OpenID

Description: Steve and Leo examine the open, platform agnostic, license free, OpenID secure Internet identity authentication system which is rapidly gaining traction within the Internet community. It may well be the "single sign-on" solution that will simplify and secure our use of the world wide web.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-095.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-095-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 95 for June 7, 2007: OpenID.

It's time to talk about security, everybody. Yeah, gather 'round the radio. Uncle Stevie's Security Hour has come. Every Thursday...

Steve Gibson: There's a new one, Leo.

Leo: How do you like that? That could be a new title. Uncle Stevie's Security Hour.

Steve: Yeah, I think we better just drop that now.

Leo: Steve Gibson's here from GRC.com, the creator of SpinRite, everybody's favorite - certainly my favorite - hard-drive maintenance and recovery utility. He creates all those great free security programs like ShieldsUP!, DCOMbobulator, Shoot The Messenger, Unplug n' Pray, the Click of Death - that was one of the first ones - and on and on and on. And today we're going to talk about - you said OpenID.

Steve: OpenID, yes. In fact, you referred to it when we were talking in our multifactor authentication, one of the two recent episodes where we talked about authenticating people. And we were talking about, remember, I was making the wish, boy, wouldn't it be cool if there was some third party, for example, that could take something really secure like RSA's SecurID token, which uses the combination of a PIN and a slowly changing number, a number that changes once every minute, and so you had, like, really strong authentication, yet not every

individual place you would want to use it would need their own authentication scheme. Essentially there would be some third-party provider who was doing the authentication with you, sort of as a proxy, and then being able to say to this target site, yes, we're sure this is Steve or Leo or whomever, and so it would sort of offload that. Well, it turns out that is exactly what OpenID is and does. And so we're going to explain it all in the next hour.

Leo: I think it's great because I'm a kind of a fan of OpenID. And I'd like to see it in more widespread use because all of these technologies, you know, aren't meaningful unless - especially OpenID, but many of them are not meaningful unless everybody uses them, or a large percent of people use them.

Steve: Well, I'll tell you, this thing has all of the feel of the way it should be done. And it's got also this - it feels very much like the way the web was originally designed. It's got the characteristics in every way I can see of something that's going to be very successful. There's a little bit of intellectual property encumbrance that seems yet still to be resolved. I think it's a Danish company, has about a six-year-old - some intellectual property patent claims over something like this. It's like, oh, no, you know, please just give it away. And in fact the OpenID.net site talks, explicitly says, you know, we want this to be as unencumbered by any intellectual property as it's possible for us to have it be. So, you know, let's hope that gets resolved. But I guess my point is that this thing has the feeling of the one that's going to win.

Leo: Shall we wrap up, before we go into this OpenID thing, wrap up anything from previous episodes?

Steve: Well, yeah. I have a couple things. I'm sort of - remember that I referred to the fact that we're beginning to get comments, I mean, really interesting and useful comments from listeners, not just questions. So I'm sort of calling this the "mailbag" portion. And we went a couple weeks...

Leo: We need a jingle.

Steve: Exactly. I went a couple weeks without opening the mailbag, just because I've been running around at my wit's end at this end and haven't had a chance really to go back and do a lot of reading. And now that I've done that, I don't know, I'm wishing we were somehow able to share more of this than we can. There's just...

Leo: Well, next week we get a question-and-answer session. But I agree with you, I think we get such great mail. It's so fascinating.

Steve: It really is. Anyway, Brad Berk- I want to say his name right - Brad Berkemier in North Sioux City, South Dakota, writes sort of very coincidentally - I tossed this in because of what we're going to be talking about today - he says, "Show 94 really sparked my imagination. I'd absolutely love to run my own OpenID server with SecurID authentication. But the problem is, how much does a SecurID installation cost? Will RSA even sell it to me without being a business? I'm guessing there's a software package you have to purchase, and then you can purchase SecurID tokens as needed individually or in packs."

Leo: I never thought of all this. But he's right, yeah.

Steve: Yeah. He said, "I'd love to know more about that, if you have any idea. If this isn't possible to do on a personal scale, Steve should start his own OpenID plus SecurID business." He says, "I'd buy."

Leo: You don't have enough to do. You need to start a new business.

Steve: That's right, I don't have enough projects in the queue. Anyway, I wanted to share that because his note was echoed by a number of other listeners. And I'm guessing that he's right. And I haven't researched it relative to RSA's SecurID, but I'm guessing they've got it locked up, it's proprietary. You need to use something that they've got complete intellectual property protection wrapped around and probably dripping in crypto so that it's not possible to do sort of a free, open version that uses their tokens.

The tokens are so cool that I just purchased a bunch of them. They've got credit card ones; they've got credit card ones with a little 10-key pad on them. I mean, I'm not going to be able to use them for anything because I don't have all the other software. But just the idea of something, you know, a little key fob that's got a six-digit number that changes mysteriously every minute, I just kind of liked that. So certainly, you know, high geek factor. But, you know, what the heck.

Anyway, Lars Solberg in Norway is using the USB fingerprint device I talked about wanting last time. You may remember that I talked about one that I'd seen that was very inexpensive. I think it was $30 or something. It had a fingerprint scanner that apparently unlocked its contents. But you had mentioned that you guys had looked at that, probably the same one, on the Lab With Leo. This one was made by SanDisk.

Leo: Yes, yes.

Steve: It happened that the day we recorded that podcast I found the box on my doorstep. And sure enough, exactly as you said, Leo, this thing was dumb inasmuch as it only worked with Windows. The good news is, I did plug it in, it did launch itself, which, you know, always gives us a queasy feeling...

Leo: U3, yeah.

Steve: ...when, you know, software on the dongle runs itself.

Leo: I think they're using a U3-like system, but not U3 itself.

Steve: Well, it's not. Because I have looked at U3, and that just gives me the willies because, I mean...

Leo: That's really scary, yeah.

Steve: ...it's really invasive. It's popping up menus and stuff, and you can't get rid of it easily. Anyway, at least the SanDisk solution is lightweight. But it still is heavily dependent on the PC. Well, Lars found one, and I will have a link in our show notes. It's at BioSlimDisk.com, so

www.bioslimdisk.com. And they've got one that is what I was hoping for. It's entirely self-contained, no drivers, OS independent. And let me just read Lars' note. He says, "Longtime listener, first mailer. During the last podcast you said that a USB stick that could just work everywhere with fingerprint authentication would be so cool. Well, I've got some good news. I have that ultimate USB stick, and I use it every day. You can find it at..." then he gives the URL, BioSlimDisk.com. It encrypts with AES 256, if I remember correctly, onboard. So there are no extra drivers or anything that needs to be installed, and it works with every OS and device that supports simple USB sticks. He said, "I even have used it on my TV, which has a USB input on it. I just love this thing." He said it was hard to find it, but he did find it, it came in the mail, and he's had it ever since.

Leo: BioSlimDisk.com. BioSlimDisk. Oh, Bio Slim Disk.

Steve: Oh, you're right, it's Bio.

Leo: I'm looking at the capitalization on the website. It's Bio Slim - you heard the music from the website. Bio Slim Disk. So now we know. Not Bios Lim Disk.

Steve: Yeah, which doesn't make nearly as much sense, although it does say BIOS, which you just sort of tend to lock onto.

Leo: BioSlimDisk. Okay, got it. Good.

Steve: So, finally, I got a really neat SpinRite testimonial that I want to share with people from, again, a Security Now! listener, Russ Suter, who wrote, "I've been a regular Security Now! listener since about Episode 15, and I've loved it. To show support for the show and for a little extra piece of mind, I purchased SpinRite a while back without any immediate need for it." And we know where this is going. He says, "It sat on my shelf for a while until a good friend of mine who's a CPA called me on Friday March 23rd, in the middle of tax season, telling me that his server had crashed with all his important client documents still on it." He said, "I rushed over and checked his backups. They hadn't run successfully in two months. But with the hustle of tax season he hadn't bothered to check them."

Leo: Of course not.

Steve: "So I dusted off my copy of SpinRite and put it to work on the poor, tired, antiquated, SCSI drive inside his four-year-old server."

Leo: SCSI, wow.

Steve: Yeah. He says, "It took until midday Monday..." - I don't know what day that was he started. But he says, "It took until midday Monday before it finished its job, and there were a whole lot of red U's showing up. But there were a lot more recovered sectors. When it had finished, I took out the SpinRite CD; and, sure enough, the system booted up, and everything was running again. It lasted the rest of the way through tax season, and by April 15th we were installing his new Dell server (with a RAID 1 mirror) and a complete offsite backup system with reporting, and copying all his data from the old server without any problems. On behalf of my friend and the hundreds of customers who now have their 2006 tax returns filed on time, thank

you."

Leo: And who probably never knew how close they came.

Steve: Well, actually you can tell that they really, you know, they saw the end of their life here. Because, I mean, then he went to a RAID 1, full mirroring, and offsite backup. And, you know, that happens often is that people will get SpinRite because they've had, like, a near death experience. And it's like, oh, thank god we're, you know, all they needed was, like, one massive save. And then they've learned their lesson; and they're, like, never going to be in a situation, even though now they're glad they have SpinRite, they just - they recognize because they came so close to complete loss of everything that now they've got to do something so that they're not as dependent on any single point of failure as they were before.

Leo: Very good. Well, that's a nice story with a happy ending. Although I'd like to see RAID 5, not RAID 1, but that's another story entirely.

Steve: Yes.

Leo: Let us move on to the subject at hand, shall we? I'm very interested to see what you make of OpenID. We've interviewed a couple of OpenID people. In fact, we did on net@nite a couple of months ago. And I use a number of different OpenID services. And so I've been intrigued by this for a while. But it seems early days. It seems like it's not quite useful yet. But you've done some research on how it works.

Steve: Well, it's certainly, I agree with you, it's early days inasmuch as I don't think I have yet encountered a single site in my limited browsing, you know, I'm not nearly the pervasive web surfer that you and Amber are, Leo. But I don't think I've ever seen a site, or at least I've never noticed it. Now, of course, it's just like anything else, you know, you learn a new word, and suddenly everyone seems to be using that new word. And you think, was no one using this word before I knew what it meant?

Leo: AOL's going to use it. Digg's using it. We could use it on TWiT. It's actually available as a Drupal module.

Steve: Well, now, wait a minute. Is Digg actually using it? Because I went to Digg...

Leo: They're supposed to be.

Steve: ...hearing that Kevin had announced he was going to earlier this year.

Leo: Yeah, they may not have implemented, yeah.

Steve: And it's still not there yet.

Leo: Yeah. I think AOL is using it now.

Steve: Well, actually AOL definitely is using it. And it turns out that the way they implemented it, and I'm going to explain how this is possible, is they've said any AOL existing users, that is, your existing log-on credentials, can be used as an OpenID authentication. So...

Leo: If you have an AOL account, you've got an OpenID.

Steve: Exactly.

Leo: Neat.

Steve: Exactly. And AOL ID is an OpenID server, which we're all about to explain. So here's the idea. First of all, OpenID allows what's known as a single sign-on capability. Without something like this, we're where we are today, where every site we go to wants our userID - sometimes it's our email address, for example - and then a password. And as we've been preaching for a long time, Leo, in order to minimize the risk of a single site being malicious, it's good practice to come up with different passwords, maybe different usernames, and not heavily reuse those because, again, the danger is you're authenticating yourself with each site you visit one-on-one.

The reason I was so stoked about the idea, for example, of a SecurID base, that is to say, RSA's secure token, is that, first of all, it gets the strength of multifactor authentication by bringing something you have, meaning the token, and something you know, which would be your matching personal password or PIN. It would bring that multifactor authentication to the concept of single sign-on, meaning that there's some other entity than the website you want to authenticate yourself with. That entity is the one that you build an identity and a trust relationship with. And then the site you want to authenticate with asks that third party, is this really that person.

Leo: So this isn't anything new. In fact, Microsoft did this; AOL has done this before. Microsoft's Passport is still around, although not particularly vital, but I still use Passport on a few sites. That was the idea; right? You'd have a Passport account with Microsoft, and then log into Expedia and various other sites using that.

Steve: Yes. So you're right, it's certainly not a new idea. What is fantastic about OpenID is that, as I mentioned before, is it is absolutely open. I mean, open, open, open.

Leo: It's not Microsoft's; it's not AOL's. Nobody owns it.

Steve: Right. The protocols are all absolutely in the public domain.

Leo: Yeah, I like that part.

Steve: Hopefully this isn't going to run afoul of anyone who comes along and says, well, we actually invented this, because you know how I feel about people inventing things that are neat

but sort of obvious. Hopefully there will be some clear prior use of this concept, even if not in this exact implementation, that would water down and hopefully prevent somebody from locking it down. But, for example, there are implementations of this in C#, in C++, Java, Perl, Python, Ruby, PHP, Cold Fusion. You know, and looking at the list I'm thinking, wait, no Assembly language? Waits...

Leo: I think you could implement it.

Steve: I think I could probably handle that. But, okay, so one of the reasons that this, for example, this differs from Microsoft's Passport is that, you know, that was all about Microsoft. OpenID is not. And in fact, Microsoft is one of the major people embracing OpenID and planning to integrate it into the system that they have got, which is CardSpace for Vista. And that's one of CardSpace and Windows will be a topic that we are going to be covering here before long . But so even Microsoft is saying, okay, Passport didn't really take. And of course nothing that is sole provider is going to.

One of the so cool things about this is that any user can have total control. There's no centralized authority. Users can choose their own authentication servers. Now, I'm avoiding all of the jargon of OpenID because there's a ton of jargon, and that's just confusing. So I'm going to use general terms that we understand. I'm sure this will not be the last time we're talking about OpenID. So we'll drill down into the specifics over time. But in general I want to explain how this works, why it's so open, why I really think this is the one, and basically what problems it has and the different modes it has of operation. So again, the idea is that the single sign-on model which is the goal, exactly as you said, Leo, with Passport, is that you would, well, as this becomes more prevalent, sites would offer you the opportunity of using your OpenID if you had one.

Leo: So you could, for instance, come to TWiT.com, and we would say use your TWiT password; but, if you have an OpenID password, you can also use that instead.

Steve: Exactly. It'd be like we will accept that, too. Now, an example of this that many people may be familiar with, for example, is PayPal. I am a heavy web commerce user, and I absolutely look for a PayPal option if I'm buying something on the 'Net from some random site that I've never worked with before. I will always try to purchase from Amazon because I would rather lump my trust with a single big provider who has a ton to lose if they ever screw up the security side of things. I mean, could you imagine a bigger disaster than Amazon's user passwords and database and purchasing patterns and credit card information and everything escaping?

Leo: Say it not.

Steve: Right. So one imagines they are being really, really, really responsible. Now, you compare that to some random Mom-and-Pop site somewhere. And I just, I do not want to give them my credit card information. Hopefully enough people feel that way that that puts pressure on them to accept PayPal. And so on sites where I can use PayPal, I never fail to do so because there I am giving them - essentially, PayPal is exactly this kind of solution for my credit information, where I end up bouncing, you know, they bounce me to PayPal, I log in there with my PayPal credentials, say to PayPal, yes, I'm me. And what happened was the site where I was sort of sent some shopping cart information along with my bounce, my redirection to PayPal. I'm talking about this because this is also very much the way OpenID works in a couple of its simpler modes.

So now I'm at PayPal. I log into PayPal, choose the way I want to pay, do I want to use my credit card, my bank account, whatever. Say yes, I want you to send money, PayPal, back to the site where I was. So PayPal takes the money from my card, verifies it, and then bounces me back to the site where I was. And that site, having got authentication from PayPal that basically the money is now theirs, says oh, good, we're going to ship your goodies to you.

And, now comparing this, I want to make one more comparison because I've recently become a fan of Google Checkout. And I prefer it to PayPal because Google has more information that they're able to provide. And that is, specifically, I don't have to fill out all the other stuff about myself because Google's got my shipping address and my billing address; whereas PayPal is limited just to the financial transaction. So if I'm somewhere and Google Checkout is offered, I find that even nicer because I log into Google, and I don't have to fill out all the other information.

And again, I mention this because that's also something that some OpenID extensions will be, well, currently do and will be used more in the future, that is, you're able to control a flow of information that you control back to that website to, for example, populate forms with all of your standard boilerplate information after you've authorized that site to receive it. So I really see an evolution here. And OpenID, I'm guessing, is going to be the one that wins because it is nonproprietary, it's completely open, and it's extremely lightweight.

Now, the way this works in practice is an OpenID, that is, this token, this string that a user uses to identify themselves is just a URL. It's a URL of a page on the web that they control, they or a friend of theirs, you know, Uncle Steve or Bob or somebody. Or maybe it's a blogging site that offers the ability to host OpenID stuff. One way or another, it's a URL of a page. Now, what I liked about it, you've heard me complain about the http://www nonsense. In the OpenID spec you specifically don't need to provide that because the recipient of this token knows to stick all that junk on because you're giving it a webpage URL.

So I'm going to a site where I want to authenticate myself using the OpenID system for whatever reason. So I put in this string, which is basically somehow to a page that I control. It can be my own server, it can be somebody else's, doesn't matter. So that site pulls that page from the Internet and looks for essentially a tag in the header. It looks for an OpenID tag which will contain the URL for the server I have assigned as my OpenID authentication server. So there's sort of a level of indirection, meaning that I give a tag pointing to my page. My page provides the URL for the server that I want this site that I'm wanting to authenticate myself to to go query in order to authenticate me.

Leo: Yeah, I've done this with - so you'd have an OpenID provider, but you could have it be your own site that you supply when you log into an OpenID page.

Steve: Well, yes. And in fact, I mean, all of this is open source. So there are already, I mean, there's a ton of servers written in PHP and Java. There are servers, you know, anybody - you could run one on your computer at home with a serving port 80, a standard web port open, and easily create your own little personal OpenID server.

Leo: Yeah, but then it's not trustworthy, is it?

Steve: Sure, because...

Leo: It's you.

Steve: Yeah, exactly, you're controlling it. Now, one of the things I want to be clear about, and we will as we go through the way this protocol works, it is a barebones protocol. And it's why I think it has a chance of succeeding, much as the original HTML was, oh my goodness, compared to today's web pages, you couldn't do anything like that. But it's because it was so simple, and it was easy to implement, and inexpensive, and nobody claimed ownership of it, those are the things that blasted the web into existence. And people could, like, copy other people's web pages and say, oh, I'm going to change the color to green, but otherwise, look, my web page is already done. I mean, it was so simple. And this protocol has all of those characteristics.

So the site that I'm wanting to authenticate myself to gets a page whose URL I have given it. From that it gets the URL of the server I have chosen as my OpenID authentication server. Now it takes my little handle and some other information about the transaction that is sort of a blob of stuff. And one of the things it gives the server is the page it wants the server to come back to, which will make sense in a second. Anyway, so basically it responds, and I should first say that there are several operating modes for OpenID. I'm going to first run through the simplest that requires the least intelligence and implementation overhead on the site that I'm wanting to authenticate to. And it's very much like the model with PayPal.

So essentially, when I submit my token, my authentication name to this site I want to authenticate to and click the button, my browser is redirected - after that site picks up the URL of my OpenID authenticator, my browser is redirected to that server, carrying with it a bunch of extra information from the site that I want to authenticate to, including that URL to send me back to when we're done. So basically now my browser is at my authentication server, and we can do whatever we want to authenticate. And this is one of the beauties of this, one of the reasons this thing is so open is that no one is saying what I, my browser and my OpenID server, need to do. It could be as simple as establishing a secure connection and using my, for example, an SSLD client-side certificate to prove, you know, which contains my own public key, which has been signed by a common certificate authority. It could be something like that. I could be prompted with another log-on username and password. I could be asked to stick my SecurID USB token into the computer or swipe my finger on my PC's fingerprint scanner. I mean, virtually anything.

So that's one of the cool things about this is that now I'm having a dialogue with whatever authentication provider I've chosen, using whatever level of authentication I care about. Remember that this is a service for me, so no one is imposing any criteria on this. Certainly, if this is an ecommerce system, or I'm wanting to authenticate to my bank, then I want a very trustworthy third-party authenticator and whatever level of trust and authentication credentials I think is sufficient. But again, it's all up to me.

So once that's done, once I've satisfied the authentication server that I really am me, then the authentication server uses some crypto stuff and the information that was provided by the site I'm authenticating to, basically to re-redirect my browser back to that original site. So in that sense it's like the trip you make to PayPal to authenticate yourself with PayPal. But in this case it's not an ecommerce transaction, it's sort of a generic authentication transaction.

So now I'm back; my browser is redirected back to the site where I want to authenticate myself. It is able to look at, essentially, the data that came back with me, which it presumes is the information that came back from this authentication third party. But there's one weakness in the system, and that is that all of this came through my browser. So we want to make sure, over on the site that I'm wanting to authenticate myself to, we need to make sure that this really came from that third-party authentication server. So now the site I want to authenticate myself to creates a direct connection to the authentication server and asks that server to validate the signature that it has that has come through the browser, and does it directly. That prevents the client-side of this, this whole little redirection bouncing, from being corrupted undetectably. At that point the site that I want to authenticate knows that the server I chose to do the authentication has said yes, this is this person who wants to authenticate, and by having a direct connection has verified the credentials that came back from the browser, and I'm

logged in.

Leo: So there's three sites - let me see if I get this. There's three sites here. There's the site I'm trying to get into. There's my site, which is where I store a credential for myself. Although this is unnecessary, but you could have your site be doing this.

Steve: Yes, you could, and that could be the same as the security provider.

Leo: OpenID site itself, yeah.

Steve: Exactly.

Leo: So I use ClaimID.com, which is one of many OpenID - but there's a lot of these guys, and they're all trying to have certain value-add features on top of the OpenID. There's Jyte.com, there's ClaimID, there's quite a few. Anyway, I use ClaimID, among others. I use Leoville.com as my server. And so let's say I'm logging into TWiT.tv using OpenID. Instead of providing it with other credentials, I would just give it Leoville.com in the OpenID space. It would then go through Leoville.com to my OpenID provider, ClaimID. Which would then put up a log-in page using the ClaimID log-in and password. I'd enter those credentials. ClaimID would then say to TWiT.tv, yup, he is who he says he is, and I'm logged in.

Steve: Exactly.

Leo: There's a little back-and-forth going on there. I'm just trying to simplify it.

Steve: Right. Now, there is a nicer mode, I mean, one of the things that's cool about this is what I just described was sort of like the simplest, least technology required anywhere approach. If you allow some scripting to go on, server-side and maybe client-side scripting, as well, you're able to make a quicker process, where, for example, in the model I just showed, you're always bouncing to your authentication server, so your whole web browser goes bounce over to the authentication server, where you authenticate yourself, and then you're bouncing back. There is extensions to the protocol which allows that process to be bypassed. It's possible for the site you want to authenticate to to create a direct connection initially to your authentication server, providing it with your ID, saying do you authenticate this person?

Now, for example, you could provide a list of sites where you just want de facto authentication, or a timeout from the last time you authenticated yourself, you know, any kind of features. And again, what's so cool about this is that it is wide open. It is a fertile ground, sort of a basic structure to support this fertile ground of innovation and value-added and services-added, exactly as you were saying, Leo, with ClaimID, where any way people want to have this work, there'll be someone providing it, and you could easily do it yourself. So the mechanism, for example, might be where for certain sites that you've identified you don't want to be forced to go through a reauthentication. And if, for example, a timeout has expired, then instead a small log-in window, a little browser pop-up window could come up where you do have a connection to your authentication server and log in in order to prove you're you; and then the site you're wanting to authenticate to is able to get that updated authenticated information, confirm it, and log you in. So it can be nicely transparent. And I'm sure that Microsoft will be moving in that direction with Windows where the transaction to the third-party server could probably use your Window log-on credentials, maybe other things on the fly.

Leo: Oh, that's interesting. That's a good point. So if you're logged on to your Windows account, you wouldn't have to enter anything more. It would just say, okay, there you go.

Steve: Exactly. And so...

Leo: Oh, that's kind of neat.

Steve: Well, it's going to be, I think, really cool because over time it's going to get easier to use, and we're going to begin to see spreading adoption of this. Just as PayPal is becoming more popular and Google Checkout is becoming more popular because people are more comfortable trusting single sources, this kind of single sign-on will end up changing the way people use the web. And of course at some point it will acquire critical mass, and so there will be websites under pressure to support this. Users are going to say, look, I don't want to have to come up with a username and password for your rinky-dink site. I want you to support OpenID. And it's going to be so easy to do so, no cost incurred anywhere, and a fundamentally very securable technology.

Now, there are some scary things. For example, this is all DNS based, and we know that DNS isn't the most secure of all protocols, which is putting it mildly. I mean...

Leo: So there could be a man-in-the-middle attack here?

Steve: Yes. And so, for example, there is the use of strong crypto where necessary. OpenID does specify a Diffie-Hellman key exchange at one point in one of its modes. And I would argue that you want an SSL connection so that you avoid man-in-the-middle attacks, and you're going to want to verify certificates. So but none of that is required, which is sort of good and bad. I've seen some people on the web complaining that it's possible to run OpenID in a nonsecure, in a non-bulletproof mode. And you could say, well, yes, fine, if I'm just wanting to go to some other blogging site to post a comment, and I don't want to have to create a log-on account for some random blogging site, I'd like to use my credentials at my main blogging site, you know, in a transportable fashion, you could say, okay, so make the technology scale with the amount of authentication security that you need.

And again, for something really secure, where you're doing your online banking and you're not wanting to have to again authenticate individually, well, there you'd want your authentication server to require contact over SSL by the site you want to authenticate to, require SSL in its interaction with you, if any. And so again, that's up to the user.

What's so beautiful about this is it is, you know, that kind of burden, that kind of cost, for example, SSL certificates, I mean, I have to buy one every couple of years from VeriSign, and I grumble every time I do, it's worth it for me because I want, of course, secure ecommerce for GRC. But the nice thing is there are free, non-SSL OpenID servers that are very useful, so long as users understand what level of security they are providing. And certainly there are commercial ones. I mean, VeriSign has one. And there is a - I've got a bunch of links that are in the show notes, Leo. So for people who are interested in this, by all means check out the show notes for this episode because there's lots of places - and again, you're already an OpenID user, so...

Leo: I have about four of them. But I've ended up settling on ClaimID.

Steve: I was going to say, that's a little against the principle, Leo.

Leo: Well, I try stuff out. You understand.

Steve: I do.

Leo: And each of these OpenID providers has, as I said, as a kind of a come-on to use their service, they offer different things. Like ClaimID, for instance, has a whole page where you can say "I claim these pages on the web; these are not mine." So I use that service. And so the OpenID is kind of a side benefit to it.

Steve: Yeah, it's one of the things that the OpenID system identifies itself, and it's very clear to explain, that this is about identity, not trust.

Leo: Oh, yes, and that's good, that's good to make that distinction, isn't it.

Steve: Yes. Because trust requires identify, but not the other way around. You have to be sure about who it is that you're trusting. But in order to identify someone uniquely, that doesn't imply anything about trusting them. That just means who they are. So by making that distinction very clear, that's one of the ways that the OpenID system has sort of kept itself at a lowest common denominator, completely open source, completely open spec. As you said, there are hundreds of servers all over the world. Many of them are, well, probably I'm sure most of them are free. Even the secure ones are free. Anyway, I think we're going to see this is the system that ends up taking hold. And it has been thought through carefully. It's got the feel of lightweight, Internet-style protocols. And it's so open that people are going to be building more onto it.

Leo: I wish Microsoft and Apple and Linux would support this in the operating system. And that would really be fantastic. Then when you log in with your secure credential on your operating system, you'd automatically have an OpenID.

Steve: I think it's just a matter of time. It is relatively new. It's only a few years old. It's catching on fast. And I look for the day when we're no longer sitting here jumbling individual passwords, coming up with personal password protocols and all the stuff we've talked about for managing the fact that, to be really secure, you want to not share your ID. This allows exactly what we've been talking about where you establish a strong identification, authentication relationship with one server; and then, using an open, free protocol, all the other sites on the 'Net are able to ask it if you're who you say you are.

Leo: Now, I just want to underscore - and you said this, but I want to say it again. This is not identity verification in any way. You can create an OpenID that says you're Steve Gibson. It's just a way of having a single password for all the sites you visit. It's not like a PGP signature. There's no web of trust, as you point out. There's no verification.

Steve: Well, now, a perfect example, though, of an extension to this would be you could imagine a commercial service that wanted to provide exactly that, you know, full industrial-strength identify verification. So it would claim that that's what it's going to do. It's not going to ever say it has identified someone where that person hasn't first gone through some rigorous

protocol for establishing their true identity with that service. So in the same way, for example, that GRC every few years, in order to get an SSL certificate, I have to really prove to VeriSign that I'm me, Gibson Research Corporation, a corporation still in good standing in California, blah blah blah and all this stuff, in order to get that GRC.com certificate that allows me to establish a secure connection so that people who connect to my server know that it's really me.

Similarly, you could have a service that is going to go put users, end-users, I mean, you know, individual people through some sort of similar wringer so that, if you use them as your authenticator, and the only way they will authenticate you is if they've proven your identify beforehand, then that's a perfect example of additional features built on top of this system.

Leo: Right, right. So that's exciting. I hope it does take off. Of course it's meaningless unless sites start using it. If Amazon starts using it, and Travelocity, and Expedia, you know, the sites that you shop at, L.L. Bean, wherever it is that you go, then that becomes exciting. I'd love - and if Amazon would get behind something like this, I think that that would be enough to kind of put it over the top.

Steve: Well, it certainly does need that kind of ubiquitous support. But I remember, as I'm sure you do, Leo, the early days of the web before virtually everyone in the world was on the web. And it was like, well, remember we used to talk about the chicken and egg. Well, yeah, maybe this will take off; but unless the things I really want are on the 'Net, then why? And why are people going to go on the 'Net, put themselves on the 'Net before it really exists? How is it going to achieve critical mass, blah blah blah? Well, it just happened because it was so inexpensive to do it. And at some point companies had to be on the Internet or they were going to get left behind. And although there isn't the same sort of you're-here-or-you're-notness to this, and there are, like, other ways to authenticate, I can see that pressure will be put on sites once OpenID becomes the way people want to authenticate themselves.

Leo: So next week we're going to answer questions.

Steve: Q&A Episode #20.

Leo: Yay. I'm so happy. I love doing those Q&As. Meanwhile, you can, of course, get a 16KB version of this show, as always, from Steve's site, GRC.com. He has transcripts there, show notes, and more. That's GRC.com/securitynow. You can also, of course, get his great program SpinRite, everybody's favorite, my favorite, disk recovery and maintenance utility. SpinRite. It's available from GRC.com. While you're there, check out ShieldsUP! and all those free security utilities.

Steve, we'll adjourn and get back together with Uncle Stevie's Hour O' Security.