SQL Injection Exploits

Description: Steve and Leo wrap up their three-part series on web-based code injection vulnerabilities and exploitation with a discussion on web-based structured query language (SQL) database attacks. They explain why and how SQL injection vulnerabilities are creating an ongoing plague of vulnerabilities besetting modern 'Web 2.0' applications.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-087.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-087-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 87 for April 12, 2007: SQL Injection.

Time to get secure. And you can never be too secure. He's the master of security, Steve Gibson, in his secure lair.

Steve Gibson: You know, Leo, speaking of getting secure, we get so much great feedback from people who, even if they thought they were really security knowledgeable before, I mean, people who've been in IT all their lives; but they invariable say that, even when we're discussing a subject that they feel pretty well versed in, they'll still get a tidbit here and there, something that they haven't encountered before. So...

Leo: Well, I don't think it's possible, even if you're – and I think if you're a pro you know this. Even if you do this all the time, stuff changes. It's hard to keep up.

Steve: Well, yes. And in fact, I had – a bunch of people wrote to me about the recent event which happened in the AACS area, remember we've talked so much about the – uh-huh. You've probably heard that WinDVD is having its keys revoked from it.

Leo: Yeah. The first example of key revocation.

Steve: Exactly. And this we talked about as something that would certainly be possible to have done. Now, it's also the case that having a software program's keys revoked is far less onerous,

for example, than having your Toshiba HD DVD player 's keys revoked.

Leo: Right, right.

Steve: That would be a big annoyance, needless to say. But still this demonstrates that Hollywood, although it's taken them many months, you know, this of course all comes from the muslix64 guy who started this, and it turns out that they've been continuing to do work on this in the Doom9 forum, and there's now a utility developed by somebody else that basically dumps all the encryption keys from any disk that you insert in a PC with WinDVD. So as we expected, they've automated and polished this approach and made it so easy. And so finally Hollywood is saying, okay, no more. We're revoking WinDVD's keys. Now...

Leo: Now, does that mean that this technique doesn't work anymore?

Steve: No, it's going to be a cat-and-mouse game. What's happened is, the InterVideo people who publish WinDVD, they've done some things to a next version of WinDVD which existing WinDVD owners can upgrade to, to hopefully – or they're hoping, anyway – thwart this kind of exploit. Maybe they're protecting it more, they're not leaving the keys out in the open as usual, they're obscuring them in some fashion. But again, this is – because the player has to have the ability to decrypt the content, and it's a fundamentally open platform, under XP this is going to be very hard for anyone to prevent.

Now, as we've talked about, Vista has this notion of protected processes, and Microsoft has put a lot of energy into tightening Vista down in ways that would make this much more difficult. You just can't run a debugger on a Vista process which is running in this protected process environment and dump its memory. So Microsoft is, you know, building walls around processes for this reason, to keep this kind of easy exploitation from being, I mean, really so easy to do.

Leo: So they're calling this a "mandatory update." What happens if you don't update?

Steve: Well, essentially WinDVD has a set of encryption keys which were given to it by the AACS authority. Those are going to be invalidated by new DVD content. So as you, for example, downstream the newer published HD-DVDs and Blu-ray DVDs, any of the technology using this AACS next-generation DRM – digital rights management – for high-definition content, they will know that the earlier versions of WinDVD are not safe because they have been reverse-engineered. So those things, that content simply will not play on the older WinDVD players. Older content might. It's not exactly clear how this is going to work, that is, whether the download of the new version will automatically remove the keys from the older versions; whether you could still play an older disk; or whether ever once letting a new disk touch your system would leave a mark behind that prevented the older disks even from playing.

Leo: I didn't realize that. So, yeah, because that's the question, is does it make your existing stuff not work? And apparently that's the case.

Steve: It certainly could make your existing stuff not work on the existing player. So the idea is you need to upgrade your player. And, I mean, this will be sort of fun to watch from the sidelines because, you know, the Doom guys, they're not going to take this lying down. They're going to say okay, roll our sleeves up a little bit further, and we're going to do this again. I mean, if for no other reason than to prove the point that what Hollywood is trying to do is, as we've said over and over, is fundamentally impossible.

Leo: Amazing. Fascinating cat-and-mouse game going on. And for those who said, oh, don't worry, nobody will ever revoke keys, well, now we know they will.

Steve: Well, yes. And in fact, again, though, I would say that this is the case for software players. It'll be very interesting...

Leo: It's different for hardware, yeah.

Steve: Oh, well, consider, I mean, you know, that's the nightmare scenario, that hardware that people have purchased, which is not as easily upgradeable as clicking on a link and getting a new version of it, that hardware people have purchased would then stop functioning because some instance of that hardware somewhere had been reverse-engineered and hacked in a way that allowed people to use that hardware in order to decrypt high-definition content.

Leo: It's my suspicion that the movie industry isn't so worried about hardware since it's software that really gives you this piracy capability, and as we see with this decryption. It's a little harder to make it, you know, so what if you crack a machine. Right?

Steve: Right.

Leo: Ah, well, fascinating stuff.

Steve: So also I received some questions about why I never posted Episode 85a on the Security Now! site. I didn't. I sort of thought it was, you know, that was our early last week announcement of the ANI cursor vulnerability disclosure. And I sort of thought of it as a little quickie, like, oh, just a little notice for people; and besides, we'll be doing a regular podcast in three days anyway. But people were saying, hey, where is that, I need that, I want that.

Leo: They want the complete set.

Steve: Yes.

Leo: Collect all 87.

Steve: So by the time anyone is hearing this, I will have added 85a to the Security Now! page. So anybody who wrote to me saying where is that, I want it, I need it, it's there now.

And the last thing was, last week I referenced some errors that people were reporting which have now been acknowledged by Microsoft and resolved. That was an error that occurred to me on one of my XP machines where I installed the ANI cursor vulnerability fix. When I rebooted the system, I got [glitch] came up complaining – the message said the system DLL, user32.dll, was relocated in memory. The application will not run properly. Well, it turns out it's because this machine actually, this happened to be a cute little HP Pavilion, little mini Pavilion that I have, it uses the RealTek HD Audio Control Panel, and that was one of the things that conflicted. There was also a German tax calculator, something called TUGZip, and a CD tag program. Those all gave this patch a problem.

So anyway, we're now talking, here we are on the 12th, which is after the second Tuesday, Microsoft's normal patch cycle, which was this most recent Tuesday on April 10th. And during that they updated this fix again to cure the problems that people were having. There was an interim hotfix that I was preparing to tell people about today, which did fix this problem for this week between the ANI release and the normal Patch Tuesday, second Tuesday of the month release. But it turns out Microsoft incorporated that already into the – and there's, like, another three or four patches, depending upon what version of Windows you have. So once again, you want to make sure you're patched. But anyone who was having this problem will have it cured here by our normal second Tuesday.

Leo: Very interesting. Okay.

Steve: And two last little quick points. Several people who have been liking and following along our discussion of eBooks and the Sony Reader and so forth made mention of two really interesting sites that have literally tens of thousands of free eBooks. And I know you'll be interested in this, Leo. One is manybooks.net...

Leo: Let me write this down, okay. Will these work with the eReader, is the question.

Steve: Yes. And in fact at manybooks.net you select which format of, like, I don't know, like 20 formats that you want the books in, and one of them is RTF. And in fact the guy that wrote to me about manybooks.net has a Sony Reader. He downloads these things in RTF format. Then, because they're in RTF, he then will use Word or WordPad to maybe enlarge the font or bolden it or whatever he wants to do, and then he says it works beautifully with the Sony Reader.

Leo: Because that was the problem we had with McCollum's books was that, even though they were PDF, they didn't really seem to work very well.

Steve: Well, yes. And I'm playing around with doing a book converter that'll just, you know, for a little quick project here, just because I want to play with the stuff before I set it aside, which will create texts in native format that allows you, for example, to resize the text so that it's comfortable for you. And of course RTF format allows you to do that. So anyway, and I did look at manybooks.net, and there's a bunch of stuff. They have, they say, at the moment 16,380 completely free eBooks, many of...

Leo: Mostly public domain, I would imagine; right?

Steve: Well, many classics, but some surprising books that look like – there was one that was, like, published back in 1920 that was – it looked like a huge compendium of English idioms, you know, and common phrases and things. I thought, oh, that'd just kind of be fun to have on there to browse around in. So there's that. And then there's also www.wowio.com, and that's another free eBook site. And I'll have links in our show notes to both of these for people who are, you know, driving while they're listening to this.

Leo: This has become the eBook show.

Steve: Not intentionally. There were some people who expressed some frustration that we

were spending so much time on that. But it's something that you and I were both interested in. And I'm such an avid eBook reader that I wanted to share this for people who wanted to know about it, so...

Leo: Thank you.

Steve: So, yeah. We don't want it to be the eBook show, but I did want to let people know what was going on.

Leo: Hey, we talk about what we're interested in. That's sometimes more than security. Not usually. All right, Steve. What are we talking about this week?

Steve: Well, we're talking about what is arguably the, well, an incredibly big problem for what's happening with the web. In fact, as I describe this to people, as I'm about to, you could almost imagine that a system could not have been deliberately designed to be worse than what we have.

Leo: Isn't that nice.

Steve: And again, it's sort of old technology which is meeting new technology and having a real problem with it, essentially. So the old technology is this database technology known as "sequel," or SQL. SQL is an acronym, of course, it stands for Structured Query Language. The good news about SQL is it was designed, you know, way pre-Internet, pre-web, as a very powerful database technology. The thing that makes SQL so powerful is that it is inherently a language-based data access system. So, for example, you generate so-called SQL queries, which are, for example, of the form select subject event date and content from a certain database name like newsletters, where the newsletter ID is 5, for example.

So it's almost sort of an English-y sort of like query language where you build up your query by putting together verbs and specifiers, variables, into this thing. And you're able to create tables. You can add items to tables. You can add columns. I mean, Mark Thompson, for example, our friend at AnalogX, he's just jumping up and down about how cool SQL is because as a programmer he's very comfortable with the idea of a linguistic interface to data. And so SQL is extremely powerful because it just allows people to have data contained in these tables with records in columns, and rows are like records of the individual database records. I meant fields and rows, sorry. And the idea is that using this linguistic approach you're able to make – basically that's your entire interface to the database. And so when you issue these queries, the result sort of comes back from the database with your answers, which is in the form of another table or additional information that you've asked for.

So SQL was there for a long time. And it made sense, as we moved from sort of static pages that we talked about before into the Web 2.0 dynamic pages, it made sense that, if you had an online forum, for example, an online bulletin board system, that as you're creating and accepting postings from people, you need to keep them somewhere. They need to go into some sort of a database in order to be stored on the hard disk in a way that they're accessible and can be efficiently retrieved. And in fact, compared to prior databases, that's really the whole concept behind SQL is over a network, for example, where you don't have high-speed connection between the client and the server, old-style databases would have required that you basically rifle through the data by sending it back and forth between you and the database, if they were, for example, over a network on separate machines.

The cool concept that made SQL so powerful is that, instead of having sort of a dumb database

where you basically have to go through the entire database, moving it across the network, checking each record to see if it's what you want, instead you send the query across the network. And now the database is smart, and it interprets the query, applies it to the database that you've asked it to, or in some cases even more than one database at the same time, and then gives you back the results. So SQL was really designed to be sort of a transnetwork solution for dealing with data in a very efficient fashion.

Leo: And there's a lot of different flavors of SQL. Microsoft makes a SQL Server. Many of us, for instance, on TWiT we use the open source MySQL database.

Steve: Which is becoming extremely popular.

Leo: And uses the same, basically the same SQL syntax. SQL is a language, and there's many different implementations of it.

Steve: Right. So last week we talked about cross-site scripting problems, where because the user's browser was accepting scripting that appeared to be from a trusted server, there were ways to inject that scripting into the user's browser to cause, you know, all kinds of nefarious things to happen. Hackers had jumped onto this as a way of creating havoc and leveraging this insecurity for their own purposes.

Leo: This is the same idea, I guess, as anyplace where you have a field that you can overrun the field with too much text and execute arbitrary code, you're doing the same thing with this SQL query.

Steve: Well, actually it's very different than that. That, what you were talking about, was a buffer overrun. And last week with cross-site scripting the exploitation is essentially occurring on the user's end, on the user's browser. What's different about SQL injection is it's a means that allows hackers to inject queries into the remote server that the designers certainly never intended. And essentially it leverages this expressive power, this linguistic power of SQL, in ways that, just like with so many of these exploits, it's very simple to get the system working. It's just so hard to have it work in a way that can't be compromised. That is, just because of the nature of the power of SQL and the connection between the client's machine and the server creates these vulnerabilities.

For example, say that you go to a site, and it presents you with your username and password that you're supposed to fill in in order to get access to the site. Maybe it's a high-value site that regular normal authorized users might pay a lot of money to have access to all the time. And so what happens is, the web browser gives you a form where it says please input your username and your password, and you click okay. You know, we've all seen that kind of front page on a site that we have to authenticate ourselves to.

Well, behind the scenes there will be – this username and password will be accepted from the user over on the server. And then the code running on the server needs to look up this username and password to see if it's valid. So, for example, you might have a table or a database called "log-ins," which is the username and password of everyone who can log in validly. So it'll say, for example, in SQL, and this is sort of a simple construct like I mentioned before, it might say "select ID from log-ins," meaning the log-ins table, where the username equals, and then the username provided by the user, and password equals the password provided by the user, meaning that username and password would be columns or fields in this log-ins database. And so the idea is that the data provided by the user is plugged into this SQL query, which is then given to the database to see whether this username and password are

valid.

Well, okay. Get a load of this. So the phrase that we're looking at is where username equals the username provided and password equals the password provided. So obviously both of those have to match. Okay. It turns out that if, instead of just putting in a password in the password field, you put in, I mean, like anything, noodles, and then you said "or" in the same field, you said "noodles or one equals one," then you submit that. Okay? So now what happens is, instead of the password that the user provided, you've put in a bogus password, noodles, and then essentially you've extended the query by saying "or one equals one."

Leo: Which is true. Always.

Steve: Which is always true, exactly. So you said, so only allow a log-in where username equals the username provided and password equals noodles, which, you know, it probably won't, or one equals one, that is to say, essentially the user of this log-in page is able to write their own SQL into the query in order to make it work.

Leo: Wow, very interesting.

Steve: And it turns out in a surprising number of sites this works. Now, I have a link...

Leo: Try it on my site real quick.

Steve: I have links to some white papers that go into more detail if people are curious because there are little things you have to do. There's, for example, the user is not putting quotes around their username and password, so those will be provided when their username and password are patched into the SQL query. But it turns out that doesn't stop you. You can still do what you want to do, but you have to be cognizant of the way quotes are probably going to be futzed in there. So that's just a simple example of a way that, without the developers ever intending you to provide your own SQL, literally, over the Internet, using a web form, you're able to do something they didn't like. Well, it turns out...

Leo: That's amazing, just...

Steve: Isn't that cool?

Leo: Yeah. Well, I don't know if it's cool.

Steve: Cool in a bad way. So it turns out that SQL is so powerful, you can use parentheses to get nesting, you're able to query other aspects of the server that are not even relevant to the application. For example, the way SQL works is there are system-level tables, sort of so-called "metadata," where the system stores its own information. For example, there's a table called sysobjects, syscolumns, sysindexes, for example. Those are tables containing information about the actual database tables. Using SQL, you're able to query those metadata tables to get the names of the databases, the columns in the database, and other information.

And in fact, if you have a website which is not very carefully cleansing and scrutinizing

everything provided by the user, it's possible, for example, on a website that's going to give you like a list of all the threads in a forum, for example, you could take the capability of dumping out from the SQL database all of the threads in the forum. In the same fashion of essentially injecting your own SQL code through a form, you could, by a series of experiments on the server, even starting out knowing nothing about what the database tables are called, what the form columns are called, you're able to sort of, over time, extract that information in order to get what you need, and ultimately put into a form a complex query which instead – where the system thinks it's dumping out a scrolling list of threads, it instead dumps out every username and password in the username/password dictionary.

Leo: Wow, that's amazing.

Steve: Basically, I mean, just giving you all that data. Now, imagine if a banking site had this vulnerability, I mean this kind of vulnerability. It's one thing, it's like, okay, someone could play havoc in some online forum. But this is, I mean, this is how serious this kind of exploit is, is if your own online banking technology that offered online banking had this kind of a vulnerability, and somebody could sit there poking at the form, extracting information about the metadata, the table names and column names, and end up tricking their server into dumping the log-in username and password of all the bank's online customers, allowing them then to log in as you and do anything that you would normally be able to do on your banking site, like sending checks to them.

Leo: So how do you prevent this? Do you look at the input and, I mean, and say, oh, you can't pass SQL commands along, or...

Steve: Well, yes. And, I mean, essentially that's what you need to do. Now, remember how last week I read the 28 new cross-site scripting vulnerabilities that had been identified just in March. This same problem relative to SQL injection has 38 new SQL injection exploits just in March. And I won't...

Leo: You'd think they'd understand this, it's so well-known that they'd understand this, and it'd be just kind of standard operating procedure to prevent it.

Steve: I know. And the problem is it's sort of like, okay, whose fault is it? Well, we know that it's the webmaster's fault. It's the person who coded this and didn't pay attention to this. But again, I mean, it is just – it's phenomenal. I'll just read a couple of these to give people a sense for this:

"VBulletin Inlinemod.PHP SQL Injection. vBulletin is an application for web site forums. The application is exposed to a SQL injection issue because it fails to sufficiently sanitize user-supplied data to the "postids" parameter of the "inlinemod.php" script file before using it in an SQL query." I mean, it's exactly what we're talking about. And here's:

"Serendipity is a web log application. The application is

exposed to this issue due to a failure in the application to properly sanitize user-supplied input."

"Angel Learning Management Suite is a learning management

application implemented in ASP," active server pages. "The application is prone to an SQL injection because it fails to sufficiently sanitize."

Coppermine Photo Gallery, Zephyr Toolbox Address Book, GaziYapBoz Game Portal...

Leo: GaziYapBoz is vulnerable? No. No, say it's not so.

Steve: SQL-Ledger/LedgerSMB Remote Code Execution, Monitor-Line Links Management, LI-Guestbook Application, Rigter Portal System, multiple AJ software products, PhpStats, Creative Files, Absolute Image Gallery, Woltlab Burning Book, blah blah blah.

Leo: We get the message. There's a lot of them.

Steve: I mean, well, and that's one month, Leo. And they're like this every month. Now, what's really a concern is you might say, okay, well, so some guestbook application is vulnerable. It's like, okay, yes. The guestbook application is vulnerable. But more than likely this is a guestbook for a site that has a lot of other stuff going on. And what this does is, this vulnerability in the guestbook app – and I think I just read about nine different guestbook apps that are vulnerable. And it's because this is sort of a low-security thing...

Leo: And it's written by amateurs. I mean, these are, you know, most of what you just described, they're kind of free open source and kind of group projects, not even high-end open source, you know.

Steve: Exactly. The problem is the guestbook application makes use of the SQL Server that runs the entire website.

Leo: Right, right. Most websites, mine included, only have one SQL program running, MySQL running. And of course if it was different databases with different users and log-ins. But if you have the same user and log-in, as many of us do...

Steve: No no no, it's worse than that, Leo. It's worse than that. A single insecure application, like your guestbook application, can use access to the table metadata to get the names of every database on your SQL Server...

Leo: That's kind of bad.

Steve: ...and then explore – yes. So it ends up being – and this is really why SQL injection is such a problem is that, exactly as you said in your case, you've got MySQL running, one instance of the server, that runs your whole site. So the guestbook application that also is using a single little guestbook table in SQL Server, that provides an entry point to allow a hacker to go and explore all the tables, all the databases that your site is using, and get, for example, much more valuable username and password data, or anything that you're keeping in that SQL database. And in fact, due to some notorious problems with SQL, it's even possible to leverage SQL vulnerabilities to allow a hacker to get access to the underlying operating system.

So, I mean, it is a very, very powerful backdoor into, well, into web servers all over the place. And the problem is that, exactly as you said, Leo, guestbook apps are written by people who this is like their first web application. It's freeware. It's downloadable. And exactly as we talked about last week, oftentimes it's not the kind of thing that you update. I mean, on all these

things I was talking about, it's like, for example, "Woltlab Burning Board is a free web-based bulletin board package based on PHP and MySQL. The application is exposed to an SQL injection issue because it fails to properly sanitize user-supplied input to the 'action' parameter of the 'usergroup.php' script. Woltlab Burning Board version 2.7 and earlier versions are affected."

Okay, so clearly they fixed this after version 2.7. But how many people who are using this thing know that there's a problem? Now all the hackers in the world who didn't know before, know exactly what the problem is with Woltlab Burning Board versions, you know, earlier than and including 2.7. So now they go around to interesting websites, and maybe they even know because this is what they do, which types of bulletin boards different sites use. Now any site that didn't update its bulletin board system, which tends to be something sort of thrown in to give your users someplace to chat among themselves, suddenly now that site's entire SQL database repository is vulnerable because there's a backdoor in through SQL.

Leo: So that gives them access to anything that's running on that database server.

Steve: Exactly. Any databases that are being managed by that instance of SQL on the web server.

Leo: You know what's really scary is that many, many people are in a shared hosting situation where there's one instance of MySQL running for many, many, many accounts. So it's not just your database. You may be exposing everybody else on the server. Which means hundreds or thousands of different websites. Wow, that's interesting.

Steve: It is. And in fact oftentimes, especially with Web 2.0 situations, the SQL Server has basically the page content, that is, all of your site is being served out of an SQL database. And so this is the way all of this website defacing that we hear about comes about.

Leo: Oh, that's why, yeah.

Steve: Exactly. People are able to go in, and you're able to delete tables. So through SQL it's called "drop table." And baby does it. It just drops the table.

Leo: It's fast, and it's easy. Yes, I know, I've used it many times. And I'm always scared when I press that button.

Steve: Yes, but imagine that somebody determines the table name, wishes you no good, and then they insert a drop table command through a web form. So to answer your question, coming back to how do you prevent this, the way to prevent it obviously is to absolutely scrutinize anything the user provides. And of course this is a problem, too, because you'd like a password to be freeform. Let them put in any kind of junk that they want to into the password field as long as they are able to type it in again. So but of course here's the danger in doing that is if they put in an SQL command extension, it's like, well, that's a valid password. I mean, no one's going to guess it. It's like an unguessable password. So you've got to really – you've got to be very clever about how to allow things that are supposed to be anything the user wants to write, how does the system know that that's also a valid SQL sequence?

So you see the problem. I mean, it's not that simple a problem to nip in the bud. And the same is true, for example, in a forum where you're inherently wanting to let people put anything in

that they want to. It's also sort of a bigger problem than scripting injection because you could argue that, as long as you don't allow the open angle bracket script/close angle bracket to be present, then you're going to be okay. Well, of course people have cleverly found ways around that by using various other means of evaluating expressions to that string. But with SQL it's much more like English language, and there isn't something as obvious as the script keyword to hang onto. So, I mean, it really is a problem.

Leo: Fascinating.

Steve: It requires extreme sanitization of anything coming from the user. And believe me, here was 38 examples from March. There's a similar number in February and a similar number in January, and it goes back through time. Because, again, we've got people who are not security experts who are using very powerful tools like SQL behind them and web servers to put together an application. The moment they get it working so that it doesn't crash when users submit data, they think they're done. It's like, yeah, I got this working. You know?

Leo: I'm done, goodbye.

Steve: I'm done. Pay me.

Leo: I'm going to go have a hamburger.

Steve: Yeah, exactly, pay me, and I'm on to my next disaster.

Leo: Yeah, wow.

Steve: So it is a very serious, pervasive problem. And it's going to take time for awareness of this to filter out into the web developer community because that's really the only way you're going to handle this. I mean, these are valid queries to issue to the SQL Server. So it's not possible, for example, for SQL Server or MySQL to say, wait a minute, this looks like it's a suspicious request. There isn't anything suspicious about these requests. They're valid requests. The problem is that people you don't trust were allowed to make them of your server by leveraging this whole concept of anonymous user-accepted input. And this creates a serious problem, one we've never talked about before, but I'm glad we have now.

Leo: It's fascinating, you know, I've heard about SQL injection. And by the way, I'm always looking for updates on the various programs we use on the site like the Drupal program, and there have been SQL injection exploits in some of these programs. I don't know about Drupal particularly. But I always look for updates, and I always make sure I apply the updates. And if you're a webmaster, that's just one more thing you have to do. You have to keep your eye peeled. Even if it's just a stupid guestbook.

Steve: In fact, next week we're going to read a Q&A note from somebody whose ears perked up when I read through that list, and he realized he was using one of the vulnerable things, and it had been exploited to attack his server. So this is, I mean, this is real stuff. This is not science fiction that we're talking about here.

Leo: Fascinating. Steve, thank you. Another great and fascinating subject explaining something many people have heard many times, and now you know exactly what it all means.

Steve: How it happens.

Leo: Yeah. Security Now!, of course, is brought to you also by SpinRite. That's Steve's day job. So I say it's brought to you by SpinRite because if Steve didn't have something to do, he wouldn't have time for this, that's for sure.

Steve: That's true.

Leo: You can find out more at SpinRite.info or on Steve's site, GRC.com. And we also want to mention that GRC.com has the 16KB versions of this show, including, what was it, 84a, 85a? I can't even remember.

Steve: 85a will be there by the time our voices are being heard, Leo.

Leo: You know, it's funny, because I made a 16KB version of it. But I completely understand why you didn't post it because it wasn't something you would want to save. It was very specific to a point in time. It was a news bulletin, really. But people want completeness.

Steve: Lesson learned. I will have everything posted as of this podcast.

Leo: You've got to remember, nerds like complete sets. Complete sets. You know that. If you're collecting comic books or Security Now!, you've got to have all 87. 16KB versions, transcripts thanks to Elaine, notes, too, at GRC.com. That's also where you'll find, besides SpinRite, Steve's free security programs. SecurAble is up there to test the security of your system. And of course ShieldsUP!, the world's most popular firewall and router test system, absolutely free, GRC.com. Next week questions and answers.

Steve: Episode 88, we're divisible by four once again.

Leo: It's good. I like that. Always enjoy those.