Virtual Machine History & Technology

Description: Steve and Leo discuss the historical beginnings of Virtual Machine technology, from the 40-year-old IBM VM/360 operating system through virtual machine language emulators and today's VMware and Virtual PC solutions. This kicks off a multi-episode discussion of the tremendous security benefits and practical uses of modern day Virtual Machine technology.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-050.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-050-lq.mp3

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

This is Security Now! with Steve Gibson, Episode 50 for July 27, 2006: Introduction to Virtualization.

Security Now! is brought to you by Astaro, makers of the Astaro Security Gateway, on the web at www.astaro.com.

We're hitting the half-century mark, ladies and gentlemen, Episode 50 of Security Now!. Steve Gibson, it's good to have you on again.

Steve Gibson: Hey, Leo.

Leo: Can you believe it? We've made it to 50 episodes.

Steve: Yeah. I mean, it doesn't feel like a year ago that we began.

Leo: No. Is it exactly – have we taken any time off?

Steve: No.

Leo: No. So we still have two weeks to go for our anniversary.

Steve: Yup.

Leo: It is hard to believe. And as you mentioned several times, we weren't sure there was enough material. Little did we know. There's plenty of material.

Steve: Yeah, there is. Well, and we get so many great feedback from our users. I've got, you know, questions coming out my ears. So in fact at some point I'm feeling like we're falling behind with our Q&A stuff. We may just switch to a – you know, and we get such great feedback from the Q&A episodes. People really enjoy the breadth of topics that we cover in one hour, or however long it ends up being. So we may end up just do, you know, maybe switch from mod 4 to mod 1 for a while, and do a...

Leo: Catch up.

Steve: And do some catch-up, yeah.

Leo: Yeah, that's a great idea. Speaking of emails, we got one from Randal Schwartz, who is a good friend, the author of "Learning Perl," the geek cruise guy. And he listens to the show avidly. And because he's very active in the security community, he's kind of a great guy to have listen and get his notes to us. And he made a note, which is something I had never heard of, but he just wanted to point it out, is that, you know, I was saying that ports above 1024 were, you know, non-canonical, and the 1024 and below were the kind of traditionally accepted ports. He said actually it's from 1000 to 1024 is also considered – what did he say? It's because some people use decimal and some people use binary that you have to be careful.

Steve: It's interesting. It did sound like...

Leo: I had never thought of that.

Steve: It sounded like it was the classic difference between a K. Is a K actually 1000, or is it, you know, a binary K at 1024?

Leo: So if you want to use a canonical port, if you want to use an official port, the safest procedure is not to use something between 1000 and 1024 but to use something below 1000.

Steve: Well, I can...

Leo: Then you're sure that everybody will see that as a canonical port.

Steve: Yeah, I can definitely, I mean, definitely tell you that my experience has been 1024. That's where – it's at 1024 that machines begin to assign ephemeral ports, the ones that clients are generally being given. So...

Leo: It may be some older or oddball software, but it's just something to be aware of. And we just – I'm glad Randal's listening because he keeps us honest.

Steve: Yeah.

Leo: So what are we covering this week, Mr. G?

Steve: Well, this is the beginning of a multi-week series on virtual machine technology, VM, virtual machines.

Leo: That's very hot right now because Intel's touting it. All their new core chips support it in hardware. And of course but there's programs, software like VMware has been around for a while. Microsoft's own Virtual PC uses it; right?

Steve: Yeah, well, I want to clarify and demystify that. What's really interesting is that VM began 40 years ago. In 1960 IBM had VM/360, which was a virtual machine technology for the original IBM 360 machine. So this notion of virtual machines is not new. In fact, it's old. And...

Leo: It was something that the big mainframes always could do.

Steve: Well, exactly. And then it sort of went away in the '80s and '90s. Now we're seeing a resurgence of the technology for very different reasons. And as often happens, you know, the capability of our hardware has grown so much that things that weren't possible or practical before have now become possible and practical. You know, so we've got super fast machines. We've got much more RAM than we used to have in machines. And we've got massive hard drives which have enabled this notion of sort of a machine within a machine.

Leo: Why don't you explain what virtualization is? And then we can talk about – there's a real – there is a reason we're talking about this on Security Now!. There's a real security use for this. But first of all, what is virtualization?

Steve: Yes. And for that reason I wanted to really lay, as I always like to do, a firm foundation of definitions and understanding so that we have some jargon that we can refer to, and that later when I say, you know, this really works for security, there'll be a reason for understanding why it is that it really works for security, rather than just going, oh, okay, I hope that works.

The whole notion of virtualization is the idea of some sort of a layer. Layers come in many shapes and sizes. The most common, perhaps, notion of virtualization is one flavor of it called "emulation," where you've got a, for example, a computer running an emulator, which is inherently pretending to be something else. In other words, say that you had – well, in fact I guess the Mac has done this when they moved from one chip to another. Wasn't there an emulation of...

Leo: That's right.

Steve: ...of one family of chips on another?

Leo: That's exactly right. They were able to run the old – well, they're doing it right now, of course, with the Rosetta technology, which...

Steve: Over on the Intel platform.

Leo: On the Intel platform.

Steve: They're emulating the PowerPC chip.

Leo: Yeah.

Steve: And so that's actually a virtual machine. I mean, by definition it is like a virtual chip. The software running, the older Mac, you know, G4, G5 PowerPC software, doesn't know that it's not being executed by a PowerPC. In fact...

Leo: Now, they've also in a way done this with operating systems. When they went from OS 9 to OS X, they were able to continue to support OS 9 software by running OS 9 in a machine. So it doesn't have to just be emulating another chip. It could be emulating another environment; right?

Steve: Well, absolutely. And in fact, you may remember the very – one of the very earlier popularizations in the PC industry, you know, stepping way forward from IBM doing this with VM/360 back in the early '60s, there was something called "UCSD Pascal."

Leo: Oh, yeah.

Steve: UCSD Pascal was based on what was called a "p□machine, " "P" as in "pseudo," a pseudomachine. UCSD Pascal compiled its – it was a Pascal compiler that accepted a Pascal source code. It compiled it into a virtual machine that never existed. They literally – they wrote up a specification for, like, okay, if we had a computer, sort of a pseudo chip that executed these instructions, it would be really good for just implementing this language, this Pascal language. And actually there was even a Fortran compiler that compiled to the same pseudomachine, the same p-code.

Leo: Same machine, yeah, yeah.

Steve: The same p-machine. And so the idea was, if you wanted to run UCSD Pascal on, like, on an Apple II, you would create a p-machine interpreter. You would create an interpreter, an emulator, for this virtual chip that no one ever actually made. And then all the software, I mean, suddenly all the software that had been written for this UCSD Pascal was running all at once on this new chip, only by creating this emulation layer, or this virtualization layer, this virtual machine.

Leo: That's how Java works.

Steve: Well, yes, exactly. I was just going to say that then there was also Forth. Forth was a wacky language which was based on the same sort of pseudomachine, or p-machine, technology, which made it very easy to port from one chip to another. Sun Microsystems, the Java Virtual Machine is the same thing. You have a Java runtime. What the Java runtime does is it is basically a virtual machine that compiled Java code is run against. So in that case it's the Java Virtual Machine is interpreting the instructions that have been compiled down into the compiled Java to do the actual work.

Leo: Unless you think this is old hat, in fact, that's how Perl 6, the next generation of Perl, is being developed. They're developing first a virtual machine called "Parrot," which will run not only Perl 6 but a variety of other languages. And it even goes back to – and I know you remember this, Steve, but you'd have to be an oldster to remember – DeskView under DOS, which allowed you to run multiple environments at the same time. Remember DeskView?

Steve: Oh, yeah. And in fact we talked about it briefly when we were talking about memory management.

Leo: That's right.

Steve: And we're going to be talking about that in a few minutes, as a matter of fact.

Leo: Yeah.

Steve: And then, but one last example of contemporary virtual machine, which certainly Windows users are beginning to see, Microsoft initiated a strong move that they called .NET. This whole Microsoft .NET framework is a – they created what they call a CLI specification, a Common Language Infrastructure. And then there's something called the CLR, which is the Common Language Runtime. And Microsoft succeeded, along with Hewlett-Packard and Intel, in getting ECMA and ISO standardization. So even now there is still new technology that is just based on software virtualization, software virtual machines that are being used for, like, hosting multiple languages. And exactly as you said, Leo, the future of Perl is moving in this direction, as well.

Leo: I think it's just fascinating. So and in the past, I think the rap against things like p-code, and even Java to some extent, was that that virtualization slowed it down. But I think because of hardware support it's really not much of a penalty anymore.

Steve: Well, actually there are two different – two very different types of virtualization. There is emulation, like we were talking about. And then there is, exactly as you were saying, a hardware-assisted virtualization. And that's the key for security because, well, I guess I shouldn't say that completely because, of course, Java was created to create a secure sandbox so that Java applications, because they were being emulated, the code itself is not running on the actual hardware. It's not actually being executed by the Intel chip.

We've talked, for example, about how stack overruns are able to allow a program's code to be executed by the chip itself. Well, in an interpreted environment you're not running on the hardware. You're, like, one level removed, and your code is being interpreted by some other software. If it's written correctly – and that's the Achilles heel in any software-based scheme – if it's written correctly, we know how difficult it is for that simple phrase, if it's written correctly,

to end up being actually true in the real world. Because in theory, then, the emulator could prevent any sandboxed code from ever having the ability to do anything dangerous. In reality, it's very hard to write these things correctly; and we end up with software layers and, as we've seen before, the opportunity for some sort of trouble.

Leo: So when you say "emulation," you were talking about, for instance, the Apple emulation of the old G5 processor in Intel. That's...

Steve: Yeah. Or, well...

Leo: That's one way.

Steve: And, for example, one of the things that killed off the UCSD Pascal – UCSD Pascal was going strong. It was a great solution. It was multiplatform. I mean, I remember using it on the Apple II. What came along? Some guy named Philippe Kahn.

Leo: And native...

Steve: Philippe Kahn had Turbo – Borland International released Turbo Pascal, and it just blew UCSD Pascal into history almost overnight. First of all, it was less expensive. But it was...

Leo: 40 bucks.

Steve: It was blindingly fast.

Leo: Yeah.

Steve: Well, actually that's the thing that started the big war between Philippe and Bill Gates was, you know, Bill was selling languages for, I think, like, 499, $500 at the time. And Philippe came along and thumbed his nose at Microsoft and said here's a really good Pascal compiler for 40 bucks.

Leo: I loved Turbo Pascal. Oh, great program.

Steve: And it was screamingly fast because...

Leo: It compiled instantly. Instantly.

Steve: It compiled to native machine language, not to p□ode, which then had to run through a layer of emulation. And the reason we're spending so much time talking about this is it is critical to understand where the costs of virtual machine technology comes in, and virtualization. Inherently there's something added, something extra added to the system to create this layer of protection, the sandbox, the virtualization.

Leo: It has to do more work.

Steve: Yes. Well, and it depends upon whether it's really being done in and by the hardware, or in and by the software. What we've talked about so far is all software emulation and software virtualization, which is very expensive in terms of performance. Now, with chips getting faster and memory becoming abundant, that's less of a problem. On the other hand, when you don't have virtualization, you're faster. Who doesn't want to be faster? No matter how fast your chip is, not having something slowing you down is always a better thing. Which of course is why, you know, the Mac apps are rapidly being recompiled into native Intel code as the Macintosh switches from the PowerPC platform over to the Intel platform. And you're going to get, you know, much better performance that way.

Leo: Right, right.

Steve: So historically what happened is Intel made some advances to their chipset. Of course, the original PC was hosted on the 8086, which was sort of a 8-bit/16-bit hybrid chip. It had a 16-bit instruction set, but an 8-bit bus. And whereas the 8088 was 8-bit all the way. The change came for the first time with the Intel 8286, which added something in the hardware which radically complicated the chip. I mean, it went from a small chip and instruction set that anyone could understand to something that began to be really complicated. It added the so-called "protected mode." Protected mode, as opposed to real mode, which was actually named after the fact because that's what chips always were before, so they had no mode, they were just there. Protected mode adds a whole bunch of hardware to the chip which begins to create this notion of interprocess isolation. Rather than all of the programs running in the system sort of sharing memory at once, protected mode protected programs from each other. It really began to create the notion of a real operating system. Remember that DOS was sometimes criticized for just how lame it was, I mean, that it was basically just sort of a program loader.

Leo: It was a pile, with all these programs running against each other.

Steve: Yeah, I'm just pissed off that the slashes go the wrong way in DOS. Because I'm still...

Leo: That's another problem.

Steve: I'm still doing it wrong. I'm never going to shake that bad habit.

Leo: UNIX uses forward slashes, and DOS wants backwards slashes for reasons no one understood.

Steve: Yeah, well, just to be different, or who knows. Anyway, so we really didn't get the hardware we needed from the 286. And it wasn't until the 386 came along that Intel added enough features to basically create virtual machine technology in the hardware. And this was a huge jump for the industry. And in fact that's where we got the memory managers, EMM386...

Leo: QEMM, yeah.

Steve: Exactly, QEMM. These things – and in fact my favorite was Qualitas. I can't remember what that one was called. It was...

Leo: Wasn't that QEMM?

Steve: No, that was Quarterdeck.

Leo: Quarterdeck, that's right.

Steve: Quarterdeck was QEMM.

Leo: And they did DeskView also.

Steve: Yeah.

Leo: Qualitas. Oh, yeah...

Steve: I think it was EMM386.

Leo: Yeah, and these would let you use more than 640 kilobytes of memory, in fact even more than a megabyte of memory.

Steve: Well, yeah. What happened was, machines at the time were having a real problem with the 640-meg limit, which is what the PC had. You know, it shipped originally with – I think you could get it, actually, with 16 megabytes. Sometimes you would expand it to 64 megabytes. But there was a physical limitation. In fact, Bill Gates famously said – and we can't really fault him for this because it was so long ago – that no one could ever possibly need more than 640 megabytes...

Leo: I have that recording somewhere.

Steve: ...of memory. You know. And in fact, you know, that was...

Leo: In his defense, I thought the same thing. It seemed like a lot at the time.

Steve: Well, you know, and here I am, still programming in Assembly language, it is a lot.

Leo: If they would write lean, mean code like you write, it would be. But nobody does.

Steve: Well, of course what happened was machines began to be used for things no one ever envisioned them for – huge spread...

Leo: Like graphics. Yeah, yeah.

Steve: Graphics. And of course media. And huge spreadsheets that just needed more memory.

Leo: The splash screen on most applications takes up more than 640K.

Steve: It does today, right. So there were several intermediate stages along the way. There was this thing called expanded memory, and extended memory, which used a bank-swapping approach, where you'd have a card with extra chips on it, and then you'd programmatically, like, swap that memory in so that it was accessible, and then you'd swap it out.

Well, what happened with the 386 was that the hardware was so advanced that literally different profiles, different configurations of memory could be shown to applications at any given time. So the first memory managers were emulators of this EMM, the expanded memory and extended memory systems. And then they began to grow. And as you said before, Leo, it was – there was TopView that IBM was working on, and then there was DeskView that Quarterdeck created. And basically they were the first people to begin to move toward using the 386 chip hardware to create hardware virtual machines.

Leo: 386MAX.

Steve: That's the name. Yay. That's...

Leo: It just came back to me.

Steve: Yes. Bob Smith at Qualitas did 386MAX.

Leo: That was the best program. I used – we were all QEMM users until 386MAX came out. And it was, like, wow.

Steve: Well, you're so right. In fact, a little bit of a segue – or not segue, an anecdote. When Microsoft first came out with Windows – I guess it was 3.1, maybe 3.0, because we had Windows 2.0, and then 3.0. The problem was it only ran with Microsoft's own memory manager.

Leo: Which was built in. You'd run it in configsys.

Steve: Well, and back then it was just crappy. I mean, it had no good options. With 386-to-the-Max, you were able to fill in from high memory into low. There were all these extra features that Bob Smith had put into 386-to-the-Max. I mean, and the world loved them. Well, so here comes Windows. And it's like, no, you have to use ours. And it just didn't, I mean, as I said, it just didn't have any of the features that we all wanted.

So I was writing my weekly TechTalk column for InfoWorld at the time. And I loved what Microsoft had done for Windows 3.0. But at the, like, the last paragraph in the column where I

raved about it, I said, but I can't use it. Because it's not compatible with 386-to-the-Max, and...

Leo: And that's all I'm ever going to use.

Steve: So, exactly, so who cares? The colors are pretty, but oh well. And so then, but I said, but next week I'm going to talk about Windows 3.0 some more. So next week I talked about other features and talked about how, you know, other cool things, blah blah blah. But I said at the end of the column...

Leo: But I'm not going to use it.

Steve: ...but who cares. You know, no one can really use this thing, this new Windows, because it won't run on any good memory manager. So I got a call after the second column came out from a wonderful guy at Microsoft, Brad Silverberg.

Leo: Oh, yeah. He's still around.

Steve: In fact, well, he left Microsoft quite a while ago. He was originally at Borland and moved over...

Leo: Right, right.

Steve: ...to Microsoft. And we had a good relationship. And so he calls me up, and he says, Steve, would you stop writing that, you know, Windows is junk unless it runs on some other memory manager. And I said, well, but Brad, it is junk unless it'll run on, you know, on, like, a real memory manager. And he said, well, I have Bob Smith here with me in the office.

Leo: Oh.

Steve: And he was on speakerphone. He said, "Say hello, Bob." And I heard Bob say, "Hey, hi, Steve." I said, "Oh my goodness. Bob, are you there because of why I think you're there? Why I hope you're there?" And he said, "Yup," he said. And I said, "Have they given you everything you need?" Because it was, you know, it's proprietary information.

Leo: Right, right.

Steve: They wouldn't tell Bob how to get 386-to-the-Max to run under Windows. And he said, "Yup, I have everything I need." And I said, oh. I said, "Okay, Brad. Look. There's one more column in the pipeline already that you're really not going to like." But I said, "Then I will tell people..."

Leo: I'll fix it after that.

Steve: "...that you've given Bob Smith what he needs, and that 386-to-the-Max will soon be Windows 3.0 compatible."

Leo: Oh, that's great.

Steve: So, I mean, those were painful times back then as we were figuring out, you know, what mattered and what was important and how to make all this stuff work.

Leo: Ah, but they were good days for geeks because it was something you had to figure out. And we shared the information. Now it's just any – anybody can use a computer now.

Steve: Okay. So what is it...

Leo: I sounded like Dvorak.

Steve: What is it in the hardware that makes this possible? What Intel added with the 386 was the ability to basically set up a supervisory code somewhere in RAM that could get notified under virtually any condition that was important. Specifically, computers – or software running in a computer has two fundamental things it can do. It can read and write to memory, and it can read and write to I/O devices. So it's got I/O, and it's got memory. And what Intel's architecture that began in full with the 386 and has continued on – and in fact the thing you mentioned, Leo, about them talking about, like, next-generation stuff, they're going further with this x86 virtualization technology, even beyond what we have today, in order to sort of go to the last step. Because Intel has recognized that virtualization is hot and is coming back, you know, gangbusters. And so that, you know, they're also looking for some way to make some more money because, I don't know if you heard, they're really getting beat up by AMD.

Leo: Yeah, they sure are, yeah.

Steve: Anyway, what they added was the ability to trap any attempt by software to access I/O and to trap any attempt by software to read and write memory. Well, that's, I mean, that's a phenomenal thing. I mean, it's a huge change in the architecture because, you know, the software is supposed to be able to read and write memory. I mean, in order to do what it does, it has to be able to do this. And, you know, what good is it unless it can communicate with the outside world, which means it needs to be able to access and talk to I/O devices.

So there is a very sophisticated set of interception hardware, essentially. There's literally a bitmap that represents the I/O addresses so that individual I/O addresses can be protected and others can be deprotected. The way memory is managed is that there are, again, complete protections and tables, called "page tables," that allow some supervisory process to literally control what a less privileged process is able to do. And this is all supported by the hardware, meaning that there is virtually – pardon the pun – virtually zero overhead with a program running within this contained or protected mode environment. It is...

Leo: How, I mean, zero? I mean, how, really, come on, there's some overhead.

Steve: Well...

Leo: You're saying there's very little overhead.

Steve: Well, there's zero overhead for a program running in that mode. There's some overhead when you change programs, change...

Leo: Overall system-wide, though, okay. Right.

Steve: ...modes, yes. And in fact there is also some overhead by, if the program in protected mode needs to call down for services that are running in the operating system, they're called "ring transitions." In the Intel architecture there are four rings, 0 through 3, and 0 being the most highly privileged, ring 3 being the least privileged. And so there were mechanisms that had to be created to allow the operating system, you know, the supervisor basically that has responsibility for everything else, it needs to be able to access all of memory and access all of the I/O and to contain programs that are less privileged. The way these are less privileged is they run in what's called "ring 3," which is just an abstraction. It's an abstraction in the same way that, you know, a port is an abstraction of an Internet connection, where it's just sort of – you just say, this program is running in ring 3, and programs running in ring 3 have the following restrictions placed on them.

Leo: So there is a little bit of overhead switching from ring to ring. But because it's done in hardware, it's virtually instantaneous.

Steve: Well, yes. It's nothing, nothing like the emulation we were talking about originally, where you've got software that is executing instructions, you know, and figuring out what instruction does, and then turning around and doing the equivalent. It is, you know, basically it allows programs to run at full speed until they do something that, not necessarily that they shouldn't, but something that requires supervisory oversight.

So, for example, I mean, a perfect example is running an old DOS program. I mean, DOS programs, as we know, still run in a window in Windows. You know, a so-called DOS box, or even that command prompt that we were using before. The DOS program thinks it is writing to an area of memory, you know, technically it's B1000 in hex or B800 in hex. There were two pages of and regions of memory in the old PC. And DOS programs wrote, actually physically wrote a byte of data into a location that represented a cell on the screen. And the display adapter took the contents of that byte, which was an ASCII character, 8-bit ASCII character, and then displayed the character at that location, at the matching location on the screen. So DOS programs just wrote – were able to sort of paint the screen very quickly with text by directly depositing data into this region of memory.

Okay, well, everything has changed since that. You know, we don't have text displays. Now we've got bitmapped graphics displays, you know, with fonts and font sizes and varying resolutions and all this. Yet you can still run an old DOS program in one of these – in a DOS box, in a command prompt. And it still writes to what it thinks is video memory, into this B1000 hex region. What happens is, the operating system has set up a trap so that anything that is written by the application in the DOS box is intercepted by the software. The software is able to say, oh, look, isn't that cute. DOS thinks it wants to put an "A" here. And so what the operating system does is turns around, figures out where that is within the window in the bitmapped region, and it then paints the fonted character "A" that the user has chosen with color and size and everything else. DOS has no idea this is going on. All of that is hidden from it. It thinks it's still running on an old PC with an actual text display, putting text out in the memory. When in fact, you know, that's not remotely true anymore. We've got graphics accelerators and NVIDIA cards with turbofans on them and everything else going on, all being hidden, thanks to this

virtualization technology.

Leo: Yeah, yeah. Now, there's some overhead in doing that kind of thing.

Steve: Yes. There's an example where you are mapping from a text display into a bitmap display and having to do some work. Of course, in this case, our machines are so radically faster...

Leo: Yeah, these DOS programs didn't expect anything like that.

Steve: Exactly. The result, you know, still looks instantaneous to us. So essentially what we've got today – and we're going to build on this next week when we talk about how this technology has been leveraged and how it can be taken further – is we've got an operating system now, every operating system that is contemporary, I mean, Linux, FreeBSD, the Mac OS, Windows, all of them, the first thing they do when they start to boot is they switch the chip from its real mode into protected mode. Real mode is the way all chips – there's one obscure chip somewhere that Intel had. It was an embedded chip that just didn't have real mode.

But all of our chips start life, when you turn the power on, they're in the original 8086 emulation, you know, real mode. And that's what the BIOS uses, 16-bit BIOS, to get things going. It loads a sector from the hard drive into memory, executes it, that bootstrap, reads some more sectors to build up enough code in memory. As the OS gets going, one of the very first things it does is it lays out all of the data required by the Intel processor. And there's a lot of housekeeping that needs to be done to get this system from real mode into protected mode because it's got a – basically it's making a huge change in the architecture of the chip on the fly. And so it sets all that up, switches the chip into protected mode. That's the first time it has access to all the memory in the system. It's still limited until then to the original 1-megabyte physical barrier, of which only 640 was RAM. So as soon as it does that, then it opens up the rest of the system and has access, thanks to being 32 bits, to the vast amounts, you know, gigabytes of memory, and then it loads the rest of the operating system and gets itself going.

So all systems today are running in this mode. And they are using features of this hardware that creates these virtualization – that allows processes to be insulated from each other. The problem is, as we've seen, that insulation between processes has got some holes in it...

Leo: Permeable.

Steve: Yes. It is very permeable. As I gave the example a couple weeks ago, the very fact that I can start a debugger and reach over and stop another application and single-step it, and even inject my own code and make changes, that's a problem when you talk about security. And as we talked about with rootkits and trojans that are going down and mucking around in the network stack, you know, remember that last week you asked would the netstat command to show you what's going on in the machine be guaranteed of absolutely always showing everything that is happening? And I said no, it can't, because it's sharing a computer with other code. And so theoretically other code could modify what we think we're seeing.

Well, it turns out that, as many people know, there are things like VMware and Virtual PC, which is the technology that Microsoft bought from somebody, which begins to create much stronger boundaries and barriers. And these things are extremely useful from a security standpoint. And we're going to be looking at specific examples of that next week and understanding, you know, what kinds of security features they offer.

Leo: Well, I think people are probably starting to get an idea that the virtualization means you can wall off part of your computer from another part of your computer. And I...

Steve: What's exciting is there's still software involved, but it's much more supported and enhanced by the hardware support than is emulation. And it is much more likely that you, as you said, Leo, that you can wall off, and you can create truly, I mean, like, provably secure, provably impervious containers. And the really cool thing is the idea, for example, of creating a virtual machine or a sandbox in which you surf the 'Net. If you do that – and you still have to behave yourself because again, you know, you could download something and drag it out of the sandbox into the rest of your machine and still get yourself in trouble. So it's, again, there's no perfect solution that, you know, can exist. But it can certainly prevent malicious things.

For example, we were also talking last week about that Windows Metafile vulnerability still being around and infecting maybe a million MySpace users. If those people were surfing in a secured virtual machine web browser that had no contact inherently to the rest of their machine, well, that Windows Metafile vulnerability could infect the virtual machine, but that infection could never get any further. And when you terminate the virtual machine, any infection is snuffed out at the same time.

Leo: That's – I use a virtual machine to run Windows on my MacBook. And we're going to talk about VMware. We get a lot of email lately, and VMware has done some neat stuff, including offer some stuff for free. And even Microsoft now is offering copies of Virtual PC for free. So there are a lot of free ways that you can do this.

Steve: Well, and I think that, you know, we've talked about how long-running themes of ours in Security Now! will be, you know, keeping touch with what happens with VPNs and what happens with, you know, various, like, long-term security technologies. Virtual machines is going to end up being in our toolkit because – for exactly this reason. I think we're just seeing the beginning of this. I would imagine a couple years from now there will be a number of competing virtual machine products. And these things will be easier and easier to use.

Leo: I agree. I agree.

Steve: Yeah. I did run across one really interesting sort of concern. I haven't had a chance to explore it fully yet. I'll know everything there is to know about it a week from now when we continue with this. But there was a researcher did some work on the notion of using virtual machines against us. That is, unfortunately, that power, as is the case with all power, can be turned to the dark side.

Leo: Oh, boy.

Steve: There's something called the "Blue Pill."

Leo: Oh, yeah, the Blue Pill. Yeah.

Steve: And that, of course, refers to our friend Neo and "The Matrix," the movie "The Matrix."

Leo: There's a security researcher, a woman who has the Red Pill and the Blue Pill.

Steve: Yup.

Leo: She's got the Red Pill, I think, is a solution to the Blue Pill.

Steve: Exactly.

Leo: As I remember.

Steve: And the idea is that it is possible to essentially, I mean, exactly, I mean, it's a perfect analogy to "The Matrix," where you believe you're in the real world; but in fact, you know, you're in a virtual machine, you're in a virtual world. And the idea is, it turns out that due to insecurities that exist in the way our current operating systems have been implemented – and again, there are workarounds for this, and one hopes they will happen soon. But due to the way these have been implemented, it is possible to literally take the Blue Pill. You run this program, and suddenly nothing you see changes, but you are now no longer using your computer. You have just been encapsulated in a virtual machine. And something going on on the outside of that virtual machine, you would have no way or knowing or seeing. And at that point it is impenetrable.

Leo: Isn't that interesting.

Steve: That's just so cool.

Leo: Well, we'll talk more about this next week because there's a lot more to say.

Steve: And all enabled by the hardware, which is where this happens with very good speed. And thanks to the 386 that has been evolving ever since, and Intel's still not done.

Leo: Yeah, yeah. Quite an amazing thing. Fascinating subject. This has been as much a history lesson as it has been a technology lesson. And I hope you've enjoyed it. We're glad you listened in.

distribution. And then what's cool, I think it's 79 euros a year, you can subscribe to their updates and get the spam filtering, the content filtering, the antivirus, the firewall, all this stuff running on this thing. So if you want to build a very inexpensive, very powerful gateway as an end-user, this is a great solution, too. Astaro.com.

And I think we should mention GRC.com, don't you, Steve?

Steve: That's where all this comes from.

Leo: Yeah. Astaro may sponsor us, but GRC sponsors everything Steve does. Of course, it's his company, the great company that gives you SpinRite, among all the other wonderful things GRC.com does. If you have a disk – and who doesn't – a hard drive, you need SpinRite, the ultimate disk recovery and maintenance utility. If it can be recovered in software, SpinRite can do it. There is no better program in the world.

Steve: It does routinely perform miracles. People can go to SpinRite.info and read some of the unsolicited, you know, actually they're love letters that we get back...

Leo: It's so neat, yeah.

Steve: ...people saying thank you, thank you, thank you, and telling their stories. I just – I love getting those.

Leo: It's kind of, you know, Steve, it's kind of neat because essentially, you know, you had a big company with hundreds of employees and a big building, and you pared it down, and you've got your focus now. It's very simple in his life: Security and SpinRite.

Steve: Yeah.

Leo: And that's it. And you're doing what you love, and you're giving everybody just a really great product and great information. So it's...

Steve: Well, and SpinRite's owners are really the sponsors of Security Now!, from my end.

Leo: Yeah, absolutely, always have been, yeah. Hey, thank you, Steve. We will talk again next week. For more information about what we've talked about, links, show notes, and all of that stuff, there are two places to go. Of course Steve's site is GRC.com/securitynow. He also hosts a transcript there. This would be a good one, I think, to get a transcript of. And I know Elaine's typing like crazy as we speak. But you can also get a 16KB version, easier download if you're on dialup. And there's a great forum there where people discuss the security topics Steve raises.

Steve: I'm glad you reminded me of that. We have a news server, and anyone who has a newsreader – most people have that built into their browser – can go to news.grc.com. And we do have a Security Now! newsgroup, which is constantly, I mean, it's a great place to discuss topics of the show and also ask questions. There's just, I don't know, we have hundreds of

really good people who hang out in our GRC forums.

Leo: And I'm going to try to be a little bit better on TWiT.tv, now that we kind of have a GRC, you know, a Security Now! page on there, to put more information and more links in there, instead of just shoveling you over to Steve. Ultimately, of course, there's always a link to Steve there. And that's really the place to go for the definitive stuff. But just things like, you know, a Wikipedia article about 386MAX, that kind of stuff, I'll start putting some of that stuff in the show notes at TWiT.tv because I know that people are looking for that stuff, too.

Well, Steve, we have really started an interesting topic. I have one more question for you, and please don't yell at me. But somebody asked me, is the VPN stuff still in progress? What's going on there?

Steve: Yes, it is still in progress.

Leo: Don't feel bad. I'm not trying to guilt you, I just...

Steve: Yeah. I've got about 20 pages to find. The problem was that I literally didn't have anywhere to put it. You know, GRC has been growing over time by my just adding stuff and adding stuff and adding stuff. And I decided I really wanted to deal with third-party cookies, that is, the issue of sort of bringing that back to people's attention, which you and I will be talking about once I get that online. And then the OpenVPN stuff is, I mean, it's absolutely going to happen. The problem is that my site has never had any navigation or, you know, or menuing or anything. I mean, I just haven't had any.

Leo: Well, you just threw pages up, and eventually – the more you get on there, the more you need navigation at some point.

Steve: Yes. And, I mean, people who love the work that GRC has done over the years, professional web designers write to me and say, Steve, you know, wow.

Leo: Want a hand?

Steve: What's the problem? And so...

Leo: But you're like me, you do it all yourself.

Steve: Well, and I've wanted to understand what CSS is, cascading style sheets. And I wanted to do a menuing system. See, the problem is, if I had anybody else do it, they'd come along with some JavaScript blob that was spit out by some web authoring tool. And, I mean, I'd just look at it and shudder. It would be nothing that I could use. So for the last month and a half, I guess, I've been working with CSS. I have now – and actually all of this has been done in public view in our newsgroups in the news.feedback forum we have there. We've got a menu system now running, no scripting, pure CSS, it runs on every browser. Actually there's nothing like it on the web. I've ended up developing up some very cool technology to allow zero scripting, pure CSS, beautiful hierarchical menus. So that'll be appearing shortly. Then I'll be finishing up the work I was doing on cookies. Then I'm going to finish up the OpenVPN stuff.

Leo: Okay.

Steve: So, I mean, I know people are waiting for it and anxious. I only want to do it once, and I want to do it right. So that – and I just had nowhere to put it on the site.

Leo: It's on the punch list. Steve's working on it.

Steve: Yup. It is.

Leo: That's all we can say. And Mr. G works at his own pace on his own stuff, and that's why we love him. You should read Dvorak's column on CSS, by the way, online at PCMagazine.com this week. I think you'll enjoy it, Steve. He's basically saying the same thing you're saying. I don't know if you've been talking together, but he agrees with you. And he's not coming to it as a programmer. I mean, he's just trying to observe what's happening in the world. And it's just obviously a problem. We have very good designers working on the TWiT.tv, and they did a very nice job. And I don't believe they – there is JavaScript on there, but I don't believe they used any JavaScript for the navigation stuff.

Steve: Well, yeah, here I am, preaching not having scripting. So, you know, how can I – how could I require scripting on GRC? And the other thing is I certainly didn't want to, since scripting is lowering your security, it would be bizarre for me to say, oh, you have to lower your security to come to my security site.

Leo: Right, right. No, I agree with you. I agree with you 100 percent.

Steve: So it made no sense for me.

Leo: The only scripting we have is for the Flash player on the site. And it's just a way, frankly, of handling the Eolas patent issue that's happened to Microsoft's Internet Explorer. You can't embed multimedia objects anymore in a simple one-click fashion. So we have a little JavaScript there.

Steve: Well, and most menus are generally JavaScript. They will still function maybe in a crippled fashion, but you need to turn scripting on in order to get a next-level dropdown to work.

Leo: Well, as you've learned, you can do it in CSS. It's just a little bit of work.

Steve: [Groaning] Yeah.

Leo: Yeah. Yeah.

Steve: It's, yeah.

Leo: All right, Steve. Hey, it's a real pleasure. We will talk again next week. I'm off for Toronto to do my TV show, Call for Help. But we'll be back next week to talk more. And you're going to come to Toronto next month; right?

Steve: Yup.

Leo: You're putting it off this month. It's a little muggy there. I don't think you want to go.

Steve: Yeah, I don't do well in...