The Windows MetaFile Backdoor?

Description: Leo and I carefully examine the operation of the recently patched Windows MetaFile vulnerability. I describe exactly how it works in an effort to explain why it doesn't have the feeling of another Microsoft "coding error." It has the feeling of something that Microsoft deliberately designed into Windows. Given the nature of what it is, this would make it a remote code execution "backdoor." We will likely never know if this was the case, but the forensic evidence appears to be quite compelling.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-022.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-022-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 22, for January 12, 2006: The Windows MetaFile Backdoor.

Leo Laporte here with Steve Gibson. We're back again. And once again, we're going to put off our discussion of how the Internet works because we've got a blockbuster story for you tonight.

Steve Gibson: Yeah. We have to delay again. Again, you know, we'll always do what needs to be done and then get back to our regularly scheduled programming.

Leo: Okay, so first let's just circle back to previous episodes and catch up with anything we left out there. Anything you want to update?

Steve: Yeah. There were a couple things sort of in my category of errata. Someone made the point that I refer to hackers as "hackers," rather than as "crackers."

Leo: Oh, we get this every time.

Steve: I know.

Leo: I get this so often. Let's address it.

Steve: Yeah, exactly. So I wanted to say something about it. You know, for me, the term "cracker" just, I don't know, it sounds like, you know, what African Americans refer to white guys as, or like a saltine. It just doesn't seem serious and evil. And so, you know, I try to say "malicious hacker." But, you know, on the fly, you know, when you and I are talking, I just say "hacker." And...

Leo: The point these guys are trying to make is that the hacking profession is an old and honorable one, and the Internet wouldn't exist without hackers, UNIX wouldn't exist without hackers, GNU - I mean, hacking is not, in and of itself, bad. And so when we talk about bad guys as "hackers," they feel like we're besmirching the hacking community.

Steve: Exactly. And, you know, I mean, well, it's two things. It's a shorthand; and also, whether these people, these purists, the hacker purists like it or not, the definition, sort of the publicly accepted definition of "hackers" is changing to the dark side.

Leo: Yeah.

Steve: So, for example, when I talk to people not in the industry, you know, who are not, like, saying, wait a minute, do you mean a malicious hacker or one of those good guy hackers, I mean, they know that hackers are bad. And so...

Leo: I think you can tell. English is a very subtle language. You can tell from the context. We don't have to tell you we're talking about good hackers or bad hackers. There are both. And just listen to what we're saying, and I think it'll be obvious. What I say when people - and I get this all the time still. And I just say, look, we're trying to communicate in a language, a common language. And if most people don't understand "cracker" and they do understand "hacker," that's the word we're going to use. You're just going to - we're going to have to trust you and your good brains to figure out what kind we're talking about.

Steve: Oh, and Leo, if we started saying "cracker" all the time...

Leo: I'm sorry, it's not going to work.

Steve: ...we'd cause a lot more confusion than just saying "hacker."

Leo: It'd be very confusing. Saltines? What are they talking about?

Steve: Yeah. Okay...

Leo: So we know there are good hackers. I consider Steve a hacker of the...

Steve: Yeah.

Leo: ...best kind. I mean, you hack away at code. And I've done a little hacking in my time. But neither of us has, to my knowledge, anyway, cracked into other people's computers.

Steve: No.

Leo: That's a different kind of hacker, and you know...

Steve: Okay. The second thing here in errata category is many people followed the advice from two weeks ago, which was Microsoft's initial advice about this Windows MetaFile vulnerability, of unregistering that shell image viewer DLL, shimgvw.dll. And after Microsoft's patch came out last week - early, as we know - they couldn't find the instructions for reregistering. So, I mean, it's simple to do. It's regsvr32 shimgvw.dll does it. In fact, it's probably in your Run command. If you scroll down, you'll see the one with the -u, which means unregister. Just remove the -u, and it reregisters it. But I've also given the complete instructions, which were on the show notes for 20. I've moved just the reregistration part over to the show notes for today's show, for Episode 22. So anybody who wants to get it directly from the web page, just go to the show notes for 22. And Leo, I imagine you might want to put that on your page, too.

Leo: I will.

Steve: Because lots of people have said, hey, I've lost all my thumbnails. Now what do I do?

Leo: And that was a bad fix from Microsoft. It didn't really fix the problem, and there's a better fix. In fact, Ilfak's fix was better, and now Microsoft's fix. And by the way, does Microsoft's fix work still? I mean, are we still happy with it?

Steve: Oh, yes. What they did was to remove this feature, which is what I think it actually is, from Windows after it was discovered.

Leo: Now, let's - you just said something, and I think we can move on. Are you ready to move on to the...

Steve: Yeah, yeah.

Leo: Okay. Let's talk about when you said "what I think it really is."

Steve: Well, remember that last week the way things ended was we weren't sure whether Microsoft was going to fix the earlier versions of Windows. They were saying that they were offering a fix for Windows 2000, XP, 64, or 2003, but not for the older versions of Windows because the problem wasn't as bad on those machines. They said there is a problem there, but it's not as bad, whatever that means. And so I made the statement, hey, you know, if Microsoft ends up not fixing this, I'm going to fix it because, you know, who wants to be using a machine, no matter how old it is, where you go to a website and display a bad image and get your machine taken over. That's not okay, even though it takes more, apparently, on those machines to display a Windows MetaFile image. You have to have, you know, something needs to have registered the handling of Windows MetaFiles. But, you know, the IrfanView program is a known way of displaying metafiles that is vulnerable, and many people use that.

Leo: Yeah. So you...

Steve: You know, and again...

Leo: So this confirms that you are vulnerable if you're using Windows 95, 98, or ME. Is that right? You are vulnerable to the WMF flaw.

Steve: No.

Leo: If you have something that can view Windows MetaFiles.

Steve: I'm still not sure.

Leo: Oh.

Steve: And this sort of leads us into what I've been doing for the last week and what I've discovered.

Leo: Okay.

Steve: What I decided to do was, you know - oh, and we should say that Microsoft - as most people probably know who are running the older versions of Windows, they've been left out to dry. What happened was, Microsoft's original vulnerability report on their page listed all the versions that were affected by this Windows MetaFile "vulnerability." And I'm putting "vulnerability" in quotes now, so you can hear those quotes around "vulnerability." What happened later is that they moved the earlier versions of Windows - 98, Second Edition, and ME and NT - out of that category. Well, it turns out that, in the process of doing that, it became a non-critical vulnerability by their definition. And their definition for what's critical is sort of amazing. I mean, and this is from a page on their website. They say a vulnerability in Windows is critical only if its exploitation could allow the propagation of an Internet worm without user action. In other words, anything else is not critical.

Leo: Well, that's surprising because many of their patches, in fact, they call "critical," although they don't allow the propagation of Internet worms.

Steve: Exactly. And so...

Leo: Why are they fudging it like that?

Steve: It's amazing. And so, get this, the next level down from "critical" is an "important severity" rating. An "important severity" is a vulnerability whose exploitation could result in compromise of the confidentiality, integrity, or availability of users' data or the integrity or availability of processing resources.

Leo: Ah. So a rootkit is actually not a critical vulnerability, it's just an important vulnerability.

Steve: Very good point. So, I mean, they...

Leo: In Microsoft's language, which is clearly, patently absurd.

Steve: Well, they've defined - yeah. They've defined this so that, I mean, almost nothing now is a critical vulnerability. And then by moving the older versions of Windows out of that category, they said, oh, well, yes, we've agreed that we will patch older versions of Windows' critical vulnerabilities, but we're no longer maintaining those older versions for non-critical vulnerabilities. In other words, this allowed them last week to say, oh, this Windows MetaFile exploit is non-critical, so the older versions of Windows we're not going to fix.

Leo: In their defense, I guess what you could say is, well, if people choose to run an old, out-of-date operating system, we really can't be held accountable for it unless it impacts the health of the Internet. In other words, if you choose to run an old operating system, we're not going to protect you, but we'll do our duty and protect the Internet. I mean, that's not unreasonable, is it?

Steve: And I also agree that, you know, it makes sense for them to set some sort of sunset provision here where they're not being obligated to, like, for example, go back and patch Windows 3.0.

Leo: Yeah.

Steve: Or, I mean, you know, or 95, the real legacy OSes that, sure, you could have put that on the Internet; but, you know, it's just so old, you know, where does Microsoft's responsibility end? So, you know, I'm not arguing, I guess, with this. But if, in fact, these machines are vulnerable, then I had committed, and I believe I should, to fix them for people because Microsoft, it was very clear then last week, was not going to. So...

Leo: And by the way, even if Microsoft doesn't consider it critical, certainly everybody else does,

including the users who are susceptible to this.

Steve: Well, and there's still millions of Windows 9x and ME systems out there, I mean, actively on the Internet, that are now in some sort of unknown limbo state. So over the weekend I rolled up my sleeves and sort of switched into what was really hacker mode. You know, normally I'm writing code. Now it's like, okay, I'm going to sort of follow in Ilfak's footsteps. And I wanted to acquire an understanding of exactly what this problem was in order to determine for myself first if, in fact, these older versions of Windows were actually vulnerable. And then, if so, I would certainly have a head start on how to cure that vulnerability.

So I started with what was known, which was the vulnerability in our existing versions of Windows, you know, 2000, XP, and so forth, and basically created from scratch my own GRC-style vulnerability testing tool. And, you know, there was, you know, code snippets from the hacking sites. And Ilfak had in fact published the source for his tester. Mine ends up working differently because, again, I wrote it from scratch. I have a different approach. But I had a hard time getting this vulnerability to trigger. I was creating metafiles. I was using this, you know, this Escape/SETABORTPROC procedure that we knew was sort of the vector of exploitation. Mine wasn't working. And...

Leo: And this is in Windows 98 you're talking about.

Steve: No, this is Windows 2000.

Leo: Oh, you couldn't even get it in 2000.

Steve: I removed the patch from my system, and I could not get the exploit to trigger using a metafile that I created with my own code. It just, you know, it came back and said it could not play the metafile, but it wouldn't run any of my own code. So, you know, I scratched my head. I looked at, you know, at the other samples of malicious metafiles. And, you know, the way a metafile is built is there's a header, a set of bytes that's the header that talks about what version of Windows it's using, how large the whole metafile is, what the size of the largest metafile record contained within the metafile is, sort of gives Windows some orientation for the subsequent processing of these metafile records. Then you have a series of metafile records where each one starts out with a four-byte size of that record in words, then a two-byte function number which is what type of metafile record this is, then followed by between zero or however many data that function requires. So it's pretty straightforward.

Well, it turned out that, first of all, the way this Escape function was working was it didn't strike me as, like, erroneous. That is, what this Escape/SETABORTPROC function does, the idea is that when an application is printing to the printer, it creates something called a Device Context. I've got to get a little bit tricky here with Windows terminology. But, you know, everyone will be able to follow along. It creates something called a Printer Device Context where things like the thickness of the pen, the color of the pen, the size of the paper, sort of all the things that are about the context of this printing page are stored. So once the application has a page ready, it turns it over to Windows and says, okay, here, go print this. And essentially it's done with that page, and it gets on about its business, for example, maybe getting the next page ready to hand over to Windows to print.

The problem is, what if the user aborted that page, that is, aborted the printing of the page, after it had been handed over to Windows? Since the application that's doing the printing has already turned responsibility for the printing over to Windows, there's really no way for Windows to say, hey, oops, just want to let you know the user canceled your print job. So this SETABORTPROC is a means for giving that printer context, that printer device context, a subroutine that Windows can call back in the application. It's called a "callback," in fact, because Windows calls back the application to notify it if the user or something causes an abort of the print job. So, you know, that's what that is. It's well understood. It makes complete sense in a printer device context.

Leo: So my understanding of it and the general understanding of it has changed a little bit. It is just simply a callback routine that's designed for aborting a printing process so that you can callback the calling program.

Steve: Yeah. Basically you're giving Windows a pointer. You're giving Windows a pointer to a subroutine in

your code and telling Windows, if the user aborts the print job, and I've given you a pointer, then call that subroutine of mine, which is a way for Windows to notify the application. That's what that is.

Leo: Right.

Steve: Well, okay. First of all, it makes no sense at all in a metafile device context. In the context of processing a metafile, setting a printer abort is crazy because it's not a printer context. You don't print metafile contexts in this way. It's just not the way it's done in Windows. So it doesn't make sense. But it's like, okay, well, so maybe, you know, it's there anyway; they didn't think to remove it or take it out. Except that, when I was pursuing this and finally got it to work, what Windows did when it encountered this Escape function, followed by the SETABORTPROC metafile record, was it jumped immediately to the next byte of code and began to execute it. That is, it was no longer interpreting my metafile records record by record, which is the way metafiles are supposed to be processed. You don't actually execute the metafile. As we said before last week, and I think the week before, it's sort of a script. It's a script of Windows graphics calls that allow you to specify, you know, draw a rectangle from here to here, draw a line from there to there. And it's in a nice sort of device-independent fashion. So you don't run the code in the metafile. But what Windows did when it encountered this particular nonsensical sequence was to start executing the next byte of code in the metafile.

Leo: Hmm.

Steve: And it's like, okay, wait a minute.

Leo: Why?

Steve: You know, that's crazy. But what's even more crazy is what it took for me to make it do this. As I said before, each record in a metafile begins with a four-byte length, followed by a two-byte function number. So in other words, each metafile record has six bytes minimum that it can possibly be in size. Oh, and since the size is in words, the smallest possible size for a metafile record would be three words long, or six bytes. Look, the reason I had problems making this exploit happen initially is I was setting the length correctly. It turns out that the only way to get Windows to misbehave in this bizarre fashion is to set the length to one, which is an impossible value. I tried setting it to zero. It didn't trigger the exploit. I tried setting it to two, no effect. Three, no effect. Nothing, not even the correct length. Only one.

Leo: And why were you experimenting? Isn't the exploit well known and documented, and isn't there exploit code floating around?

Steve: No. I mean, what we've got, Leo, is a bunch of misunderstanding and sort of strange half explanations. I mean, you know, and frankly...

Leo: So none of the hacker sites have exploit code up.

Steve: Oh, no, many of them do. But no one is really looking - see, they don't care about how Windows is working. They just want to get their code to run.

Leo: Right.

Steve: And so, you know, because I'm a developer when I'm not being a hacker, I wanted to understand - oh, and the other thing is, I want to write a robust testing application, you know, that always works all the time. So I wanted to know, like, okay, what bytes have to be set which way, what matters, what doesn't. Because, you know, that's the way you get something that is as solid as, you know, the code that I put out from GRC. So what I found was that, when I deliberately lied about the size of this record and set the size to one and no other value, and I gave this particular byte sequence that makes no sense for a metafile, then Windows created a thread and jumped into my code, began executing my code. Okay, Leo? This was not a

mistake. This is not buggy code. This was put into Windows by someone. We are never going to know who. We're never going to know - well, actually I'm going to find out when because we're going to know when this appeared because this appeared - I'm guessing this is not in older versions of Windows, which is why this function - or if it is in older versions of Windows, it's done slightly differently. I'm still on the hunt.

So this is not my last report on this. I expect to have a much better sense for this a week from now. But the only conclusion I can draw is that there has been code from at least Windows 2000 on, and in all current versions, and even, you know, future versions, until it was discovered, which was deliberately put in there by some group, we don't know at what level or how large in Microsoft, that gave them the ability that they who knew how to get their Windows systems to silently and secretly run code contained in an image, those people would be able to do that on remotely located Windows machines...

Leo: So you're saying intentionally or - Microsoft intentionally put a backdoor in Windows? Is that what you're saying?

Steve: Yes.

Leo: Well, that's a pretty strong accusation. Could this not have been a...

Steve: Well, it's the only conclusion...

Leo: It couldn't have been a mistake?

Steve: I don't see how it could have been a mistake. Again, I'm going to continue to look at it. But from what I've seen now, this had to be deliberate. It was not what we were led to believe. Well, and it's funny, too, because then I thought, okay, wait a minute, Microsoft has lied to us. I reread the original vulnerability spec in, you know, their vulnerability page. And they never say this isn't the case. I mean, they describe it as a vulnerability, which it certainly is. Nowhere, you know, is even what I'm saying contradicted by their page.

Leo: So you're saying Microsoft, or people at Microsoft maybe unbeknownst to Microsoft, intentionally put code in Microsoft Windows that will allow anybody who knew about it access any Windows machine, to get into any Windows machine and run any arbitrary code on it.

Steve: Well, it's not like a trojan, where they would be able to contact a remote machine. But, for example, if Microsoft was worried that for some reason in the future they might have cause to get visitors to their website to execute code, even if ActiveX is turned off, even if security is up full, even if firewalls are on, basically if Microsoft wanted a short circuit, a means to get code run in a Windows machine by visiting their website, they have had that ability, and this code gave it to them.

Leo: And there'd be nothing anybody could do about it or - and in most cases detect it. So it sounds like - and I really want to be careful here because this is a very serious accusation. It sounds like this was done on purpose by Microsoft or somebody at Microsoft. It sounds like it was accidentally discovered. Microsoft reacted and has pulled it out now.

Steve: Right.

Leo: Could there be other backdoors like this?

Steve: Well, yes. I mean, that's the problem with a closed source operating system like...

Leo: I have to say, before we go any further, you're not an open source advocate. You're not a

Macintosh advocate. You've been a Windows user. And frankly, you're my staunchest friend who's a Windows advocate. I mean, so this is not some plan on your part to discredit Microsoft.

Steve: Well, no. And in fact I'm sure, I mean, I'm hoping that we're going to see corroboration from other people who didn't think about or didn't look closely at this. I mean, frankly, if last week Microsoft had patched the older versions of Windows, I would have had no cause to look closely to understand how this exploit worked that was discovered. I believe that some very clever and industrious hacker figured this out, started using it, and Microsoft was caught off guard and thought, whoops, we've got to close this backdoor down. Now, you know, to say that Microsoft did this, I mean, on one level it's clearly true. But we don't know who knows about this in Microsoft.

Leo: It could have been a renegade programmer working for Windows who just thought he'd throw this in for fun.

Steve: Yes. I mean...

Leo: Let me ask you one more time, though...

Steve: But that's dangerous, too.

Leo: Well, of course. But let me ask you one more - you're convinced there's no way this could have happened by accident. It can't be a programming error or bad design.

Steve: No. No. I mean, you know, again, this is as much a surprise to me, Leo, as it is to, you know, anyone who hears this. I did not expect to see this. I expected to find, for example, that the way this exploit worked was that the SETABORTPROC was working correctly, and that I would give it a pointer to my own code a few bytes lower, then I would do something to force the metafile to abort, and then the metafile processing would use the pointer, the legitimate SETABORTPROC pointer, and then basically run the code that was located right there in the metafile. That's what I thought I was going to encounter, something that sort of made sense, like we were originally led to believe. Or actually I think, you know, Microsoft didn't say anything at all. So we just all kind of presumed this was another one of those coding errors that Microsoft now famously makes and corrects on the second Tuesday of every month. This wasn't a programming error. And, you know, so it's like, whoa. When I give it the magic key on the size of the metafile record, then it jumps directly into my code.

Now, again, I will know more in a week. I have to say that, you know, I want to call this preliminary. But I don't see any way that this was not something that someone in Microsoft deliberately put into Windows. And, you know, the other thing, too...

Leo: Could this have been at the request of a government agency, let's say? I guess not because, as you point out, it's not a trojan horse. You have to go to a site. You have to go to the site of somebody who knows about this exploit to be taken advantage of. And in fact, the scenario you describe is really the only scenario I can think of, Microsoft doing it so that, if worst case happened, they would be able to update a machine. They'd be able to say, go to the Microsoft site and we'll fix you or something. I mean, the NSA wouldn't put this in because they couldn't guarantee access to any computer.

Steve: I've looked back over all the documentation. I can't find anything about this documented anywhere. Okay, then I said - I played my own devil's advocate. Okay, so code is running in the metafile. Wouldn't that be useful? Wouldn't it be useful if a metafile could contain executable code, sort of as an undocumented feature? Microsoft never got around to writing about it; but they said, oh, this would be cool, and we'll use the SETABORTPROC. Notice that SETABORTPROC, it was just, I mean, this has nothing to do with printer aborting. It was just sort of a - it was a value that they had handy from other processing, and they sort of reused it. But this has got nothing to do with aborting printing. So it almost helped with the obfuscation and sort of, you know, the plausible deniability, except that this wasn't a coding mistake. And, you know, you even had to put the magic key into the length of the record in order to get this to work. And that was protection from somebody's metafile having a SETABORTPROC metafile record in it and tripping over this

backdoor by mistake.

Leo: This is exactly what you would do, if you were going to write a backdoor, this is exactly how you would do it.

Steve: Yeah. Yeah. So I asked myself, isn't there, like, a constructive purpose for putting code in a metafile? And the problem is, code running in the metafile doesn't have access to the context of the metafile. It doesn't know what to do with it. It's, you know, it's powerless to use the objects that Windows is using. And there seems to be no way to get back to Microsoft's code from this. Again, I've got some more work to do, and then the timing of this Security Now! podcast coincided with, you know, I've known this for a day now. And I've been going back over it and trying to come up with a reason, I mean, a benign reason for this. And I just don't see it.

Leo: I suppose we should contact Microsoft and ask them what they think about this. But I doubt that we'd get a straight answer.

Steve: I've tried doing that before on other issues like this, Leo. And, you know, it's not useful. So...

Leo: In this case, you know, Microsoft often says, well, don't reveal this stuff because give us a chance to fix it because it could be a security issue. But the security issue has already been revealed. We're not revealing anything that isn't already known. We're just asking why is it in there.

Steve: Yeah.

Leo: And the answer is not very encouraging. It does make you think maybe open source is a better way to go for an operating system. At least if it were in there, somebody would have had a chance to see it.

Steve: Well, and setting this length of the metafile record to one, that breaks the metafile processing. It's not possible to, like, execute some code, then go back and finish things up gracefully. I mean, you've got a thread running on your own code, in your own image. And everything is, like, over at that point. I mean, it just makes no sense.

Leo: Have you contacted anybody in the security community to ask them about this?

Steve: No, everyone hearing this will be learning about it from our podcast for the first time.

Leo: We've got a bit of a scoop here. And I want to reserve judgment and give Microsoft the benefit of the doubt. Prove to us why this isn't an intentional backdoor in Windows and reassure us that there aren't any more.

Steve: Well, I mean, as you've mentioned a couple times here, I mean, one of the advantages of an open source system is, you know, and I'm finding myself gravitating more and more toward open source solutions because of their transparency. And so, you know, but an advantage of that is that all kinds of people are looking at the code, and there's just no opportunity, especially when you build the system yourself from source, there's no opportunity for anything evil to get stuck in. And also, about this what appears to be a Windows MetaFile backdoor that's always been in Windows from 2000 on, you know, they've done recently serious security reviews of all their code. You know, they took that whole timeout from all the work they were going to be doing and said they were rereading all their code. And this is not the first time metafiles have had a problem. There have been what are probably real bugs in metafile processing in the past, I think two of them. So the whole metafile system would have come under the scrutiny of someone, you know, very deliberately.

Now, you know, if Microsoft had said last week, whoops, this was an undocumented backdoor or means for us to run code in a metafile, we never documented it, our security sweeps didn't find it, blah blah blah - but nothing was said. They allowed the industry to believe that this was just like all their other code mistakes, but this wasn't like all their other code mistakes.

Leo: Well, it's a very serious indictment, if not of Microsoft, maybe of a renegade programmer inside Microsoft. If you were doing a code review, would this kind of thing stand out? Would it be fairly obvious that something was going on?

Steve: Yeah. I mean, I've seen Microsoft source code. In the old days they used to publish the source for what's called the DDK, the Device Driver Kit. And, you know, they're very cautious about, you know, on a module-by-module basis, there's the person's name or initials and when they made changes and what they made to the code that follows. So, I mean, again, Leo, we're never going to know for sure. I mean, I've been in this position with Microsoft in the past, or similar positions. And, you know, it's very difficult to get a straight answer from them. So I don't know what their source says. But it seems to me that somebody had to have seen relatively recently, certainly since Windows 2000, had to have looked at the code, seen that this was something that was there, and just kind of nodded to himself and said, yup, that's what we want to have in our metafile processing code.

Leo: Wow. Well, I'm sure we'll hear more about this. I think you probably are going to stir up a hornet's nest here. And if Microsoft would like to come on the show and respond, you absolutely are welcome to do so. I'd like to hear an explanation.

Steve: I'm going to continue looking at this. The unanswered question is, when was this installed in Windows? My hunch is it actually wasn't ever in the earlier versions of Windows. I'm going to look for it and see what I can find. But it feels to me like this was something added later and that the older versions are, in fact, not vulnerable and have never been vulnerable.

Leo: Hmm.

Steve: But again, I haven't looked there. I don't know for sure. I haven't also looked at Windows code itself. So far my work has just been from the outside, you know, poking at this, trying to get the behavior from Windows that I expect. So, again, it may be that a week from now I come back with my tail between my legs and say, Leo, you know, I told what I believed to be the case at the time. I now see how this makes sense, and something that I see in the code didn't occur to me. I haven't done that yet. So that's what I'll be doing. We'll certainly know more in a week. But everything to me looks like this had to have been put in Windows, in many versions of Windows, for a long time, and that someone just discovered it, so Microsoft had to take it out.

Leo: Wow. Well, that's a blockbuster. That's a real bombshell, Steve Gibson. Of course at this point the good news is Microsoft has released a patch that fixes it, and so we don't have to worry about at least this backdoor.

Steve: As a matter of fact, my little tester is finished. It's still going to undergo more development. But since I believe this is a backdoor, my tester is called KnockKnock.

Leo: And you can get that at GRC.com/securitynow.htm. That's where you can also get the show notes, the transcripts, and the 16KB versions of this show. Of course we thank the folks at AOL Radio and their podcast channel for broadcasting the show and for providing us with bandwidth for the download. Quite a blockbuster, Steve Gibson. I don't know what the reaction is going to be. But I can tell you, there's going to be one. And we'll continue to cover this.

Steve: Well, this is the way it looks to me. I do want to mention that this KnockKnock.exe is still in development. It currently only runs on the newer versions of Windows because that's all that my research has extended to so far. And it's when I got that running that I realized the implications of what I had to do in order to make that work.

Leo: Just to reassure us, you can't use KnockKnock as a hacker tool. It will just let you know the vulnerability is there.

Steve: It's not a hacker tool. It works similar to Ilfak's vulnerability test, although it executes the code in a different way. I retained control of the program, whereas Ilfak's terminated the program after presenting the user with a little dialogue box. So mine actually validates the code, that a thread was created by Microsoft and allowed to run in my metafile image. Also mine will not trigger people's AV scanners, whereas other vulnerability testers do.

However, I want to make sure people understand, this is not in our normal freeware files list. This is not linked from my homepage. This is for people who are listening to Security Now! who want to play with this little tester that runs under Windows 2000, XP - oh, and it has been checked under the 64-bit version, both with a vulnerable and non-vulnerable machine. Several people wrote saying that, after applying Microsoft's patch, they still believed they were vulnerable. So I think it had to have been pilot error, like they forgot to reboot Windows afterwards or something, because you do have to reboot Windows after applying the patch. But my little program, this little KnockKnock.exe, will tell you for sure, on the later versions of Windows, if you're vulnerable or not to this. And then I am going to be doing the research, finally, to extend this to the earlier versions. And as you said, we may find out more about how this happened in Windows. I don't think we've heard the whole story so far, Leo.

Leo: Headline reads: Security Expert Steve Gibson Says Microsoft Intentionally Put a Backdoor in Windows 2000 and XP. Film at 11.

Steve: 2000 and 64, I mean, basically all the versions from the time it first appeared.

Leo: Wow. Stay tuned for more on this one. We'll be back next Thursday with another edition of Security Now!. Steve, I'd make sure you lock your door tonight.