The Evercookie

Description: After reviewing the past week's security updates and news, Steve and Leo examine Samy Kamkar's (http://samy.pl/evercookie/) clever suite of JavaScript Hacks, collectively used to create an "Evercookie" for tagging web browsers in a fashion that's extremely difficult to shake off.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-270.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-270-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 270, recorded October 13, 2010: The Evercookie.

It's time for Security Now!, the show that covers and protects your security online, and privacy, too. And here he is, the man of the hour, the man who has done more, I think, to protect our security than almost anybody else, Mr. Steve Gibson of GRC.com, creator of ShieldsUP!, SpinRite - the world's best hard drive maintenance utility - and a great many security utilities, including the very first antispyware. Hey, Steve.

Steve Gibson: Hey, Leo. Great to be with you again, as always.

Leo: Nice to see you.

Steve: Yeah.

Leo: Today we're going to cover something that a lot of people have been asking about.

Steve: Yes. In fact, it was - I almost sort of sidestepped it because I thought, well, we talked about the EFF effort with Panopticlick and the idea that just sort of passively our web surfing offers enough hints about our identity to be trackable, which is what our Panopticlick episode of Security Now! [SN-264], I don't know, a few weeks ago, a month ago or so, covered. But the Evercookie is different enough that - and so many people were writing in about it. And then when I saw, a friend of mine sent a link that The New

York Times had picked up on it, I thought, okay, I mean, this is what the podcast is for is to talk about these kinds of things.

And when I looked at it closely, I was very, frankly, very impressed because there are some very clever things that the developer, Samy Kamkar is the guy who did it, at samy.pl - S-a-m-y, one "m," S-a-m-y dot pl. And his page is samy.pl/evercookie. Basically he's come up with a cookie that is frighteningly sticky, frighteningly difficult to get rid of. And some of the things are generic. Some are super clever. So definitely a great topic for the podcast, and that's what we'll be talking about this week.

Leo: The evercookie. I wish there were - it sounds like something that Willy Wonka might invent, a cookie that you can never finish, but in fact it's not so nice. Before we get to evercookies, I imagine you have some updates and so forth.

Steve: Yeah, some interesting news. First of all, we're just past the second Tuesday of October, which of course is everyone's favorite Patch Tuesday. Microsoft broke their own record, which they had previously set. I guess they would always be previously setting their own record. Anyway...

Leo: [Laughing] No one else is setting it, let's put it that way.

Steve: However, what's interesting is it was last October, last October of '09 was the previous record that Microsoft set for their own number of vulnerabilities fixed. They have exceeded that exactly one year later, October of 2010, with 49 security vulnerabilities fixed, in a set of 16 bundles, 35 of which are remote code execution vulnerabilities. So there's various types. Obviously remote code execution is bad guy gets to run his own code in your machine without your knowledge or permission. Then there's privilege escalation and information leakage and different sort of classes of problems.

This one, 35 of these 49 were things that, when exploited, would allow someone to run their code in your machine, which is never a good thing. They're not helping you. They're not fixing, they're not upgrading apps that you forgot to. So many of them were publicly disclosed before the patch, so everyone is breathing a sigh of relief that Microsoft has caught up. Several of them were zero-day surprises to Microsoft, where Microsoft first learned of them by them being found in the wild, being used. There were two zero-day vulnerabilities which the Stuxnet worm/the zombie trojan, whatever, that we've talked about extensively, was using. They fixed one of those two, so that's good.

IE, both versions 6 through 8, all of those versions, got 10 security holes fixed. And interestingly, several of them were critical, even under Windows 7 and IE8, which generally has been more resistant to these problems. So, in fact, it is more resistant to these problems, but several of them got by anyway. And then one thing that will affect users aside from all this is that, as part of this bundle of updates, Microsoft updated their MSRT tool. And they've finally updated it to detect the Zeus Trojan, which is that online banking, capture your credentials trojan that's been causing so much havoc.

And so I wanted to remind people that normally the software removal tool, the MSRT, is only run monthly by Microsoft. That is, at some point after it gets updated they'll run it. But anyone can run it on demand, and it just sort of feels good to do it. It's not MSRT, it's just MRT. So if you just go, like under the Start Menu, and choose the Run... option, you'll get a little field there saying, what do you want me to run? And just put MRT in,

with no extensions or anything else, and hit Enter, and that'll fire up the software removal tool, which will then offer to scan your machine right there, sort of while you're waiting. And you can ask it to do a deeper scan than it otherwise normally does. And it just might be a good thing to do, after they've made a major update to it, as they have this month.

Leo: Yeah, it's important to note that they don't do a full, thorough scan every Tuesday, every Patch Tuesday. They only do kind of a quick one.

Steve: Right.

Leo: So it's a good idea once in a while to do a thorough one. I presume that gives you a different result.

Steve: Well, it potentially finds things that the overview scan won't find. So definitely worth doing.

Leo: Right.

Steve: We also had a major Java technology update. In fact, Oracle, at the beginning of this month, I think it might have been Tuesday or Monday, had a major, corporate-wide, huge update of their software across the board. Most of us aren't going to be affected by that, except for those of us who have Java installed. They're now at Version 6, update 22, which fixed 29 different security holes, so it was a big one. And among other things, they did fix their TLS/SSL renegotiation hole, which their own implementation of the TLS technology, the TLS protocol had not yet fixed. So they fixed that. They added some new root CAs to their own private store of root certificate authorities, and otherwise just fixed a bunch of stuff. So I did notice, on my main machine somehow I've managed to survive so far without anything installing Java, which I'm pleased about.

Leo: You know, it used to be it came with Java. And even if it didn't, you'd almost always have Java. And now that's really changed. Java is not everywhere anymore.

Steve: It's not. And I'm seeing people saying that it's becoming an increasing problem from a security standpoint. So, I mean, I'm glad it's not on my main machine. And there have been a number of people who, following other security experts' advice, have removed it from their system, and nothing's broken. It's like maybe something that they once installed brought it along, and then they removed it, or they stopped using it, and it sort of stayed behind, which it wants to do. It wants to be an extension to your OS, to provide its own virtual execution environment, the whole JVM, the Java Virtual Machine, which is able to run in browsers and also execute apps natively on your platform. But it's one of those things where, if you don't need it, you're better off without it. In any event, it's updated.

And then I did want to mention that Foxit Reader is now at version 4.2. Many people have switched away from the Adobe PDF Reader over to Foxit, as just something much less lightweight. Boy, I noticed the other day how big Adobe Reader has become. It's

huge now. I mean...

Leo: Yeah. I use Foxit. I really like it.

Steve: Tens of megabytes of bloat from Adobe. And they had like a classic buffer overflow problem where PDF files that contained a title longer than 512 characters, and we of course know what that number is, that's 2^9, so yeah, some programmer said, oh, 512 characters, no one's going to have a PDF title longer than that. And so whoever that was statically allocated a buffer to contain the PDF title and apparently didn't check to see whether the title might actually be longer than that. Turns out that it will crash the reader if you load a PDF with a title larger than 512 characters, which potentially opens a door to a buffer overrun exploitation. So that's been fixed at v4.2 and hopefully from there on.

Leo: Excellent.

Steve: In news, one thing caught my eye that I just thought I had to bring up because it was like, uh-oh. And that was our well-known major content distribution network, Akamai, had one of their employees charged with wire fraud...

Leo: Uh-oh.

Steve: ...because - his name is Elliot Doxer. His name's been in the news. A few years ago, actually in '06 when this began he was 42. He approached a U.S. consulate of an unnamed foreign government, offering them insider information. Now, the good news is the consulate immediately contacted our FBI and said, uh, this guy is offering us information from his employer, Akamai. And the FBI set up a classic sting operation. Apparently an agent, about a year after that first contact, got back in touch with Elliot and said, hey, I'm with the whatever-it-is consulate, and we're interested. And so they recorded videos of him doing dead drops at a location which the FBI had set up and ended up finally arresting this guy.

The thing that brought it to my attention was that it is the case that any cloud-based services are, like Akamai living up in the Internet, being a content-distribution network, are vulnerable to insiders. And so the thing to just remain conscious of is that you really do want a technology which encrypts anything you're storing out in the cloud, always, before it leaves your machine. All the cloud is doing, if possible, is storing a bunch of pseudorandom data, and not having access to what that content is.

Now, that's really not possible unless the application that's running is designed correctly. And for purely web-based applications, it's arguably not possible at all. I mean, for example, the Google stuff is understanding the contents of what you're saving, and so we're trusting them with that. I'm now assembling the outline for our Security Now! podcast every week in Google Docs and then exporting them as a PDF which I share with you. But I'm trusting Google with this content.

Now, I'm only trusting them, frankly, because I don't care. It's public anyway. It's going to be public. We're showing it, I mean, I'm reading it now. You're going to be posting it on various feeds. And I would be reluctant, frankly, to do sensitive corporate, really

sensitive corporate documents this way at this point. I don't know how you do that safely. But it is the case that insiders have access. And security people have been saying for a long time, I mean, like, well, like the FBI, when they're looking at things they're seeing that people inside companies represent a bigger threat to companies, arguably, than outside attackers.

Outside attackers have the advantage of being anywhere in the world, operating over the Internet, being anonymous, being able to attack with impunity in the middle of the night. Inside employees have the advantage of access. I mean, there may be a door that's left ajar, or they've turned somebody's keyboard upside down to get the password and then logged in and so forth. So it is the case that all of these dangers are not from the outside. I think on this podcast...

[Interruption]

Leo: Just ignore that [laughing]. I didn't mean to put that in your feed. Facebook is streaming something live from Microsoft, and I just wanted to check in to see if it was anything that we cared about. I don't think it is.

[Interruption]

Leo: Yawn. Okay, we're done. Moving right along. Continue, Steve. Where were you?

Steve: So anyway, I think on this podcast we tend, I tend to focus more on the external threats and the technology because it interests me, and what are you going to do about the internal threats? Although for companies, certainly I think it's very necessary for them to remain diligent and do what they can. I mean, you want to trust the people that work for you. The problem is you're extending your trust to all your customers when you're someone who's offering cloud-based services; and somebody on the inside who has access can do tremendous damage to major proprietary interests that are moving their data out to the cloud. So that just - it's something we're going to be talking about in the future because I'm sure there'll be more problems with it in the future.

And something that is sort of cloud-related, also, I noted that Charles Schumer, our senator from New York, has introduced a bill into Congress where it wants to extend existing electronic funds transfer consumer protections, which cap the liability for electronic funds transfer fraud at $50.

Leo: Our liability, not the banks' liability.

Steve: Correct, caps our liability, consumers are capped at $50. Chuck's bill would extend that to municipalities and schools.

Leo: Oh, because lately that's where that spear phishing attack has been against municipalities, hasn't it.

Steve: Well, yes. In fact, I pulled some numbers together here for this. Earlier this year, for example, $378,000 was transferred from a town, not probably coincidentally, in Poughkeepsie, New York, which is Chuck's home state. And that $378,000 went to Ukraine. Also $450,000 was stolen from Carson City, California; $600,000 from Brigantine, New Jersey. Not far away, $100,000 from the Egg Harbor Township, also in New Jersey. $3.8 million from Duanesburg Central School District, also in New York, Chuck's state. And a chunk of that they were able to get back. They were able to get $3.3 million of it, but still lost half a million, which was never recovered.

So in most of these cases it appears that the banking credentials were obtained by what we were talking about before, the infamous Zeus online banking credential-stealing trojan. Now, the question is, can banks absorb that kind of loss? I mean, what Chuck's bill is proposing to do is to limit much larger entities than individual consumers, that is to say, cities, towns, and school districts who have been having their money transferred off and out of their accounts due to electronic funds transfer fraud, limiting their liability the same as consumers.

And so of course the banking industry lobbying groups are saying, whoa, whoa, whoa, wait, whoa, wait a minute, how can you make us responsible for these things? From their perspective, they're seeing people logging in, presenting valid credentials, and asking to transfer money. And presumably these entities are routinely transferring money in and out of their accounts, and so this just looks like another valid transfer. So it'll be interesting.

It's not clear that this is going to get through our Congress in this session. It may well be not till next year, in which case Chuck's going to have to reintroduce it. But it does remind me of the advice I've given to our listeners, which is - and I know that you heard this before, Leo, and may have taken action, and I have on my own corporate accounts. It's possible to explicitly disable those features from accounts typically with your bank. You say, we do not use, do not need, and do not want electronic funds transfer to and from specific accounts. So disable them.

And it's the kind of thing where, hey, they're on by default, typically. And unless you tell your bank to turn them off, they haven't. And so it just makes sense, from just a pure security standpoint, disable those things that you don't actively need. And you may be glad you did. Because, I mean, it would be nice - oh, I should also mention that this does not extend to small, medium, or large-sized companies. So Chuck's bill is extending this to municipalities and schools, but not to businesses. So businesses will still be vulnerable. And I did want to credit Brian Krebs, who used to be with the Washington Post, now is blogging on his own. He did a lot of reporting on this. And so that was useful when I was pursuing these details.

Leo: I was an avid reader of his Security Fix column in the WaPo.

Steve: Oh, and we talked about it often. I mean, I was often saying, hey, here's something that Brian brought up.

Leo: But I'm glad that he's still blogging and still talking about it. That's great.

Steve: Yup, he'd doing a great job. Facebook has now added one-time passwords.

Leo: Yay.

Steve: Yes. In a very recent, last couple days, blog post, they announced one-time password support, which would be very useful in situations, for example, where you want to log into your Facebook account, somewhere where you don't have 24/7 control of the security of the machine. The way it works is, you need to register your cell phone number with your Facebook account. So add your cell phone phone number to your Facebook account. And when you are somewhere that you want to log in using a one-time password, you text the string "otp" to the number "32665," and you will immediately receive a password that can be used only once, to log into your account, and which in any event expires 20 minutes after that. So they've said they'll be rolling it out gradually, meaning it may not be available immediately upon your hearing this on the podcast. But they said it would be widely available within the next several weeks.

So that's a nice step forward. I'm glad they're doing that. And I also saw in the same posting, although I feel like this is something we've talked about, or maybe you talked about on another podcast, Leo, that they're adding the ability - maybe it was Google, and they're following Google - to log you out of other sessions which you may have left logged in, in other locations. So, for example, if you were over at a friend's house and logged into your Facebook account using their system, if you went back home and then thought, ooh, shoot, I forgot to log myself out over there, you're able to log yourself in and, using your account settings page, you can see any other places that you're currently logged in and then explicitly log those out there. So that's additional good security. I'm glad to see them pursuing those things.

Leo: They've done a number of things. I know I get an email now every time I attach another device to Facebook for sending information back and forth using OAuth. I think Facebook's paying a lot of attention to this. I might mention that there is a press conference going on right now, I've been watching it, at Microsoft headquarters. Facebook is streaming it on their Facebook live streaming page. And it looks like Facebook and Microsoft Bing are going to do a deal, and there's some speculation that perhaps Facebook will start using Bing as part of its Facebook Connect. But I do think that it's really important that Facebook pay attention to security since they are in fact now the interface for people to the web in many, many, many cases.

Steve: That's why I felt it was worth bringing it up here. I mean, I know you and I have been rough on Facebook from a privacy rights standpoint and enforcement standpoint.

Leo: Well, we'll continue to be that way.

Steve: Yes. But they just, I mean, you can't get away from them.

Leo: Right. And I will praise them when they do the right thing, which they are doing here, yeah.

Steve: Yeah. And then, in a little bit of a mystery, the UAE and BlackBerry, that is to say

RIM, the makers of BlackBerry, have reached some sort of agreement. Their disconnection of BlackBerry services that had been threatened and widely reported has been canceled. And everybody's happy, and no one is saying why or how.

Leo: That scares me more than the original deal.

Steve: And Saudi Arabia and India have also both backed down.

Leo: Really. Hmmm.

Steve: Yeah. So the problem is, as I understand it - and maybe we'll do a podcast on this. I pulled up all the technical documentation. I have the PDFs which explain how the RIM technology works to be secure, so that the data is available for understanding this. And if RIM is to be believed, and certainly I believe them, they have been saying all along that they are unable, they are technologically unable to provide the kind of access that these countries want because the way they originally designed the system didn't allow any sort of man-in-the-middle eavesdropping, which is what the UAE, the Emirates have said that they require. So, I mean, I would like to believe that the RIM execs sat down and explained this and said, look, we can't do anything about it. You either allow it or not. But here's the technology. Or maybe there is some way of establishing a man-in-the-middle server. It's not clear. But when deliberately asked, point blank, RIM has said we will not discuss what was decided. It's proprietary.

Leo: I can only think that's bad for everybody.

Steve: I know.

Leo: I'm sorry.

Steve: I know.

Leo: I'm just a - maybe I'm paranoid or a pessimist.

Steve: The good news is there are a lot of people who have time on their hands, apparently, to plow into this. I will keep my ears peeled for any news of what's going on. I imagine that people who look closely at the BlackBerry technology would be able to detect whether, for example, certificates have changed, or certificate authorities have changed, or what was going on. So I don't know. But I wanted to bring that to people's attention.

And in my errata, we haven't had any errata for quite a while, but I got a kick out of the fact that the news that jailbroken Kindles - it's now possible to jailbreak a Kindle - are able to run the original Zork.

Leo: I love it.

Steve: From Infocom.

Leo: That's a text adventure.

Steve: Exactly.

Leo: That we nerds love. That's great.

Steve: Exactly. And it's perfect for a Kindle because it needs something that's textual and sort of slow, where you're typing instructions in, and then it's telling you what you're seeing and what's going on, and you're able to explore your environment. I didn't realize that there were some apps. Amazon apparently has a couple free applications which they've developed and made available. And then Electronic Arts has released Scrabble for $5 for the Kindle.

Leo: Oh, that's neat.

Steve: Yeah. Now, the problem, of course, is the Kindle is very UI challenged. And apparently the Scrabble, EA's Scrabble game is pretty good except you can't get to it because of the rather sad user interface that the Kindle has. I mean, it really needs a touch screen if you're going to do things that are very fancy. And frankly, typing on the keyboard is a painful process, too. It's not something that you want to do that much. So the Kindle's got up, down, left, right navigation and a couple buttons, but really not much more than that. So I don't think we're going to see a big active gaming environment for the Kindle. And when Amazon in fact announced that they'd created an SDK and opened it up for people, it's like, oh, okay, what are they going to do with this? But, you know, maybe we'll see something clever. It would be sort of interesting to me. I just think it's a perfect eBook reader, so I'm happy.

And then one other little bit of news in errata is that an analysis of where people use their iPads have found that they spend one fifth of their time in bed.

Leo: Hmm. That's about right.

Steve: Yeah.

Leo: Don't we spend a fifth of our life in bed? More than that. A third.

Steve: So apparently 20 percent of people's use of their iPads is they tuck themselves under the covers, and then they turn it on, and they read or they surf or they check their email or whatever it is they do. Who knows? But I thought that was sort of interesting

and appropriate, yeah.

And then always on the outlook for some sort of a new testimonial for my own company, GRC and SpinRite, I have something different that I've never mentioned before. And in fact the subject was "GRC's Tech Support: A Different Slant on a SpinRite Success Story." From Melbourne, Australia, Russell Phillips wrote: "Hi, Steve. I would like to say thank you for providing such a great product in SpinRite. But thanks especially to Greg and the rest of your tech support team." Well, Greg's probably looking around, saying, "What team? It's just me."

Leo: Well, that's a team. A team of one.

Steve: Which is true, a team of one. Russell says, "I purchased SpinRite a few weeks ago" - oh, sorry, a few years ago. "I purchased SpinRite a few years ago after listening to you and Leo on Security Now! (love that show!!) and have used it on all my PCs and my kids' PCs and have never had a hard disk problem." Okay, well, he's the poster boy for this is the best thing you could possibly do is believe me when I say, if you use SpinRite, your hard drives won't die, or they won't have these problems, which is exactly what has been his experience.

So he said, "What prompted this email was that I purchased two new PCs this year, a Dell desktop and a Dell Netbook, and attempted to use SpinRite on them as soon as I received them. Unfortunately, I could not get them to boot into SpinRite, whatever I tried. I purchased the desktop first, and after a number of attempts to run SpinRite I contacted Greg in your 'tech support department.'" We'll put that in quotes because that's Greg at home. "After a couple of emails back and forth, and following his advice, I had SpinRite running on the desktop without a hitch. Apparently the problem was due to the configuration of the hard drive settings in the PC's BIOS, which Greg knew all about and was able to help me fix.

"Fast-forward a couple of months, and I purchased a Netbook. Again, I could not get SpinRite to boot from a USB drive since this Netbook has no CD. So I contacted Greg again. He responded immediately, and again his advice was spot-on. And I am now sitting here with SpinRite running on the Netbook without a hitch. I just needed to configure a bootable USB drive correctly.

"So again, a big thanks to you, not only for producing SpinRite, but also for having such a helpful and prompt tech support 'team' standing behind you. Keep up the good work at GRC and also with Leo. I look forward to each episode of Security Now!, and I'm always learning something new. Thanks again from a fan in Australia. Sincerely, Russell Phillips."

Leo: How sweet.

Steve: Very neat. Thank you very much for sharing that, Russell.

What's cool about the evercookie is that its designer, Samy Kamkar, has produced an open source, freely downloadable, available API which exploits in some clever ways essentially what can be done with scripting. So The New York Times picked up on it, as I mentioned, on the binary day, 10-10-10, had a story titled "New Web Code Draws Concern Over Privacy Risks." And they focused on some features primarily of HTML5, which actually does have a number of features which are of a concern from a privacy

standpoint.

But what Samy did is he developed a suite of 10 different ways - many of them clever, that we're going to talk about in detail - of inducing our computers to accept, store, and return an immutable token. So we know what that means. That means a means of tracking us, a means of following us as we move around the Internet, which is traditionally what standard HTTP browser cookies have done. Many people understand the concerns about browser cookies. They've got third-party cookies turned off. They flush their cookies when their browser stops. I mean, the original HTTP cookies have been around for so long that all kinds of tools have been developed to aid people who are annoyed by that behavior to get some control.

What Samy did was look at beyond what the Panopticlick guys at the Electronic Frontier Foundation did. He said, okay, what's possible to do using scripting and all the features of a contemporary browser? So in his own FAQ, his Frequently Asked Questions on his page, he asks the question of himself, what if the user deletes their own cookies? And Samy replies, that's the great thing about evercookie. With all the methods available, currently 10, it only takes one of those cookies to remain for most, if not all, of the rest to be reset again. For example, if the user deletes their standard HTTP cookies, their LSO - that's the Locally Shared Objects data that Flash uses - and, for example, all HTML5 storage, the PNG cookie and history cookies will still exist. Once either of those is discovered, all of the others will come back again.

So the first thing I want to make sure people get is that he's storing a single token in - he's basically squirreling it away in many different places, every place he can think of. And then when you revisit a page that is using this evercookie technology, which is now freely available, the script on the page will look in each of those different little squirrel holes to see whether that immutable token has survived, that cookie, essentially. And, if so, it says, ah, good, here's an instance of it. I only needed to find one. And then, if any of the nine others had been deleted, it refreshes them, that is, it reestablishes them so that this thing basically holds on really hard.

So let's look at where he's storing this stuff because this is where the fun and the cleverness is. First of all, he does use traditional HTTP cookies, so standard web browser cookies he will take advantage of. And as we know, unfortunately, even today, third-party cookies are enabled by default, and first-party cookies are. So in general your browser will accept a cookie from a page you visit. With this evercookie code, what that means is that, upon that happening, immediately that cookie value is spread throughout your system using scripting on the page, squirreling it away in case you should deliberately flush your cookies, disable the cookies, delete that cookie, it doesn't matter. It's already gone many other places within your browser.

It of course uses Flash cookies, the so-called LSO, the Local Shared Objects, which is technically configurable using the Macromedia domain. You're able supposedly to turn that off. And we were talking just recently on the podcast, a week or two ago, that the UI seemed a little tricky for me because I thought I had turned it off, and I looked at the other tabs in the UI, and then I went back, and it hadn't taken, the offness. However, a day later I looked, and it had stayed off. So maybe it just needed to be told twice. Who knows.

The one thing that is interesting about Flash cookies that's worth noting is it bridges browsers. Since you've got a single instance of Flash installed in your system, which surfaces on different browsers - for example, IE has Flash, Firefox has Flash, Opera has Flash, and of course Safari - the idea is that that represents a single point of contact. So if the evercookie had stored itself, among everywhere else, in Flash, and you brought up

a different browser and went to the same site, then on that domain multiple browsers are sharing the same instance of the Local Shared Objects in Flash, which means that the evercookie would be able to jump into a different browser. So that's worth noting, as well.

Silverlight, which of course is Microsoft's next-generation, essentially sort of competitor to Flash, Silverlight offers something called "isolated storage," which is essentially local shared objects from Microsoft. And on Microsoft's page they say, "In Silverlight, there is no direct access to the operating system's file system, except through the Open File dialogue box. However, developers can use isolated storage" - that's their name for it - to store data locally on the user's computer. There are two ways to use isolated storage. The first way is to save or retrieve data as key/value pairs by using the IsolatedStorageSettings class. The second way is to save or retrieve entire files by using the IsolatedStorageFile class." So here we have...

Leo: Is that kind of a sandbox, the isolated storage?

Steve: Well, it's actually - it's Microsoft's solution for wanting some means for allowing their Silverlight technology...

Leo: [Indiscernible] storage.

Steve: Yes, to store - to be persistent. And the bad news is, it's tracking. I mean, it's absolutely tracking technology. And I haven't discovered any sort of a UI that Silverlight has. I don't know if there is one squirreled away somewhere. But I haven't found it. And so, I mean, it would be nice if there was some way of disabling that, or examining it, browsing it, looking at it, filtering it, controlling it somehow. Otherwise we don't even have as much control as the little control that we have with Flash cookies.

Leo: Well, those two are the obvious methods.

Steve: Okay.

Leo: Now it gets sneaky.

Steve: I love this one. So Apple first introduced something in WebKit which they called the HTML Canvas, which was used for dashboard widgets and also in Safari. An HTML Canvas is a scriptable rectangular area of space on your browser page where JavaScript is able to draw. Now, we've had scalable vector graphics, SVG, for a while. And that's cool, that's vector based, sort of like Adobe Illustrator or Corel Draw. Instead of being pixels, it's lines and arcs and curves and so forth. And what's cool about that is, as the name implies, it's physically scalable. You're able to stretch it out if your screen is larger, or it's able to squeeze itself down by doing everything with vectors.

Apple wanted to essentially have the same sort of power, but with bitmaps, with regular pixel images. So they introduced it. Then it was adopted by the Gecko browsers, so Firefox has it, as do Opera and Chrome. It's then moved into the HTML5 standard. IE9

doesn't quite have it. It's not clear whether it's going to get it or not. But I wouldn't be surprised because Microsoft's making a lot of noise about IE9 being so standards-compliant and passing all the various torture tests and so forth.

Leo: And it's HTML5. Everybody's - right? Canvas, it's the same one as HTML5 Canvas; right?

Steve: Yes, yes. Okay.

Leo: If you going to support HTML5, you've got to do it. You don't have a choice.

Steve: So here's the idea. A web page using this technology has an image on it. And it asks the server, hey, what's the value? It makes a standard request for the image. The server that has obtained or knows what the cookie is for this session, as it would, designs an image where the value of the cookie is stored in the RGB, the red-green-blue pixel values of the image, which it returns to the browser with a 20-year expiration. So this PNG image, for all intents and purposes, doesn't expire. It turns out that scripting is able to load a browser image into the canvas, this HTML Canvas. And the HTML Canvas API is a full pixel-drawing API that allows you not only to set pixel values, but to read them. So this is a means of storing a cookie in an image which the script has access to. And by reading literally the color values...

Leo: Oh, my god.

Steve: [Laughing] By reading the color values in the image, it's able to extract the cookie's value.

Leo: You mean they store it in the RGB.

Steve: Yes. Yes.

Leo: That's steganography.

Steve: Yes, it is, exactly.

Leo: Wow.

Steve: Exactly. So you could, again, as Samy indicated earlier in his FAQ, you could flush everything else. But if you forget one thing, if you didn't also flush your image cache for that domain, living in the image cache is a tiny PNG with the cookie stored in it. And when you come back to the page, script will load that into the HTML Canvas and take advantage of this pixel-level drawing API to obtain the RGB color values. And those are the value of the cookie, which then recreates all the other ones that you did delete.

Okay, now that was good. This next one is beyond good. And this is the one where I said, okay, this guy gets an award.

Leo: It's pretty - actually, I can see why he published this because it's like, look what I did. I'm not happy about it, however.

Steve: Now, back in '06, Jeremiah Grossman, and we talked about this then, four years ago, came up with a very cool hack where he could tell you what sites you had visited because CSS has this notion of visited links. And as anyone using a web browser knows, oftentimes you'll look at a page, and some of the links will be colored differently. And you look at it, and it's like, oh, yeah, I've already been there. So it's a nice visual cue to allow you to see what links, what URLs you have already gone to. So CSS colors them differently.

JavaScript, being very powerful and being a full language, the JavaScript language allows the programmer to query the color of a URL, thus telling you whether you have - telling the script whether you have been, sort of by implication, to that URL or not. JavaScript doesn't even have an explicit way of querying that, so this is a hack. This is like information leakage that was never intended as part of JavaScript. But by saying what color is this href on the page, the script can infer.

So here's what Samy figured out. He says, okay. How can I store a cookie with that information? And he figured out how. Once he has the cookie value that he wants to store, he converts it, he uses base 64 to convert it just to ASCII, so that the cookie could be like a URL. Just upper and lowercase A-Z, 0-9, and a couple other characters gives you 64 different - an alphabet of 64 characters. He has his script manually access a URL. And for the sake of example, we'll say google.com/evercookie/cache. And then say that, for example, after he's converted the cookie into ASCII, say that the first character, and I'm using his example because it's simple, is "b." So he attempts to access, with his script, Google.com/evercookie/cache/b. Then he accesses the same thing /bc, if the second character of the converted cookie was "c." Then he accesses /bcd. Then he accesses /bcde, and finally /bcde-, that being the cue that that's the end of the cookie.

Now, so what he's done is he has set history, he's said, I have visited those four URLs ending in /b, bc, bcd, and bcde. And then - I'm sorry, five URLs, and bcde-, those five URLs. He's set the history memory in the browser so that, if it ever encounters those URLs again, CSS will color them differently. And this can all be done, you're not seeing any of this on the page, this is done sort of at the script level. So now here's what he does. When you next come to the page, he successively tries /a, okay, wrong color. That is, his script builds a URL, then checks the color of it.

Leo: All behind the scenes, none of it visible.

Steve: Right, none of this is visible. His script builds the URL with /a, checks the color of it. And that's like, oop, that's the non-visited color. So he disables, he deletes that one and builds the same URL /b. Oh, that's the visited color one, which means /b is the beginning of the cookie. Then he says, okay, I got the first character. So he takes that URL apart, and now he goes to /ba. Nope, that's not visited. /bb, nope, that's not visited. /bc, oops, that's visited.

So now he has the first two characters. And you can see the reason he visited those URLs

in succession, he's left himself a trail of breadcrumbs through this history that allows him to essentially brute-force explore, but character-by-character reveal, the ASCII value of the cookie, until he gets to bcde, and then he gets - so he tries the hyphen, and he says, ah, I've reached the end. Now he's got the ASCII which he uses to, with base 64 again, to turn it back into what it was before, if it wasn't already ASCII; and he's recovered the cookie using the history of where you have browsed.

Leo: So if I, right now, and we've only covered, what, four out of the 10...

Steve: Uh-huh.

Leo: But if I cleared cookies, cleared Flash cookies, cleared Silverlight storage - I don't know what I'd do about PNGs.

Steve: Flushed your browser cache.

Leo: Flushed cache and flushed history. I've now deleted those four techniques.

Steve: Yes.

Leo: But we're not done.

Steve: And you have to do all of them because any one, any one that survives...

Leo: That's what's interesting.

Steve: ...reconstitutes the rest.

Leo: Repopulates it.

Steve: Yes.

Leo: So only one of the 10 has to survive.

Steve: Yes.

Leo: Wow.

Steve: There's also a tag, a standard HTML tag which I don't think we've ever had

occasion to talk about. It's been around for a long time. I've seen it. Anyone looking at browser headers will have noticed it. It's called the ETag. The idea is that browsers want to know if their copy of something that they have cached, like images for a page, is still current. The way that's normally done is that, when the server issues a resource, like an image, there will be an expiration date associated with it. We were just talking about expiration dates relative to this PNG cookie. There'll be an expiration date associated with it, and it'll say - and it's called the Expires Tag. And it'll say "Expires 01-01-2020," 10 years from now. And so the browser stores not only the image, but the expiration date, which allows it to say, okay, I've got a copy, and I don't need to ask for it again.

It's also possible for the browser to make a request and store the date that it cached this object, and it's able to make a request called If-Modified-Since, where the browser then says, here's the date that I have this. Give it to me if it's been modified since. And if it hasn't been - the server checks. And if it hasn't been modified since that date, the server responds with a return code of "304 Not Modified," telling the browser, hey, it's good to go, use the one you've got, we don't need to take up time with me sending it to you again.

Well, there was a concern that it would be nice also to have something more like a signature so that, if something was changed, even though, I mean, the date thing ought to be enough. But the developers said, hey, let's create like a digital signature that's called the ETag. And so it's just an arbitrary string which the server is also able to provide. So when the browser says, hey, I need this image, the server says, well, here's the Expires Tag, telling you formally how long I think you could keep it. But here's also an ETag, which is just an opaque token. And the server can generate it any way it wants to. It could be like a digital signature, like a hash of the contents of that image. So if anything changed in it, and then it was rehashed, it would create a different signature.

So the browser stores the image, the expiration date, and this ETag value all together. And when it's later bringing up a page, it sends the ETag back to the server with the header If-None-Match. So the idea being, this is the one I've got. If it doesn't, if the ETag I have doesn't match the ETag you have, then I want a fresh copy.

Well, what Samy realized - and I guess it seems, in retrospect, obvious, but apparently no one's done it before. It's an opaque token. It's a cookie. I mean, and this one scares me, you don't even need scripting for it. So far the stuff we've talked about is heavily scripting based. HTTP cookies are not. And, well, and to some degree Flash and Silverlight are because, if you had scripting disabled, they're not going to run. And the other things are heavily scripting based.

But this ETag is just like an HTTP cookie. The browser asks for something, and the server sends it an ETag which it stores. So you would need scripting to read the ETag value. So it wouldn't work without scripting, but it definitely is an opaque token which we're not thinking about right now, I mean, we haven't been for all these years. So Samy thought about it and added it to his JavaScript, which allows it to be used for tracking. So there's another one.

Leo: Wow. So I'm sure at the end we'll talk about ways to fight this. But I guess NoScript would fight the ability of a site to retrieve that cookie. But they could set it without JavaScript.

Steve: Correct. Correct. It could be set. And then if at any time later you had scripting on...

Leo: There you go.

Steve: Oh, but wait a minute, no. That would only be if the script cared, for the purpose of reconstituting all the other ones. Even without...

Leo: Ah. But at least you could get the cookie without rebuilding.

Steve: Correct. Well, even without scripting, a given instance of the browser would be sending back that same ETag. So for tracking, the ETag is 100 percent effective, just like an HTTP cookie is, and you do not need scripting. So even if NoScript was enabled, unless something was explicitly filtering ETags, you're trackable. You can't - the scripting on the client side, on the browser, would be necessary for repopulating all the other cookies and keeping all these multiple ways of tracking you synchronized. But the ETag by itself is all you need for tracking. And we have no control over it, no control through the user interface at all. So there, only flushing your cache, flushing the browser cache, would flush the images and the associated ETags.

Leo: Wow, amazing.

Steve: Yeah, yeah.

Leo: And we're not done.

Steve: We're not done.

Leo: There's more. But wait, there's more.

Steve: Turns out that the Document Object Model, the DOM, for standard state-of-the-art browsers, has an actually not very often used property for windows called "name," where a window can be named. Most windows - and by "window" we mean, like, the surface that programmers call, like, the page you're seeing a "window" from a standpoint of, like, that's the terminology used in the scripting internally. And it's not very useful because no one's ever really cared before. But someone realized, hey, you know, that's sticky. So, like, it would be a way of creating a session cookie. It doesn't persist across browser restarts. But it persists as you move around in a site. So it's not useful for, like, intercession tracking, but it's very useful for tracking a user moving through a website who might have disabled first and third-party cookies. Most people have first-party cookies enabled because so many things don't work at all if you don't enable first-party cookies. And so this is sort of an alternative first-party cookie.

The one thing that's a concern about this is that the name of the page, like the tab, or the page, is what's named. And if you click a URL to go to a different domain, the page name doesn't change. So it does create an opportunity for cross-domain leakage, and that's a bit of a privacy concern; whereas, for example, cookies at least are domain specific. If Amazon.com gives you a cookie, then there's no way that Microsoft.com is

able to read it. It's only using third-party cookies with assets that are appearing on a page that someone like DoubleClick.net, for example, is able to track you across domains.

But again, Samy's taking advantage of it. And if, for example, you were on a page, and then you deleted a bunch of stuff, and you thought you'd cleaned yourself up completely, well, he's named the page the value of the cookie. So the next thing you do on that site that's using his evercookies would - it might say, wait a minute, I don't seem to be finding any of my cookies here. But if it looks at the page, that would be where the cookie had - the one place you couldn't get to, and actually you can't. There's no way a user can delete that when they're sitting there on the site. The page has been named sort of secretly. It's not the title, not the title that you see. It's sort of an internal, programmer-level handle that scripting could use. And we have no - we as users have no access to it.

So you might think that, whew, I just successfully deleted everything. And then everything would just instantly come back because the page has a name, and it's squirreled away the value of the cookie there.

Also, Internet Explorer has its own UserData extension. And for years I've just sort of turned off in IE, back when I was an IE user, UserData Persistence. And you may remember, Leo, like in the Advanced tab of Internet Explorer, there would be, you know, you could disable UserData Persistence. And it was like, okay, that sounds like a good thing to do.

Leo: Yeah. Well, yeah, if you knew what it was, maybe.

Steve: Thank you very much. I don't know what it is, but I don't want it to be persistent.

Leo: Yeah.

Steve: We'll turn it off. Anyway, it's disappeared from the UI.

Leo: Oh, can't turn that off anymore.

Steve: No longer available to turn it off.

Leo: I presume it's a cookie-like function, though; right?

Steve: Oh, yeah. It's not sinister. It's a way, well, the good news is it's Microsoft only. No one has adopted it. It hasn't gone into standards or anything. And maybe it'll fade away, sort of the way VBScript has, something Microsoft tried to do, and it didn't take. So it's just one more way. And Samy is using it because most people are still using IE in the world, and now you can't even turn it off through the user interface. So it's the same sort of thing. It's a site-specific place where script is able to squirrel away some information, and it's persistent, as its name implies, UserData Persistence, which allows it to identify you or basically whatever information it wanted to store about you and obtain

it at some later session.

So there's all of those. I think we're at six now. There are four more that we can kind of lump together. Unfortunately, they're part and parcel with HTML5, HTML5 in the formal spec. So far everything we've talked about has sort of been, well, they've been out in left or right field somewhere.

Leo: Proprietary in some way.

Steve: Yeah, well, or...

Leo: Not standards based.

Steve: ...screwy, like setting bit values and pixels and things. These are formally part of the HTML5 spec. There's session storage, local storage (which is inherently persistent), global storage, and then even an SQLite for people who have installed SQLite, you know, SQL database on their system. In the HTML5 spec is the SQLite subset for allowing scripting to do database things on your computer. Now, the good news is most people probably aren't installing it. On the one place I was playing with this, this morning, it said, okay, I don't know about that.

Oh, in fact, I should mention, Samy's page lets you do these things. He has an "Explore the Evercookie" feature. So you can go samy.pl/evercookie. And up at the top of the page is a button that you can press to set yourself an evercookie, which is just...

Leo: On your system.

Steve: On your own system, which is just - so basically he's using his own script on his own page, at your request, to establish a cookie. It's just an integer-value 1-1000. And he says, don't worry, I'm not using this to track you. This is just for demo. And then you can press another button to try to read back the cookie that's been stored. And it all works. I found several things that it - it didn't get my PNG image value correct. It was off by 300. Actually my cookie value was 447, and the PNG recovery was 147. So there's a little bug in his code somewhere. And I'm using a high-color system, so I don't think that's what it was. But I'm sure he'll fix it. And I had just, like, yesterday he added the Silverlight technology. So this is still a little bit of a work in progress. I think he's at, like, 0.4 of his beta or something. So...

Leo: Oh, I'm so glad it's going to get better.

Steve: Yeah, isn't that nice. And it's public domain and open source, and everyone's free to grab it and use it to log on...

Leo: I'm sure they all have by now.

Steve: Uh-huh. And so in his FAQ, at the very end, he asks the question, can it be stopped? And he did say that private browsing in Safari will stop all evercookie methods after a browser restart. So Apple Safari private browsing is robust enough to just shut all this down. I mean, private browsing creates a sandboxed environment such that nothing persistent leaks. And that's good news. And for the rest of us, for example, Firefox users, our good old friend NoScript is highly effective.

Leo: Oh, good.

Steve: But, for example, it doesn't deal with ETags. I've never seen anything that does. So ETags look like a very nice, well, nice in a...

Leo: For him.

Steve: Yeah, nice for anyone who wants to track you, means of tracking people. And now that this has spread, I'm sure we'll see it popping up as a tracking means all over the place. Maybe our NoScript friend will add that and...

Leo: I was going to ask. So you could, in theory you could protect against all of this, now that people are aware of it, by building it into NoScript. Or something like that.

Steve: Well, yes. Now, okay. So here's the real takeaway from all this. I mean, because what we've seen is a bunch of very clever things that a script can do. I mean, this notion of hiding a cookie in RGB values of an image, which the HTML Canvas API allows you to access, or very cleverly hiding it in a hierarchy of visited links, which again scripting allows you to access, what this is really telling us is we've lost the war.

Leo: Yeah.

Steve: That, if you've got scripting running in your browser, you've got code which you've accepted from the sites you're visiting. And they can pretty much do what they want to, if scripting is allowed to run. It's code. And if it wants to store things on your system for the purpose of identifying you when you come back later, it's pretty much going to be able to.

Leo: Wow.

Steve: Yeah.

Leo: Well, it's a fascinating subject, I have to say. And I guess you can't really fault Samy for releasing the information because presumably others could have figured this out.

Steve: Yeah. It's better that it's in the public domain.

Leo: Better that it's out there.

Steve: And that we all know what can be done so people don't have a false sense of, oh, look, no one's going to be able to track me. I'm sneaky. It's like, yeah, well, I mean, what it really says is something like a fully sandboxed browser, or as we've talked about before, booting an environment from a CD and doing your surfing, I mean, if you're really that concerned about it, doing your CD in an environment where nothing is written to your hard drive, but it's all in RAM. And then when you shut that session down, you've disappeared. Nothing persistent from one visit to the next. Basically this raises the bar to the point where that's now the only way to be safe. Or I guess a virtual machine, too, if you...

Leo: Well, let me ask this question. So when you say "safe," what is the risk of this kind of thing?

Steve: Good point. Good point. And it's absolutely worth reminding people, is all we're talking about is tracking. We're talking about some site knowing that they saw you, you uniquely you, a week ago. And maybe not who you are, but just same entity came over and visited the site. Which many people kind of shrug off. It's like, eh, I don't care.

Leo: Well, no. In fact, it's kind of functional for a lot of sites. That's why persistence...

Steve: It can be very useful.

Leo: I mean, right from day one Netscape created cookies. They called them "persistent client-side state information." "Cookies" is probably a little bit more colloquial. But persistence is something a browser wants. It's convenient.

Steve: Certainly, yes, certainly within a browsing session, today you virtually need it.

Leo: Would have to, yeah.

Steve: I mean, the whole concept of logging onto a site is one of establishing state and identity. And now as you move through the site, and we think Amazon or eBay or MSN or Facebook...

Leo: You wouldn't want to log into every page as you go.

Steve: No, you'd literally, well, it wouldn't work at all. I mean, you couldn't do anything we use the web for now unless there was a way of you identifying yourself, even for that

session, and then having that be sticky. Because remember, each display of a web page and a query for the next web page, when you click a button or click a link, that's a whole separate transaction that isn't linked to your prior page unless - I guess the only way to do it would be to encode your identity in the URLs. And that's...

Leo: Right. Yeah, that's even worse.

Steve: ...a nightmare.

Leo: But so it's important. People need to understand that. And if you don't know how the structure of surfing works, you may not understand that each transaction is separate and unattached. So you need something persistent across transactions, even within a single visit to a website. Otherwise it's hideously inconvenient or unusable entirely.

Steve: And frankly, it is also useful to then be able to say, leave me logged in for today, for 24 hours.

Leo: Right, or just remember my password.

Steve: Right. And so you're able to...

Leo: Or my email or something.

Steve: You're able to come back within a reasonable time and say, hey, it's still me, and without having to go through the burden of being logged in again.

Leo: So some persistence is necessary. Persistence in and of itself is not necessarily bad.

Steve: And so here is the problem. We would like, in a properly working world, to give the power to the user, to say - for users to be able to say, I only want specific sites that I permit to track me. And we don't have that today.

Leo: Right, right.

Steve: That's what you'd like. You'd like to say...

Leo: And that's NoScript. That's what NoScript does. It says, "No scripts unless I say okay."

Steve: Right.

Leo: But again, against this it's only partially effective.

Steve: Yeah. And not enough.

Leo: Yeah. Very, oh, just fascinating. I'm glad you decided to cover this. Samy's got it all. We've got links to everything. Samy's got a discussion of it all, and you can go there, it's samy.pl. He's a prolific son of a gun. I mean, I'm looking at all his little projects. He's a very busy guy. I like his website, too. It's like a Windows start page. It's very clever. Samy.pl.

Steve Gibson doesn't use so many scripts on his page. So it doesn't look like a Windows start page. If you go to GRC.com, you'll see what I mean. But let me tell you, it's all there including SpinRite, the world's finest hard drive and maintenance utility; all of the great free programs Steve gives away; and, of course, this podcast, 16KB and 64KB versions available. Full transcriptions and notes, as well. And we do it, as well, because it's a TWiT podcast you'll find it at TWiT.tv/sn. And you can watch live. We record live every Wednesday, 11:00 a.m. Pacific, 2:00 p.m. Eastern time, 1800 UTC at live.twit.tv.

Next week, because Apple is having an event on a Wednesday, we're going to flip-flop with MacBreak Weekly. So Security Now! will be Tuesday at 11:00 Pacific, 2:00 p.m. Eastern. And MacBreak Weekly will be in this slot next week so that we can cover the live Apple announcement, whatever it might be. Steve, great to talk to you.

Steve: Always, Leo, a pleasure. Talk to you next week, on Tuesday.

Leo: On Security Now!. Bye bye.