SECURITY NOW!

Transcript of Episode #193

Conficker

Description: Steve and Leo discuss the week's security news; then they closely examine the detailed operation and evolution of "Conficker," the most technically sophisticated worm the Internet has ever encountered.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-193.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-193-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 193 for April 23, 2009: Conficker. 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!. We're ready to cover your security butt with Mr. Steve Gibson from the Gibson Research Corporation, creators of the SpinRite, the fabulous SpinRite disk maintenance and recovery utility, and discoverer of spyware, and our security guru. Hi, Steve.

Steve Gibson: Hey, Leo. It's great to be with you again this week.

Leo: Welcome back.

Steve: As always. As we approach our 200th episode. We're at 193 and counting.

Leo: Wow.

Steve: So actually I'm excited about 208, since that will be four times 52, meaning that it's the end of our fourth year as we go into our fifth.

Leo: And only you can do that math because only you do a show every week. You're amazing.

Steve: Never missed one.

Leo: He's the Iron Man of podcasting.

Steve: And it's so funny, too, because when you first suggested this years ago, I thought, well, Leo, I kind of like the idea, but there's no way we're going to have enough to talk about. And now we've got people complaining that I have promises for future episodes backed up.

Leo: You, too. We're backed up.

Steve: Just can't get to them.

Leo: I love it.

Steve: But we will. We will. We will.

Leo: Well, let's start - we're going to talk about, I think, the number one security topic of the month, the year, who knows, maybe the decade: Conficker.

Steve: Well, yes. I don't think we've ever really gone into great depth about any previous worms or, for that matter, viruses because there really hasn't been that much to them. I mean, it's like, okay, so MSBlast sprays the Internet with packets trying to spread. Well, Conficker is interesting to me and to our, I'm sure to our audience and the broader Internet because it is a phenomenally sophisticated worm. It's defying all attempts at eradication. It is managing to survive. The author is dynamically updating it, literally in lockstep with all attempts to thwart it that have been made by the industry and the so-called Conficker Cabal, which is a group of whitehat companies, Microsoft and the AV companies that are getting together to deal with it. But there's so much to it. And so it just, you know, it would make for a really interesting and meaty episode. So I decided let's really talk about exactly what Conficker does. And my feeling is, by the time we're through with our listeners today, Leo, there'll be a greater sense of respect for how bad and sort of deeply bad these things can be. I mean, there's just so much this thing does.

Leo: Well, as you say, you couldn't really do it ever before. I mean, the stuff was...

Steve: There wasn't much to talk about.

Leo: It was, I mean, I guess you could say, wow, isn't it interesting that they're using email now to spread, or that they're able to live on the 'Net. But there wasn't great programming involved. This sounds like this is pretty sophisticated.

Steve: Well, it digitally signs its transmissions...

Leo: Incredible.

Steve: ...to prevent anyone from being able to spoof them.

Leo: State of the art. This is state of the art.

Steve: We're going to talk about it all today.

Leo: So what is the security news? What's going on in the wonderful world?

Steve: Lots of stuff here. You probably heard that a verdict came down in the Pirate Bay case.

Leo: Oh, yeah. Oh, yeah.

Steve: The four guys that were the defendants were found guilty of breaking Swedish copyright law for their involvement in the Pirate Bay website. Three were the maintainers of the site, and one was the financier. They were sentenced to one year each in prison and ordered to pay 30 million Kroner, which is about 3.5 million U.S. dollars, to various media companies who brought the suit. They plan to appeal the verdict, so we don't know how that will turn out. They were as defiant as ever, I mean, these are the most defiant guys you've ever seen. But it was a significant verdict in that, despite the fact that their defense was that they are not hosting copyrighted content, the argument was, yes, but they're making it - they're facilitating the clear violation of the copyright holders' rights. And that was enough to find them guilty.

Leo: They are probably judgment-proof. They've already said we don't have the money, we don't intend to pay it, we're appealing. And really, however you feel about them, the bottom line is it hasn't shut the Pirate Bay down. It won't because, as they say on their page, we are from the Internets. You can't stop us. And, you know, they make an interesting point. I wonder what you think about this? They say we're just a search engine, in the same way that Google's a search engine. Are you going to take Google to court because you can search for pirated stuff on Google?

Steve: That's, I mean, that is a good point. My - yeah, I mean, I guess it's gray. I'm trying to think whether I've ever needed anything from there.

Leo: No.

Steve: I don't think so. I don't think so.

Leo: I mean, let's face it. The name kind of says it all. They're not really saying we're just a...

Steve: Yeah, but again, you know, we can't, I mean, part of free speech is you get to name things what you want, and that's too tough if people don't like what you chose.

Leo: Well, the courts ruled, and that's the bottom line. And I know that the recording industry and the movie industry are happy as can be.

Steve: Yeah. Well, in another interesting bit of news, Amazon.uk has been the first of a number of high-profile companies to announce that it is going to block Phorm from scanning its pages.

Leo: Right on.

Steve: Yeah. Phorm, of course, we've talked about. We brought up some news about it last week, in fact. P-h-o-r-m is the really sort of nasty, very invasive technology that ISPs are still toying around with deploying, even though its technology is completely unproven. I mean, it's amazing how much negativity is being generated by this company where it's not even clear that what they're doing is going to be effective. You know, they end up planting their own cookies in every single website you visit. So your browser ends up just stuffed with their cookies because they add them to every site you visit by intercepting your connection to ISPs and dynamically seeding your web browser.

So anyway, the reason they scan pages is when people go to a site - like somebody who's using unfortunately a Phorm-enhanced ISP would go to Amazon.uk. Well, the Phorm servers would be notified of that. They perform a keyword search of the site you go to, like Amazon, to figure out what kind of site it is. And then on the fly they inject their own advertising, which is supposed to be germane to where you are. So Amazon is saying we're going to block Phorm from doing keyword search scans on our website and actively resist it. And there's an open rights group based in the U.K. that has asked, you know, high-profile websites like Amazon, AOL, Microsoft, eBay, YouTube and so forth, to opt out of participating with Phorm. And so Amazon in the U.K. is the first one to do so.

Leo: Good.

Steve: So congratulations for them.

Leo: Big victory for all privacy advocates.

Steve: Yeah, that's just - again, it's that there is no informed consent. I have no problem, and most people have no problem, if a user formally says - and that's not p-h-o-r-m phormally, it's formally...

Leo: Officially...

Steve: ...with great formality.

Leo: Yes.

Steve: If they formally allow Phorm to do this, then fine. Then get tracked and have your browser filled with cookies and so forth. But the whole problem is that this is just, well, in fact the suit that's being brought by the European Commission is due to the fact that there was no consent provided during British Telecom's, BT's, previous secret testing of this technology. It was just being done to users without their knowledge or permission. So that's not okay.

Leo: Yeah.

Steve: One other little bit of news is that - or actually two more. One is that there was a report put out by Verizon Business that said that it had responded in '08 to at least 90 confirmed data breaches involving on the order of 285 million consumer records. And what was worrisome about this, I mean, this is a huge number, 285 million consumer records. What was most worrisome was that the number of breaches and the size of the breaches in total was larger than all of the breaches in '04, '05, '06, and '07. So '08 dwarfed the sum of breaches in the previous four years.

Leo: Holy cow. I mean, I knew it was bad, but I didn't know it was that bad. Wow.

Steve: And interestingly, it turns out that the breaches at banks and financial institutions were responsible for 93 percent of all such records compromised last year. So, I mean, these are high-value targets. Now, in a strange little twist, there's a side effect of this. So much of this material, stolen consumer records, is now available on the black market, that the prices have fallen on the black market, and the bad guys are not any longer making as much money as they used to.

Leo: I guess that's good.

Steve: Because they're wading around in all of this stuff.

Leo: I guess that's a silver lining.

Steve: Yeah, I'm not so sure, but...

Leo: It's not that good.

Steve: Yeah.

Leo: Wow.

Steve: And then the last little bit of news, I'm sure you picked up on this one, too, is that the Pentagon found spies in the network for the Joint Strike Fighter project. This is the $300 billion program that the Pentagon is running. It's the most sophisticated weaponry we have yet. The network was hacked. And get this: Several terabytes of files, which were encrypted by the bad guys before leaving the network, so no one knows exactly what it was that was taken, but several terabytes of data over the last about a year and a half.

Leo: This is unconscionable.

Steve: They know that it was the design and the avionics material were siphoned off and sent somewhere. They don't know where.

Leo: Awful.

Steve: It's believed to be China. But again, as we've said, it's impossible to really nail down full accountability on these things. So but it was a big concern and a black eye for the Pentagon. And, I mean, and I really hope that we begin paying attention to this because it just seems like this is rapidly on the rise. This is the most publicized, worst such incident we've seen so far.

Leo: Steve, how do you explain this? This is not something that is hard to protect. These are valuable, hundreds of billions of dollars' worth of valued state secrets. First of all, why is this stuff even on Internet-connected computers?

Steve: Yeah, I know.

Leo: Why is it not secured, if it is?

Steve: The Wall Street Journal's report indicated that the most sensitive of the material is on a separate network which is not connected to the Internet. So there is some sense of that. But clearly this material, which is on Internet-connected machines, should also not have been in that network. I mean...

Leo: It's not rocket science. We know how to protect this stuff; right?

Steve: Yes, yes. I mean, we do in theory. And in fact that's why I'm so very nervous about the push that we're seeing towards medical records being put online. It's like, oh, goodness. I mean, I recognize we want to bring our healthcare costs down in the U.S., and it's one of the current administration's major pushes. But it's like, you know, there's just no demonstration that government knows how to do this. And I haven't yet seen a smart government person. I mean, that's sort of an oxymoron, I mean, in terms of, like, technology and real security protection.

Leo: Well, come on, the NSA must have smart people. There must be some smart people. You think these people are just dufuses?

Steve: But then is it bureaucracy that prevents, I mean...

Leo: I can't figure it out.

Steve: Yeah, I agree, the NSA down deep in some think-tank behind locked doors with all kinds of authentication, they've got really, really good people. But it's very much like you don't put your good people on tech support. You put them on development. And so the people doing tech support, it's too expensive to have a good person and have them talk to customers. So you have sort of an okay-enough person who deals with customers, and maybe they're able to escalate that to somebody who's more capable. Similarly, the NSA is not going to have their really good guys doing IT networking because there's, like, really more important things that they need to be doing.

And but again, I mean, this sort of comes back to my rant, which I will not recap, from last week, where I was talking about how much we have grown to put up with Windows. And I was reminded that, speaking of Conficker, that it knocked the, I think it was the Sheffield Hospital chain in the U.K. off the 'Net, I mean, out of operating mode for some length of time because the equipment in the operating theater was running Windows and was on the Internet. So, first of all, it's worrisome that critical care equipment would have Windows as its operating system, and also critical that it would be on the Internet. It's like, oh, well, we turned off Windows Update because the machines used to reboot in the middle of an operation. Like, oh, what, how many things can you have wrong with the picture?

Leo: But this I understand. I understand this. But this is the nation's most critical military secrets. And they're not only sitting on the Internet, but they're apparently doing so with no really good protection.

Steve: Yeah.

Leo: I mean, if you had these military secrets in your house, Steve, you could lock them down. Right?

Steve: Yes, I could, actually. Although...

Leo: Yes, you could.

Steve: ...my world is much simpler. In fairness, my world is much simpler.

Leo: I guess they have contractors who are looking at the plans online, or there's some sort of, I mean, there's something going on.

Steve: It's like, oh, don't worry, we'll just put this little website up that allows you to do vendor agreements or something, and so there's some cross-site scripting vulnerability that allows them to get into the server. There was a guy many, many years ago, maybe ten years ago, who used to call me through, I mean, literally like sci-fi mode, linking through multiple satellites and jumping around between different points so that I couldn't track him back. And he protected his identity. And he used to tell me in detail how he was roaming around inside of Microsoft's most sensitive networks, and how he would find some entry point in the U.K. through an affiliate, that he was able to bridge across network interface cards from an external network to the internal network and then jump from there over through two other offices to Redmond, and convinced me through what evidence he produced that he was really doing this.

And it's because, as you say, Leo, these networks are incredibly complex, lots of interconnections. And at some point doubtless this network was established by people who knew what they were doing, and it was really bolted down and secure. But then over time stuff got added. I mean, all it takes is for someone to stick an infected USB thumb drive into any machine on that network. And unfortunately, if it's running Windows, and it's processing autorun.inf files, and it may have been stuck in a laptop before that that had Conficker B on it, because Conficker version B would jump over to removable drives when they appeared, that's all it would take to suddenly have the malware on that machine. Or as we know, social engineering attacks are highly effective. So some Excel document or plans or something was sent to somebody who was expecting them. And sure enough, there was a virus that rode in, some sort of malware came in.

And so it's not that hard to set these things up so at the start they're secure. But it's really difficult over time to maintain that level, that initial level of vigilance. And I think that's what happens is it's like, well, we'll just connect this up briefly, or I'll just open a port in my firewall just for some specific event, and then we forget. And it stays open, and something crawls in.

Leo: It just seems to me this isn't rocket science. And it should be something that, look, I understand if a bank gets hacked. I understand if a hospital gets hacked. I don't want our military secrets to get hacked. I mean, there are just certain things - or our infrastructure, our grid. I hope we've learned - I think we've learned something here, ladies and gentlemen.

Steve: The experts who were asked about this said that the nature of this breach is such that it arguably makes - whoever it is who received the information would have received enough to do a much better job at defending against what this technology is meant to do for us.

Leo: You know, we did the Blackbird, the stealth bomber, the stealth fighter. All of that was secret. We got the jump on them. We did the Manhattan Project. We got the jump on them.

Steve: Yeah. The good news was there was no Internet back then. I mean, this global network really is a mixed blessing. I mean, I don't have to tell any of our listeners that. It is...

Leo: Well, take it off the freaking public Internet, then, if you can't figure it out. Right?

Steve: And, see, that's the other problem, too, is that because it is - the positive side, the Internet can be so useful that not having connectivity starts to become an increasing problem where it's like, wait a minute, we can't not be on the Internet in order to be in business. I mean...

Leo: Well, couldn't they make - I mean, look. If it's defense contractors, I mean, okay, legitimately they might need a network to see this stuff. But they could do a VPN and secure it and not allow public Internet access. You shouldn't be surfing the 'Net on a machine that has the plans. I mean, it just seems like there's ways to do this. Maybe I'm wrong. Maybe it's not a nontrivial thing.

Steve: I'm sure that this system was far more sophisticated than the typical corporate or home network.

Leo: Oh, I would hope, yeah.

Steve: I mean, there's no doubt about it. But clearly, whatever, however it was this actually happened, and we have no details about the actual details of how this happened, but I'm sure that it was somebody on the outside really wanting to get access to this who spent time working on how to do it. And the problem is that we've seen digital technology is a little more analog than we wish it were. Sure, everything is a one or a zero. But there are ways around absolute protections like firewalls. Which you'd think, okay, that's an absolute protection. It's like, well, yes. But what if the firewall itself, I mean, Cisco just did, as we reported, did a recent patch of IOS. There were a bunch of problems with the Cisco IOS. So anyone who's using defensive technology is depending upon the defensive technology itself to be safe and accurate and correct. But if it's not, and we keep seeing instances where it's not, then that creates a way in. I mean, unfortunately, complexity is the enemy of security. And we do keep making these systems more and more complex. Which makes them harder and harder to secure.

Leo: Yeah, yeah. Okay. I'm going to get - I'm going to calm down here.

Steve: I had something interesting happen that I just wanted to bring up to our users as a little bit of errata. One of the things that Microsoft did in XP that was very nice was they limit file and printer sharing to your local network. That is, it's like, okay, why didn't they do this a long time ago? But it's nice that they did it. So if you look under the firewall configuration, when a default install of Windows XP, that is, everyone's XP, and Vista for that matter, that is in use will have file and printer sharing enabled by default. And what that means is that you've got frightening services listening on ports 137, 139, and 445, the standard really scary ports that Windows has. Well, when you're behind a router, as we know, your network is protected by the router. Nothing can come in through those ports in the normal case. Well, we're going to be hearing about an exception to that when we talk about Conficker in a minute. But normally you're safe.

Well, Microsoft enhanced the protection by not allowing packets to leave the LAN, the Local Area Network. And so file and printer sharing is on by default. But even if you had access to the global Internet, file and printer sharing is protected. So it will not allow traffic to come in from a global address, only from a local address. So that seems fine. Except it occurred to me the other day when I was at Starbucks that, with their change to AT&T - they used to be T-Mobile, they're now AT&T - we lost all encryption. So Starbucks, for example, corporate, is an open WiFi network. You need - there's, like, an intercept page. It's not an open hotspot in that someone could just walk in and get on the Internet. So you've got to - there's all kinds of rigmarole they've got with register your Starbucks cards and so forth. I still have my T-Mobile account, so I'm able to use my T-Mobile account through the AT&T interface. But it's an open, that is, it's a nonencrypted network.

What occurred to me is, ah, right, that means not only is all the traffic sniffable, but everybody's laptop in a given Starbucks location is by definition on the same LAN. Which means all of our file and printer sharing ports are open to each other by default. Which is, for example, exactly what something like Conficker wants because it scans the local network looking to make TCP connections on port 445. So I don't know what people's habits of use of their laptops are. But I wanted to remind people that it is very simple, if you do not need file and printer sharing for your WiFi connection, it's easy to unbind it, is the term, from the WiFi adapter. Leave it bound to your regular hard-wired Ethernet adapter so when your laptop is plugged in, physically plugged in at home, given that you...

Leo: Oh, that's a good idea.

Steve: ...plug it into a wire, there you've got file and printer sharing. But unless you really need file and printer sharing wirelessly, I absolutely strongly recommend that it's just a matter of turning off the checkbox for file and printer sharing, and also to turn off NetBIOS, which is still in there for some reason, turn that off. And then those ports are not open, those protocols are not available to wireless, which is - because you never know where you're going to be hooking into a non-encrypted network. You figure you're secure because you're behind, like, the corporate firewall or a NAT router. But you are participating in a LAN with everybody else on the LAN. So you're implicitly trusting every other machine on the LAN not to be going after yours. And it might well be. One or more might well be.

Leo: Right.

Steve: Certainly that's the case if you've got Conficker anywhere on the LAN, as we'll be discussing.

Leo: That's a nice little fix, simple little thing to do, though. Just disable it on the WiFi.

Steve: Yes, exactly, just unbind it from WiFi. And I have an interesting little report of success from someone named Jerry who sent us email saying, "Just a thank-you note." He said, "Steve, I bought SpinRite v6 last year, and it saved my butt then. But I'm writing to let you know how much I appreciate the product now. You see, I couldn't make a disk image of my laptop's hard disk drive. I kept getting write error or disk full error messages. Well, the disk wasn't full, so it had to be a write error. I ran chkdsk /f and /r, but no errors were found. So I ran SpinRite v6 in mode 2, and it also found no errors, either. A retry of creating a disk image was still unsuccessful. So I turned again to SpinRite and found that I could change the mode, SpinRite's mode, while it was running mode 2. I changed the mode where it reads and rewrites the data. That was just what was needed. I guess some of the data was weakly stored. The data was strong enough to be read properly by chkdsk and SpinRite mode 2, but not strong enough to pass a verification test between it and its image. By having SpinRite I saved myself untold trouble of having to buy another hard disk drive and transferring data to it. Who knows how that would have gone? I just had to let you know how much I appreciate this product. Grace and peace, Jerry."

Leo: Isn't that nice.

Steve: So another happy SpinRite success story.

Leo: Happy SpinRite customer.

Steve: Love those.

Leo: All right, Steve. Let's talk Conficker.

Steve: Now, I want to - I need two ground rules laid first. Or, well, okay, one ground rule and a little bit of technology. I'm sure people understand, they know me well enough to know that when I say I'm impressed by something, it doesn't at all mean that I'm endorsing it or thinking it's a good idea. There is a lot of state-of-the-art impressive technology in Conficker. It's not bleeding edge, by any means. It's not something no one has seen before. It's that somebody who was not your typical script kiddy, who was not taking stuff somebody else did and just sort of mindlessly duplicating it, but whoever is the author or authors of this series of, this genus of worms, because we've had now four of them, and there's maybe a fifth one on the way, they really understand this technology.

So it's certainly the case that anybody who really understands networking, I mean, I could write Conficker. There's any of the smart guys in networking, security companies could write Conficker. I mean, you know, it's not like this is rocket science. But this is unique for what it is, that is, that it is - it's been done in a way that is really reacting in lockstep to the industry's attempts to counteract it.

Leo: How good would you say the guy or guys who wrote this are? I mean, you said you could do it, any competent security professional could do it. But can you look at the coding and say this guy knows what he's doing? Is he a professional? Is he a kid? Do you have any sense of that?

Steve: Yeah, I would say - I guess I don't know, I mean, first of all, somebody can be good at any age. So we don't have any sense for their - there isn't any nonsense in it. For example, some of the early bots had, like, used four-letter-word slang.

Leo: Variables, right, yeah, yeah.

Steve: Which sort of made you think, okay, we were a little maturity-compromised here in this case. There isn't any of that. It's...

Leo: Can you see variable names? You can't, can you?

Steve: Well, no, but...

Leo: We don't have the source code, but you can disassemble it.

Steve: They weren't variable names. They were, like, embedded strings in the executable where it was just like, okay, this is not somebody we need to take too seriously. Although their tools were oftentimes potent. But because they had patched these, patched codes that they got from somewhere else. This is clearly being written by somebody who knows what they're doing. And as I said, by the time we're through discussing this in detail, I think that our listeners are going to have a strong sense for, it's like, okay. In many ways this raises the bar. Conficker has gotten a huge amount of press. It's been dissected by really smart guys. And so there's now, like, okay, anybody else who's going to do a worm is likely going to do everything Conficker does.

Now, it is worth giving this guy credit for several things. There is new technology in this, for example, the way it generates its domain names, that we haven't seen before. It's like what some good guy who, I mean, someone who's really networking-aware who sat down and said, okay, how can I have malicious code scattered around the Internet somehow find a server to update itself and prevent somebody from reverse engineering the code to see what the domain is that I'm going to be contacting?

For example, back when I was tracking down the IRC-driven botnet that was attacking GRC many years ago, I was able to look at the traffic, see what the IRC server was that the bot was contacting. And then I wrote my own pseudo-IRC client and logged into the same channel on the IRC server and watched all the bots talking with sort of my own version of an IRC client. Well, so I was able to do that because there was a static domain name that all of the bots in that particular network were contacting. Well, Conficker doesn't do anything like that. Conficker has a whole - several aspects of next-generation-ness to it. So while the technology is not surprising, the fact that it has been employed is arguably surprising and unique. So that's really what's new.

Okay. The second thing is I need to explain what a "thread" is because Conficker is highly multithreaded. And I realized as I was preparing my notes for what I want to discuss that, I mean, I live in thread land. Threads are one of my favorite abstractions in programming.

Leo: Threads R Us.

Steve: But if people don't understand what a thread is, for me to say, oh, and it spawns three threads to do this, they're going to be, like, what? What's the spawning of a thread?

Leo: Right.

Steve: So a thread is an abstraction of computer execution. Everyone's sort of familiar probably with the notion that a computer does one thing at a time. It executes one little instruction - add two things together - and then maybe another one: Oh, if the result is greater than something, then jump to here, and then do something else. So the point is, as we know, computer programs are one thing at a time. And it's because the computers are very fast that all those little things add up to something substantial like recalculating your spreadsheet or 3D rendering at Disney, I mean, phenomenally amazing stuff comes out of just lots of little additions and multiplications and decisions being made one at a time.

Well, as computer science has evolved, it's been nice to have a program, a single program being able to sort of do more than one thing at a time. Windows had one approach, which is a so-called messaging paradigm, where you'd have a so-called message loop, and it would go and do something, then come back to the message loop and get the next thing to do and go do that and then come back. And so it kind of kept checking back in. Well, that was one way of creating sort of a feeling of asynchronous events.

A different way is through something called a thread. So if you imagine this series of steps I was talking about - doing one thing at a time, add, compare, jump, store, load, one thing - now, if you imagine that's a chain of instructions or so-called a thread of execution, then it's possible to have one thread spawn, that is, start another thread, so it sort of forks into two chains of execution. Now, we know that the computer itself can only be really doing one thing at a time. Now, that's evolved a little bit as we have, like, a multicore processor where we actually have multiple cores. But if we just take the case of a single processor, this model actually works well no matter how many cores you have.

What happens is a thread is going along happily doing its thing, and then it's preempted. That is, the operating system says, okay, you've had enough time. We're going to switch over to another thread, the other thread, for example, if there were two, and let it run for a while. Well, this switching happens so quickly and so often that the effect is that two things are being done at once. And in fact there's no practical limit to how many you can have. At some point, if you have thousands of threads, or maybe tens of thousands of threads, well, switching among them all becomes a problem because it takes so long to get back to any one thread, and then you begin to have some overhead associated with switching threads. So you don't want to have a bazillion.

But something like Conficker is doing many things at once. It's checking to make sure that you're not running antivirus programs, and that's one thread's doing that. So one thread, sort of a separate, a little spawned-off worker, its full-time job for that one thread is making sure that anything that you start up that might be used to shut it down doesn't have a chance to get going. Then another thread is camped out on some listening TCP and UDP sockets, actually one thread per socket, so that if anything comes in and attempts to establish a connection, that thread will wake up and say, oh, hi, glad to see you, come on in, send me your data, and we'll see what's going to go on.

So I wanted to explain that's what a thread is. It's a beautiful abstraction. I call it an abstraction because, in the case of a single processor core, the processor core is only doing one thing at a time. And so in Windows we've got multiple applications, and the multiple applications probably have multiple threads. So this one processor is jumping all over the place, not only between individual applications, but between parts of the application where each part is a thread. And again, it's because the system is so fast that it all sort of seems like everything's moving forward and alive and running simultaneously, when in fact literally it's timesharing. So this thread-jumping is a sort of a form of time-sharing within a single application.

What's cool about multiple cores is, if the system has the job, like a contemporary operating system has a bunch of applications, and they all have a bunch of threads, well, then, the unit of execution is the thread. And if you've got four cores, like a quad-core processor, well, you can literally be doing four things at once. So it scales very nicely. You add cores, and instead of having one processor that's madly flying around, trying to keep all of the threads moving forward by giving them all a little slice of time, now you actually have two or four cores that are able to simultaneously be running from this myriad of threads in the system, pushing them all forward in time. So it's a nice way of actually leveraging additional processing power.

Okay. So with that bit of foundation, we know that Microsoft identified and patched on October 23 of 2008 a flaw in Windows which was one of the many dreaded remote execution flaws, meaning that, if you had an open port, and your computer was just sitting there with this port exposed, a packet could come into the port, and this is a TCP connection over port 445, which would create a connection to the so-called RPC service, the Remote Procedure Call. And it was then able to take advantage of a small defect in Windows that would cause the payload that it provided with a packet to be executed.

And in the case of Conficker, what Conficker did with this packet was it actually caused the computer that had received this packet to open a reverse connection in the other direction, back to the IP provided in the packet, and establish a connection to a service that Conficker was also running in that attacking machine that would cause the victim to download all of Conficker. So the first thing was not Conficker. That first arriving infection was not Conficker. It was just - it was a packet that only had enough code in it to cause that victim machine to reach out and essentially download Conficker from that source target.

So, okay. Right off the bat, a number of machines are going to be protected. First of all, since port 445 has been a source of so many horror stories through Windows history, I mean, it is the Windows file and printer sharing port. And many other things are overloaded on that port. Many other services are available. So it's a ripe port for exploitation. The good news is, many ISPs have responded by blocking it at their own borders, so no 445 traffic is able to transit into the ISP's network.

Now, it's not clear whether you are blocked from other systems within the ISP's network. That is, it's not clear how fine-grained that blocking is. It's not clear that somebody nearby, like literally on your block if you're using a cable modem, would not be able to reach your 445 port from their machine. But it is the case, for example, that many ISPs are blocking incoming traffic from further out on the Internet into their internal customer network. So that would prevent incoming infections. Also, any properly configured NAT router would prevent incoming connections. And I say "properly configured" because how many times, Leo, have you and I told people, begged them, advised them, implored them to disable...

Leo: Universal Plug and Play.

Steve: ...Universal Plug and Play.

Leo: So this opens it up?

Steve: Conficker does. Conficker is a Universal Plug and Play client which will reach out and open incoming ports through your firewall and router if you have not disabled Universal Plug and Play. So it's a perfect example of why Universal Plug and Play was a really bad idea from a security standpoint.

Leo: So just to underscore this, we say a router is a firewall. It is a firewall. It will protect you. Except that, if something does get on your system, and you allow Universal Plug and Play, it just opens the ports and says c'mon in, guys.

Steve: Right. Right. Universal Plug and Play allows you to do, through a network protocol, all the kinds of things you can do through the user interface on the router, like open static ports and set up a DMZ. And in fact the Universal Plug and Play interface is even more powerful than what is surfaced on the web-based user interface of most...

Leo: Really.

Steve: ...of standard consumer routers.

Leo: Wow. Wow. And without warning, without any notice, it just does it.

Steve: Right. Completely silent. No pop-ups. No security. No passwords. I mean, this was just a ridiculously insecure thing for - Microsoft of course was pushing it because it was part of their - plug and play was a prior technology that we saw for many years in Windows that allowed Windows to recognize when you put something in. It's like, oh, look, la new piece of hardware has appeared. Let me go find a driver for it if I can. That was Plug and Play. And this was Universal Plug and Play that was sort of awkwardly named because it is a completely different technology. But it was the same goal. It was to allow discoverability so that...

Leo: Universal open my ports so I'm insecure. That's what they should have called it, yeah.

Steve: Yeah, exactly. The idea was it would be a zero-configuration sort of thing, so that if you ran some software on your computer that was intended to automatically configure your firewall or your router, it would be able to send out a broadcast to your network and say, hi there, got any routers out there? And the router would, through Universal Plug and Play, say oh, yeah, hey, I'm over here. And then the malware, if it was in this case malicious, would say, oh, good. Lower your shields, please.

Leo: Yeah. Let me in. Let my friends in. Let 'em all in. Wow.

Steve: Yeah.

Leo: Well, but it's interesting, it's almost like the guys who wrote this listen to this show.

Steve: Well, they're definitely...

Leo: They're up on security.

Steve: ...tuned in. Again, this is taking advantage of every available facility. Now, it's worth explaining also, just to make sure, just another definition, that we understand the difference between a worm and a virus. Because this is a worm inasmuch as that, if left alone, it would infect all the machines on the Internet that are infectable. That is, it needs no user interaction at all. Once it's launched onto the 'Net, it finds vulnerable hosts, infects them with no user interaction, and they turn around and start trying to infect others.

Now, one thing that's different about Conficker than, for example, MSBlast or Code Red, is those worms, we may remember, really brought down or seriously challenged big chunks of the Internet because they were so rapidly reproducing. They were pouring packets out as fast as they could. So it had two consequences. One was they tended to rapidly find other infectable machines and infect them. But also it was like little local denial of service attacks. And so if a network had a whole bunch of Code Red infected in it, it would pretty much go off the 'Net, just because its own infections were so actively trying to find other machines.

By comparison, Conficker is very patient. In my own instance of it here, and I've seen this confirmed in other analysis, it sends maybe, oh, three to four packets a second. Which, compared to what it could be doing, is really slow. I mean, it's very patient. It just sort of pokes away.

Leo: Is that so that you won't notice it, that it's using a lot of bandwidth?

Steve: Yes. I can't see any real other advantage. For example, mine's been running...

Leo: One thing people do to see if they're infected is they look at the lights on their router. And if it's flickering when nothing's going on, they go, oh, somebody's using my connection.

Steve: Right. If it's going crazy. On the other hand, mine, I'm using a hub so that I'm able to monitor Conficker with another machine. And, I mean, if I look at the lights, it's going blink blink, blink blink, blink, blink blink blink, you know, it's...

Leo: Nothing to worry about. Normal.

Steve: Just a few packets a second. But it's not going [jackhammer sound], just like crazy. And that's what we have seen in the case of other malware infections. So it really - it's staying under the radar that way. It also does make it less easy to find the clients. Anything that's out there, malware which is pouring traffic out at random IPs, its own IP is going to end up being known by anybody who's curious because all they have to do is put a packet monitor on a block of IPs, and they're going to see all of this searching traffic coming into that block of IPs, that is, from all of the different infected machines on the Internet that are searching for other machines to infect. So by being much more slow about this, although it means that it's going to be slower to find another machine, it also kind of keeps it under the radar.

Leo: It's got all the time in the world.

Steve: I've got to say, too, that as we go through the way these Conficker variations have changed over time, there's this desire to find meaning in these changes. It's like, okay, what's the guy thinking? Why has it done this? For example, Conficker A would immediately abort if the keyboard layout of the computer it had entered was Ukrainian. So it would check the keyboard layout. And if it was a Ukrainian layout keyboard, it would not infect.

Leo: That makes you think it might be a Ukrainian that spread it.

Steve: Yes. There are several reasons to believe that. There's one company in particular, Baka Software, B-a-k-a, is a well-known sort of shady operator who's been responsible for all kinds of mischief in the past. There was one connection that was caught by some folks that were analyzing Conficker. And they set up a big honey net in order to look at traffic patterns and levels of activity on the Internet. There was one packet that they found where it was Conficker B that was - I'm trying to remember. It was a cross-version packet. So it was Conficker B that was set up to infect Conficker A. And that's never the case in any version of Conficker, that is, the versions always, with A and B, they had defenses against anyone malicious taking them over.

Leo: Interesting.

Steve: So A would use A's protocol to spread version A. B would use B's variant protocol to spread version B. Well, this was one connection that was deliberately using A's protocol to spread version B. So it would be upgrading A's to B. And it happened that that came in from a block that is known to be used by this Baka Software Group in the Ukraine. So there's some reason to suspect that there's some connection there.

Leo: Interesting.

Steve: But again, unfortunately so much of this is just - it's, you know, you're having to divine intent and what's behind the design decisions that are being made. Okay. So we know how it originally started. It originally started by taking advantage of this vulnerability which was patched on October 23rd. I think it was November 10th, so not long afterwards was the first appearance of Conficker A, the first variation of Conficker, which says that a lot of this code was already ready. That is, we hadn't seen this worm before. But this is too much for someone to write in 17 days, or 18 days, between the 23rd and the 10th of November. So, I mean, and a lot of this had to be perfected. When we get a sense for the technology in here, you'll see that it's just way too much. So somebody had this and was waiting for a vulnerability to surface.

Now, of course, the embarrassment is that the patch was issued on the second Tuesday - oh, actually not. In this case it's October 23rd, so it was an out-of-cycle patch, not the second Tuesday of the month, that Microsoft patched it because they recognized this was important enough to talk about. And we talked about it on Security Now!, of course, back then because this is, you know, you don't want to leave any wide-open remote wormable exploits available for any longer than you have to. So Microsoft did an out-of-cycle patch to close this. And still, months later, many months later, there are machines that have not been patched. And as we were talking about this before, it seems that an analysis of the population shows that the density of Conficker infections, which are determinable by looking at the incoming IPs into a honeynet, that is, a block of IPs that have set up sort of like an Internet telescope in order to see what's going out on the Internet, the incoming IPs generally are highest in concentration in geographic regions of the world where piracy is more prevalent. So it does look like there's a correlation between unpatched machines and pirated copies of Windows.

So the actual payload of Conficker is it's a DLL, a Dynamic Link Library, which is first compressed using a well-known compression tool, UPX. It's the one I use myself. It's a very nice executable packing program that makes Windows executables much smaller because the format that Microsoft designed is inefficient in terms of the EXE size. So it's possible to use standard compression techniques to make it much smaller. Then it is further obfuscated so that, even if you decompress the EXE, it still doesn't look like regular code. It needs to get into RAM, and then it sort of self-decrypts itself.

So in order to do an analysis of it, it's necessary to actually load it into memory and then take a snapshot of memory in order to see what's going. It installs itself in the svchost.exe process. Anyone who's used Windows and is security aware has looked at their list of running processes, and they'll see a bunch of svchost.exe. The idea is that in Windows an executable program, you always sort of have to have an EXE as an anchor. But then the EXE can either have with it or can load dynamic link libraries into that executable process space.

So what Conficker does is it injects itself into an existing instance of svchost.exe by injecting a thread and causing the thread to run load library, that loads the DLL into the process. It does a number of other clever things. For example, the way a DLL, a dynamic link library loads is there's an initialization sort of stub at the beginning of the DLL that Windows calls in order to let the DLL set itself up and sort of do an internal housekeeping. That stub is always returned from. And after that returns, there's like a return code, success or fail. So it's possible for the DLL to say, whoops, whatever it is I needed I didn't find here, so terminate me. Do not load me. Or the DLL is able to say, hey, everything's fine, I'm ready to stay resident here in this process. So Windows waits for that return in order to list the DLL among those that are part of this process.

Well, Conficker, cleverly, never returns from that initialization. It accepts the fact that it's running, and it spawns a bunch of threads to do all kinds of things, never goes back to Windows. So Windows never lists it as a DLL that's part of the process. And it's one of the ways that Conficker stays invisible. It also has a null string name when it registers as a process. It does so with an empty string name, and it flags itself as "Make me invisible," which is one of the status bits that a process is able to set. So again, it works on remaining sort of off the radar.

Leo: But it's not a rootkit, though, is it?

Steve: Well, I wouldn't call it a rootkit. But it does a lot of things in order to hide. I'm going to run through a bunch of the specific things it does to hide. And it's also evolved over time. One of the things that it needs to do is it needs to know its public IP. If it has infected a machine behind a NAT router, the only IP it has is, like, 192.168.1.1 or 1.5 or whatever, you know, the nonroutable IP. But when it sends its packet out to infect another machine, and the way the infection works, is that a reverse connection is made from the victim back to the attacking machine. The attacking machine has to know the public IP.

So get a load of this. It uses well-known IP-checking sites. It connects to getmyip.org or getmyip.co.uk or checkip.dyndns.org. It knows all - the A variant knows all three of those. So it chooses one or two at random in order - or it actually chooses them until it gets the answer that it's looking for, and uses that remote site whose job is to tell you your IP, it parses the returned page to get the IP that is public for its router. It also downloads a geographic IP database from Maxmind.com. The GeoIP database relates IPs to locations. And it uses that in order to avoid attacking any IPs in the Ukraine.

Leo: Again the Ukraine. Again the Ukraine.

Steve: Yes. So when it's generating random IPs, it carefully filters out any Ukrainian IPs. Now...

Leo: Now, if I were writing a virus, and I lived in the Ukraine, that would be a very handy thing to make sure I didn't infect myself, my friends and family with my virus.

Steve: And to make sure you don't upset the local authorities.

Leo: Oh, yeah, because that's the jurisdiction I'm in, isn't it.

Steve: Exactly.

Leo: Oh, very good point.

Steve: And we know that it's much harder to get cross-country cooperation than it is to upset the police station around the block. And so it's been, again, it's been surmised that they're not attacking anybody in the Ukraine because they don't want to rouse the local authorities. Which again I think is very clever.

Leo: That's smart, yeah.

Steve: Now, one of the new technologies that we have not seen in previous worms, that the A variant of Conficker starts, is this notion of using a pseudorandom number - essentially it's a pseudorandom number generator that maps to pseudorandom domain names. Conficker version A, and this is one aspect that has changed a lot because it was one area where it was vulnerable to being blocked, Conficker version A every day would generate 250 domain names based upon the UTC date. It would get the UTC date by querying from among a large number of well-known public websites. One of the headers that comes back when you request a page is the current date and time in universal time. So that way it knew sort of globally, that way all the Conficker instances all over the world would be synchronized to the same UTC date, which would mean that on a given day they would all use the date to seed the pseudorandom number generator which was used to generate domain names. And the A variant of Conficker would generate 250 domain names based on that pseudorandom generator and then perform DNS lookups to look up the IP of that domain name using the standard public DNS system, and then attempt to make a connection on port 80 to a server, a web server because it's port 80, a web server running on that port. If it succeeded in downloading a binary file, it would then go through a substantial process, which I'll describe in a second, to verify the validity of that file.

Now, the B variant made some changes. For example, the B variant dispensed with the keyboard detection so it would no longer abort if you had a Ukrainian keyboard layout defined in Windows. But it still did the GeoIP data in order to filter out Ukrainian IPs. And that has remained to this day. So Conficker really doesn't want to upset, apparently, the Ukrainian authorities. B also began the task of terminating many popular antivirus, and it began blocking DNS lookups to prevent you from going to, like, Symantec or Microsoft or Windows Update to do things that were related to maybe wondering if your computer might be infected or, finally, getting the update that would cure this problem.

On the other hand, all versions of Conficker have closed the door behind them. Conficker got in by using this MS, what is it, 08 dash - can't remember the number. It's, like, dash 68 or something, Microsoft's ID for this exploit. What they did was they would modify, by patching in memory, they would modify the vulnerability so that only they could subsequently use it to prevent somebody else from coming up with something malicious that would knock Conficker out of the system. So after they got in, they didn't completely close the door, but they filtered any other incoming traffic to make sure that it was them. So, I mean, a lot of thought was given to this.

B also began to incorporate extensive anti-debugging and anti-reverse-engineering defenses. This is technology that's been known and done for years, a lot of it in the hacking community, to prevent malware from being reverse engineered. So these concepts were not new. But it's another layer of defense that Conficker was employing. For example, it's possible for software to tell if it is being single-stepped through, which is one of the typical things you do when you're reverse-engineering code. You go step at a time in order to see what the code is going to do, sort of running it under supervision. But that always messes up, of course, the timing. So not only timing, but there are other means that can be used to see if breakpoints are being set in the code.

So there's much that could be done for code to protect itself against analysis, and Conficker does a lot of that. It uses an additional set of public services to determine its IP. It uses getmyip.org, also whatsmyipaddress.com, whatismyip.org, and additionally, as did version A, checkip.dyndns.org. So there has been some evolution and variation in Conficker's behavior over time. Oh, and whereas the A variant went to Maxmind.com to load the GeoIP filter, version A incorporates it internally. It uses RAR to compress...

Leo: You mean version C.

Steve: I'm sorry. No, no, B.

Leo: B.

Steve: Version B uses - it incorporates the GeoIP list internally. It uses RAR to compress it, and then RC4 to encrypt it. And so it's part of the payload, it's built into the body of Conficker version B.

Leo: Does it seem sensible, it sounds to me like this is the case, that it's the same guy doing all three versions?

Steve: Oh, yeah, yeah. There's no doubt that this is the same - that it's the same guy. And it's clear that it's, wow, look how A is succeeding. I can make it even better. So it's probably some notion of, wow, you know, it's succeeded beyond my wildest imagination. Now I'm motivated to put more time and energy into it. The way...

Leo: Oh, great.

Steve: Yeah, exactly. The way Conficker protects itself is really interesting, also. I had mentioned before that it digitally signs itself. So when anything is going to be accepted by an existing version, like an upgrade to Conficker, taking it from A to B or B to C or beyond C potentially, and that appears to be happening now, the block of executable code is hashed using a digital signature algorithm. A used SHA-1. B, Conficker B, used MD6. What's interesting is that Ron Rivest of RSA, who designed MD6, publicly disclosed and announced and released MD6 just two weeks before Conficker B incorporated it. So...

Leo: Wow. Wow. That's amazing. I mean, this sounds like this guy's, like, a genius.

Steve: Well, he's in the game.

Leo: Yeah.

Steve: I mean, he's actively watching what's going on in the industry and, on some level, participating. A little side note, the very first release of MD6 had a bug. There was a buffer overrun glitch in MD6 which was found and corrected. This guy was so quick to get MD6 into Conficker B that he incorporated that bug, although the nature of the way it's used would not allow anyone to take advantage of that in order to, like, take over Conficker. So it didn't represent a weakness in his case.

Okay. So the code is hashed to create a 512-bit hash. That hash is used as the key, the symmetric encryption key for the RC4 stream cipher that we've talked about at length in previous podcasts. RC4, you'll remember, was the cipher used in WEP encryption, which when used wrong is a bad thing. In this case it's used in a sort of a noncritical fashion. So the 512-bit hash is used as the key to encrypt the binary. Then it is signed using public key encryption. The hash is raised to the power of a private key, taken mod n to create a signature. And that signature is appended to the end of the package.

That's the package, then, which is sent to a potential recipient version of Conficker. So it reverses the process. It takes the public key which it contains, raises the signature to that value mod n, and due to the miracle of public key encryption, that produces the hash. So it then uses the hash to decrypt using RC4. Remember that RC4 is just - it's just a pseudorandom stream. So it generates the same pseudorandom stream as was used to encrypt it, XORs the stream with the body of the payload, and that produces decryption. It then hashes that and compares it to the original hash. Only if they match does it know it was signed by somebody who had the private key, meaning the author, and nobody else is ever going to have that, so only the author is able to produce new payloads which would be injected into the Conficker system. The A variant uses a 1KB RSA modulus. The B variant uses a 4K. Again, just because why not? 1K was good enough; 4K, well, that's even better.

So, you know, you begin to get a sense for the amount of technology, I mean, state-of-the-art crypto technology which is in this and is serving the purpose of keeping this thing alive, preventing it from being commandeered, and maintaining this mysterious owner of this thing in control of this network.

Now, so we talked about how domains are being generated. 250 domains per day were generated by the A variant. But they only had the top-level domains of .com, .net, .org, .info, and .biz. So, you know, pretty much the five most popular domains. The problem is that 250 a day in those top-level domains, it was easy for the so-called Conficker Cabal, the anti-Conficker folks, the white hats who were trying to protect us from this, it was easy for them to generate the 250 domains for tomorrow and the day after and the day after and go preregister them so that they were able to block Conficker from being able to expand.

Well, the first thing that happened - I'm sorry, to block Conficker from being able to basically phone home in order to get an update to itself. Because the idea would be that the malware author would go and register a domain sometime in, like, a domain that would be used by Conficker next week. They'd register the domain and set up a web server, point that DNS address to some IP that they controlled, set up a web server there with a signed package, signed using the technology we just talked about, and so then what would happen is on that day all the Conficker worms that knew what day it was would generate 250 domain names and try them all. One of them would be a hit, and it would only take one. They would find that one, look up the IP, connect to that TCP server and download a binary payload, use their public key to verify the signature and to generate the hash used for decrypting it, verify that, and run the code.

So, I mean, lots of technology here. B added three more top level domains to that approach: .ws, .cn, and .cc. The other thing that B did was it expanded the domains that it uses for determining the date. It used W3.org, Ask.com, MSN.com, Yahoo.com, Google.com, and Baidu.com, B-a-i-d-u.

Leo: Yeah, that's like a Google for China, Baidu, yeah.

Steve: Right, right. Okay. Then the big change that we talked about several weeks ago was the one that also made a lot of press, unfortunately. I mean, a lot of sort of Y2K scare stuff was what would happen with Conficker on April Fools Day, on April 1st, because the C variant was designed to have its behavior change. We did talk about this...

Leo: Is the C variant the one that you've been using, or you've been playing with?

Steve: Yes, C variant is the one I've got. And...

Leo: It ran, but did it ever get any data on April Fools?

Steve: No. Well, no. What happened was its behavior changed. It suddenly began querying, it began generating 50,000 domains, up from 250, so to 50,000, from which 500 would be randomly selected. And not only that, but whereas A, the A variant used the five most popular top level domains, and the B variant added those three more - ws, cn, and cc - the C variant uses 110 different TLDs. I mean, just about everything you can think of. And that creates a huge problem because these are TLDs literally spread globally and under the control of a phenomenal number of registrars. Beforehand, all you had to deal with was the registrars who were registrars for .com, .net, .org, .info, and .biz; and then later ws, cn, and cc. Now, if you're going to preemptively register, you've got a big problem. Not only do you have to preemptively register 50,000 domain names per day, but you've got to do them with all the registrars controlling these 110 possible top level domains. So with the C variant this whole notion of - this cat and mouse, basically, really got escalated. Oh, it was MS08-067 was the original variant, the original vulnerability which was being used for exploitation. So...

Leo: So...

Steve: I'm sorry, go ahead.

Leo: Go ahead, no, no.

Steve: Okay. So, okay. So A propagated. So we have the way Conficker phones home by generating domains and trying to contact a server at a pseudorandomly generated domain name which has been, you know, is knowable in advance, but has been made much more complex after April 1st when we switched to variant C. The only way that the A variant caused infections was the same way it got infected, that is, it would send out the packet, the so-called Server Message Block, an SMB protocol packet, to port 445 using the TCP protocol. It would connect to the server surface and take advantage of the vulnerability. It was the original vulnerability that was supposed to be fixed. So a machine that got infected that way would attempt to reinfect other machines. And what it would do is it would just generate IPs at random and attempt to make a port 445 connection to them, although it would avoid the Ukraine, any IPs that were physically geographically located in the Ukraine.

The B variant added two additional propagation techniques which, oddly, were removed from C. So it hasn't always been escalating. It's like its behavior has been changing for one reason or another. And in fact C is less effective because these other two approaches were removed. B would use NetBIOS shares to propagate. It would look for other machines on the Local Area Network, and then it contained a list of 240 common passwords. And so it would attempt to connect to any other machines, any other shares on other machines, and get into them that way. And, oh, and this had an interesting side effect, too, because within corporate IT where policies could be enforced, if Conficker got in and began scanning the network, finding machines and attempting to log into them and guessing wrong, that would trigger the account lockout policies. And so the actual users were unable to log into their machine because their machine would say, sorry, you've had too many failed login attempts. You're locked out until you talk to your IT administrator.

But B did something else. If it saw a removable drive arrive or in the system when it got there, it would copy itself to the removable drive and edit the autorun.inf file to cause itself to be run whenever that drive was plugged in somewhere else. So it would use USB propagation in order to move from one system to another. So C removed those two other strategies, but also changed - it added another approach using the SMB, essentially, using a so-called "named pipe," which is one of the APIs in Windows that allows you to essentially connect a - to establish a connection between two machines anywhere on the Internet and send data back and forth through the so-called "pipe," which is really just a connection between those.

So in terms of hiding itself, Conficker has always gone to some extremes to hide itself. It would give itself a random name in the Windows system32 directory and set its time and date stamp to the same time and date as kernel32, which was clever because with Service Pack updates and security updates there's normally a bunch of things that are going to have the same time and date stamp. But rather than, for example, not doing that, one of the first things people who are, like, used to looking for malware in machines do is they'll sort the directory listing by date and time and look for the most recent changes in the directory, thinking that something may have, if it got into the system recently, it'll have a current date and time stamp. Well, Conficker says, not so fast. We're going to set our own file date to the same thing that we know a lot of other files will be set for, which is the data and time of kernel32.dll.

It also sets up multiple threads. One thread provides a constant security service disable, so that if any security services are running, like Windows Update, it'll shut that down, or any of the other third-party services. It looks at a whole bunch. There's, like, autoruns, avenger, confick and downad are both cleanup utilities, filemon, hotfix, regmon, tcpview, wireshark, it knows about all these different processes and terminates them if you try to run them. It also is getting very smart about the IPs that are returned from DNS lookups. If any DNS lookup returns more than one IP, it says, eh, I don't think so, and it just ignores it. Or if it's a stub IP, like 127.0.0.1, which is a localhost IP, that's something that, for example, I imagine the antimalware guys were doing was they would register, instead of sending Conficker off to some other server, they may have been setting them up as just setting the IP to 127.0.0.1, causing it to try to connect to itself, which would fail, but it was just sort of a nice way of stubbing that lookup.

Well, Conficker over time became smart. It also maintains blacklisted addresses. And if it ever got an IP from one DNS lookup that it got from another DNS lookup, it would note the collision and not bother to connect to that same IP. So the other behavior that the good guys might have had is to aim, when they were, like, preregistering all these IPs, they would aim them, like, at some monitoring location, saying, okay, we're going to preregister all these to XYZ Internet address. Well, later on Conficker began remembering all IPs it had received. And if it ever got the same one a second time, it says, well, I already contacted that, and I don't want to be tricked, because it knew that its own secret phone home IP would only be listed at one DNS. Or that it would have contacted it, and there's no reason to contact it again. So it was getting smarter over time.

It also had a long list of /8 networks, that is, the first byte of an IP address. It knew that, like, 0, 1, 2, 5, 10, 14, 23, 27, 31, 36, on and on, are invalid IPs. And we know that, for example, 5 is an invalid - anything starting with 5 because that was what the Hamachi peer-to-peer system used because it was IP space that had never been allocated. So Conficker was evolving over time, making a better use of the resources that it had. And it's just a very robust, strong piece of malware.

I wanted to read the final paragraph in a report from SRI International that did an analysis of this. They said: "Conficker C is in fact a robust and secure distribution utility for distributing malicious content and binaries to millions of computers across the Internet. This utility incorporates a potent arsenal of methods to defend itself from security products, updates, and diagnosis tools. It further demonstrates the rapid development pace at which Conficker's authors are maintaining their current foothold on a large number of Internet-connected hosts. Further, if organized into a coordinated offensive weapon, this multimillion-node botnet poses a serious and dire threat to the Internet."

Leo: Wow. Well, it sounds like it does. Now, I was just looking, and I saw that, at least according to Symantec, that some Cs had updated themselves to E after the April Fools thing, in the last couple of days.

Steve: Yes. That was what finally motivated me to install my own Conficker in a honeypot.

Leo: Are you letting it update?

Steve: Yes. Yes. I'm not allowing it to attack anybody else, but I am hoping it's going to go - I want to see it discover somebody and get itself updated.

Leo: Oh, interesting.

Steve: It was shortly after April 1st. The news went around the security community that there was an encrypted package that was being acquired by C. And so it was like, okay. At that point I thought this thing's not going away. I've got to get in the game and be watching it myself.

Leo: Very interesting, I've got to say. It's very sophisticated. Do you think it's a team of people? Must be a team of people.

Steve: I don't think so.

Leo: Could be one guy.

Steve: I mean, not necessarily. One person could easily do this. One smart network-aware author could easily do this.

Leo: Could be his life's work. His great achievement.

Steve: Well, who knows what the goal is, or the plan.

Leo: The E variant put one of those creepy antivirus things on there; right?

Steve: Yes, I was going to say there is now some scareware that is being downloaded by the most recent Conficker. And so it may be that they have decided, well, we might as well commercialize this now because we've established ourselves. We're in millions of machines worldwide. Basically we've got a technology that can live as long as it's able to. We're able to give it encrypted, digitally signed payloads anytime by just knowing which domain we want to register ahead of time, grabbing that, aiming it at our web server, and some percentage of Confickers will find it. The ones that don't will be establishing a peer-to-peer network among themselves, and they'll be able to pass it back and forth that way.

So we've got two things. We've got Confickers interlinked through a peer-to-peer network. That is, they're sending these packets out, trying to find other copies of themselves, that interlinks them in this peer-to-peer network. And they're also periodically, daily, attempting to use DNS lookups on pseudorandom number-generated domain names to basically phone home in order to get payload updates that way. So it's a very sophisticated network designed to survive what anybody tries to do to it to shut it down. And since it's using public key encryption for its digital signatures, we don't know the private key. We cannot know the private key. Only the author knows. And so that prevents anybody who might want to, even good guys, from taking advantage of that and leveraging that in order to somehow deal with this problem.

Leo: It's really an interesting study, isn't it. I mean, this is...

Steve: It is. It's a perfect case study in how the technology can be used by a sophisticated author creating a sophisticated, state-of-the-art piece of malicious software. I mean, there really isn't anything that this guy hasn't, or team hasn't come up with. And the other thing is, thanks to the fact that they've got this dynamic update facility, they're able to respond to what the industry does. And that's what we've seen them do. Anything that the Conficker Cabal have come up with in order to thwart Conficker, the author said, okay, fine, I'll just bump the domain names up from 250 a day to 500 a day, chosen from a set of 50,000. Let's see you preregister all of those in 110 different top levels.

Leo: I guess the other thing we learn from this is how to protect ourselves. I mean, it's demonstrating all the holes, all the things that you might want to be paying attention to.

Steve: Well, yeah, I mean...

Leo: Like Autorun and Universal Plug and Play, I mean, there's a lot of...

Steve: What I liked about it is that everything we've talked about in the approaching four years of this podcast are things that, if our listeners were diligent about doing, would be one less way that they could be bitten by this.

Leo: Right, right.

Steve: Because if they've got Universal Plug and Play disabled they're going to be in better shape. And if they've got Autorun disabled on removable drives they're going to be in better shape. So, yeah, I mean, it's an example of why security matters and how you can be protected by security.

Leo: It surely does, Ollie. Well, thank you. Very interesting expos. You could read more about this on Steve's website. Security Now! is at GRC.com/securitynow. Show notes there, 16KB versions, full transcript of all shows, all online at GRC.com, along with SpinRite, the world's best hard drive maintenance and recovery utility, and all of Steve's freebies, too. There's lots of great free stuff there.

Steve: More stuff coming soon.

Leo: Very good. Next week, Q&A.

Steve: Yup.

Leo: Submit your questions to:

Steve: GRC.com/feedback.

Leo: There you go. Thanks, Steve. I appreciate it. This is kind of like a ghost story. You scared me.

Steve: It's all true, too. I mean, it is impressive. This author or team have really done something. And now the question is, what's it going to do next? I mean, this thing is alive. It's a creature of the Internet. What's it going to do next?

Leo: Thank you, Steve Gibson. We'll see you next Thursday on Security Now!.