Vista Security Bypass

Description: Steve and Leo discuss some recent revelations made by two talented security researchers during their presentation at the Black Hat conference. Steve explains how, why, and where the much touted security improvements introduced in the Windows Vista operating system fail to prevent the exploitation of unknown security vulnerabilities.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-159.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-159-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 159 for August 28, 2008: Is Vista Safe? 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! Episode 159, the one I've been waiting for for some time. Ladies and gentlemen, may I introduce you to our majordomo of security, the king of lockdown, Mr. Steve Gibson.

Steve Gibson: Yay.

Leo: Host of GRC.com, Gibson Research Corporation, creator of SpinRite, and a guy who's really done a lot to forward the idea of security. He's the guy who put Microsoft's feet to the fire over raw sockets. He discovered spyware and coined the term. And on and on and on. Of course also the creator of ShieldsUP!, which is, as of right now, the most used test of firewall security in the world. Hi, Steve. How are you today?

Steve: Hey, Leo. It's great to be back with you. And we've got more goodies coming. I'm working, I'm just having the time of my life writing code. There's nothing I like better. So I'll have another chunk of code result ready for our listeners here in another week or two.

Leo: Oh, that's exciting. That's really exciting.

Steve: That's going to be very, very cool. This week we're going to talk about what we mentioned, I think, first two weeks ago, and then reminded people of last week. There was a presentation other than Dan Kaminsky's, believe it or not. Dan, of course, told us about what his discovery was earlier this year on DNS spoofing. There was another presentation which I'm not - let's see. The actual title of the paper was "Bypassing Browser Memory Protections." And I call it "The Vista Security Bypass." So that we're going to talk about. I've got a bunch of security news. And, Leo, the SpinRite testimonial to end all SpinRite testimonials. In fact, I'm a little concerned that no one will ever again bother sending one in after they have heard this one.

Leo: Wow.

Steve: I mean, I really - I still want them, folks. But this one, nothing I could say could prepare you for this. It came in over the weekend. Greg, my tech support guy, sent it to me. And as I'm reading it I'm thinking, oh my goodness.

Leo: Wow. Can't wait to hear that.

Steve: Believe it or not...

Leo: Go ahead.

Steve: That is not too much of a buildup, believe it or not, so.

Leo: Okay, I'll be the judge of that. I don't know, that's a pretty big buildup.

Steve: That's fine. I accept your judgment.

Leo: We are going to talk about Vista and is it really safe or not, is it really locked down or not, and what does this discovery mean. So, you know, it's funny, I don't know if you noticed, but I came in this morning, this is not a Patch Tuesday, I got an update, a critical update this morning.

Steve: Yeah, well, Microsoft - remember I mentioned last week that some seem to be drifting in a little bit late. It was confirmed that Microsoft messed up some of the patches from the first shot on Tuesday. And so they were amending them and making some further changes. So, yes, there have been - some other things have been drifting in more recently.

Leo: I didn't look - it didn't require a reboot, but I didn't look and see what it was. So in other words, others may have gotten that last week, it's just they're kind of rolling them out slowly?

Steve: Yeah, yeah. The big, well, there's a couple interesting things. The potentially most interesting news of the week was a story that was picked up by a number of computer journals. I don't know if it's made the popular press because I know that InformationWeek, ZDNet, and the Register in the U.K. picked it up. There was a Polish security researcher by the name of Adam Gowdiak, G-o-w-d-i-a-k. He claims, and Nokia has since confirmed, that he found two serious security vulnerabilities in the Java mobile technology J2ME, which is deployed on Nokia Series 40 handsets, of which there are more than 100 million.

Leo: Wow.

Steve: Now, you'll remember last week I said radio is bad.

Leo: Yeah. Anything broadcast is risky.

Steve: Any, yeah, radio is bad. I mean, yes, it's useful. Yes, we use it, you know, because it's so handy and so potentially beneficial that it's like, okay, well, how bad is it? Well, in this case Nokia's very unhappy. This guy actually was holding them and Sun ransom to the tune of 20,000 euros. He wanted 20,000 euros from each of them, which is just shy of about 3,000 U.S. dollars at the moment, before he would tell them what he found. But he told them enough. Apparently there are remotely executable exploits in more than 100 million Nokia Series 40 phones. Those tend to be the low-end handsets that are using that particular run of the Java system. Which, if you know the phone number, you can, without any permission required or knowledge from the handset's owner, you are able to install and run any code remotely that you want. So that's not good.

Leo: That isn't good.

Steve: That's not good.

Leo: Matter of fact, that's terrible.

Steve: It's terrible. So Nokia's scrambling around, running around, trying to figure out how they can mitigate it, what they're going to do. I mean, this is in 100 million low-end handsets.

Leo: Unbelievable.

Steve: So, you know, this is what happens. And again, I did smile a little bit because I thought, well, okay, this is bad, but so is radio. I mean, it's just - it's risky. So the other news, and when I saw this I checked, Opera is getting a lot of updates. And there's another one. So anyone running Opera, check your little, under the Help deal, check for updates. And you'll find, if you're not already on 9.52, that there is now 9.52 ready for you. On the Windows edition that update fixes seven flaws, five in the Mac, and six in Linux. So there's a bunch of stuff. They've been - most of what they fixed they've revealed. But they're being coy in one case where there's some sort of a cross-site scripting vulnerability - we did a whole episode on Security Now! a while back on cross-site scripting - that essentially allows various sorts of exploits to be pulled off with the benefit of stuff from the browser getting into a website and confusing the code that runs on the website in a way that allows you to, like, get your credentials messed up or spoof who you are to another site and so forth. Anyway, it's a bad thing, so you definitely want to get that fixed.

And one last little bit on the DNS flaw that I've picked up, I thought was very interesting. The DNS flaw that we've spoken about now for several weeks is, Dan says, is actively being exploited, although there's not a lot of people talking about it. It's the kind of thing now that ISPs are not going to be probably anxious to blab about. But in the case of a Chinese, a large Chinese ISP called China Netcom, there was an exploit against their name servers that I thought was rather clever. And that is, somebody is injecting mistyped domain names, similar to Amazon or Google or Microsoft or those. And if you think about it, everything we've been talking about was the notion of overwriting valid domain names with something else. Well, there's nothing to prevent you from injecting, like making up domain names that don't even exist in, for example, in the dotcom servers. The same spoofing attack would allow you to deliberately create every kind of misspelling of Microsoft that you can imagine someone might make, and just stuff an ISP's name server with all those misspellings. And in this case, in every case they direct your browser to an IP that brings up a page which attempts to exploit all the most recently disclosed and fixed, but still maybe not patched in, for example, Chinese machines, which we know tend to lag behind the patch levels of Western machines, just as a consequence of the machine's heritage and software and so forth. So I just thought that was an interesting twist on, I mean, they're using the Kaminsky approach of jamming records into a DNS server. In this case they're not replacing valid ones, they're filling the server up with all kinds of mis-typings. So that someone makes a typo, and rather than getting "Sorry, we couldn't reach that site," they are exploited.

Leo: Right. That makes sense. And this isn't really a new idea. People have been exploiting typos for a long time. You mistype Amazon you're sure to get something else related or an advertising site or something like that. But this is a new switch, a new thing on it. They're really kind of, that's right, amazing. Boy. I have a security story that I just saw that I thought you might be interested in. And there's nothing you could do about this but shake your head. According to eWEEK, a laptop sold on eBay contained personal information about more than one million customers of the Royal Bank of Scotland, American Express, and NatWest. Information included historical data related to credit card applications - and you know what's on a credit card application - and data from other banks. They wouldn't say more. It's just stunning. Just stunning. Sold on eBay. A former employee sold the computer earlier this month without removing the information. It was an employee of a third-party archiving firm. So these...

Steve: Ooh, no kidding.

Leo: Yeah, yeah. Isn't that good?

Steve: I wonder what it was doing on a laptop? I mean, why would - because if they're a third-party archiver, then they're archiving...

Leo: It shouldn't be on a laptop.

Steve: ...highly, yes, highly confidential information on behalf of their clients. And, oh, boy.

Leo: According to the Daily Mail, it was being held by a company called Graphic Data. They're an archiving firm. They copy paperwork. Oh, I know why. They scan the paperwork. This is paper that they scan in. Maybe the laptop was being used to do that.

Steve: An intermediate, ah, that's still...

Leo: It contained names, addresses, mobile phone numbers, bank account numbers, sort codes, credit card numbers, mothers' maiden names, even signatures.

Steve: Everything you need.

Leo: Every - it's a kit.

Steve: Can you say "identity theft."

Leo: It's a kit. Thank goodness the person who bought it said, whoa, and didn't use it.

Steve: Yeah.

Leo: Unbelievable.

Steve: Okay, Leo.

Leo: I'm going to calm down. I'm ready. But wait a minute. Okay, go ahead, do that, and then we'll do the - then we'll talk about Audible. I don't want to keep people in suspense any longer.

Steve: Okay. Now, the subject line, it's like, okay, maybe this is a little over inflated. Subject says: "SpinRite 6 Saves 8 Lives."

Leo: Okay.

Steve: It's like, okay. "Because of the nature of this email and what it contains, I would like to stay anonymous. Thus the email address I used." Actually this was sent - and I won't even read that because he might not want to get a response. "You can use this on Security Now!, though there are no names, and the geographical location is not important. Just know it is terrorist country."

Leo: Oh.

Steve: "We are 50 miles from the safe zone and have to cross a quarter-mile deep by 10-mile wide danger zone containing land mines. But we aren't really sure where they are. We have two injured men, one who can't walk and has to be carried on our backs. We take turns doing this. Here is the story. I am part of a six-man team deep into terrorist country on a rescue/snatch-and-grab mission."

Leo: Holy cow.

Steve: "It has been 10 days, and we made it 45 miles on foot to our destination, where we had to complete the mission, then cross the quarter-mile danger zone lying somewhere ahead in the next five miles. We started out with three Panasonic Toughbooks, the tablet PC style, and one was quickly shot by small arms fire three..."

Leo: Even a Toughbook isn't going to survive that.

Steve: Oh, no. Get this. Half the screen did. Anyway, "One was quickly shot by small arms fire three days into the mission, and the screen was hit. Even though it still worked on half the screen, we trashed it" - that is a Toughbook - "and took the hard drive out of it for security reasons. Another one was used to defend against an attacker with a large stick in his hand. But the laptop still worked, or so we thought. It was powered on at the time of using it as a club to hit someone over the head, and the hard drive was damaged. The third one was lost when we had to egress quickly, leaving us with only one laptop, the one used as a club with an unknown-to-us damaged hard drive."

Leo: Holy cow.

Steve: "On the rescue, like most missions of this nature, to say they don't go as planned is an understatement at best. We were down to one laptop for the mission, and we badly needed it for communication and GPS navigation. Although we have handheld GPSes, in the infinite wisdom of the people in command of this, the solar panels we used to charge everything only worked for the laptops and radios, not the handheld GPSes. So they were out days ago. So we completed the first half of the mission. Now we had to get back.

That's when the problems arose." It's like, oh.

Leo: That's when they had problems?

Steve: So far this is business as usual.

Leo: It probably is for these guys. You think this is legit?

Steve: I do. You'll see. There's a lot of facts here. And this is a SEAL team, he mentions in a minute.

Leo: Oh, my goodness.

Steve: "So 30 miles into the trip back, the laptop crashed, locking up. And when it was rebooted, we got the 'Please insert boot media' message, and we were screwed. We had the hard drive from the other laptop that was damaged three days into the mission. And since it was from the same model laptop, it worked fine when placed into the last surviving machine. At first. Then it gave us the BSOD." Of course, the infamous Blue Screen of Death. "Now, remember, there are eight of us - two men injured, one who has to be carried, and the other six of us. This mission had to be completed. A SEAL never leaves a man behind, even if they are dead. So we kept moving because we had to.

"One of the men knows of a village two miles from our current location that might be able to help. We go there, having no other choice in the matter. We have to have that GPS, or we have to go at least five miles out of the way closer to enemy soldiers, or cross the mine field with no GPS to guide us through the safe parts. Not good. At the village, we find someone with satellite Internet." He says, parens, "(This is where the small size of SpinRite is a blessing.)"

Leo: Yeah, no kidding.

Steve: He says, "And we use a VPN to try to communicate with command to let them know what the problem is. But we can't establish an uplink since we're getting such low bandwidth." Well, he's going to be needing CryptoLink here before long. But for the moment, we don't have that. So because they're getting such low bandwidth, they cannot establish an uplink. "And we didn't have the cables we needed to plug in the satphone and use its modem. So we had to call command and let them know what the problem was. There was nothing that could be done. We weren't even supposed to be in this country in the first place."

Leo: Oh, man.

Steve: "So I decided to try to buy SpinRite."

Leo: Did he have a credit card with him?

Steve: "And try it right there on the spot."

Leo: Oh, my goodness.

Steve: He says, "I am the geek in the bunch. So the others, two of which are 'jocks,' don't believe it will work. We didn't have a credit card of any kind, or anything else that might identify us, of course. So we asked the owner of the house, and he said yes; but 100 U.S. dollars was the max he could spent a month and had already spent most of that on supplies for their store. So we were screwed again. Or were we? I went to a dangerous, illegal, underground pirate site and found an illegal copy. Sorry, GRC."

Leo: At least he tells the truth.

Steve: "At the satphone's very low data rate, it took eight minutes to download the pirated copy of SpinRite." And he says, parens, "(Sorry about stealing it, but it was a life-or-death situation, and I bought it later.)"

Leo: I'm sure you don't mind, Steve.

Steve: Of course not.

Leo: In fact, had you known, you would have sent it to him immediately.

Steve: I've been wondering if we need to put on the eCommerce system an emergency SEAL rescue team free discount button.

Leo: Wow.

Steve: So, he said, "We had blank CDs, so using SpinRite to create a boot CD was no problem. I booted the machine with SpinRite, ran it at Level 4 on the 20GB hard drive, and it found about 12 red U's and recovered 10 of them. After nearly four hours of repair, SpinRite was finished. The computer booted up correctly, and we were able to use it to navigate across the minefield to safety."

Leo: Oh, my goodness.

Steve: "Later we learned that the enemy was only two miles behind us and surely would have caught us if we had to slowly navigate the minefield, probing for mines while covering the quarter-mile distance, or if we had to go completely around the minefield.

There is no doubt that, had we been captured, we would have been tortured and killed for sure. So SpinRite 6 saved eight lives that day, in my opinion. Thank you for your great product."

Leo: Yeah. Okay. You're welcome.

Steve: "It is now an unofficial piece of the toolkits many SEAL teams have with them on every mission, purchased by the team leader since the higher ups don't want to buy a site license for whatever reason. It is run on every single computer we use before we leave, and any hard drive having even one red 'U' is discarded and replaced after we get back, just to be sure this never happens again. The IT guys here know about this and have been pushing this for some time now, but it just 'isn't in the budget.' Well, now maybe they'll reconsider that. Again, you can use this on Security Now!. I have left out names and locations on purpose. Please don't try to guess the location or anything like that on the show, but you can talk about it."

Leo: Wow. You think that's real?

Steve: I don't know.

Leo: Wow. Somebody has a very good imagination if it's not.

Steve: Yeah, and lots of details and so forth, so...

Leo: Man.

Steve: Yeah. I mean, it certainly, I mean, a lot of the information, you know, Panasonic Toughbooks, obviously they needed them for communications and navigation, so they had redundant Toughbooks. They had three. And as he said, these things don't go - they never go the way you expect. And, you know...

Leo: That's the truth, yeah.

Steve: ...plan got shot.

Leo: Every plan is immediately just thrown out when you...

Steve: And he had to "El Kabong" somebody with another one.

Leo: That's amazing. I just checked. Somebody in the chatroom sent me a link. The Navy does use Toughbooks. So, I mean, I'm sure they have mil-spec stuff. But...

Steve: Yeah, and wouldn't you think maybe solid state drives would make sense in this case?

Leo: Would they be more resilient to use as battering rams?

Steve: Yeah, I mean, you've got to bonk someone over the head, it wouldn't hurt a solid state drive.

Leo: No moving parts. But then they couldn't use SpinRite, so...

Steve: Well, they wouldn't need SpinRite.

Leo: Okay.

Steve: I mean, I'd rather that they stayed alive.

Leo: Yeah, no kidding.

Steve: Oh, my goodness.

Leo: What a story. That is very dramatic. And if it's true, thank you for sharing it with us, and thank you for your contribution.

Steve: Well, and I presume the person who wrote this, hopefully truly a Navy SEAL, will be hearing his story read by us, since he clearly is also, as he said, he's the geek in the squad and a Security Now! listener, so I thank him for the mission, and I am sure glad that it worked out.

Leo: That must make you feel pretty good.

Steve: That's very cool, yeah.

Leo: Your software was used to save lives, eight lives.

Steve: Now, as I said, I still want to hear from people whose photos we save.

Leo: Holy cow.

Steve: Because that matters, too.

Leo: You know what, I thought you were building it up too much, you know, there's no way you could live up to what you - but you were right, you couldn't possibly build that up too much. That's an amazing story. Thank you for sharing that with us.

All right, Steve. It's time to get to work here. We have a lot of people who heard about this presentation at Black Hat. Or was it at DefCon?

Steve: It was Black Hat.

Leo: DefCon follows Black Hat, but Black Hat's like the more serious one for security professionals; right? And DefCon is kind of open to the public.

Steve: Yeah. And this was, okay, this is an extensive paper. I'm going to run through and summarize essentially what these guys have done. Their work, however, is very solid. I first picked up on this from a little blurb that someone sent me from Bruce Schneier's log where he had picked up on this. And he said this is big. There were some follow-on where people were sort of pooh-poohing it, saying this really isn't that big a deal. But once again, this demonstrates in my opinion that Bruce gets it, as he always does about security. This is big. This is still sort of leading-edge fringe capability. But that's the way these things always start, and they always evolve. I'm going to read just the introductory paragraph from this paper to give our listeners a sense for the content of this. It says - because again, this is serious, corporate-level, really beautifully put together work.

"Over the past several years Microsoft has implemented a number of memory protection mechanisms with a goal of preventing the reliable exploitation of common software vulnerabilities on the Windows platform. Protection mechanisms such as GS, SafeSEH, DEP and ASLR" - all of which I'll explain in a second - "complicate the exploitation of many memory corruption vulnerabilities and, at first sight, present an insurmountable obstacle for exploit developers. In this paper we will discuss the implementations and limitations of all aforementioned protection mechanisms and will describe the cases in which they fail. We aim to show that the protection mechanisms in Windows Vista are particularly ineffective for preventing the exploitation of memory corruption vulnerabilities in browsers. This will be demonstrated with a variety of exploitation techniques that can be used to bypass the protections and achieve reliable remote-code execution in many different circumstances."

Leo: Oh, boy.

Steve: And they have done that. So stepping back a little bit, first of all, I wanted to remind our listeners, or maybe those who have been listening since Episode 67, meaning they never heard 66 - 66 was our episode entitled "Windows Vista Security." And we laid out during that podcast - that netcast. I'm going to try to use the word "netcast" from now on. I see you're using it uniformly, Leo.

Leo: Well, I'm not using it uniformly. I use "podcast" all the time because that's kind of colloquial. But I do prefer it because what are we doing on the TWiT Live thing? It's not a podcast.

Steve: Right.

Leo: It is a netcast. So I just like the idea better, you know.

Steve: So anyway, on Episode 66 we sort of basically covered in very good depth and detail the two main new features in Windows Vista: DEP, which is Data Execution [Prevention], and in fact I have mentioned that in some subsequent podcasts, so if you just did a search on the Security Now! page or search GRC in our own little built-in search tool for DEP, Data Execution [Prevention], you'll find our references to that in Security Now! podcasts. The other mechanism is ASLR, Address Space Layout Randomization. Both of these are intended to go a long way toward thwarting exploitation of various types of vulnerabilities.

Now, over and over and over, the problems we see in vulnerable software are problems with pointers not being checked, or the very common buffer overrun problem where essentially some software allocates a certain amount of buffer space, sort of dynamically, that is, the code jumps into the routine, and it wants to read something or get something, fetch something from outside. So it will allocate some buffer space. Many times it's on the software stack, which is a variable size sort of scratchpad area that software is able to use. And then the program will say, give me the data that I'm expecting. Well, programmers typically say, oh, they know what expected data size is. And very, very careful programmers will go to some lengths to limit the amount of data that they copy into the buffer. But, I mean, it only makes sense. If you say, okay, I'm setting this much space aside, then I want to bring the data into this buffer. And naturally you don't want to bring more data into the buffer than the space you've made available.

Well, it turns out that, for whatever reason, this is probably the most common of all vulnerabilities is an attacker will find an instance in some vulnerable code, a particular function or some way of invoking a function and giving the code data that it doesn't expect, data that it isn't prepared for, data that causes it to behave in a way that allows other data that the malicious hacker supplies to be put somewhere in memory where it wasn't intended, and then after that to cause that code to be executed.

So basically what we have is an ongoing cat-and-mouse game with Microsoft and the developers of Windows versus the bad guys. And so one thing gets done, someone figures out - a bad guy figures out how to exploit something. Microsoft looks at the exploit and says, okay, we're going to fix that. Well, so all the patches we're constantly getting, these things that are every second Tuesday of the month, all these patches, these updates are specifically new problems of exactly this sort that have been found in specific instances of Windows.

The problem is, Windows is so vast and is so large that it's - well, and frankly so capable. Many of these things are resulting from the flexibility that Microsoft has often deliberately, sometimes inadvertently, put into Windows. For example, and we've talked about this in many episodes of Security Now!, it's possible for a browser to invoke an ActiveX control that never was useful in a browser context. That is, it isn't something

that you would normally expect a browser to load. But Windows will do it for you. So if anywhere else in the system an ActiveX control, which is really just sort of a specific type of DLL, if a vulnerability is found in one of its functions, then someone can create a web page that tells that to load and give that function some data that it isn't prepared to accept the way it's given. And that will allow them to inject their own code through the Internet, through the browser, into this vulnerable object which has been found in Windows, anywhere in Windows, essentially. And due to the fact that there's this vulnerability, they can end up running their own code, thus take over the system remotely. That's a remote code exploit.

Okay. So for years Microsoft was in this, okay, look, we're going to fix everything that we can find that's wrong. The problem is, they're always playing catch-up. That approach is always reactive. Microsoft finally began to get actively proactive, first with XP, further with Service Pack 2 of XP, and then again with Vista. By, essentially, by incrementally making these things harder to exploit. And so as you heard in the beginning of this paper, what these research hackers have done is to carefully look at the various mitigation efforts Microsoft has implemented over time and looked at, okay, how do we get around those?

So, for example, Data Execution [Prevention], DEP, what it does is it is a technology available in microprocessors since 2004, that is, it's a software technology, but it relies upon a feature called the NX bit, the No eXecute bit, which was not available in flat memory model, which is what Windows runs in ever since Windows 3.1. The 16-bit versions of Windows were segmented model architecture. In the segmented architecture you could mark segments as non-executable. But due to sort of a design oversight in the Intel architecture, in a flat-mapped memory model, which is what we have in 32-bit Windows, you are not able to mark individual pages of that memory as non-executable. So we had to wait for another generation of microprocessors from Intel that would offer the - well, and AMD - that would offer the so-called NX, the No eXecute bit.

And so the idea with DEP is that only the memory that is expected to have code in it would be marked as executable. And so data memory, like for example the stack that I was talking about where you allocate buffer space and variables, well, you don't run buffers, you read and write them. You don't execute them. So those are, you know, three different operations: read, write, and execute. So the stack would be marked No eXecute, meaning that any attempt to overwrite the stack, to overwrite a buffer and execute on the stack would immediately result in the processor terminating the - or in the operating system receiving notification of this violation. And the OS would just terminate the process instantly, bang. Just the attempt to execute that instruction, the first instruction that attempted to be executed in that space would never complete.

Leo: What does that look like to the user? Do they get a blue screen? Do they get...

Steve: What they get is a sort of annoying warning. And in fact, anybody who has tried to turn on DEP - there are four ways that DEP can run. And this is one of the problems with its actual implementation in the real world. And we talked about how Microsoft has an opt-in policy for DEP rather than an opt-out policy. And of course we talk about opt-in and opt-out relative to cookies and spying and everything. And this is sort of similar. The idea is that you can, if you turn your OS - if you boot it with DEP in opt-out mode, meaning that it is normally on, but you opt out specific things, every so often you get dialogue boxes. In fact, I got Internet Explorer itself - I'm sorry, Windows Explorer itself gave me a DEP error just yesterday because on my laptop, I mean, I try to run with DEP enabled. And I can vouch for the fact that it's painful. There are programs that are not

DEP compatible. Thus in XP and even in Vista the Data Execution [Prevention] that is so valuable is opt-in.

Now, programs can be assembled with the so-called "NX compat flag," meaning that when they are assembled and linked, they can have a flag built into them that says, I am DEP compatible. So Microsoft has been very good, to their credit, threw out all of the OS components and even their own applications to rebuild them after checking to make sure that they were DEP-friendly with this NX compatibility flag. So Windows itself is running with DEP enabled, as are Microsoft's apps. The problem is, there's a lot more in Windows than the code that comes from Microsoft. And many other vendors, the vendors of many critical components, for example, like Flash from originally Macromedia, now Adobe, and even the Sun virtual machine. Their technologies, especially in the case of Java, Java has, deliberately has readable, writable, and executable memory because of the way it operates. So it's a big target. And so many of these third-party things, which you could pretty much depend upon, you know, Flash player is installed in the high 90 percentile of Windows machines so you can count on it being there. So as I mentioned that DEP operates in either - you can either have it completely off, in opt-in, which is a little more secure than off completely, but not much; in opt-out, which is certainly more secure, there you're saying by default anything that isn't marked specifically compatible, we're going to assume it is, but if it blows up on you, you know, when you get an annoying dialogue box that says, oop, DEP execution error, Windows is terminating the execution of this application. And it's like, oh, okay. I mean, and again, it happened to me in Windows Explorer the other day. So it's not quite as DEP friendly as Microsoft was thinking.

And then the fourth is always on, which is to say, DEP is on, it is enabled, nothing can turn it off. Well, it'd be wonderful if we could run our machines that way. The problem is, they won't run that way. There are known programs that have to have DEP disabled. And so part of the problem that these guys have talked about is the fact that there are, even in this day and age, using the latest versions of everything in a contemporary Windows system, Windows Vista, the most recent service pack, running the latest and greatest, there are enough programs that have - well, and Vista by default is, remember, is opt-in. So there are enough programs that have not yet deliberately marked themselves as DEP enabled, that Vista will not enable DEP for them since you have to explicitly ask for it because there are just too many incompatibilities still. Because those programs are not enabled that way, there are plenty of opportunities for exploit even though we're at Vista Service Pack 1, and Microsoft talks about how powerful DEP, Data Execution [Prevention], is. It turns out it's not really.

Leo: It would be if you could leave it on and it would run all the time on every program. It is an old habit of programmers to put code in the data areas, which is essentially why DEP doesn't work for these guys; right?

Steve: Well, it's, frankly, I mean, it's...

Leo: It's a bad habit, but it's a habit.

Steve: Well, actually it's very useful. There have been - there are situations where you actually need to execute data. I mean, that's what interpretation is. A good, fast interpreter can actually execute some of the data that it's reading. Back in the old days of Windows, back before we had display acceleration, where we didn't have a so-called

"hardware blitter" that would blast rectangles around the screen, which is so important for these, you know, everything in Windows is a rectangle of some sort. Microsoft's original GDI, the Graphics Display Interface code, it would look at the job it was being asked to do. And so much of it is repetitive and involves loops and counters, when you're moving data around, and you're having to, like, shift the bits by strange amounts in order to cause them to be bit-aligned on the screen, the original GDI actually built the blitter on the fly on the stack and then ran it. It executed it. So there Microsoft went, I mean, they were going to all kinds of trouble to make Windows run at an acceptable performance on an old 4.77MHz PC, an old 8088 or 8086. And they had to pull out the stops. So there are times when executing on the stack, or executing data that is deliberately variable, is very useful. And when you think about it, technically, anything coming over the wire is data. You know, so if something downloads into your browser and says hey, I'm an ActiveX control you just loaded, would you give me permission to run, it's like, well, okay, that's data, but we're executing it. So...

Leo: Right, right, okay. So you make a credible case that there are times. And as an Assembly language programmer, is that something you do a lot? Not really.

Steve: I'm not - I can't think of a place where I have. But...

Leo: So we're not going to see the end of it? I mean, doesn't functional programming or, I mean, don't more modern programming techniques make it less necessary to do this?

Steve: Well, it's certainly dangerous. And you're right that we're moving toward an era where DEP will be turned on, where there will be increasing pressure on people. Certainly after this paper has come out where these guys demonstrate clearly the exploitability of Flash, which is not DEP compatible, it's like, okay, Adobe, if you want your code in my machine, you make it safe. Because we've seen a bunch of Flash exploits here in the last few months. And, you know, this wouldn't be possible if Adobe would do the work. I don't care how hard it is, it's certainly possible to code around this. That is to say, I guess to answer the question you were asking, Leo, is it absolutely necessary to execute data, and I would argue no. It is possible for an application to explicitly - you're able to use something called VirtualAlloc, the virtual allocation commands in Windows. If you want to allocate executable pages, you can. But you could set it up so that the software knows it needs to execute something that is in data, so it explicitly sets itself up with permission to do so.

Basically this is laziness. In this day and age, for Flash still not to be marked as DEP friendly when it is in a highly vulnerable environment, it's not like it's something down on your tray, it's in your browser. And we know what a target browsers are just by their very nature. I mean, in fact, the whole focus of this paper was specifically browser vulnerability.

Now, another thing that has been done that Microsoft did to further complicate things is this ASLR, Address Space Layout Randomization. One of the key things that hackers do is they're able to use code that's already in the system. They, like, they look at the disassembly of Windows, and they say, oh, look at this, this code, if I jump into it right in this weird spot, where it was never designed for anyone to jump in, I can get it to do some work for me that I need done. And so, for example, sometimes the first thing that a malicious hack will do is it will jump to somewhere in Windows in a way that Windows

was never designed for and execute a little chunk of code that then hits a return instruction that comes back to where it came from. So the hacker, the malicious exploiter, has just gotten somebody else's code, Windows code typically, to execute on their behalf in a way that Windows never intended. Then maybe they jump somewhere else, and they make something else happen. They don't even have to supply their own code. They just use code that's already in the operating system that's known to be in a specific location.

And that's the key of Address Space Layout Randomization. Microsoft figured, okay, we're getting hacked all the time because we always load Windows in the same way. We load ntdll.dll, then we load csrss, you know, we stack these DLLs in a specific, totally predictable fashion. Therefore we're setting ourselves up as a target. Let's randomize the layout so that literally every single time the system boots, the main core Windows DLLs load in a different place. So it's no longer possible to have a fixed pointer, which you know is going to point at a useful piece of code that exists in Windows, it's going to be in a different place every time.

Now, the problem is the details, of course, the implementation. What Microsoft has done is they take an 8-bit value from the timestamp counter. All Intel processors have an instruction, which I do use all the time, called rdtsc, Read Timestamp Counter. That is, it's a screamingly fast counter. It literally is a counter of the clock, which means three giga counts per second. And so, I mean, this thing, there's just so much variation with the rotation rate of the hard drive, keyboard clicks, mouse clicks, there's no way that the same count is going to be pulled from this little counter which is screaming at three giga counts per second. So Microsoft pulls the eight bits from the counter, multiplies it by 64K, and uses that - oh, except they never use a value zero. There was a bug in the first version of Vista where it would use - where if it got a zero, it just made it a one, which meant that there, I mean, this is a small bug, but it meant that there was twice the chance that you would be at the location one versus two through 255 because you turned a zero into a one. They fixed that so that they completely smoothed out the statistics of what value you were going to get when you pulled this value and multiplied it by 64. So they multiply by 64K, and that sets - that value is added to that executable's preferred load point in memory. So essentially, as Windows now boots, in Vista and in Windows 2008, but not in XP or 2003, so in Vista and Windows 2008, as it loads it's randomizing these things.

Okay. Here's the problem. Address Space Layout Randomization only works if everything in the machine is randomized. That makes sense. I mean, you don't want any code to be in known fixed locations. Turns out that only newer executables, that is, EXEs and DLLs that are explicitly marked as ASLR compatible will be randomized because Microsoft also learned the hard way, no doubt during the early testing of this, that there was code that expected to be loaded where it said it wanted to be loaded. Individual executables contain what's called the Image Base Address, which is where they want to be loaded.

Well, it turns out that, unfortunately, some programmers have made their own programs dependent upon the load location that they specify, and they don't work if you load them anywhere else. And, unfortunately again, as is the case for Data Execution Prevention, this is the case for tons of non-Windows components, that is, non-core OS components, which is what Microsoft has successfully randomized. But that leaves a wide-open field of other things that are very popular, known to probably be in the system, and can be invoked through the browser. So anything, for example, provided by ISVs - ActiveX controls, browser extensions, protocol handlers for browsers, and especially image parsers or codecs are generally not yet ASLR compliant, so they won't be randomized. And again, a hacker who knows nothing about your computer can inject something into your browser that will cause it to cause something else to load in a known location, and

that could be exploited. And again, these guys demonstrate that.

Leo: That's pretty incredible. So what do we do?

Steve: Well, essentially this is a wakeup call.

Leo: Yeah.

Steve: For, again, it's a cat-and-mouse game. This is a wakeup call, I would say, for developers everywhere. For example, when I'm developing my forthcoming VPN product, CryptoLink, I will absolutely always be linking it with ASLR enabled and with the NX compatibility flag enabled all throughout development. So that if at any point I do something which is position dependent or writing into its own memory, I'll instantly know and go, whoops, and fix that. So that when I ship the product it will fundamentally be as secure as it can be. I certainly don't want my own code to be exploitable by some malicious hacker somewhere through the web browser. I mean, I can't imagine how that would happen, but this is the way these things happen.

So really it's now - it's like Microsoft has done really everything they can. When you look at the details - and I skipped over tons of details. But Microsoft, I take my hat off to them, they have really bent over backwards to thwart every possible exploit mechanism based on looking at exploits, and not only just chasing after them reactively, but now really getting proactive. The problem is, Windows is a rich environment that runs - it's extremely heterogeneous, with bits and pieces coming from all over the place. And it really is necessary now for all of the other players to work with Microsoft, to work, I mean, they don't need Microsoft's permission. They just need to turn these things on, take the time to stop being lazy about this and get their stuff to work so that these latest protection measures can be employed.

Leo: All right. So but again, there's nothing - in the interim is there something to worry about? Are these exploits in the wild yet? Are we...

Steve: Well, okay. This was a theoretical paper that said, okay, you guys are so impressed with DEP and ASLR - and, I mean, there's other things too that I didn't talk about. Structured exception handling, stack cookies, heap spraying, there's just all kinds of stuff. Oh, I have to say this, though. Get a load of this. Okay. We know with IE7 that it's much more secure than 6 was. One of the nice things about IE7 that I really appreciate is you get a pop-up notice when any ActiveX control that you haven't seen before is downloaded, and the web page you're visiting wants to run it. Well, Microsoft is so fond of their new .NET framework. And because it's supposed to run in a sandbox, very much like Java does, and be safe, they said, oh, we don't need to warn anybody about .NET. So they don't. And it's exploitable in the same way that ActiveX has been. But we get no notice and no warning, no pop-up in the default configuration of Internet Explorer.

The good news is you can go in and turn off .NET permissions. So one takeaway from this, the good news is .NET doesn't have much penetration today. I'm not aware of anything browser-wise that runs it. But it is, as a consequence of this paper, these guys demonstrate .NET silently exploiting Internet Explorer under Windows. So I would

immediately turn off .NET execution in Internet Explorer. Just say no to that.

Leo: Wow. I mean, that turns off a lot of stuff.

Steve: Well, .NET? No, I'm not sure that it does.

Leo: Doesn't Silverlight rely on .NET?

Steve: Oh, darn.

Leo: It turns out, I mean, the Olympics are over, thank goodness, but I think that turns off a lot of stuff. I may be wrong.

Steve: I've got it turned off, and I haven't run across anything so far. But you're right...

Leo: Well, for sure Silverlight, yeah.

Steve: Oh, I guarantee you Silverlight is, I mean, that's a classic Microsoft .NET, oh, here you go, give this a try. I mean, and it'll be exploited tomorrow.

Leo: Well, just as Flash was. I mean, in some ways that's kind of similar, isn't it.

Steve: Yes. And again, the problem is that Microsoft assumed, as they always do, that they're not going to have any problems. They have never not had any problems.

Leo: It's very clear that in security you should assume the complete opposite, that you absolutely will have problems.

Steve: Yes, exactly. I mean, you have to. So essentially the OS is providing the tools. Now, to answer your question, Leo, one thing users could try is what I have tried. And that is to switch your Windows to opt-out so that rather than...

Leo: Opt out on DEP.

Steve: On DEP. So and you do that with a boot.ini switch. We've talked about DEP in length, in detail. And in fact my little SecurAble freeware app shows you instantly whether your processor supports the NX hardware bit, which is necessary in order for hardware DEP to be useful. There is something called "software DEP" which is, well, I mean, they were unhappy when they first released Data Execution Prevention because at that time there wasn't a huge install base of processors that supported it. And so they

said, oh, well, but we have software DEP, and it doesn't require any hardware. It also isn't very useful. I mean, which is - even that is giving it too much credit. So...

Leo: I don't even know how you would do software DEP. Doesn't it have to be in hardware? And isn't this in the processor? Doesn't Intel have the DEP processor in there?

Steve: Oh, yeah, well, Intel and AMD - since 2004 all processors have had this NX flag. So it's only really old...

Leo: Oh, AMD, too, yeah, yeah, yeah. In fact, AMD I think did it first.

Steve: It'll only be really old machines that will not have it. And so one thing users, I mean, really security-conscious users could do, as I do, is to run with DEP in its opt-out mode, where then by default applications will have it turned on. As you learn that specific apps are not happy, then you are able to put them in a list of "leave DEP off for only these apps." And you can then sort of evolve. It's sort of like the way a personal firewall works, where by default it's blocking everything. Then you learn, oh, wait, email needs to get out, IE needs to get out, Messenger needs to get out. So you add exceptions to the "prevent all" rule so that those things which you know and want to have access to the Internet are able to. Similarly, with DEP operating in opt-out mode, where it's on by default, you would be learning which apps were not DEP compatible. So you say okay, fine. I mean, and you could decide, like, whether you care about this particular app and want to run it or give it permission or not.

And so over time you evolve. And that's certainly going to be more secure. But it's not perfect. I'm impressed with Microsoft having given us now with Vista the tools to make these exploits much more difficult. It is very common applications like Silverlight, like Flash, commonly used components, or even Media Player, that are invokable by the browser and still not yet safe, that is really now the main target of exploitation.

Leo: Okay. I guess we've got to get ready for a whole wave of exploits on this. And how quickly do you think Microsoft could fix something like this?

Steve: Well, the problem is this isn't Microsoft's problem. I mean, for example, these guys demonstrate using Java exploits, using...

Leo: So it's just everywhere.

Steve: Using Flash exploits. Yes, it's people - it's the third parties that have very popular, often-present code that now need to belly up and make their stuff work as well as Microsoft has the core Windows components. They need to be very careful with Data Execution Prevention and the ASLR randomization. They need to be compatible with those things so that they, too, will be loading in random locations, not known fixed locations, and that they're less exploitable by malicious code. Right now those are - essentially, Windows used to be the so-called low-hanging fruit. Well, basically we ate all of that. And now we got it all. It was low hanging, and it was easy pickings. So now it's

the very popular non-Microsoft components, which are lagging behind even Microsoft in taking advantage of the security features that are now in XP and Vista, they need to get going and make themselves secure.

Leo: Wow. So sounds like a really pretty big mess out there. I mean, we all reasonably assumed that these two technologies, and others also named in the paper, would be the panacea, the magic bullet for security.

Steve: Well, and just to finish, let me read the conclusion of the paper because it sums it up beautifully. They say:

"In this paper we demonstrated" - I mean, and they did, they've got code samples - "we demonstrated that the memory protection mechanisms available in the latest versions of Windows are not always effective when it comes to preventing the exploitation of memory corruption vulnerabilities in browsers. They raise the bar, but the attacker still has a good chance of being able to bypass them. Two factors contribute to this problem: the degree to which the browser state is controlled by the attacker. and the extensible plug-in architecture of modern browsers. The internal state of the browser is determined to a large extent by the untrusted and potentially malicious data it processes. The complexity of HTML, combined with the power of JavaScript and VB Script, DOM scripting, .NET, Java, and Flash, give the attacker an unprecedented degree of control over the browser process and its memory layout. "The second factor is the open architecture of the browser, which allows third-party extensions and plug-ins to execute in the same process and with the same level of privilege. This not only means that any vulnerability in Flash, for example, affects the security of the entire browser, but also that a missing protection mechanism in a third-party DLL can enable the exploitation of vulnerabilities in all other browser components." And then they finally say, "The authors expect these problems to be addressed in future releases of Windows and browser plug-ins shipped by third parties.

Leo: So "future versions of Windows" sounds a little bit like it ain't going to happen tomorrow.

Steve: Yes. And the problem is, Microsoft has tried to turn these things on, and they've gotten bitten. You know, I mean, this is always the case, and this is what we talked about when we first talked about DEP and ASLR, why they were generally off by default. But Microsoft was sort of creeping forward, trying to have them be on. Oh, and get this. You know how Vista 64 is substantially more secure. For example, they have DEP on for all 64-bit processes.

Leo: Oh, interesting.

Steve: And it cannot be turned off.

Leo: Oh, interesting. So that's possible.

Steve: But...

Leo: Oh. I got excited.

Steve: ...IE, Internet Explorer, is a 32-bit process.

Leo: Turned off, right.

Steve: It's a 32-bit process under Win64. And so it is policy driven. It's driven by the DEP policy in the system, not turned on and enforced.

Leo: That's really interesting.

Steve: So it's like, okay, Microsoft, get going.

Leo: Yeah.

Steve: You know, fix this.

Leo: Do other operating systems like Linux and OS X, are they also using these kinds of techniques to protect themselves, and are they also vulnerable?

Steve: Well, yeah. And in fact Address Space Layout Randomization is not a new thing. Windows is like the last OS to come along and implement this. This has been around for a long time. This is a classic case of Microsoft being the big target. I mean, for example, we're seeing an increasing number of Mac exploits and vulnerabilities now. I mean, those are happening. We're getting security patches from Apple at an unprecedented rate compared to the past because hackers have Macs now. It used to be that they predominantly had Windows machines. Now they've got Macs, and they're poking around in there. I mean, fundamentally we're dealing with phenomenally complicated systems. And complexity is the enemy of security. Any time it's complex you can't really be sure about every side effect and feature and bug. And bugs will bite you.

Leo: Yeah. Apparently IE8 will have DEP turned on because it's going to be, I guess, a 32-bit process. But and there's some debate over whether Silverlight uses .NET. I remember, I was pretty sure it did use .NET. But...

Steve: Actually it's IE8 and - I'm glad you mentioned that, Leo, because I wanted to mention this. IE8 and Firefox 3 do, the IE8 beta and Firefox 3 both have the NX compatibility flag. So they are opting in to DEP. And so that's - I know that there have been some people that have said, eh, Firefox 2 is working just fine for me. It's like, okay.

Leo: Oh, no, get Firefox 3, yeah.

Steve: Firefox 3 makes a lot of sense, if only because the developers developed it with DEP on. It's been vetted with DEP on. And it marks itself as DEP capable. And so all versions of Windows, from Vista forward, will turn DEP on in Firefox 3, and you want that buffer overrun protection for anything that might be going on in Firefox 3.

Leo: So your recommendation is to use the opt-in setting of DEP now. Is that right?

Steve: I would - it's a mix...

Leo: In booting?

Steve: It's a mixed blessing. I mean, it will cause you some problems. It's why Microsoft didn't set it up that way. They would love to have Windows be more secure. But there are things that will cause problems. And in fact in this case where even Windows Explorer gave me a DEP execution error yesterday, I just restarted it and it was fine. It was, you know, Windows.

Leo: It doesn't crash, it just restarts. You restart.

Steve: Well...

Leo: It just says you can't do that.

Steve: Okay, right. All of Windows does not crash.

Leo: No. The program stops.

Steve: Yes. Just Windows Explorer, I clicked "yes," it shut down, and of course Microsoft saw, oh, look, Windows Explorer is no longer running, that's bad. And it restarted it for me.

Leo: Right. That's not so bad. And when it comes to securing your computer, there's one guy. There's the man, Mr. Steve Gibson from GRC.com. He saves lives.

Steve: How did I know you were going to use that segue, Leo?

Leo: I'm the king of segues, baby. Yes, I am. So Steve, it's really great to talk to you. I'm glad to get this particular subject out of the way because I think there's been a lot of questions in the community, even in the expert community, what does this mean? Does this mean we have to freak out? Is Vista no longer secure? What do we do? Where do we go forward? And at least understanding what was a very

technical talk. It really helps to kind of get some understanding in there so...

Steve: Yeah, it's not big news. It's mostly just sort of a wakeup call.

Leo: Yeah.

Steve: It's a reality check that says, okay, these things are available, but they are not being taken advantage of universally. And it's not until they are that we'll be increasingly safe.

Leo: Are and can be.

Steve: Yeah.

Leo: I mean, it's not - in many cases we can't. And that's kind of frustrating, too. Okay, Steve. Hey, it's great to talk to you. Next week we've got a Q&A session. How do people get their questions to you?

Steve: They want to go to GRC.com/feedback, where there's a form they can submit. And by all means, I love people's questions. It's great to get them, and then it helps us build a really great Q&A episode every other week. So...

Leo: Yeah, yeah. While you're at GRC, don't forget you can get the 16KB versions of the show there. You can get of course the transcripts that Elaine does, all the show notes. It's all available at GRC.com, along with SpinRite, the program that saves lives. Saving more lives every day. It's an amazing program, the ultimate disk maintenance and recovery utility, GRC.com. Also a lot of free security stuff, as I mentioned at the beginning of the show, including ShieldsUP!, Shoot The Messenger, DCOMbobulator. And watch for some new stuff coming soon.

Steve: Thanks, Leo. Talk to you next week.

Leo: Take care.