had two years to go. So it doesn't seem logical that they would have replaced their certificate.

Or actually Google signs their own. Google is a certificate authority. So it would say the last certificate was signed by Google. Now we're getting a Google certificate, apparently a Google certificate signed by somebody else, by this random DigiNotar certificate authority. So that would have alerted any really security-savvy user in Iran that something was going on here. Instead what happened was, while that certificate was in use and being delivered to users in Iran, DigiNotar revoked that certificate. The user's browser checked for revocation, saw that it was revoked. That's what popped up a notice that caused this user to make a forum posting, and that's where the world began to first know that something was going on that was not copacetic.

Okay. So we've now heard from the guy who did this. He said...

Leo: Well, the guy who takes credit for it. We can't prove he did it.

Steve: Well, and it's confirmed.

Leo: Oh, it is confirmed, okay.

Steve: Yes, because he used DigiNotar's private key to sign an executable, which he posted.

Leo: Okay. That proves it.

Steve: And only he - yes, exactly. So he posted to Pastebin. He said, and I'm going to share this because it's interesting, and there's lots of details in here. And we get some sense into his psyche:

"Hi again. I strike back again, huh? I told all that I can do it again."

Leo: The ego of these guys, god.

Steve: Oh, yeah. "I told all in interviews that I still have accesses in Comodo resellers. I told all I have access to most of CAs, you see that words now? You know, I have access to four more so HIGH profile CAs, which I can issue certificates from them, too, which I will. I won't name them. I also had access to StartCom CA. I hacked their server, too, with so sophisticated methods, he was lucky by being sitting in front of HSM for signing." And what he's referring to is that the StartCom CTO did actually see a hack in process and was able to stop it.

And so this guy, going on, says, "I will name just one more which I still have access: GlobalSign. Let me use these accesses and CAs. Later I'll talk about them, too. I won't talk so many detail for now, just I wanted to let the world know that ANYTHING you do will have consequences. ANYTHING your country did in past, you have to pay for it."

Leo: Oh, boy.

Steve: "I was sure if I issue those certificates for myself from a company, company will be closed and will not be able to issue certs anymore. Comodo was really, really lucky." Actually, Comodo responded immediately to the nine that this guy made. We were able to - the browsers were immediately updated to block those, and essentially this leak was plugged. DigiNotar, of course, the breach was vastly larger and longer. And they didn't take responsibility for it. So they are, essentially, out of business.

So he says, "I thought if I issue certs from Dutch Gov. CA, they'll lose a lot of money. But I remembered something, and I hacked DigiNotar without more thinking in anniversary of that mistake. When Dutch government exchanged 8,000 Muslim for 30 Dutch soldiers and Animal Serbian soldiers killed 8,000 Muslims in same day, Dutch government have to pay for it. Nothing is changed, just 16 years have been passed. Dutch government's $13 million which paid for DigiNotar will have to go DIRECTLY into trash. It's what I can do from KMs away," whatever that means. "It's enough for Dutch government for now to understand that one Muslim soldier worth 10,000 Dutch government."

Okay. And he says, "I'll talk technical details of hack later. I don't have time now. How I got access to six-layer network behind Internet servers of DigiNotar, how I found passwords, how I got system privilege in fully patched and up-to-date system, how I bypassed their nCipher Net Hardware Security, their hardware keys, their RSA Certificate Manager, their sixth-layer Internal "CERT NETWORK" which have no any connection to Internet, how I got full remote desktop connection when there was firewalls that blocked all ports except 80 and 443 and doesn't allow reverse or direct VNC connections, and more and more and more.

"After I explain, you'll understand how sophisticated attack it was. It will be a good hacking course for hackers like Anonymous and LulzSec." Then get a smiley face. "There was so many zero-day bugs, methods and skill shows. Have you ever heard of XUDA programming language which RSA Certificate Manager uses it? No. I heard of it in RSA Certificate Manager, and I learned programming in it in same night. It is so unusual, like greater-than sign in all programming languages is "<", but in XUDA it is "{". Anyway, I'll talk about DigiNotar later.

Leo: You don't have to read the whole thing. I mean, I get the gist of it. What do you think of this guy's skill?

Steve: Well, he does show one thing, which is disturbing. He says, "By the way, ask DigiNotar about this username/password combination." So he says, "Username is production\administrator, which is the domain administrator of certificate network." The password, he claims, is - it would be prodadmin, p-r-o-d-a-d-m-i-n, except that the "o" is changed to a zero, the "a" is changed to an at sign, and the "i" is changed to a one [pr0d@dm1n]. So if that was truly the password...

Leo: That's pathetic.

Steve: ...that this certificate authority that the whole world was trusting was using to protect their domain, their network domain, then that's very sad.

Leo: Yeah. It's not a good password.

Steve: Not a good password.

Leo: It's basically the login with the standard LEET replacements.

Steve: Exactly. So GlobalSign, because they were named in this first posting, immediately suspended the issuance of any further certificates. They posted in their blog, under what they called an "incident response," although at this point it's only a potential incidence response, they said: "On September 5th, 2011, the individual/group previously confirmed to have hacked several Comodo resellers, claimed responsibility for the recent DigiNotar hack. In his message posted on Pastebin," which is what I just read, "he also referred to having access to four further high-profile certificate authorities and named GlobalSign as one of the four. GlobalSign takes this claim very seriously and is currently investigating. As a responsible CA, we have decided to temporarily cease issuance of all certificates until the investigation is complete. We will post updates as frequently as possible. We apologize for any inconvenience."

And to their credit, they hired the same third party to come in, this Fox IT, since Fox IT now has internal confidential information about this attacker and how they were able to penetrate DigiNotar. GlobalSign said, okay, we're willing to and want to leverage what you've learned. Let us know if we actually have been hacked by this guy. So I salute them.

Leo: But again, does it strike you that this guy, I mean, these are a lot of boasts, typical kind of kid hacker boasts.

Steve: Yes.

Leo: Does it strike you that - it strikes me just looking at it, this guy might actually know what he's talking about.

Steve: I think he does. I think his English obviously is not his native language.

Leo: You can't judge it on that.

Steve: No. And if anyone's curious, I won't drag our listeners through his second posting, but he's made another posting, and we have the link in the show notes, where he gives additional details and essentially responds to, on a Q&A-like basis, to the other things which have been posted about him in the interim. So my take is, the fact that he did prove that he has the private key that DigiNotar used to sign their own certificate - he signed an EXE using it, which he posted - means that he's the guy. Or he got it from someone who is. But very likely he's the guy.

Leo: And there's a lot of bravado in this. You know, at the end he says, "Never forget, I'm just 21. You have to see much more from me." And there's a lot of boasting and bravado.

Steve: Right. But that's the nature, I mean, it's why...

Leo: It's pretty common. That's why people do this.

Steve: That's why we coined the term "script kiddie" was because, yeah, I mean, it's...

Leo: Well, this guy's more than a script kiddie, I would guess.

Steve: Yeah, he is. Well, depending upon what he - he obviously found a way in. Remember that years ago, Leo, I stumbled on just the number of root CAs which were in our browsers. And that's when I just said, okay, stop the presses. We're in trouble.

Leo: Right. So there's a vulnerability, obviously.

Steve: Yeah. This is trusting too many people. And there's no way...

Leo: And this guy sounds politically motivated. I think that's a little scary, too. I mean, he sounds like he might be a terrorist, in effect, a digital terrorist. He wants to be called "unstoppable genius digital hacker."

Steve: I don't think that's even a good acronym.

Leo: UGDH.

Steve: And then the question is, how did one of the certs that he made get put into service at a border for the Iranian network traffic? Because that...

Leo: And he addresses that. He says that people say this is a governmental hack. He says, no, it's just me, all by myself. No governmental hack involved.

Steve: Yeah, except that somehow the certs did, that one cert got used. And, I mean, so much so that 300,000 IPs of users, 99 percent of whom were in Iran, were having their traffic filtered through this bogus certificate. So the pieces don't still add up, but there's more pieces.

Leo: He gives a login and a password that he created on a Berkeley LMI.net server. He says, "Ask them about that." And he says, "I owned all their Linux boxes and got access to all their DNS servers. I am really sharp, powerful, dangerous, and smart." Really creepy, to be honest with you.

Steve: Yeah. So Mozilla, following up, said, "Earlier this week we revoked our trust in the DigiNotar certificate authority from all Mozilla software." Now, that's different from what we discussed last week, where, as I said, they made sort of a surgical, we're going to suspend trust in the following certificates, which we know have been compromised. Now they're saying, get a new copy of Mozilla stuff, we're pulling the plug. They said, "This is not a temporary suspension. This is a complete removal from our trusted root program. Complete revocation of trust is a decision we treat with careful consideration and employ as a last resort. Three central issues informed our decision:

"One: Failure to notify. DigiNotar detected and revoked some of the fraudulent certificates six weeks ago without notifying Mozilla. This is particularly troubling since some of the certificates were issued for our own addons.mozilla.org domain.

"Two: The scope of the breach remains unknown. While we were initially informed by Google that a fraudulent *.google.com certificate had been issued, DigiNotar eventually confirmed that more than 200 certificates had been issued against more than 20 different domains." Of course now we know it's way more than that. So, and this says, "We now know that the attackers also issued certificates from another of DigiNotar's intermediate certificates without proper logging. It is therefore impossible for us to know how many fraudulent certificates exist, or which sites are targeted.

"Three: The attack is not theoretical. We have received multiple reports of these certificates being used in the wild.

"Mozilla has a strong history of working with CAs to address shared technical challenges, as well as responding to and containing breaches when they do arise. In an incident earlier this year we worked with Comodo to block a set of mis-issued certificates that were detected, contained, and reported to us immediately. In DigiNotar's case, by contrast, we have no confidence that the problem had been contained. Furthermore, their failure to notify leaves us deeply concerned about our ability to protect our users from future breaches."

And then they talk about the subsidiary, which is Staat der Nederlanden Certificates, saying "DigiNotar issues certificates as part of the Dutch government's PKIoverheid," that is, the PKI government program. "These certificates are issued from a different DigiNotar-controlled intermediate, and chain up to the Dutch government CA," which is this Staat der Nederlanden. "The Dutch government's Computer Emergency Response Team (GovCERT) indicated that these certificates are issued independently of DigiNotar's other processes and that, in their assessment, these had not been compromised. The Dutch government therefore requested that we exempt these certificates from the removal of trust, which we agreed to do in our initial security update earlier this week." That is what I was talking about, about this surgical removal. That was the 6.0.1 update to Firefox.

"The Dutch government has since audited DigiNotar's performance and rescinded this assessment. We are now removing the exemption for these certificates, meaning that all DigiNotar certificates will be untrusted by Mozilla products. We understand that other browser vendors are making similar changes." Except Apple, apparently. "We're also working with our Dutch localizers and the Bits of Freedom group in the Netherlands to contact individual site operators using affected certificates. The integrity of the SSL system cannot be maintained in secrecy. Incidents like this one demonstrate the need for active, immediate, and comprehensive communication between CAs and software vendors to keep our collective users safe online." Signed by Johnathan Nightingale, Director of Firefox Engineering.

So this has been a significant event in the history of the way the Internet works because of the nature of this trust model which we have for SSL. All over the world are authorities that we are trusting to issue and sign certificates which web servers hold in order to assert their identity to we users when we connect to them. Our browsers then trust those signers of those web server certificates. If that trust is broken, then our recourse is to stop trusting anything they do.

The problem, of course, is all the people who legitimately purchased DigiNotar certificates are no longer trusted by Mozilla, and increasingly by any other browsers because they bought their certificate from somebody who turned out to be untrustworthy and, sadly, who didn't accept responsibility for this immediately, which is what has really shaken people. And basically this company is probably gone. They're now out of business.

Microsoft's emergency update, their out-of-cycle update, did exactly the same thing. This is advisory 2607712, where Microsoft said in their very sort of generic bland way, "Microsoft is continuing to investigate this issue. Based on preliminary investigation, Microsoft is providing an update for all supported releases of Microsoft Windows that revokes the trust of the following DigiNotar root certificates by placing them into the Microsoft Untrusted Certificate Store." And so this is the DigiNotar Root CA, DigiNotar Root CA G2, DigiNotar PKIoverheid CA and two others, basically those ones I just mentioned where the Dutch government said, okay, it looks like we can't keep trust even on those. So I imagine that Windows users with the currently supported security platforms will have received that update, and that's something that you will want to apply. And you read through a bunch of those. There was a *.*.com. So, I mean...

Leo: Wow, that's everything.

Steve: That's pretty much everything - *.*.org, *.android.com, I mean, it's just - it's everybody, essentially, 500-plus certificates. So the good news is anybody who has received the update from Microsoft and rebooted Windows, anybody who's updated to Firefox 6.0.2 on whatever platform and Chrome are all safe. Inexplicably, Safari users are not, unless you're on Windows. But over on the Mac, I just can't understand why we've heard nothing from Apple. But that's the case.

Leo: Do we know how many - I guess we couldn't possibly know how many attacks are based on this.

Steve: The EFF has this SSL Observatory which is used to track the use of certificates. And that's what the Mozilla people have used in order to give them some handle on the use of certificates. So that has been queried a lot in order to track the use of these certificates. But remember, to use them you have to arrange to have them delivered to a user's browser, rather than the website, the legitimate website that the user's browser is trying to connect to.

So it is all about man-in-the-middle attacks. It's about intercepting traffic. So you have to be in a place where you can intercept traffic going in both directions. And that's pretty much an ISP, either at the user's local level or at the nation-state border level. That's how these certs end up being used. I really think the guy went overboard to issue himself so many certificates. One wonders how many...

Leo: Really? He went overboard? What a shock.

Steve: One wonders how long this could have been, I mean, the only way this is valuable is if the CA that signed the fraudulent certificate continues to be trusted by the browsers. So you really want to just sort of sneak in and do this without being detected, ideally.

Leo: Well, it's pretty clear that he's self-aggrandizing, and he wants the attention. The good news, probably, is that he's not doing this - maybe he is - to hack. Now, it's also possible, and Eon pointed this out in the chatroom, that this whole Pastebin manifesto could be disinformation. It could be a government in fact doing this, but wants people to think it's a kind of cracked up individual.

Steve: That's a very, very good point. We don't know otherwise.

Leo: We have no idea.

Steve: And we do know that somehow 300,000 Iranian users, or 300,000 IPs inside of Iran did have one of these certificates used to allow someone to essentially hack into all of their Google accounts, which is what this meant, because...

Leo: Which the government might well have wanted to do, looking for dissidents.

Steve: Yeah. And remember that the way Google maintains login is that there's a cookie which any attacker who had this certificate and a man-in-the-middle position on the traffic flow, they would have grabbed their Google login cookie and then been able to impersonate them across any Google services that that user was using. So that's a very powerful hack.

Leo: None of the people that he gives out passwords to have admitted that this, well, I guess, I mean, we know that, because he used a key, a private key, that he had access to this data.

Steve: Yeah.

Leo: So we know that this is a legitimate Pastebin posting. We just don't know if it's disinformation or misdirection or really...

Steve: Exactly. We don't know who the individual is, if it is an individual. But we know that whoever posted this has proven that they were inside of DigiNotar.

Leo: Right. And as Zephyr I think posted in our chatroom, the guy doesn't really matter. What matters is what the hack was, how it happened, the failure to protect us, all of that, much more so than who did it.

Steve: Yeah. And again, I got a tweet from someone who communicates with me through Twitter, Walid Damouny, who tweeted...

Leo: We've talked about him before, yeah.

Steve: Yes. He tweeted from Lebanon, saying, "The question on my mind is how trustworthy are other CAs. Like DigiNotar, other CAs are not uncompromisable." And my thought is, well, it's worse than that because, for example, in the United States we have now - we know that we have legislation, the whole Patriot Act stuff, where our government is able to issue orders for corporate entities to behave as the government wants them to and to prevent them from saying anything. So who's to say that VeriSign, that Network Solutions, that GoDaddy, that other trusted certificate authorities haven't issued certificates on behalf of our government or other governments.

I mean, again, I've said, I've poked fun at the Hong Kong Post Office. But the fundamental problem here is that our browsers are trusting a huge network of third parties. And the nature of this architecture is every single one of them must be perfect in order for the system to work. Well, that is a bad model. So in two weeks we're going to revisit this. We're going to look at the Certificate Authority Trust Model: Past, Present, and Future because people are beginning to talk about alternative ways of setting things up.

Leo: Yeah. And in fact that's what this kid says, if this is a real kid, in his post is this breaks SSL. This breaks the Internet. And if I could do this, then you really have to - and he also, of course, asserts he's got a perfect system that would work, but he's not going to share it with us. Okay.

Steve: Okay. So in other news, I did want to update our listeners about my Off The Grid project. Last week you'll remember that I mentioned I was going to be working on what I called an ultra-high-entropy pseudorandom number generator, the reason being that there are so many Latin Squares possible, I want this system which is online to be able to get to them all, potentially. And if we only use a standard pseudorandom number generator, there just isn't enough bits of state in the pseudorandom number generator to allow it to access all of those Latin Squares.

So I did write a pseudorandom number generator in JavaScript, which is much faster than what I had before. It has 1,536 bits of entropy. And it has just successfully passed multiple batteries of independent testing by the denizens of the GRC newsgroup, over in the GRC.thinktank newsgroup. They ran it through the Fermilab, the Diehard, and the Dieharder tests, and it passed with flying colors. In some cases they were giving it multiple gigabytes of - that is, giving these tests multiple gigabytes of data generated by this pseudorandom number generator, and it's passed. So we now have a very solid new high-entropy pseudorandom number generator which I'll be bringing online. There is a new link underneath the block of links on the Off The Grid pages, if anyone's curious, to a page where that new ultra-high-entropy pseudorandom number generator is located. And of course it's free for anyone to use who has an application for it.

Leo: Not easy to say "ultra-high-entropy pseudorandom number generator."

Steve: No. I've been practicing that.

Leo: I think an acronym is called for here.

Steve: And I did also want to mention the cool little embedded ARM Cortex-M3 board, which was sold out last week when we talked. They got 50 in. They sold them out immediately. They got 50 in again. And let's see if there's - last time I refreshed the page there were four left from the second batch of 50. Oh, now there's three at this point. So if anyone listening live wants one, it's LPCTools.com, and it's the LPC1769 LPCXpresso board, and there's three left. And I imagine they'll get some more when those are gone. And I have to say I'm tickled that our listeners are hardware hackers to the degree that they are. I'm sure we'll have some fun together playing with that, as soon as I'm able to spend some time over there.

Leo: Pretty cool. And by the way, they now say "As talked about on Steve Gibson's Security Now!, Episode 315." So actually there is much deeper discussion of what you can do with these on 315.

Steve: When they were sold out, they said, "Well, we're sold out again, thanks to Steve Gibson. So we'll have more in soon."

Leo: We've really created a run on the Xpresso board. That's great.

Steve: Yeah, well, it's very cool. And for $29.95 it's just - it's a beautiful little thing with all the software being free. I wanted to share a note I received on August 31st, so about a week and a half ago, from a Terry J. LeBlanc, who shared his SpinRite success. He said, "Thanks, Steve. I haven't worked with SpinRite in quite a few years. Then I needed to run chkdsk on my Dell OptiPlex 755 over the weekend, and chkdsk would hang at 36 percent during Phase 5, checking the available free space. No matter how long I let it run, chkdsk would not continue. It would not finish.

"So I thought of SpinRite, accessed your website, bought it, and downloaded it; created the ISO image; burned a bootable CD. I booted from the CD and got an opcode error. Researching the error on the Internet, I found references to others with the OptiPlex 755 boxes, trying to run SpinRite, getting that error that indicated that changing the BIOS from RAID-AHCI to RAID-ATA would eliminate the opcode error. So I changed the BIOS setting and restarted from the CD.

"Sure enough, SpinRite came right up with the familiar DOS interface I had not seen in years. SpinRite ran successfully at Level 2. I restarted the workstation, ran chkdsk successfully, and booted up normally. Problem solved. I'm using it right now. SpinRite and the other tools from GRC have helped me so many times over the years. I'm happy that you're still around and still making these tools available. I'm happy to have SpinRite back in my toolbox. Great job, folks. Keep up the good work. Thanks, Terry LeBlanc."

Leo: That's nice. So did you know that little bug, well, it has to run in ATAPI mode, not AHCI? Is that...

Steve: Yeah, what's happened is over the years bugs have crawled into BIOSes, and they're not being found because there are not that many programs any longer like SpinRite that depend upon the BIOS. So Greg has a whole toolkit of available things to try when people run across a problem. And so he could have written to Greg, and Greg would have told him the same thing. Or he Googled and found other people who had solved the problem. So that was perfect.

Leo: Apparently KD5AMB in our chatroom says that, oh, that fixes my problem, too. Well, a lot of people with those Dell motherboards, that's for sure. All right, Steverino. Let's talk about - now, usually we say TCP IP, TCP/IP. What is that? What is TCP, Terminal Control Protocol?

Steve: Transmission Control Protocol.

Leo: Transmission Control Protocol/Internet Protocol. How does TCP relate to IP? Why do we bundle them together like that?

Steve: Yeah, it's interesting. I remember in the beginning sort of being curious about the same thing. It's like, okay, well, what about UDP/IP and ICMP/IP?

Leo: Right, right.

Steve: And I think that is really - that demonstrates what I was saying earlier, which is the importance of TCP. TCP is a peer, really an equal peer with ICMP, UDP, and other protocols that are all riding on top of the IP protocol. So it's called the TCP/IP stack traditionally. And we talk about TCP/IP. But in fact, I don't know why, but ICMP, UDP and the other guys just don't get equal billing.

But so the idea is that, as we originally talked about, the Internet exists as a confederation of interconnected routers where routers connect users and services and servers to each other through links that individual packets move around. We talked a couple weeks ago about the UDP protocol because it's very simple. Essentially the IP protocol thinks in terms of IP addresses, and then that carries other protocols sort of on top of it, UDP being one.

UDP adds the notion of ports. So you can have a source port and a destination port, each which are 16 bits. So that's where you get the 65535 ports because zero is not a valid port number. So one through 65535 are the number of combinations of 16 bits. So UDP just adds that to IP. So you have a port number, a source port number, and a source IP; a destination port number and a destination IP; and then whatever payload the UDP packet wants to carry with it.

The problem with UDP is that that's all it is. I mean, that's all it does. So a sender who puts a packet onto the Internet, aimed at a destination IP, doesn't know what's going to happen. As we've talked about, the genius of the architecture of the Internet was that routers made their best effort to forward packets in the direction of their destination, but that's all. And if their queues got too full, if too many packets were, like, coming in from four different places and all trying to go out through one, you could imagine that there would be a congestion there.

Well, the routers have the right by agreement to just discard packets that they don't have time to send, or for whatever reason they choose not to. The router might receive the packet and then crash. Routers could go down. Links could drop. So all kinds of things could happen that could cause a packet not to arrive at its destination. The other thing that can happen is that packets could arrive out of sequence, that is, because routers have, again, the right to choose which links they send packets from or forward them to, one link might be busy. So the router says, oh, I've got another link here. So it would send a second packet down a different link. Well, that one might have a shorter path to its destination so that it arrives before one that was actually sent prior to it, which is to say that the order of the packets arriving is not guaranteed the way the Internet was designed. So there's a problem with reliability and with out-of-orderness.

So the designers of the Internet said, okay, we've created this wacky kind of strange foundation, packet-switched networking. But what happens when applications just want to send a file? You want all the pieces of the file to arrive. You need them to arrive somehow and be reassembled in the right order. You don't want pieces of your file scrambled around. So we need, on top of this admittedly, and even by design, sort of rickety foundation, we need to take responsibility for its ricketyness.

And so that's what they did with TCP. And that's the brilliance of this is because that foundation was assumed to be error prone just by its nature. Even when it's working perfectly, it might not deliver packets at all, or they might deliver them out of sequence. So, I mean, that's part of the genius is that they then said, okay, now we have to design around those problems. So we need to somehow impose order from chaos.

So they said, first of all, we're going to create an abstraction called a "connection." Remember, traditionally, a connection was actual wires connected between here and there. And you had your own pair of wires connected. That's the way the phone system used to work is, if any of our listeners ever remember, like, pictures of the old phone-switching rooms, they were huge banks of relays that, as you dialed, stepped up and stepped over to take your pair of wires and connect them to another pair, which then stepped up and over to another pair, and so forth. So you're actually connecting your wires to somebody else's wires.

The Internet broke that model with this concept of packets of data. But for an application, for example for a web browser, a web browser doesn't - it's not its job to worry about the vagaries of packets. It simply wants to view the Internet as a reliable connection between it and a server, where it's able to send something and know that it's going to get there. And the receiver of a query, for example, sends a reply, and it knows it's going to get there.

So what we needed to build was we needed to build a layer on top of IP, with all of its known and designed-in problems, to essentially insulate anything running above from all of the problems of what goes on below. So this notion of connecting endpoints was what was created, a pure intellectual abstraction. The way this was done is that every byte which is sent is numbered. They're counted. And the sender numbers the bytes, and the recipient acknowledges the bytes that are received. So if packets are lost along the way, the recipient won't acknowledge those.

What happens is the acknowledgment is the highest numbered packet received in order. So if packets did come in out of order, then the recipient would hold that packet, assuming that the missing one might be arriving here any second to fill in a gap. In which case the recipient would acknowledge everything it had received up to that point. But if it just sits there waiting for a gap to be filled in, which isn't, it will not acknowledge. And at some point the sender notices that it's only got acknowledgment up to a certain point. So it thinks, hmm, okay, enough time has gone by, and I haven't heard from the other end, so I'm going to resend this again.

So one of the things that TCP does is it autonomously resends lost data in order to make up for that loss. What that means is it has to buffer down sort of in its layer, it needs to buffer all outbound data until it has been acknowledged by the other end. Which is to say that the application running above sends data down to TCP to be sent and then doesn't give it a second thought. The application is able to trust that TCP will achieve its goals of guaranteeing that it gets sent. The TCP protocol layer thereby receives the data and holds it until it has been affirmatively acknowledged by the other end, at which point it's able to let go of that data because it knows that it's been sent. It will not have to resend it again.

So how does all this happen? The first thing occurs when a connection initiator wants to connect to a connection receiver. Typically, for example, in the model we're all familiar with, we're sitting in front of a computer using a web browser. And so we want to initiate a connection to a server, and that server is listening for incoming TCP connections from anywhere in the world. The Internet is global in the typical case. Anyone, anywhere in the world, can initiate a connection to a server anywhere in the world, barring nation-state firewalls and Chinese firewalls and so forth. In general it's just an open global network.

So the connection initiator sends what's called a synchronize packet, or it's also called a SYN, S-Y-N, the first three letters of synchronize. The initiator sends a SYN packet to the connection recipient, the one who is open, waiting for incoming connections. What that packet does is it contains a declaration of the endpoint. If we're going to have a connection, we need to have endpoints. So the endpoint is identified by the one endpoint, the initiating endpoint, by the source IP and source port; and the other endpoint is identified by the destination IP and destination port. IPs are 32 bits; ports are 16. So together that's 48 for each endpoint, and you merge those, and you get 96 bits. So essentially it's the uniqueness of a local port and IP and a remote port and IP is the identity of a connection. And it's one of the reasons we have ports on our computers is we might want to have multiple connections to the same location.

For example, my browser might want to open three or four different simultaneous connections to a single remote location, like to Google.com. So we would always be connecting to Google's IP at Google's port for services, like port 80. That would be constant. But on our local end we would always be using our own IP, but we would then - we would disambiguate the traffic by using four different local ports. So the ports being different allows this four-tuple, containing source port, source IP, remote port, remote IP, they all have to match up in order to identify a connection. We use four different local ports to essentially create four different connection abstractions.

So this SYN packet is the connection initiating packet. It contains the local port, the source port that we will be connecting from, and our IP, and inherently contains the remote port, which would be typically port 80 if we were connecting to a web server. It might be 110 if we were connecting to a remote POP server, or 143 for a remote IMAP server, and so forth. 25 for remote SMTP server. Those are the so-called well-known port numbers. The idea being that a remote IP could have different services listening on different ports for incoming TCP connections. And each of those ports implies a different service. SMTP, listening on port 80, implies the use of SMTP protocol over the TCP connection, where the SMTP protocol is layered on top of TCP.

So the SYN packet, the most important thing the SYN packet has, in addition to declaring this is the source port and source IP and destination port and destination IP, is a sequence number. Thus the word "sequence" and SYN, which is to say it declares to the recipient, I'm going to start numbering all of the bytes that I'm going to be sending you from the following count. And that's a 32-bit count. So that's - we know that 32 bits is 4 gigs. So the sender says just make a note on the receiving end that as I start sending you things, I'm going to be counting the bytes I'm sending you, starting from this number.

So the server receives the SYN. And if there's something listening on that port number, like on port 80, and it's willing to accept a connection and ready to do so, it will reply with a SYN ACK. This is a synchronize and acknowledgment. It's actually, it's like a two-part packet. The SYN part is its own numbering scheme, that is, it is telling the connection originator that, okay, I've received your SYN, and I've made note of how you're going to be numbering your bytes. I'm assuming you're going to want something back from me because a TCP connection, although it doesn't have to be, it is inherently a full duplex connection, meaning that each end is able to, at any time, at will, send data to the other end.

The connection can be brief. It could be connected. Something is sent, and it could be dropped. Something can be sent and received and then dropped. Or it can be persistent, and it can last as long as both ends agree that they have a connection because - and this is sort of a cool thing about TCP. The reason I say this is an abstraction of a connection is it's just their agreement that creates this. It's a virtual connection. The endpoints are connected only if and because they agree, they both agree that they are.

So the server side sends back this SYN ACK packet. The SYN portion, as I said, is its corresponding starting numbering of the bytes, of the data bytes it's going to be sending, and the ACK is its acknowledgment of the receipt of the initiator's SYN. Upon receiving the SYN ACK packet, the connection initiator sends its final packet, acknowledging the receipt of the server's SYN side. So really there's a SYN and an ACK, a declaration and an acknowledgment, a SYN and an ACK, which passes in between both endpoints in each direction. And because they're overlapped, we only need three packets to make that happen. And so that's called the TCP three-way handshake because the endpoints are establishing each other and the numbering for the bytes which follow.

Well, several important things are also going on because we know how crazy the Internet is with routers and sort of this very ad hoc means of routing packets. It's conceivable, with the way routers are set up, that traffic flowing from endpoint A to endpoint B will flow over a given path, jumping between routers to get there, and that the reverse traffic coming from endpoint B back to endpoint A, nothing says it has to go the same route. It could, for whatever reason, there might be a faster link in a different direction or who knows what. But nothing guarantees that, like, data in each direction is going to be passing each other over the same link. They could easily be going on different routes. But it also says that there may not be a route back. There may not be any way for traffic to get back, just because of something being broken.

So the beauty of this TCP three-way handshake is that it validates to each endpoint that the other is able to receive its traffic. When the sender sends its SYN packet, the initiator, the connection initiator sends its SYN packet and receives a SYN ACK from the server, it knows the server got its request to open a connection, and it knows that it's able to receive traffic back from the server. Similarly, when the final ACK packet is sent by the connection initiator, the server knows - of course it knows that it originally got the SYN packet from the connection requestor. But it sent off a SYN ACK. It doesn't know, unless it gets that final ACK of its SYN back, that the connection initiator was able to receive its traffic. So getting that final acknowledgment back does verify that.

So now each endpoint, they've found each other; they've verified that they're able to exchange traffic back and forth; and, importantly, they've each declared the starting point for the numbering of their bytes. Now, why don't they just use zero, you might ask? Like, okay, what's with this 32-bit counter? Why not just always start numbering from zero? The reason is, again, the craziness of the Internet. As I mentioned, connections might be persistent or might be short-lived. And we might be connecting between the same ports, that is, the same two endpoints, over and over and over. It might be that we grab a local port, we connect to a remote web server, make a query, receive a response, drop our connection. But then we decide, oh, we've got something else we want to ask it. So we grab the same endpoint and connect again and so forth.

The problem is, if we always started numbering our data from zero, it's possible that data from a previous connection could be confused with data from this connection. That is, if we always started at zero, because the Internet is as ad hoc and prone to crazy behavior as it is, for example, say that a packet was delayed. And so, because of the delay, the receiver that was waiting for the packet got tired of waiting and sent out the data again. So we know that there can be duplication of data on the Internet in addition to data being lost. So what could conceivably happen is a short-lived connection is brought up and then torn down and then brought up again. Yet, due to the vagaries of packet delay, it's conceivable that old traffic from the prior connection could be confused with new traffic from a reconnection.

So, smart as these guys were who originally designed this, they recognized that problem. And they said, okay, let's agree that we're going to use a 32-bit count. And they just used that so that they had plenty of margin, especially compared to speeds before. And the idea is that, as we send data to the other side, we're going to be advancing this counter with the number of bytes sent in each packet so that the sequence number continues to advance in an upward-pointing direction. And if it hits the 32-bit limit, it'll just be allowed to wrap around back to zero in the same way that binary numbers wrap around when they get maxed out. And that's fine.

What we want to make sure, though, is that the number is big enough, that is, there are so many combinations, binary combinations of this sequentially numbered byte ordering that it's just inconceivable that traffic would be still floating around the Internet long enough that it could ever be confused with earlier traffic that just hadn't died during a connection. And so they said, okay, let's just make it four bytes. We'll make it 32 bits of traffic. That's four billion bytes of traffic. Certainly that'll give us enough headroom. And it turns out that it did. That has not turned out to be a problem.

And so after the SYN, and SYN ACK, and the ACK packets, then this full duplex connection is established, and either end is able to send the other data over this virtual connection whenever it wants to. It looks at the amount of data it's sending and advances its sequence number that it's sending in that packet to point to the last, the number of the last byte that it is sending, and it puts that packet on the Internet, and routers send it towards its destination. And it then takes another packet and adds data to that, advances that same sequence number again, up by the amount of data that it has put into this next packet, so that the last byte of the packet is the number which it is declaring as part of this packet, and sends it off.

The recipient of that data has the job of acknowledging the, as I said earlier, the last-in-order byte that it has reassembled of that sender's data. So if packets are lost, it has no way of knowing that they're lost. The recipient doesn't know that more data is being sent because there's never a declaration ahead of time at the TCP level of how much data is going to be sent. It simply receives it as it comes in. And it looks at how much data is in the packet and then moves this numbering forward as far as the amount of data in the packet would indicate, and acknowledges back to the sender that it has received up to this much data correctly. So if it gets a packet out of order, it waits to see if the other one is going to come in.

Whether or not it does, because it's TCP, it knows that if the sender does not receive an affirmative acknowledgment within a certain expiring length of time, the sender has buffered that data and is able to send it again. And it will send it again and again and again, trying over and over and over, until that data finally gets through, and the recipient of that data has acknowledged the last byte which has been sent.

All of these TCP packets contain both a synchronized field and an acknowledgment field. And the beauty of that is that, this being a full duplex connection, they are able to simultaneously send data in each direction. And the acknowledgment field of the packets going in each direction is always acknowledging how much in-order data has been received up to that point, while the SYN field is always setting the expected byte numbering of the data in the packet.

So this flow in each direction is having a dual role. It is able to send new data to the other side, numbering that as expected. And the same packet contains an acknowledgment field of any data that's been received. And so that's in a nutshell the way endpoints identify themselves; how ports are "open," which is the term we use for one that is accepting TCP connections; how connections are established and how data is numbered and how duplicate data is ignored; how lost data is retransmitted if that occurs in order to guarantee the receipt; and ultimately how each end is able to reassemble a coherent block of data that the sender has sent. And this is all the responsibility of the TCP protocol. The application, the web browser, the email client, the web server, the email server, whatever the application level is, it knows it's able to trust TCP to get the job done.

Next week, or actually in a few weeks from now when we pick this up again, because we're going to talk a little bit in two weeks, after next week's Q&A, we're going to talk about certificate authorities, as we said, looking again at the whole CA structure. When we come back to TCP, which we will in our next available slot, I'm going to talk about the hacks which have been created as a consequence of this. We've talked about denial of service, SYN floods, and the problems those create. And then we'll look at what was done in order to make this all go faster because, as the Internet has gotten bigger in terms of its diameter, the number of router hops, bandwidth has been strained, and it's necessary to come up with ways of allowing the sender to know how much data that has not yet been acknowledged it's able to send ahead. And that allows it to essentially not have to wait for everything it sends to get acknowledged. So there's lots more cool stuff to talk about in the way the TCP protocol operates. And we'll pick up where we left off.

Leo: It's very elegant, very straightforward, and it gets the job done, as we can see.

Steve: It works. It works.

Leo: It works, yeah.

Steve: We're using it all the time.

Leo: Yeah. Well, Steve, this is fun. So next week we do Q&A, which means if you want to ask a question of Steve, you can go to GRC.com/feedback, and we'll pick 10 good questions. Steve will answer those. You know, we've been doing, Steve's been doing all along with Security Now! these go-deep explanations of how things work - how a computer works, how the Internet works. So if you like this, and you want to know more, I mean, you've got 316 other episodes to listen to. And if you look at the name, it'll be very obvious where the ones where we explain, where he - I say "we" - "he" explains how stuff works. So the chatroom gives you a standing ovation, by the way, Steve.

Steve: Oh, no kidding.

Leo: They enjoyed this quite a bit, yeah.

Steve: Neat.

Leo: So you'll find all the shows at GRC.com, along with SpinRite, Steve's bread and butter and the world's best hard drive maintenance utility, his free stuff like Off The Grid and the Perfect Paper Passwords, all the things that Steve does such a great job with. Do read the Password Haystacks article. That's a great one that will change the way you do passwords. And of course lots of free software like Wizmo and - I guess nobody really needs Shoot The Messenger or DCOMbobulator anymore, Unplug n' Pray...

Steve: Not so much. They're still there for old-timers.

Leo: Trouble In Paradise. If anybody has a ZIP disk, it's clicking - hey, it's possible. Somebody finds some ZIP disks in the attic and says I wonder what's on there, and gets the Click of Death? You know, things like that.

Steve: Yup.

Leo: So GRC.com. He also puts 16KB versions there for people who don't have a lot of bandwidth and full transcripts, too, if you like to read along, which is frankly a great way to follow along what Steve's saying. I know I've talked to many people who actually use Security Now! in the classroom. Lot of college courses, Security Now! is required listening. So the transcripts are useful for those folks, as well. GRC.com. Of course we have it all, too, at TWiT.tv.

Steve, thanks so much for joining us, and we'll be back. We flipped days again for Paul Thurrott. Next week we will be back on our Wednesday, regular Wednesday schedule. That's 11:00 a.m. Pacific, 2:00 p.m. Eastern, 1800 UTC at live - I keep saying "live.twit.tv." Don't need that anymore. Just our new site now, it's right there on the front page, TWiT.tv. Thanks, Steve. We'll see you next time...

Steve: Thanks, Leo.

Leo: ...on Security Now!.