SECURITY NOW!

Transcript of Episode #179

Cracking Security Certificates

Description: Steve and Leo delve into the detailed inner workings of security certificates upon which the Internet depends for establishing the identity of users, websites, and other remote entities. After establishing how certificates perform these functions, Steve describes how a team of security researchers successfully cracked this "uncrackable" security to create fraudulent identifications.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-179.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-179-lq.mp3

INTRO: Netcasts you love, from people you trust. This is TWiT.

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

This is Security Now! with Steve Gibson, Episode 179 for January 15, 2009: Cracking Security Certificates. This show is brought to you by listeners like you and your contributions. We couldn't do it without you. Thanks so much.

It's time for Security Now!, the show where we cover all things secure. Like your security blanket, your binky? No, no, no, Internet security, online security, privacy. And here he is, the man who knows it all, Mr. Steve Gibson. And if he doesn't know it, he'll find out.

Steve Gibson: I don't think I know what a binky is. So right off the bat I don't think I'm quite up to speed on this, Leo.

Leo: You don't have kids. If you had kids, you'd know a binky is one of those little pacifiers that kids suck to make them feel secure.

Steve: Oh. Well, you're right.

Leo: But you're...

Steve: I hope your kids, which are now teenagers...

Leo: No longer, no.

Steve: ...no longer are binky enabled.

Leo: No, no binkies anymore. So it's good to talk to you. You know, since we talked last I've been exhausted because I did Macworld Expo, and then of course CES. And this is traditionally the week of the year that most of the tech industry collapses in a puddle.

Steve: Well, you did CES by remote control.

Leo: I know.

Steve: The way most people want to do it, is you basically interviewed people who had actually been there on the floor.

Leo: It was so much easier.

Steve: Yeah. And it was neat that you had LeVar Burton on. That's, you know, Geordi La Forge from "Star Trek: Next Generation."

Leo: Oh, man. You know, I was a little reluctant because I don't want to put celebrities on just because they're celebrities. But LeVar is a complete geek. He knew, you know, he was completely up to date on all that stuff. I told him, LeVar, you could come on as many times, any time you want. He says yes. And we have some interesting ideas of things we might want to do with LeVar. So, yeah, he's great. Wasn't he fun? He was just really a neat guy.

Steve: Really a good show, yeah.

Leo: Thank you. So what are we covering today on the program?

Steve: Today the title of the show is Cracking Security Certificates. As I promised, I want to explain, really for the first time ever. We've talked about security certificates in passing many times. But I've never gone through the process sort of step by step of what someone does to get one, how they're created, how they're signed, what it really is that a security certificate does for us. So I want to really lay a foundation of understanding of how they work. And then we're going to talk about some detailed operations of the MD5 hash, which is damaged to this point beyond recovery, and how these incredibly clever security researchers used this broken MD5 hash to create their own fraudulent certificate. So this will be high up on the propellerhead scale, but I think very accessible and really interesting for our listeners.

Leo: Security certificates are really a fundamental technology on the Internet. I mean, there is no...

Steve: Oh, yeah. I mean, more so all the time. They give us both the ability to encrypt our traffic; but, more importantly, our browser is making sure that the certificate it is given by a website when it connects to it over a channel that it wants to be secure, there's an authentication process that happens invisibly underneath the - sort of like behind the scenes so that it's easy to take for granted. But that's always happening. And sometimes users may see their browser note a problem. For example, it's surprising, some sites forget about the expiration of their certificates. And you'll connect to a site, and you'll get a pop-up notifying you that this security certificate of whatever site you're trying to go to has expired. And you can typically look at the certificate, and you may notice that it, like, expired yesterday.

And so you can imagine that they're getting lots of people complaining that, hey, did you realize your server certificate has expired? And in fact they're probably already scrambling, trying to get a new certificate reissued while this notice is popping up and, like, no one's trusting their site. So there's this mechanism going on in the background which we all depend upon to an increasing degree because of course we depend upon the Internet and the Internet's security to an increasing degree.

Leo: Yeah, yeah. Microsoft, I remember when Microsoft wanted to use certificates for security for ActiveX that it was a little bit controversial.

Steve: Well, yeah, because you need to pay somebody in order to get one. The idea is that - and we're going to discuss this in detail. There is an implied trust at some level within the certificate structure. That is, you know, for example, VeriSign is generally the company I get mine from, although I used GoDaddy this summer when I was messing around with a wildcard certificate because they were so much less expensive. But the point is none of them are free because, in return for getting the certificate, you have to go through some procedure to prove you are who you say you are. They are supposed to, in turn, verify that information, make sure that this is coming from Gibson Research Corporation. I mean, they check our Dun & Bradstreet number, they place phone calls to phone numbers that are known to be associated with Gibson Research that come from sources other than us saying, oh, here, call this number. So they independently verify as much information as they can, really as a function of how secure you want this to be. So they want to get paid in return for doing all that. And that's sort of the ecosystem that we have for certificates.

Now, people have been upset by that because they've said, hey, wait a minute, why do I have to get my ActiveX control signed? That means I have to pay someone in order to be able to sign my ActiveX controls. And in fact, for example, Microsoft calls this Authenticode, and I now have an Authenticode certificate. I need to purchase it every, I think it's every two or three years. I just checked recently because I'll be doing some work with signed EXEs in the future. But, for example, the most recent little piece of freeware I did, Securable, is a signed executable. And so it prevents some of the pop-ups that you get, for example, under Vista when you try to run something that you downloaded from the Internet. It's able, instead of saying we don't know who this software is from, good luck - actually it says do you want to run this anyway. Instead it'll pop up and say, hey, this is from Gibson Research Corporation. Do you want to run this? But so the environment, Windows, the operating system, the Mac, whatever, is able to assert who this software is from because the executable itself contains a certificate that binds our identity information to the executable.

Leo: You know, I also think that maybe some people don't really trust this chain of trust notion so much.

Steve: Well, exactly. I mean, remember, I mean, I was the one who first, when I looked several years ago at the number of root authorities in my browser - I hadn't looked for years. And there used to be a handful. And in the meantime there had been this explosion of root authorities. And my immediate concern was, whoa...

Leo: Who are these people? Who is this Hong Kong Post Office?

Steve: Yeah. The more you have - essentially what that means is that any certificates signed by any of these people will implicitly be trusted by our browser. So no warning, no pop-up, no notification, nothing.

Leo: And that was the issue.

Steve: The browser just goes right on ahead without - assuming that everything is legitimate. And so in the case of this cracked certificate that we talked about a couple weeks ago, it was a valid, solid certificate authority that was issuing the certificates. There were a number of things that they were doing that were insecure, even aside from the fact that they were signing their certificate with MD5, which is what we'll talk about. But inherently, just the law of numbers, the more people you're trusting, the greater chance there is for one of them to be mistrustful.

Leo: Right, right. Okay. Well, we're going to talk about that, how it works and what's been going on with these cracking techniques. Before we do that, any news in the world of security? Errata from last week?

Steve: Oh, we've always got some news because we've always got Microsoft and Windows in the picture.

Leo: There's always something.

Steve: Always something happening. So we are past our second Tuesday. That was a few days ago. And there was a January release, a small one actually in terms of number of problems. There was only one problem that Microsoft released a patch for. However, it's potentially significant. It is in the Windows filesharing protocol, that so-called SMB, the Service Message Blocks. And it unfortunately is a remote code execution flaw. It affects all currently supported versions of Windows at the critical level, that is, a critical remote code execution, except Vista and Server 2008. Those guys are only affected to a moderate level because there's a somewhat lesser exploit against them, not something that Microsoft wants to grade as critical. But for any people using any versions of XP, both 32- and 64-bit, and also Windows 2000 before that, this is a problem.

The exploit comes in over ports 139 and 445, which are the standard Windows filesharing and many other services ports. The good news is anybody with a router is blocking that by default. Anybody with a personal firewall is blocking that almost certainly. And even many ISPs are blocking those for us. Now, that doesn't prevent, for example, behind an ISP. ISPs' own customers might be able to infect each other. And similarly, computers within an organization are all behind the corporate firewall. If there was a policy of sharing those ports within an Intranet, within a corporation, then you've got the possibility that something could infect one machine, and it could spread using this remote code execution exploit.

And in fact we are seeing something like that. There was a report that companies that had, that is, corporate entities that had not applied that out-of-cycle October patch from late last year, that they were getting infected by worms that were using that exploit. So we've seen this problem where companies are less willing to upgrade, that is, their IT staffs really want to vet these changes before they turn them loose on the whole infrastructure. That introduces an implementation delay in patching these problems. And some companies are being bit by the fact that they're not doing it immediately. So the word to end users, our listeners, typically, is get these things fixed as quickly as possible. It's annoying, for me at least, to have to reboot my system in order to get these patches operating down in the code. But it's something you don't want to hold off doing for too long.

So Microsoft had that happen. And of course we also got a new version of the MSRT, the Malicious Software Removal Tool. And it's funny, I was looking at that, seeing if there was anything noteworthy about it. And we were talking last week, one of our Q&A questions was some guy asking, gee, should I be running MSRT all the time? And we explained that it was something that was run once a month, essentially, when you get a new one, and you reboot in order to get this thing installed. It's during the reboot that it runs, and not subsequently. There are ways you can induce versions of it that you can get from Microsoft to run all the time, if that's what you want. But in the context of that I was noticing that one of their questions was, is this something I can use instead of AV, because we were talking about how Microsoft does not want to step on the toes of the antivirus vendors. And Microsoft explicitly states this is not a replacement. The MSRT, the Malicious Software Removal Tool, is not a replacement for AV. It is explicitly a post-infection removal tool.

Leo: Oh, so it doesn't block anything ahead of time.

Steve: Exactly. So Microsoft is looking for evidence of infection in the system, not even latent code or files that haven't yet infected. So they're actively looking for infection, and it's a post-infection removal. Of course you don't want to get infected. You want to prevent that. So the way to think of this sort of in tandem with AV is the AV is your frontline guard, preventing things as they come in, scanning your email and websites and so forth on the way in, preventing infection. But Microsoft has said, look, if people aren't doing that, if something gets in under their radar, or there isn't a signature available yet for something, but Microsoft has it, we want to be able to disinfect systems. And they're not saying they do a perfect job. They're just trying to do the best job they can.

Leo: Yeah. And I think they don't want to step on the toes of commercial vendors; right?

Steve: I'm sure that's...

Leo: You don't want to kill that ecosystem. It's too important.

Steve: Yeah, I mean, and we've seen Microsoft sort of slowly creep forward. They now have a personal firewall that has certainly damaged the sale of personal firewalls to some degree, third-party tools. It's like, well, I've already got a personal firewall built in.

Leo: But they had to do it.

Steve: Of course.

Leo: They had to do that.

Steve: Yes.

Leo: And some would argue that, well, nobody better than the operating system vendor to put antivirus in there. But no other operating system does that. I think that you kind of like the idea as - in fact, remember Apple put out that note saying it's healthy when you use different antiviruses. If there's only...

Steve: In addition to your own.

Leo: Yeah. If there's only one, then it's easier, and it's an easier target for the bad guys.

Steve: That's exactly right.

Leo: Yeah. So maybe Microsoft, it's not entirely economic or are worried about getting sued, it really is better for the platform.

Steve: There was an interesting note also in the security news this week. A company called CheckFree, which is an electronic bill-paying service that's used by many banks sort of as their backend electronic bill-paying service, their Network Solutions account was hacked. And this is not the first time we've recently heard of Network Solutions accounts being hacked. And their assigned DNS servers were changed to point to somewhere in the Ukraine. So what happened was, unfortunately, this poor company - well, poor, I mean, maybe it's their fault. Who knows how they were hacked. But they're having to now notify nearly six million users that, because they don't know who was affected, anybody could have been affected. And essentially what happened was this fraudulent site in the Ukraine attempted to install password-stealing malware into the machines of anyone who visited believing that they were CheckFree. So this was a DNS attack. It was not a spoofing attack, though. It was actually the source of the DNS was changed.

So the takeaway message is I just wanted to raise this issue to our listeners and say you really do need to make sure that your login credentials for - anyone who's maintaining web servers and websites and web domains, that your login credentials for your registrar are really strong, that you've got a really good password. It's a perfect instance or a place to have a really strong password because it's not something that you need to do every day. Typically you log in maybe no more often than every few years to update your domain name registration. But it's the kind of thing that, without adequate safeguards at the registrar's end, somebody could just sit there and patiently try to crack into a high-value domain like CheckFree's domain. And ultimately they did. They got in, and they were able to redirect their DNS. So that's sort of like a legitimate change to DNS because the registrar's account was hacked.

Leo: This has been happening a lot. In fact, Clickjack, one of the uses of clickjacking was to get people's Gmail account, and then once you had the Gmail account request a new password from the domain registrar. So even if you had a strong password - you have to be really careful all around.

Steve: Yeah.

Leo: It's, boy, that's a terrible thing when you lose your website. And it's a disaster worldwide if it's somebody like VeriSign or, you know.

Steve: Yeah, yeah. And in one other little bit of news I thought was interesting, there were articles both in the Boston Globe and the Washington Post. Actually Brian Krebs picked up the story in the Washington Post from the Boston Globe, that people were noticing 25-cent charges appearing on a large number of credit card statements. And so there were two theories. One was - and I think this was probably the more nave theory, that Brian also didn't subscribe to, was that somebody was trying to make money by hiding lots of little small charges on credit cards. Well, in order to do that you've got to have the ability to charge to the card. And so Krebs, who is more technical, thought, you know, I doubt that. What this sounds to me like is somebody offering a service for the bad guys to verify that stolen credit card information is authentic. And of course what you first see is a 25-cent charge. And then what happens some length of time later is serious money starts getting charged. So I just wanted to also put that on our listeners' radar, that they just want to scan their credit card statements for any suspicious small charges that they might otherwise not have made, or just make sure there's not that going on. Because that would be, could be an early indication of trouble to come.

Leo: Huh. So there's a way of them verifying - you know, PayPal does that, and other systems do that. They do two - for instance, the way they figure out if it's really your account, they'll do two small deposits.

Steve: They use it as an authentication loop, essentially.

Leo: Right. And then you tell them how much it was.

Steve: Yup.

Leo: So but this is different. This is a debit, not a deposit.

Steve: Yes. This is somebody charging cards a very small amount of money. And the number I saw in the article was 25 cents.

Leo: Yeah, it'd be a way of seeing if you had access, if you were able to - if you got the quarter...

Steve: If they could get a quar- exactly.

Leo: You're in, yeah. It's amazing. I'll have to look at my statement.

Steve: We've got some interesting errata, too. There's a contributor to the GRC newsgroups who goes by the handle Bill_Mi. And I finally ran across a posting of his in our Security Now! newsgroup at GRC, which he said he had posted several times, but I just hadn't seen it before. And he commented that PayPal only shows the plug-in menu option after you have downloaded the plug-in. Which I was like, oh, my god.

Leo: And how would you get that plug-in if...

Steve: PayPal bites us again. So we talked about this also last week. Somebody was saying - I think he was in Australia saying, hey, I can't find anywhere to generate these one-time-use credit cards. And so I said, oh, yeah, it's right there on the menu. The first item is Plug-in. And even though it's strangely named, you wouldn't think that that's where it is, you click on Plug-in, then you go in, and you don't need to use the plug-in. You can still have the website generate a secure card for you. Well, it turns out you may not have the plug-in option on your menu. PayPal knows whether you have ever downloaded it, even if you're not using it, and you don't have to ever use it. You just have to download it. And then your account is tagged as having downloaded the plug-in. Then the plug-in item appears on the menu. And you don't have to ever use the plug-in. But that allows you to get to the ability to generate secure cards.

Leo: That's just silly.

Steve: It's just ridiculous. I mean...

Leo: Because I can't use it. The plug-in is for Windows, so I can't use it on the Mac. So, but you know what's funny, I must have downloaded it. I think I did at some point.

Steve: And there is a plug-in for Firefox, also.

Leo: Ah. Okay. Because I do have the option.

Steve: I'm sure you and I both did.

Leo: Sure.

Steve: And I didn't like it because it kept popping up and asking...

Leo: Yeah, I didn't want it.

Steve: Yeah, exactly. So I stopped using it. But since PayPal knows that I once downloaded it, now I've got the option any time I want to generate secure cards using their web interface, and not with the plug-in.

Leo: That's just goofy logic. I'm sorry.

Steve: So there, for anybody who after last week went looking for the plug-in menu item and still can't find it, try downloading the plug-in. You don't have to use it, just download it. And then the menu item apparently appears.

Leo: [Chuckling]

Steve: Another friend of mine sent me something that I thought was pretty funny. Just it was just some humor that's clearly security related. Timothy McSweeney has a website, McSweeneys.net. And he posted some secure website authentication questions that had been put together by someone named Joel Gunz. And so these were typical questions, or nontypical questions that I guess this person was recommending as something that would be hard to guess. So we have: What is your older sister's favorite Monopoly game piece?

Leo: There you go. There you go. That's good.

Steve: Who did your paternal grandfather vote for in the 1956 presidential election?

Leo: I don't know that one.

Steve: Why did you choose a liberal arts degree when your entire family urged you to go into finance?

Leo: I ask myself that every day [laughing]. Now it's comedy.

Steve: In what year did you begin working on your novel? How many weeks away was graduation when you dropped out of college?

Leo: Oh, well, that one I know. That might be in my Wikipedia article.

Steve: What was your score on the civil service employment exam? Where were you sitting when your girlfriend told you she was pregnant? Where did you never end up going for your honeymoon? In what year did you begin working for the post office?

Leo: Oh, this is so good.

Steve: What is the name of the hedge fund manager your ex-wife married?

Leo: So this is obviously comedy. But it raises a serious point, which is...

Steve: How many hours did it take you to drink that bottle of Jack Daniels yesterday? And what time was it when, in a drunken rage, you threw your novel into the fireplace?

Leo: There you go. And how many pages long was it?

Steve: And then the last one is, if you could do it all over again, what would you do differently?

Leo: Oh, I love that.

Steve: That's a good security question.

Leo: So these are questions that you would know, but no one else.

Steve: Yeah, I mean, and they're just meant to be funny also.

Leo: But it's a legitimate point. If you're allowed to generate your own question. Some places do; some places don't.

Steve: Right.

Leo: I wish they would because that's a much better way of doing it.

Steve: Much better. And then lastly, we've talked of course many times about one of our favorite security authentication gizmos, the YubiKey, where you plug it into a USB port on your computer, and it cryptographically generates keystrokes. Well, our good friends at ThinkGeek have come up with something that takes advantage of the USB port in a different way. It's called the Phantom Keystroker Version 2. It's a little dongle-y like thing with a USB connector on it. There's three little potentiometers, three variable adjustments on this, which determine the delay between things it does that are meant to annoy you. It will generate random mouse movements on your screen. It will toggle your caps lock. Or it will type out odd garbage text and phrases.

So the idea, of course, would be that you set this appropriately and plug it into the computer, like an unused USB port in the back of someone's machine. And their mouse just sort of twitches or moves around from time to time, or their cap lock toggles. And so they're typing along, and suddenly everything's in caps. And they go, it's like, oh, how did that happen? And they toggle it back off again. And they're typing along, and after some length of time that you can control, it toggles caps locks on again. So after a while you begin to think that your computer were possessed. Or unfortunately you might think it was infected and go off on a wild goose chase, when this in fact is just something that's using USB to simulate keyboard and mouse.

Leo: That is pretty funny.

Steve: Oh, it's pretty...

Leo: It's a practical joke.

Steve: Just a practical joke. And in fact, since I was first there, I went to their site just to verify that it was still there because I'd made a note of this several weeks ago, and we'd had so much content that I kept bumping this little note about this down. A new warning has appeared on their site, in red. It says, "Warning: The Phantom Keystroker never hits the return key, and it never clicks the mouse button." Meaning that it's not going to actually do any damage. It's not going to select anything or do something, it's just going to annoy you. And then it says, "However, you should not use it on anyone's system who is doing critical work where disruption could cause serious consequences. The Phantom Keystroker is a joke. Like any joke, you need to use prudence and judgment when executing it. You have been warned."

Leo: I like the - there's switches on it for time delay, caps lock, keyboard and mouse. So you can switch - you can change its behavior around.

Steve: Exactly. And then they suggest that you not set them to be too quick because it's sort of going to give itself up, if it happens. You would like to have it happen infrequently enough that someone's like, huh? What? Did I bump the mouse? Or is it, like...

Leo: That's mean.

Steve: Is it drifting on my desktop?

Leo: Oh, it's so mean. It's only 12.99.

Steve: Yeah.

Leo: They have - I don't know if you've seen it. They have another thing called the Annoy-a-tron.

Steve: Oh, I've got two of them.

Leo: No.

Steve: Oh, yeah. I'm a sucker for ThinkGeek stuff.

Leo: So the Annoy-a-tron just makes this little beep; right?

Steve: It's a little twitch, yeah.

Leo: Every once in a while. And we all have things that do that when they run out of juice or whatever.

Steve: Right.

Leo: And you just hide it, and it drives people crazy.

Steve: And in fact it's got a magnet on it, so to make it easier for, like, sticking it on the back of something. And so it's like, what? You're looking around, what's making that sound? And it's a high-pitched little chirp, and it never lasts very long. So it really just annoys you. Because it's like, okay, wait a minute. What's doing that? And so you look around, but it's not on enough to allow you to even zero in on it. And of course the high-frequency sound is hard for us acoustically to locate anyway.

Leo: Can't hear it anyway.

Steve: Yeah.

Leo: That is mean.

Steve: And I have a nice sort of, in the spirit of the holidays, which aren't very far behind us, SpinRite story I wanted to share before it became dated too far. Garry Weil, W-e-i-l, I think that's how you pronounce his name, Garry Weil sent us a little story about SpinRite saving Hanukkah. He said, "Steve and Leo. I'm a long-time listener to Security Now! and other TWiT podcasts. I bought SpinRite several years ago and use it at work and home." And actually his email address is Intel.com, so he's - that's where he's using it. He says, "My daughter requested a couple of PS3 games for Hanukkah, and I dutifully bought them for her. It was the fourth night of Hanukkah, and it was time to give her the new PS3 game, Mirror's Edge. A little while after lighting the menorah and giving her the game, she told me that the PS3 was broke. She had enlisted her 19-year-old brother's help, but the Restore option on the PS3 would not take. So I pulled the disk drive out and put it in my notebook and booted to SpinRite. I used Level 2, and it quickly found a single unrecoverable sector. After SpinRite finished and I replaced the disk in the PS3 and selected the Restore option, within a minute the PS3 was working perfectly, and my daughter was able to start playing her new game. I received a great big hug and thanked GRC silently. Thanks for a great utility, and keep up the podcast."

Leo: Wow. I didn't know it would run on a PS3. You're just...

Steve: The reason I wanted to share this with listeners is it will fix anything. If it's magnetic and spinning, SpinRite can fix it.

Leo: That's pretty amazing. Wow.

Steve: And the other thing, I noted that he said it found a single unrecoverable sector. One of the other things that SpinRite does that is truly responsible for I think a lot of its capability in recovering things is, as far as I know, it has always been unique in its ability to recover all but the unreadable little bit of a sector. That is, for example, if a little more than a couple bytes, it depends on the total number of bits that are damaged from the - the distance between the first wrong bit and the last wrong bit is the so-called "burst." And if that's longer than a certain length, and like 11 is typical, well, okay, a byte is eight bits, so that's less than - it fits within two bytes, or maybe it could straddle three. But that means that if only two or three bytes are bad, nothing else in the world will give you any of the sector, that is, any of the other 510 bytes. And SpinRite will. It's able to say, okay, no matter what we've tried, and believe me we've tried, we cannot get this sector perfect. But we can give you most of it. We can give you 510 out of the 512 bytes. Well, very often, for example, if that's in the directory, or if it's a boot sector or something, you've gone from getting nothing to getting everything you need in order to get the system to boot. And so that's one of the reasons SpinRite ends up having such data recovery gain is that it can give you most of a sector. Whereas nothing else will give you any of it.

Leo: Well, that's very interesting. I didn't - yeah. Because normally you'd say, well, if I can't get the whole sector, forget it.

Steve: Right. But it turns out that most of it can be useful.

Leo: That's all you need. Very cool.

Steve: Some parts are edible.

Leo: You're in a mood today. It's the comedic stylings of Steve Gibson, I swear. All right, Steve. It is time to talk about certs.

Steve: Certificates. Cryptographic certificates.

Leo: Yes.

Steve: Okay. What a cryptographic certificate is, that is, what it essentially does, is it offers a public key. What a certificate is publishing, essentially a certificate publishes someone's public key. So what you want to know is, you want to know that the person whose public key you believe it is, is true. That is, so the certificate essentially binds together an identity of someone or something with a public key. And, for example, a web server certificate, it's binding the domain name to the public key. So, for example, www.GRC.com, we've got an SSL certificate. And the certificate contains GRC's public key. So what the certificate is asserting is that these two things go together, the identity information and the public key.

Well, that assertion is being made by a third party, as we were talking at the top of this episode, a third party who has taken responsibility and has put their reputation behind that assertion. So someone like GoDaddy or VeriSign or Equifax or Global Trust or one of these certificate authorities has gone through some sort of process to be able to assert, to attest that this assertion being made by the certificate is true. So the way a certificate is created is you create what's called a "certificate signing request." And certificates are just files, they're just data files like anything else. You know, a few K in size. The certificate signing request is something that an individual who wants to create a certificate generates on their computer. And that process involves creating a public key pair. And we've talked in years past in detail about public keys. So I'll just quickly remind our listeners how they work.

The idea is that a so-called public key pair produces, or consists of, two different keys. That is, they're different, and so we use the word "asymmetric." We've talked about symmetrical encryption where you use the same key to encrypt and to decrypt. With an asymmetric key pair, which is normally called a public key pair, one key encrypts, and the other decrypts, although it doesn't matter which is which. So you could use either key to encipher something, and you just use the other key to decipher it, or to decrypt it. So somebody who wants to generate a certificate first creates a public key pair. They then fill out essentially the fields in the standard form. There's a format for all this called an X.509 certificate. X.509 is sort of the worldwide accepted standard of format for public key certificates. So you fill out this information. Then, using your private key, you sign that information.

Now, we've talked also before about signing. Signing is the process of generating a hash value for that information, which ends up essentially digesting any amount of information into a small fixed amount, typically 128 bits, which is 16 bytes, or maybe 160 bits, which is 20 bytes. Some relatively small digest, as it's called, of the input text. And then that little blob is encrypted using, for example, if we want to generate a certificate signing request, that would be encrypted using our private key. So we never disclose our private key, even to the entity that we're wanting to have sign our certificate. We don't need to because what we do is we take that result, that signed certificate request, and our public key, and send them off to someone to sign, like to VeriSign or GoDaddy or someone. What they're doing then is, because it doesn't matter which of the keys, the public or private, that you use to encrypt or decrypt, they don't ever get our private key. But they're able to decrypt our signature with the public key that we gave them, that is, the other key is really all that matters. That gets back the MD5 or whatever hash was used to sign the form. They then, essentially, they hash it themselves. They rehash what we gave them, which consists of all the information we provided and our public key. And they make sure that they get the same resulting hash value as we did, which we encrypted when we sent it to them.

So what does that prove? That proves that we are in possession of the matching key that we sent them, which in this case we're going to call the private key because we're not letting anyone have it. So they know that we sent them this request containing the identity information and a public key, and that we're in possession of the private key, because that's the only way that we could have encrypted the signature that we sent them. So then they say - they do whatever they need to do, depending upon what kind of request this is, to verify that the identity information really is us. You know, for example, in the case of a website, you have to go through some hoops, phone calls, verify IP addresses or that you're in control of the website, do something to say, to prove to this authority that I'm somebody who has the authorization to request a certificate, for example, for a server at GRC.com. So once that's done, then the certificate authority creates a new certificate, which is our server certificate. That's something which they sign in order to prove that they've done this work, they've verified the identity and the public and private key. And so they're signing this saying that, okay, we've done our due diligence. These people are who they say they are.

That certificate is then sent back to the requestor. For example, in the case of a web server, it's installed on the web server. And now anytime a third party wants to establish a secure connection, at the beginning of that connection the web server provides that certificate to the web browser, saying this is who we are. And somebody who you trust has agreed with that. Somebody who you trust has signed our certificate. So what happens is the web browser looks in its store of certificate authorities for the matching entity that has signed it. Now, the certificate was signed with their public key. So the public key is available in this store of certificate authorities. So the web browser that wants to verify that this is a proper signature is able to duplicate the signing process, verify that they get the same result. And the only way that's possible is if the certificate has not been changed in any fashion, that is, if the hash value, the so-called fingerprint on that certificate has not had a single bit changed. So that's sort of the process by which all this operates.

Now, in this example there's only a server certificate and the root, or the certificate authority. It's possible to have a chain of certificates. And essentially it works in the same fashion, where, for example, there might be a so-called intermediate certificate authority, or an intermediate authority. And you could have, in fact, several of those, essentially forming a chain where each certificate is signed by somebody, and then that one is signed by somebody, and that one is signed by somebody. Ultimately you come back to the root. And it's interesting because the root certificate is signed by itself. That is, it's a so-called "self-signed certificate." They sign their own certificate, meaning that there's nobody here left to trust. I mean, you have to trust them because you've got no choice. Ultimately...

Leo: There's always somebody; right? There's got to be one final end to that chain.

Steve: Well, and of course we're trusting our OS vendor, for example, Microsoft, or maybe Mozilla, that provided this set of self-signed certificates. So those root certificates are self-signed. They're just signed by themselves to sort of form an anchor to this chain of trust. Okay. So that's sort of the process by which a certificate is created and issued to a server and then presented to the browser for authentication. What happened back in 2004 is some researchers discovered some problems with the MD5 hash, which is one of the oldest hashes. There was an MD4 before it. In fact, there was an MD2, not surprisingly, before that one, which was pretty well destroyed. MD4 had some problems. And Ron Rivest at RSA came out with MD5.

Well, as often happens, as our cryptographers analyze and poke at these things over time, some problems surface. What happened in 2004, so more than four years ago from the time that we're recording this here at the beginning of '09, what happened in '04 is some very clever attacks were designed against the way MD5, Message Digest 5, functions. Now, what MD5 does - we've talked in detail in the past about various cryptographic algorithms. I'm going to sort of explain in overview how MD5 actually functions because it's important to understand that in order to get sort of a visual idea of how it's been broken. MD5 outputs a 128-bit result, that is, 16 bytes of data, 128 bits. So essentially, from the outside of this, if we think of it for a minute as a black box, we pour any amount of data into it. It's able to digest texts, plaintexts of any size. So we pour any amount of data into it. And when we're done, no matter how much data we've put in, we always get 128 bits.

And what's special about a hash, there are key characteristics that make a hash a good hash as opposed to a useless hash. And they are that it is supposed to be extremely difficult - computationally infeasible is, like, the technical term, which means you need a huge amount of computation time to deliberately put something in that creates a deliberate result. So you're not supposed to be able to precompute a complex input that ends up giving you a specific output. Meaning that you cannot, for example, if you had one document which you hashed into a 128-bit result, you're not supposed to be able to produce, for example, a document with a few changes which will give you the same result. So it's meant to be sort of a fingerprint, a unique identifier.

But the fact that so many bits are reduced to just a few, that is, if you had a long document, it's always being crunched down, in the case of MD5, to 128 bits. Well, that means that there are many, many, many different patterns of input bits that will give you the same 128 output bits. There have to be because you're reducing a huge number of inputs into a much lesser number of outputs. So there have to be so-called "collisions." A collision is two different things that result in the same output hash. But the idea is that none of those different things that might, that would result in the same output, are useful to you. They're just, you know, they're cryptographically random. They're not going to be something that is a useful collision.

Well, the way MD5 functions is it starts out with a particular fixed set of 128 bits. And this is part of the spec. In the spec it says - and that's 16 bytes of data, or you can think of it as four 32-bit blocks. So it starts out with these four 32-bit things that are fixed values, specific hex values. And it digests the data 512 bits at a time. So it takes - and 512 bits is 64 bytes. So it takes 64 bytes in and runs an algorithm on those that ends up producing a different four 32-bit blocks. And so what happens is, as the input comes in, it takes the input 512 bits at a time and runs it through this algorithm. It's four rounds that takes the data in 128 bits at a time. So four times 128 is 512 bits that it takes in as a whole block. And every time it ends up sort of taking that input four 32-bits, and ends up producing an output of these four 32-bit blocks.

Now, the problem is that, because we want to be able to take something of any length, but MD5 only operates in 512-bit chunks, it's necessary to do some padding of the end of what we're feeding it, if it doesn't end up being an exact multiple of 512. And in fact what happens is it's padded out to - the last block is padded out to 448 bits, and then a 64-bit final piece of data is added, which is the binary representation of the original message size. So that's how MD5 always ends up with an even 512-bit block size multiple, which is what it needs. So that's the process of hashing something through MD5, is it's first padded out so that it's a multiple of 512 bits long, the end of it being the size of the original message before the padding. And then 512 bits at a time are fed into this algorithm which runs through four rounds. And at every phase it takes in these initial four blocks of 32 bits, or 128 bits. And then after that there's a resultant 128 bits. Well, once you're all done, that final resultant 128 bits is the hash value.

Okay. So what the researchers figured out is that there was a way to choose any beginning input. Imagine that you take the so-called - this plaintext. And you're able to specify two different source plaintexts. Your goal is to have them hash to the same value. Well, these guys, these security researchers figured out how you could take two different deliberately created source plaintexts and at some point before the end, essentially, make some changes and then add some of your own data to the end that would essentially bring you to identical hashes. Well, this is never supposed to be possible. That is, it was absolutely supposed to be that you could not deliberately create a resultant hash value using a secure hashing algorithm. These guys figured out essentially how, by appending their own carefully designed data to the bottom of two different texts, they could create a common hash. And that we knew about two years ago, in 2007. So essentially the technology for damaging MD5 more and more has been advancing, which is exactly, historically, this is what we see in crypto work.

So then the researchers said okay. I mean, and this work was published in '07. And people said, oh, gee, that's interesting. Very theoretical. Very academic. And some guy in fact used this as his master's thesis in cryptography. But it's like, okay, we're not worried about that. Well, so these researchers in the middle of last year, last summer of '08, they decided, okay, we've got to bring this to everyone's attention. We've got to explain to people very clearly how badly MD5 is broken, how much of the security industry is still using MD5, despite the fact that we are now able to take two different inputs, and if we are allowed to extend their length, if we're allowed to add our own stuff on the end, we now have the computational ability to make the hashes end up the same. So how can we shock people? And they figured out a way to create a fraudulent certificate, a fraudulent security certificate.

It turns out that doing this was extremely difficult. Essentially they needed, in order to pull this off, they needed to be able to get a security authority, somebody who was going to sign and issue a certificate, they needed to get them to issue exactly the certificate they wanted because they needed to provide a certificate signing request with exactly the data that they needed, where they were going to have a fraudulent certificate match the resulting certificate that they got back. The problem was that there were some things they could not control. For example, there's a serial number in certificates that are issued, as well as a first valid date and an ending valid date, which is always based on the instant that the certificate is made. Normally it's like, it's valid for one year or two years or three years. But it's, like, it involves minutes and seconds based on when the actual signing takes place.

So they sent - they created a bot to go out and scour the 'Net, which over the course of a relatively short time collected 100,000 website certificates. They analyzed the website certificates for a number of characteristics. And it turns out that one particular certificate authority, RapidSSL, they were using a sequential serial number. That is, it just - it incremented literally by one for every time a certificate was issued. And to verify that they asked RapidSSL for some RapidSSLs, in rapid succession. And they got back certificates with serial numbers either one off or a couple off, if somebody else had asked RapidSSL for a certificate in the meantime. So they were able to verify that the serial numbers were being issued sequentially.

They were also able to - and they did this over time by asking for one certificate every so often. They were able to look at how rapidly the serial number was incrementing and guess, essentially, what serial number would be issued at a certain time in the future. They also verified that from the time they clicked the Okay button asking for a certificate, that it took eight seconds. I think it was - it was either six or eight seconds for the certificate to be issued. So by deliberately clicking the Okay button on Yes, please issue us a certificate at a certain time, and they had to synchronize their clocks, of course, they were able to deliberately choose the timestamp on the RapidSSL certificate that would be issued. So they looked at where the serial number was. They decided it would take them about three days to get everything together, that is, essentially to generate the matching data to be appended in order to make the hashes match.

And so what they did was they created the certificate that they wanted to have forged and the certificate that they wanted to be issued. They then figured out what needed to be added to the end of those in order to get the MD5 hash to match. The reason that was important is that the certificate authority was going to sign the resulting certificate that would be issued to them. Well, they would never have the private key used to sign the certificate. But if their forged certificate had the same hash as the one that was signed, then the signature would be good for both.

So three days before, they created this pair of certificates, the forged one and the one that they were going to get signed from the certificate authority, RapidSSL. They cranked this array of 200 PS3s many times in order to do the math to figure out what to add to the certificates. And they had figured out how to hide that, the added stuff, in the certificate body itself. And then they started asking for certificates to manually advance the serial number counter until it was one shy from where they wanted. And then at exactly the right time they said we want a certificate. And so the serial number jumped to the next number that they had anticipated, issued at the time that they had designed, to issue exactly the certificate that they wanted. That's what they had to go through. It turns out the first three times something went wrong. Somebody else asked for a certificate just before they asked for their final one. Or the timing of their clocks wasn't quite right so that, like, even one second off would cause the timestamps not to be predictable. But on the fourth try they got it. They were issued a certificate by RapidSSL with exactly the serial number that they had anticipated and required, on exactly the timestamp to the second that they required, and all the other information that they had provided. So they got back from RapidSSL the certificate that they had designed to be given, for which they already had a matching fraudulent certificate.

Leo: So clever.

Steve: And where both of them had the same MD5 hash. And that meant that they were able to lift the signature off of the certificate, which they didn't even care about. I mean, they threw that away. It's the signature they wanted. They were able to lift the signature off of the one with the matching hash, stick it on the end of their fraudulent one, and it was valid.

Leo: Wow. That's so clever. That's such a great hack.

Steve: It's just, I mean, it's just incre- it's brilliant. And now, of course...

Leo: But of course it only works because RapidSSL is really flawed in the way it's assigning these certificates.

Steve: Well, okay. So several things. Many, many other certificate issuers are issuing a random serial number. In fact...

Leo: This is very much like that DNS attack we were talking about.

Steve: It's very much like that, yes. And we know, for example, listeners know that the certificate authority could actually have a counter which is just incrementing by one, just like RapidSSL does. But all you have to do is encrypt it.

Leo: Right.

Steve: You know, just run it through a simple symmetric encryption. There are, I believe it's 20 bytes available for - maybe it's bits. I can't remember. Anyway, there's a big chunk of space available - I think it is bytes - for the serial number. Because that's 160 bits. So if you did 160-bit encryption, then every single one would be completely different than the next one. And that would have blown this hack out of the water. I mean, and there's even the ability to put private data in a certificate. RapidSSL wasn't. But you could have your own fields with some random gibberish. That would have blown this out of the water. It was just the fact that they were able to very carefully induce the issuance of exactly the certificate they wanted, where they had precomputed what its hash would be and had their fraudulent certificate ready, so they were able to essentially lift the signature from a valid one and stick it on theirs, making it look valid.

Leo: Very clever.

Steve: And if anything else had been done, it wouldn't have worked.

Leo: Right, right.

Steve: So it's the fact that - so, but they really drove their point home finally by saying, okay, sure. This is theoretical. This is academic. We have just created a fraudulent certificate. Oh, and the other cool thing is, one of the - there's a bit flag in the X.509 certificate spec which says is this a CA or not? Is this an authority, or is it sort of like an end-user certificate? Well, in their fraudulent certificate, they set that bit. So they didn't just create a single fraudulent certificate. They created their own signing authority that was signed by RapidSSL. So now they were able to sign any one, any certificate that they wanted to. Now they could create website certificates all day and night, and they signed them. And again, due to the way a chain of trust works, because the root SSL CA was trusted, so was the intermediate certificate authority, which signed the final website. So they created their own fraudulent, but valid for every web browser on the planet, certificate authority.

Leo: Now, how repeatable, though, is this? I mean, it sounds like it's really a lot of work.

Steve: Well, they have, in their careful dissertation, I mean, they wrote a really well-written paper where they take us through this. They also offer all the files that they used, so people can see them for themselves. They have not gone into the details of how they pulled it off. They just said, you know, lots of math, lots of PS3s, this is what we did. And but they said some number of months in the future they're going to reveal this. They've commented, though, that to their knowledge nobody else has done this. But we don't know that nobody else has done this.

Leo: Right, right, right.

Steve: People who could do it for malicious reasons certainly aren't going to advertise. It's the academics. And, see, this is a perfect example of why we cannot criminalize this sort of work. I mean, this is what unfortunately the DMCA, the Digital Millennium Copyright Act, has really created a problem because there are many academicians that are now afraid to do research because technically they're breaking the law. But we need people to poke at these things. And it's worth mentioning, too, that the more secure, or I should say currently more secure, SHA-1 hash is also under attack. It hasn't been wounded to the degree that MD5 has. But there's some concern that it may - it's better than MD5 right now. But it may not last that long. So, I mean, we really need to allow cryptographic research to operate in a way that is not fettered by ridiculous law, which is well-meaning but wrong-headed.

Leo: Well, I think there are some companies that would prefer security by obscurity. They don't want this stuff to be exposed.

Steve: Right.

Leo: Which is really a problem because of course the hackers will find out, and they aren't going to tell anybody. I thought it was very interesting they used PS3s, too. They're really computational powerhouses.

Steve: Yeah, well, the way that - there's some types of algorithms which you can run in parallel, and where you don't need, like, lots of branching decisions. So you're able to write code that just does mass number crunching. And they were able to map the algorithmic computational work that they needed to have done, they were able to map it into the architecture of the PS3 cell processor and get this bank of PS3s to do a lot of their work.

Leo: That's really cool.

Steve: That's very cool. The idea that they were able to basically create a pair of certificates, one that they anticipated receiving, and one that was that of a fraudulent certificate authority. They were able to engineer those two so that they would have the same MD5 hash, then induce a real certificate authority to issue them the certificate they needed that would have a matching signature so that they could take the signature off of the valid one, then throw that one away - it's only the signature that they wanted - tack it onto their fraudulent one, making it a valid certificate authority.

Leo: Now, we don't know how many times they had to do this. This could be quite costly if you had to buy a bunch of certificates.

Steve: They said they spent about $700, just shy of $700. However, RapidSSL, unfortunately it's highly automated. So they were able to - there is a reissue certificate. And I think they were able to reissue a certificate, like, 22 times, some high number of times. So they did spend just shy of $700. But they generated and then canceled and reissued many, many times so as not to just rack up the bill.

Leo: Oh, interesting. And that's clever, too.

Steve: Yeah. But again, if anybody were - if there was any oversight over this, that's another thing that is, you know, this would raise red flags by anybody who's saying, wait a minute, what are these clowns doing? They're issuing certificates and canceling them and issuing and canceling and issuing and canceling. And so it's like, whoa, stop. So it was many characteristics of this one certificate authority. But these guys succeeded in demonstrating that MD5 can be cracked. And it's worth noting, too, that historically the kinds of limitations that they faced may not stand a year or two or three from now.

Leo: Why not?

Steve: So we need to get away from MD5.

Leo: You mean in terms of the speed of processors, the technology?

Steve: Well, for example, if more - no, no. If more were learned about MD5...

Leo: Oh, I see, yeah.

Steve: ...it may be possible to take existing certificates and lift the signatures from them and move them over. Right now this attack required a very precise issuance of a precomputed, predesigned certificate to be signed. But imagine if you, like, had godlike understanding of MD5, and much more than we have now, where you could just take anyone's signature and move it onto your own fraudulent certificate. Then the world is over.

Leo: Oh, boy. Oh, boy. Well, I'm glad - this is really interesting stuff. And I just admire the, you know, it's graduate students. They've got a lot of time on their hands. I admire just their persistence and their cleverness. I mean, there's a huge amount of clever, smart work being done here. It's really interesting.

Steve: Oh, absolutely so. And this is what we need them for.

Leo: Yeah, yeah. That's why open systems work, because people can bang on them.

Steve: Yeah, yeah.

Leo: Steve Gibson, you're the greatest. Thank you for making that all crystal clear, as usual. You can find more about all of this in show notes. And we have a new wiki, wiki.twit.tv, and we're putting show notes there for public consumption. We will migrate them. It's really nice because a community is working on them. We'll migrate them, of course, over to our regular show notes on TWiT.tv. And then Steve has his show notes. There are many places to get more information. Steve's show notes are at GRC.com/securitynow. Also that's where you'll find transcriptions. He does full-length transcriptions. You've inspired us, Steve. I think really, it's really clear that from the point of view of Googling things and finding things, you've got to have text. So I think we're going to talk to Elaine and do more and more of our shows because the transcriptions just really help this show in particular. This is the one you really want a transcription of, of course. And 16KB versions for low-bandwidth people. It's all there, GRC.com. When you're at GRC...

Steve: And the feedback page, GRC.com/feedback. Next show is going to be a Q&A. So if you've got questions, you've discovered things, you want to make sure we know things, you've got ideas, whatever it is, GRC.com/feedback.

Leo: We want your questions. Also at GRC of course SpinRite, the world's best hard drive maintenance and recovery utility. And many great free programs like ShieldsUP!, Wizmo. GRC, the Gibson Research Corporation, GRC.com. Steve, have a great week. Let's hope there's no major security backlashes this week, and we can talk next week and answer people's questions.

Steve: Either way, we'll keep it covered, Leo. Thanks very much.

Leo: That's what we're here for. Thank you, Steve. We'll see you next time on Security Now!.