DTLS - Datagram Transport Layer Security

Description: After catching up with lots of interesting security news, updates on Steve's Acoustic Dog Training project, and lots of other miscellany, Steve and Leo examine a recently developed and increasingly popular Internet security protocol, DTLS, which combines the advantages of UDP with SSL security.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-380.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-380-lq.mp3

SHOW TEASE: It's time for Security Now!. I'm back, baby. So is Steve Gibson. We're going to talk about a new, well, not that new, but an interesting encryption and security protocol for the web, DTLS, in just a little bit. All the security news, too. It's all next on Security Now!.

Leo Laporte: It's time for Security Now! with Steve Gibson, Episode 380, recorded November 28th, 2012: Datagram Transport Layer Security.

It's time for Security Now!, the show that protects you and your privacy online, with our Explainer in Chief, he's here right now, Mr. Steve Gibson, the man - that's him. "That's me," he says.

Steve Gibson: Welcome back from Australia, Leo.

Leo: Well, thank you. I've missed you. I really did.

Steve: Likewise. Only Tom, I have to say...

Leo: This morning I was thinking, oh, goody, it's Security Now!. I get to see Steve.

Steve: Oh, goody. I've got to wake up and take a shower and get my butt in to work, yeah.

Leo: No, I was excited. But I do thank - go ahead.

Steve: You've already been here. You've already been back for a week. You just barely missed being able to do it.

Leo: Oh, yeah. I've been back for a week. I'm acclimated. But I got back, yeah, later on Wednesday. And I decided to just skip Wednesday because of jet lag and so forth.

Steve: I was going to say, that time zone switchover is like that, I know.

Leo: You know, it was much better than it had been before. The last time I went to Australia, a couple years ago, was in the spring here, fall there. And that's the maximum difference because we are in summertime and they are in wintertime. And this is the minimum difference. It's only - we're only five hours earlier than Sydney. So it's not, I mean, Sydney's only five hours earlier than us. So, for instance, right now it's 5:18 a.m. in Sydney. Five hours is not - that doesn't phase me. Twelve hours phases me.

Steve: Yeah.

Leo: Five hours I can handle. So I'm fine, you know, I'm good. I was waking up a little early, going to bed a little early, that's about it. Or, I'm sorry, waking up a little late. So you today are going to be talking about the Datagram Transport Layer Security, whatever the heck that is.

Steve: Well, the last question that we entertained last week was from a listener who said, hey, I just got the option under optional upgrades on Windows 7 to install something called DTLS, Datagram Transport Layer Security. Should I do that or not? What is that? And I thought, well, that's a perfect topic for next week. So we...

Leo: Do that, do that.

Steve: Basically that's how I answered the question, and now we'll do a full episode to talk about it. Turns out it is very cool. It's been well done, implemented just right by people who knew what they were doing. It's had a couple RFCs. It's made its way into the security packages, so it's real. And you know if Windows, Microsoft are saying, oh, let's add this to the OS, then it's made some sort of cut. I mean, Microsoft must have some plans for it, for some reason. And so we're going to talk all about what that is. There's all kinds of news and paraphernalia and updates and miscellanea and fun stuff. So I think we've got a great podcast for your return to Security Now!, Leo.

Leo: Thank you. And by the way, hearty thanks to Tom Merritt, who made my trip

possible. Everybody pitched in here, you know, Sarah and Tom and Iyaz and everybody. But Tom took three Security Nows. So that was quite a few. Thank you, Tom.

Steve: It was very comfortable. And what are his moving plans? Is he moving down at the beginning of the year, or some time next year?

Leo: Yeah, I think what he told us, you know, Eileen, his wife, my former producer, got a great job as a senior producer for YouTube in L.A.

Steve: Wow, neat.

Leo: So I think she's already started. I think she's moved down there.

Steve: I think he said that last week, yeah.

Leo: And so he is going to wrap things up here. They've got a house and so forth. And I think he plans to be down there early next year. But we're going to, I mean, we'll figure this out. He probably won't be able to fill in for me on Security Now! anymore, but I think he's still going to be able to do TNT and all of that stuff.

Steve: Good.

Leo: We're hoping, anyway, because it's his show. I can't - no one else can do TNT. So, yeah. So anyway, thank you, Tom, for doing that. But I'm glad to be back. I missed - some shows I missed more than others. But I missed this show a lot. I'm sorry. Didn't mean to distract. Steve Gibson, let's start with the security news. I know there's something.

Steve: So we got, yeah, a number of things, sort of observations, and a few events. Brian Krebs had an interesting note on his site that I just thought was sort of interesting as some background about what's going on. And that's one of the things that Brian really does well, is he's, like, I guess he's infiltrated, would be the right word, many of the hacker underground where he keeps an understandably low profile. But it allows him to have a finger on the pulse of what's going on in that world, which, unless you really made an effort to be part of that community, it wouldn't be easy to have. He picked up, essentially, an ad from a hacker who was offering a new zero-day flaw. Yes, a new zero-day flaw, which actually sounds redundant. And I was wondering, why did I write the word "new"? It's because it's a new...

Leo: And there've been many before this.

Steve: Well, it's because it's a newly introduced flaw that doesn't exist in earlier versions

of Java, meaning that they just created this flaw...

Leo: Oh, lord.

Steve: ...in the most - I know - in the most recent update. So, and the hacker is expecting to get a five-figure sum. Brian wrote, "According to the" - I love this. He used the word "vendor." This is a zero-day vendor. He said, "According to the vendor, the weakness resides within the Java class Midi" - as in, you know, M-I-D-I, music - "MidiDevice.Info, a component of Java that handled audio input and output. 'Code execution is very reliable.'" This is the hacker speaking. "'Code execution is very reliable, worked on all 7 version I tested with Firefox and MSIE on Windows 7.'" So he meant Java v7, which is the current one. And Brian says, "...the seller explained in a sales thread on his exploit." It's not clear whether Chrome also is affected. He said the hacker said, "I will only sell this one time, and I leave no guarantee that it will not be patched, so use it quickly."

Leo: Oh, wow.

Steve: And Brian said, "The seller was not terribly specific on the price he is asking for this exploit, but set the expected offer at, quote, 'five digits,' unquote. The price of any exploit is ultimately whatever the market will bear, but this is roughly in line with the last Java zero-day exploit that was being traded and sold on the underground. In August, I wrote about a newly discovered Java exploit being folded into the Blackhole exploit kit, quoting the author of that crimeware tool as saying that, quote, 'the price of such an exploit, if it were sold privately, would be about $100,000.'"

Leo: Hmm. Wow. That's six figures.

Steve: So think about it. There are entities on the 'Net that have a use for vulnerabilities such that it will generate a profit if they pay $100,000 for this knowledge. So there's the question of how quickly the window will be closed, how quickly it'll be discovered, how quickly it'll be patched, how quickly it will no longer be useful. But obviously they're not going to buy it if they're not going to make money. So, I mean, it gives you some sense for the economy which is now operating on the Internet in this world, where Java, as Oracle keeps bragging about three billion devices have it, but many of them that are being - here in this case, this is the most recently updated patched versions have this problem.

Now, I just tweeted another nice piece that Brian put together titled "How to unplug Java from your browser." If you just put that phrase into Google, "How to unplug Java from the browser," actually, that'll take you right to his blog posting. Or you can just go to Twitter.com/SGgrc, my Twitter stream, and you'll find the link that I tweeted because he does go through what we've talked about before, sort of an oh, yes, yeah, just go do this, specifically for all the various browsers. And a couple people tweeted back saying, well, that must not be very recent because Brian refers to the wrench icon on Chrome, and apparently Chrome has now changed it to the more standard three lines to represent a menu. And I'm thinking, well, okay. But Chrome is about as dynamic as anything has ever been, and last time I looked at Chrome it had a wrench. So I presume if I open it up today it'll have three lines. But that was only a few days ago that I saw the wrench. So it

doesn't mean that this is old information, it's just that Chrome is ridiculously fluid.

Leo: One of the things that's kind of interesting about this is how hard it is to take Java out of Internet Explorer.

Steve: I know.

Leo: It actually doesn't need an article for most browsers. But for IE, there's registry keys.

Steve: Most of the posting is, like, 90 percent is getting it out of IE. And of course the takeaway is just get yourself out of IE.

Leo: Yeah, no kidding.

Steve: Don't use it.

Leo: Although there's a good article, I just read an article by our friend Peter Bright, who's a Windows expert, writes for Ars Technica, about Mozilla and why they've abandoned the 64-bit version for Windows, and they're sticking with 32-bit, and there's real security flaws with that, including an inability to do good ASLR. And it was just - it was very interesting. It really sounds like the most secure browser in Windows, frankly, is Internet Explorer 10, if you're a Windows 8 user. So it's a mix; right? It'd be nice to be able to just check a box, as you can in Safari and Chrome and Firefox, that says disable, you know, don't enable Java. But on the other hand there's other security benefits to IE. So...

Steve: Yeah. Microsoft is moving forward. I mean, I'm happy with the changes that they're making. The podcast two weeks ago, that is, the non-Q&A podcast that I did with Tom, we covered in detail a keynote offered by an executive VP of Microsoft explaining the nature of the browser and privacy ecosystem and why it was that they chose to turn the Do Not Track on by default.

Leo: Oh, how interesting. I want to listen to that one, yeah.

Steve: So it was an interesting take. And the bottom line, if you want to really shorten it, it's they asked people, and three quarters of their customers said yes, I want that on by default. And so Microsoft said okay.

I ran across an interesting CAPTCHA. CAPTCHAs are always - we talk about it whenever something comes along that seems fun. This one is a JavaScript jigsaw puzzle. It's called a "Key CAPTCHA," so KeyCAPTCHA.com. Now, the problem is, even tweeting it, as I did to put the link out there so that people could grab the link from my Twitter feed, brought the site down an hour or two ago.

Leo: Oh, dear.

Steve: So it's gone right now because I just said it for our live audience, who all tried to go there. And even this morning, when I was playing with it, it was dog slow. So Lord knows what they're running it on. But for anyone to take this seriously, they're going to have to get a stronger site. But it's kind of a cool idea. It shows you a picture with some chunks missing, and then a little graphic of what it's supposed to look like, and the pieces over on the upper left. And your job is to drag - is to, like, recognize how the diagram should look when it's complete. And then there's a demo...

Leo: It's kind of hard to do, actually. It wasn't easy.

Steve: Yeah, yeah. And then drag the pieces over to fill in the diagram. So I was like, okay, well, there's yet another CAPTCHA. Prove that you are not a bot.

Leo: Yeah. I'm sure somebody could do this algorithmically. But the point really should be made that most of the time what people now do is just copy this CAPTCHA to a porn site and get humans to do it. If they really wanted to get in.

Steve: Right. Well, and I would argue that the more different kinds of CAPTCHAs there are, the better, because this is one area where, if everyone used the same one, then it would make much more sense - it's very much like if we were all using only one OS, that would be less secure than if everyone wrote their own because then none of them would be identical. And it's the fact that so many people are using Java, for example, that creates the problem that Java introduces. Otherwise people would have to start from scratch on every single machine they wanted to infect. But that's not the way the world works. Still, the more CAPTCHAs we have, the better.

Now, Kaspersky has a very nice five-minute video about securing Facebook. I also sent the link, I put the link in my Twitter feed, so anyone can go to Twitter.com/SGgrc to find it. Or, again, you can Google "how video securing Facebook" is the text in the blog. That'll probably bring you to it. It's the kind of thing that our listeners ought to watch, but mostly just to preview it before sending it to everyone they know who uses Facebook. It's a little high end, that is, it's only five minutes, so it goes through a lot. But it is really, really good how-to for the settings you can change in Facebook to lock it down, for example, to require a higher level of authentication if you ever log in from a machine you haven't before; how to set up a text loop to your phone; and turning on secure.

One of the bits of news that you missed, Leo, is that Facebook is now rolling out HTTPS by default globally. They're doing the U.S. first and then worldwide. So first they gave us a setting we could turn on for that. Soon it will just be on by default. So that's a really good move forward. But this has lots of - because it's a video, it's pretty easy to follow. Someone could stop it and, like, follow along with it, and of course backspace to watch it again. But I would like to commend it to our listeners to send to their Facebook friends who think they might be able to take advantage of some better security. It's a nice little five minutes.

Leo: I'm watching it right now. Problem is it changes from minute to minute. So Brian Donohue, who did this for Threatpost on Kaspersky, will have to redo it every five minutes.

Steve: Yeah, you're right. There are details. For example, he mentions there when he's talking about turning on HTTPS security, he says, "Soon this is supposed to be on by default."

Leo: Right, right.

Steve: Well, that's now the case. Still, I think lots of good tips are in there. The other thing that I saw that caught my attention was just the problems with SCADA, S-C-A-D-A, which is the acronym for Supervisory Control and Data Acquisition. You and I talked about, on the last podcast we did together before your trip, we were bemoaning that, like, nuclear reactors were, like, connected to the Internet,, and what a horrible problem that was. That prompted one of our listeners, who works for a company that have solved that problem, to tell us about it. And so Tom read that. Might have just been last week. It was really interesting. They had what they called a DualDiode is their term for this.

The idea is they have two machines, one inside the security perimeter, one outside. They're connected only by a fiber optic cable, and the interface on the protected one can only transmit. It only has a fiber optic emitter. There is no receiver. And similarly, the one on the outside can only receive. It's unable to transmit. So that the hardware itself is only one direction. There's no way to send data in the other direction over that link. So I thought that was interesting.

But we've often talked about how, fundamentally, how difficult security is; and that, unless it's always been written from the beginning to be secure, coming back and looking at it from a security standpoint retroactively is almost guaranteed to be a disaster. And I saw several places in the last week were comments about the problems with SCADA. And once again, this is something that has been in place for a long time, and it really wasn't until the Stuxnet worm event that people realized, whoa, you can do bad things with this.

And so Kaspersky picked up on the story and wrote: "It's open season on SCADA software right now. Last week, researchers at ReVuln, an Italian security firm, released a video showing off a number of zero-day vulnerabilities in SCADA applications from manufacturers such as Siemens, GE, Schneider Electric, and others. And now a researcher at Exodus Intelligence says he's discovered more than 20 flaws in SCADA packages from the same vendors and other manufacturers, after just a few hours' work."

So this is the problem is that all of this software was written sort of pre-security concern. Much of it was written before networking happened. And in the same way that Windows got stuck on the Internet when it was not secure, and we all know how well that turned out, similarly, this SCADA software has been, you know, it's like, oh, well, wouldn't it be nice if we hooked this up to a network. And it's like, no, it wouldn't, because it'll never be secure. And so what's happening is software that was written when network security wasn't even on anyone's radar is now being examined, and it's just being found to be Swiss cheese of security. So I am glad that there are solutions like hardware-enforced optical one-way connections between secure systems and insecure systems, as one means of allowing monitoring of a facility, but no reverse direction control for the sake of security. It's like, whoa, I mean, we do, we need something in order to keep these things

safe.

And I also, just for the sake of people who tweeted this and were wondering about it, GoDaddy has had more problems. They reported, I think it was either yesterday or today, they acknowledged some strange problems that people were having who were getting infected by malware when they were visiting the websites of GoDaddy's customers. So it took a while to figure out what had gone on. What it was, was that GoDaddy's customers were spear phished. Using what they know of GoDaddy, some spear phishers pretending to be GoDaddy management phished those customers. Those customers, as a consequence, got their accounts hijacked.

Those accounts then allowed the hijackers to modify the DNS records that GoDaddy was hosting for them, which meant that people, unwitting regular visitors who visited the sites of those GoDaddy customers, were redirected to servers that were hosting the Cool Exploit Kit, which was dropping the Reveton ransomware onto their machines. So here's another instance where running with your shields up, running with NoScript on, blocking scripting by default is just really the only way you can fly if you want to poke around. Or maybe wrap yourself up in a virtual machine so that you're in an environment that you don't care about if it gets infected. You really do need some sort of protection because this stuff is out there. And so GoDaddy is apologizing and saying that they're changing the passwords on the accounts that have been hijacked and restoring the records the way they were before. It's like, okay, well, good.

And I saw another little note that ties into this. Symantec did a study where they were wondering, again, sort of like, what's the economic incentive here? Turns out that this ransomware, which is sort of one of the newer phenomena of malware that we're seeing, is netting its authors as much as, in Symantec's computations, about $33,000 a day. So again...

Leo: I'm not sure that they should publicize this. That's kind of like, hmm.

Steve: I think, boy, that's more than I'm making on SpinRite.

Leo: Maybe I should do that. Hmm.

Steve: Whoa, yeah. So there's definitely some incentive for the bad guys to find zero-day exploits, to spear phish. I mean, and the overall thing we see is that we now understand, our systems have gotten so complicated, that security is not black and white. You could argue that in the same way that no software is bug-free, no really complex software is bug-free, similarly, no really complex system has perfect security. And we keep seeing people being able to pry up the corners and find little glitches and weak spots and so forth. And so if there's strong economic motive behind doing that, people will be incented to do it and leverage it in every way that they can. So, yikes.

Leo: Yikes.

Steve: Now, a couple weeks ago I promised my analysis of CloudBerry. CloudBerry is sort of the alternative cloud storage, CloudBerry Backup, which now supports the Amazon Glacier storage, which is so cost effective for any instance where you just want

to stuff things up into the cloud, and you don't need real-time immediate access to it, which is the case for most backup. There wasn't - we already have a solution for the Mac that we've talked about. We didn't have one for Windows-based machines. I got in contact with the CloudBerry guys and got email back from their crypto guy and had a chance to look at it.

Now, there's one thing they didn't specify that I have asked about and have not yet received an answer. Everything else, though, they did right. And the question I have, and it's not crucial, but I wouldn't - and so I wouldn't be surprised if they did it right. But the bottom line is, from everything I've seen, they did everything right. They have a suite of ciphers that the user can choose among: AES, 3DES, single DES, or RC2. Now - so just choose AES. I'm not sure why anyone - I guess maybe export reasons, or if there was some reason you had to soften it, then you might just use, I mean, no. No one wants to use one DES. Maybe triple. But so you have a choice of ciphers.

Every single file or portion of a file, if you use the block level option, gets its own initialization vector. Now, that's important because remember that the way you encrypt a block of data is not just to encipher each block, each cipher size, like 128 bits for AES, by itself. If you did that, then any time you enciphered the same 128 bits, you get the same cipher out, and so patterns could be seen. So they solved this problem with something called a "cryptographic mode" where you chain these together. And we'll actually be talking about this a little bit later with regard to the DTLS protocol because this comes into play there.

But similarly, if you're going to encrypt multiple blocks with the same key, then the problem is, if you encrypt the same data, you'll get the same result in terms of the whole block. So to solve that problem we use a so-called initialization vector, which can be provided. It doesn't have to be kept secret. It just has to be pseudorandomly derived and different. So they do that correctly.

They also use - they take the user's provided password, which never leaves the user's machine. And so what I haven't used yet is the acronym, TNO. And they pass. This is fully TNO safe. Trust No One. So they take the user's password. They run it through a thousand iterations of an HMAC SHA-1 hash in order to slow down the process of turning the password into the hash. And they use an 8-byte random salt per file. So every file which is hashed under the same password ends up with a different and unique encryption key because it starts with a random salt, which is mixed in with the password key derivation function. So that was done correct. And then with every file they store the algorithm that was used, the encryption mode that was used, the length of the key that was chosen, which also by the way is user settable and configurable, the initialization vector, and then the password-based key derivation function used, and the iteration count, and the salt.

So, I mean, this is everything you want in what should be done before the data leaves your machine so that everything that's being stored in the cloud is just irreversible pseudorandom noise to any authority and any entity, bad guys, good guys, anything looking at it. And the only way to make sense of it is to bring it back as it is and then do the reverse process that you did on your machine.

So I've not switched to it, but I think I'm about to. So I will give a little more of a user-interface features sort of look once I've had a chance to do that. But a number of people have been saying that their 15-day evaluation period, or whatever the evaluation period is, is running out, and what should they do? And my advice is this looks like the real deal. I think these guys did it right.

The one thing that I mentioned they didn't talk about is something which we're beginning to understand in the security crypto industry is more important than we originally thought. And that is the idea of a message authentication code, or so-called MAC. The idea is that you want to prevent tampering. And so it's one thing to have privacy, but you also want to know that what you get back is exactly what you originally stored. And based on everything I've said so far, somebody could tamper with the data.

Now, it'd be very difficult for them to do anything useful in terms of tampering. But you would like to know when it comes back that it's exactly what it was that you sent. And that requires message authentication to sort of wrap this entire thing. And that's one thing they didn't mention when they gave me their technical readout, but neither did they say they don't have it. So I sent back a note saying, hey, what about that? And maybe they don't, which is not a huge problem, but I imagine that it's something that they ought to add if they haven't. So I'll update our listeners as soon as I know. But I'm impressed with it. I think they did a good job.

Leo: Neat.

Steve: For what it's worth, I just thought I would - I tweeted this a few days ago when I saw it, but I thought I'd let all of our listeners know. People are always asking about the blinky lights running behind me.

Leo: Yeah. Every five days, every five episodes we should explain what those are.

Steve: And those are our PDP-8 mini computer replicas, which I built from a kit, very nicely done, using a chip that was produced by Harris Semiconductor, I don't know, back in the '80s. It is a silicon version of - that is, it's a single-chip PDP-8 mini computer. I talk about the PDP-8 on my site. It's under one of the menus somewhere, under probably Miscellaneous, or GRC.com/pdp-8 will get you there. The PDP-8 had a very minimal instruction set. It was a 12-bit mini computer. Anyway, it's sort of a blast from the past. The point is there is one of these on sale from somebody who knows me because he has a video of it running my blinking lights program, which is what those things are running behind me.

Leo: You don't know who it is?

Steve: His name was familiar from - there was a forum that we were all participating in a few years ago.

Leo: Right, right.

Steve: But I posted the link on my Twitter feed, and the link is here in the show notes. You can probably search SBC6120. SBC is Single Board Computer. 6120 was the number of the chip which was used. Currently there's only been one bidder. The price is $399, which was probably the starting price - well, it would have been the starting price - and the auction has four days to go. It closes in the late morning Pacific time, at 11:35 in the morning this coming Sunday, December 2nd. So I know that at the time there was a

huge demand for these. People were always upset because the kit's not continuously available. The whole thing was prepaid. The kits were built in batches, and then people would come along later and say, hey, I want one. It's like, well, sorry.

Leo: Right. Too late.

Steve: Not available anymore. So there is a working one with all the bells and whistles. It's got battery backed-up RAM and virtual disk drives and the OS/8 operating system that was the operating system for the PDP-8. You can compile...

Leo: Is that open source or public domain now?

Steve: It went from DEC to Compaq to HP. And so it's sort of got a muddled past. There is a software version, a software emulation of the PDP-8, where DEC apparently, DEC or one of the derivative chained owners, did allow the operating system to be put into the public domain for hobby use. And then I think when Compaq had it, there was a hobby use exclusion. So people can certainly use it without feeling any concern.

Leo: Cool.

Steve: Now, also when you were gone, Leo, I had at one point, on one of our podcasts, mounted on a tripod behind me, an experimental, one-of-a-kind, return of the infamous Portable Dog Killer.

Leo: [Laughing] You mentioned you were working on this.

Steve: Well, what happened was, the way my life works is I sort of run based on demand. And it was last summer that I was lounging with my best friend, who has owned a home in Aliso Viejo for three years. We were on the patio, sipping on some Cabernet, and this dog next door was just yapping nonstop. And so, oh, and, I mean, it just - it ruined the afternoon. It was difficult to talk. It wasn't like a yap or two. It was just yap yap yap yap yap yap. And that's why I then said, okay, we've got to deal with this. Now, Mark has tried to talk to the owner. She's a nice lady. She's a pilot for American Airlines, so she's gone a lot. Her child has a babysitter. Apparently she, like, leaves the house with the dog in the back, and it just yaps up a storm. I mean, and so if Mark complains, the problem is solved for a while, and then they forget. So anyway...

Leo: We should explain, for those who don't know, this is a nonlethal, despite its name, a nonlethal solution.

Steve: Yes. It was named when I was 15. And I thought it was kind of a fun name.

Leo: Seemed like a good idea.

Steve: Never killed anything. But it was designed originally, a variation on this was designed to tame a virtually rabid dog in the neighborhood that was attacking people through a fence and causing all kinds of havoc. So we have an episode which we've aired twice which is probably the No. 1 favorite episode of all time, called The Portable Dog Killer, where we explain what all this is. Last summer I launched a Google Group and started in on the design of a development platform for looking at this. But then the problem went away. I don't remember what it was. They, like, apologized, or the next-door neighbor got a new boyfriend or something, I don't know what it was, but it just stopped being a problem. And so it's like, okay, well, boy, if I don't - and then there was also the problem of the blackbirds or crows that were incredibly loud.

Leo: Oh, those things are so annoying, yeah.

Steve: And there was a tree - I was sitting out on the patio at Starbucks in the mornings. And at one point this tree would just be blackened with crows, all competing to see who could be louder than the other. And it was crazy. So I thought, okay. I remembered, of course, as we know from the adventure of the Portable Dog Killer when I was 15, that birds are affected by this also. So I thought, if I could just sweep the trees free, just tell the crows, give the crows the idea, because they're smart birds, to go choose some other tree, further away from Starbucks. But then that problem went away, too. So I sort of lost steam on the project. But it wasn't before many people registered their needs for this product or a solution to this problem, too. So the pictures that I showed, you have links to them, you can put them up. I also tweeted them, and it's not far back in my Twitter stream, if people haven't seen it.

Leo: Okay. This is the speaker system. It's on a tripod. Is that four tweeters? What is...

Steve: It's four piezo tweeters. They're very efficient.

Leo: Because you're doing a high-frequency sound here. This is not human audible; right?

Steve: Well, barely. While I was working on it, I had it pitched at about 10KHz, which is audible. And in order to stand it, I had to take the lid of that box, which mounts the four tweeters, and I put them in two laptop bags stuffed inside each other and zippered it closed. Just, I mean, it was so loud. And it is really loud. Now, in Mark's setup, he's got an upstairs bedroom window that looks down into sort of the run along the side of his neighbor's house. And the dog stands there and just yaps. And you can see an antenna on the top. Mark now has three remote controls. One has a belt hook, so he can wear it if he wants to.

Leo: This is really interesting.

Steve: And so when you press the button, it gives either a one-second or four-second blast. I had it for a while where you'd press it once, and it's marked "toggle" on the front panel, you can see. But I changed that to one second or four seconds because it turns

out four seconds is a long time. Anyway, this got its first use last Tuesday. I dropped it off on Monday. And he called in the afternoon...

Leo: And he's using it on crows, not dogs.

Steve: No, this was this neighbor's dog.

Leo: This is for the dog, okay.

Steve: The problem was back.

Leo: Okay.

Steve: And I had to wait for him to stop giggling because - and I realized what this was. This was a giggle of relief because this has been a - he owns the home. This has been a problem off and on for three years that he's been there. And finally he has a solution. It's like everything he's done, pleading, imploring, I mean, what can he do? And so many people have this problem, Leo.

Anyway, the reason he was giggling was that he gave the dog from the second floor window down to the alley along the side of the house just a short little blip, just, I mean, nothing. And the dog jumped about three feet in the air and has never been seen since.

Leo: I shouldn't laugh. But it's probably not painful. It's just shocking.

Steve: Oh, no, no, it just startled the dog. It's just outside of its experience.

Leo: It's no different really than yelling at a dog.

Steve: Right. If you - yeah. But yelling, you know, your throat gets hoarse, and it's much less high-tech and fun. Anyway, so the problem is, in response to these pictures, a number of our listeners tweeted, well, that doesn't look like something I can go jogging with. And the other thing is that it really is overkill. I didn't want to do underkill.

I should also mention that I have purchased for Mark, I have said this before on the podcast, about 20 different commercial things that exist. I mean, the problem is big enough that other people have tried to solve it. None of them work. And that's been the reports that I've seen in the Google groups and in feedback mail and so forth, is people have bought things, and they don't work.

Well, so I went a little overboard with this first thing. So, I mean, it is so loud. But so I have a new design. And it looks like this. And I don't have a picture of it yet to show, but people who are seeing the video can get a sense for it. So it's just one of those piezo tweeters mounted on a nice little sort of easy-to-hold hand-size box. I learned a lot building that first one, which is - well, okay, the first one from when I was 15, and then

this latest overkill one. I have an amplifier which is only five components. Technically it's called a Class E amplifier. It's a digital switching amplifier that uses two inductors and two capacitors. But it turns 4 volts into 60 volts, peak to peak, which drives the tweeter strong and, in doing so, only consumes about 200 milliamps. So it's easily battery powered. Probably three AA cells will just run it.

I've chosen a frequency of 15KHz because that's pitched so high that it doesn't disturb people, but it will definitely be audible to a dog. And remember, Leo, I'm sure you do because I know you've got good hearing, when we were growing up, televisions, you could tell when they were on even if, like, the volume was turned down.

Leo: Yeah, there'd be a flyback transformer whine.

Steve: Yes. That flyback was - that generated the horizontal sweep for the CRT. And that was 15750, so 15750KHz. So this sounds like that. It's 15KHz. So it's that same barely audible, I mean...

Leo: Have you heard of the Mosquito?

Steve: No.

Leo: This is an electronic device that is used on young people to deter loitering.

Steve: Oh, yup.

Leo: And it is at 17.4KHz and 108dB and typically can only be heard by people under 25 years of age. You have to have acute hearing. And I know this because you can also get this Mosquito tone for your cell phone as an alert tone. And kids are using it in class for their text alerts because the teacher can't hear it, but they can. And we actually played with it a little bit, and it drove the younger people in our live audience crazy, and I couldn't hear a thing.

Steve: No kidding.

Leo: Yeah.

Steve: Interesting.

Leo: You have to have acute hearing. And of course as you get older, the high end drops off.

Steve: Drops off, yes. And there were two reasons I chose 15KHz. One is that canine hearing also drops off as you go higher. And of course the speaker efficiency drops off

quickly. It's rated out to 20KHz, but that isn't - it's not flat all the way out to 20, so it's beginning to drop off.

Leo: Now, in France this was ruled illegal.

Steve: Really.

Leo: And a private individual was fined 2,000 euros after operating the device outside their house. Now, that's France. In Belgium they've been trying to prevent it. In the Republic of Ireland, under The Non-Fatal Offences Against the Person Act, it is illegal. So the legality is kind of interesting. I don't see anything in the U.S.

Steve: Well, and I would never suggest that this thing ever be...

Leo: Not used on humans.

Steve: Well, A, not used on humans, and not ever used in more than just a very short blast. I mean, no more than that is necessary. I guess one of our listeners is a jogger who has something ferocious chasing him if he, like, jogs in the wrong direction. And so he'd like to be able to use it to deter an animal from chasing him. And I can't tell you what a problem barking is, Leo. It's just amazing. My mom got new neighbors up in San Mateo, and they were both a young couple working during the day. They'd leave their dog out in the backyard, and off they would go to work. My mom and her husband were at home, retired, reading, and this dog was just barking because it was bored. It had nothing else to do. It just barked. And so, I mean, it is just such a problem for so many people.

Leo: Now, how loud? How many decibels?

Steve: I don't - we're north of a hundred.

Leo: Okay. That's pretty loud.

Steve: I mean, it really is loud.

Leo: This Mosquito that you leave on all the time, right, it just keeps kids away because they can't...

Steve: Yes, it's supposed to be an irritant for dispersing crowds, as if you were, like, spraying a foul smell in the air, that sort of thing. I chose...

Leo: By the way, it won an Ig Nobel Prize in 2006.

Steve: Ig Nobel?

Leo: Do you know about the Ig Nobel prizes?

Steve: Unh-unh.

Leo: They reward the quirkier side of scientific endeavor, achievements that first make people laugh, then make them think. So you should keep this up. You might win an Ig Nobel Prize.

Steve: Well, this is...

Leo: Oh, look at that.

Steve: This is 15KHz, a 15KHz square wave. It's also not exactly 50 percent duty cycle. It's about 60-40, which is optimal for driving this incredibly efficient - I should also mention the amplifier is about 97 percent efficient. Nothing generates any heat. All of the battery power is turned into audio energy. In order to generate the square wave, I chose an extremely cool little microprocessor. It's the MSP430, which is made by Texas Instruments. Back when I was looking at this a year ago, everyone may remember that I'd found something, the Espresso, was an inexpensive development board based on the ARM processor. And I was thinking, oh, it'd be fun to - that had so much power that you could program it in C and so forth. Well, there my goal was to have sort of an R&D platform, and I was looking at a full range power amplifier and a much fancier but inherently much more expensive solution. This is really - this is sort of the next generation, solve the problem with something lightweight, easy to carry, handheld and so forth.

Leo: Now, how directional is it? I know your original Portable Dog Killer was supposed to be very directional; right?

Steve: Yes. And this is. When I was testing it on myself, as I rotated the speaker in front of me...

Leo: You can tell.

Steve: ...it absolutely peaked when it was aimed at me. So you get some off axis, but not nearly, I mean, but it's very directional.

Leo: Should I play this 17.4Hz tone for people?

Steve: There's no way it could get out through compression.

Leo: I'll play it and see if - you're right, I mean, I would think it would be rolled off.

Steve: Yeah. I would be almost sure.

Leo: I'm playing it now. I don't know, does anybody hear that? I don't hear anything.

Steve: Yeah. And if anyone has any tinnitus at all, then that's also going to be - it's like, wait. Is that a tone, or am I...

Leo: Is that me or my ears? Yeah, yeah.

Steve: Exactly.

Leo: Wow. People in the chatroom saying they can hear it. No.

Steve: Eh.

Leo: All right. I'll tell you what. I'm going to play it sometime in the next 30 seconds. But I won't tell you when. And if you can hear it, then raise your hand. I don't know. No, it's not - this is silly. I think there's a psychological impact.

Steve: So I needed to generate a stable 15KHz square wave with a specific duty cycle, and I wanted to do it with very few components. This MSP430 is really cool. I actually first discovered it in March when I was considering automating my coffee pot. Because you may remember that I went through the whole drip coffee phase, finding the right grinder and the right beans and all that.

Leo: They heard it. They can heard it.

Steve: No kidding.

Leo: They heard it. I snuck it in there when you said "drip coffee," and everybody went, "I hear it." We've got some young listeners with some good ears.

Steve: Wow.

Leo: I'm surprised that the compression doesn't roll it off.

Steve: I really am, too.

Leo: It's 17.4KHz, yeah. Isn't that interesting.

Steve: I really am.

Leo: I'm sorry.

Steve: So this microprocessor is very cool. This is the development board which is available for it. It costs $4.30.

Leo: Is that the Espresso?

Steve: No.

Leo: This is something different.

Steve: Yes, this is the chip that I'm using for this next-generation, very efficient, very lightweight, very effective, I believe, dog trainer. The chip, since the chip is the MSP430, TI sells this complete development kit for $4.30 with free shipping. So you can just, anyone who wants to start playing, it's $4.30. I went through the whole purchasing process to verify that the shipping was free. It's got a socket. It comes with two chips. And my plan is to - I'll end up with a bill of materials with the parts everyone needs to get if they want to build one for themselves, and the software for the chip, which you just saw running there, generating that square wave, and I'm just going to put it all up on my website so people can solve this problem for themselves. And we'll sort of play it by ear. If there are people who can't build it, I'm sort of thinking maybe I'll generate, I'll, like, build 20 and let people experiment with them. I'd like to have some beta testers...

Leo: That's so cool.

Steve: ...who can relate what their experiences are. And actually one of my Starbucks buddies said, hey, it's too bad we don't have a video of when Mark gave the annoying dog next door a little blast.

Leo: Yeah.

Steve: And I thought, oh, that would be perfect. So what I'm thinking is the...

Leo: You going to build a camera in?

Steve: Well, no. Everybody's got camera phones now.

Leo: That's true, yeah.

Steve: And so if our listeners who borrow one of these from me could make a video of their problem being solved, that would really be wonderful. So anyway, I will update people as this moves along. I just wanted to let everyone know that there have been people who have this problem, have continued to have the problem, and it looks like I'll have a solution with a complete kit that's very easy to build and will also be very inexpensive.

And you and I mentioned, I think before we began recording, Leo, that we had recently donated to Wikipedia. I did tweet that. And I wanted to thank people for buying SpinRite because it's because of people buying SpinRite that I am able to donate some of that money back to Wikipedia. I gave them a hundred dollars the other day because it popped up.

Oh, and by the way, speaking of Wikipedia, Wikipedia has a beautiful article on this microprocessor, on the MSP430. So if you just Google "MSP430" - by the way, it stands for Mixed Signal Processor. If you Google "MSP430," I think the third link down is the Wikipedia article on it which is a really nice comprehensive overview of the chip. And there's lots of things I didn't say about it. It's 16 bits. It's a beautiful architecture. Very few instructions, I think 27 instructions. A von Neumann architecture, not a Harvard architecture, so it's mixed data and code in a unified address space. It's little endian byte ordering, which I prefer, the way the Intel chips are. Sixteen 16-bit registers, just a joy to program in assembler, which is what I'm doing, but also in C. There are free - the GNU tool chain supports it. Eclipse-based GUI supports it. So there are multiple free development systems. And I should also mention that it uses less power when it's idle, but still alive, than the battery's own self-discharge rate.

Leo: Microamp. It's amazing.

Steve: It's just a super, super, yes, it uses a microamp of power.

Leo: That's amazing.

Steve: Which is less than the battery's own chemistry loses in energy sitting on the shelf.

Leo: But it wakes up pretty quick, too.

Steve: Yes. It's able to come up to speed in a microsecond. I ought to say that where I'm thinking of heading is, and the reason I also have chosen this, is I'm considering going further and doing what I would call a "dog sitter," so that you can put this thing on a fence or on a tree or in your backyard or wherever, and it will hear the bark, recognize it, realize that it's not a motorcycle driving down the street or a car backfiring or someone piling wood on the side of the house, I mean, it will really be a bark recognizer, and then give a little toot of the horn in order to respond to a dog's bark and hopefully train it quickly not to bark.

Leo: I love this. I love this. This is great.

Steve: So we'll be following along.

Leo: Love it. Good. You've got it all because I think the ad is done. So it's all yours from here on end.

Steve: I talked a few weeks ago about a Kickstarter movie that a listener of ours was trying to get launched called "The Root Kit." And his site is TheRootKit.com. And it doesn't look like it's going to make it. He's got 192 backers who have, as of this point, this morning when I wrote this down, he'd raised $13,652 out of his goal of $50,000, and he's got nine days remaining. Now, I don't know, I've not looked to see what the profile of dollars and days remaining is. It's not like eBay, where a lot of people bid at the last second in order to not have people outbid them. So I would imagine the dynamics are different here.

But for what it's worth, Jonathan Schiefer is our listener. He's trying to get this "Root Kit" movie going. He says, "Rogue computer hackers discover a plot to monitor everyone on the Internet. They fight back." So he is a movie maker, a commercial maker. He's been interviewing lots of people in the industry. I've been getting links to him. I think I've got 10 so far because I am a supporter, a backer. And anyway, it doesn't look like it's going to happen, but it was a cool idea. So I just thought I would mention it again to help him make the cut, if he can.

And I did have a nice note, an anonymous note from a listener who said, "I must be honest and tell you I borrowed a copy of SpinRite, which a friend of mine let me use to recover 120GB of data on a drive which had suddenly quit. As a systems admin by trade, I should know better, and back up regularly. But you know how it is. Backups are always on a backburner, not something we think of until problems occur. Needless to say, SpinRite saved my bacon. I was able to recover all of my data to a secondary hard drive. I was so thankful and happy, I just bought my own licensed version of SpinRite. Thank you so much for an amazing product. $89 U.S. for 120GB of data recovered is cheap. That's $0.74 per gig. I would say that falls in the bargain bin." So thank you for that. And as a consequence of people who buy SpinRite, I'm able to do everything else that I do. And I'm glad to.

Leo: Yeah. And I pledge to give a hundred bucks this year, as I did last year and the year before, to Wikipedia, as well, because I think that that's a very good cause.

Steve: Cool. So, DTLS: Datagram Transport Layer Security. We've talked about the way

the Internet moves data, how the brilliance, the genius of the 'Net was the idea that data did not have - that data could be turned into packets. Packets could be sort of blindly sent out onto this network. And each packet, aided by routers along the way, would sort of bounce from one router to the next from its originating location to its destination. And that was the way data would flow.

Prior to that we had a, you know, you called somebody, and you had a phone with modems at each end, an actual almost switched connection between the two endpoints. This was a whole re-conceiving of the notion of how data moves. And one of the brilliant parts was that they were relaxed about recognizing there would be problems. You might have routers that were congested, where all the data was, like, too much data was coming in for it to go out. And so that would be a problem. And so packets might get lost. Routers might be offline, so packets would have to route themselves around some other path. They solved the problems with a set of protocols, which were layered on top of the underlying technology. And "layer" is a term we're going to use a few times here in the next half-hour because it is the concept of layering of sort of a hierarchy of protocols that's another one of the fundamental right ways that these guys did that.

So one of the sort of lowest level protocols is one which simply tries to send data from, that is, a packet of data from one point to another. And that's the so-called UDP, which is a "datagram" is the term used for a packet. So there's that sort of send the packet onto the 'Net, hope that it gets there. The benefit of UDP protocol is that it's very lightweight. That is, you simply launch the packet at its destination and hope for the best. If you don't get a confirmation back that it's been received, then you, at your option, if it's important that it get there, could send it again. For example, this is the way DNS, the domain name system, operates, where you make a query, and if you don't get a response, you make it again because maybe the packet got lost to the server that you were querying, or its reply got lost back to you. One way or the other, you're not satisfied, so you ask again.

Another place where UDP, this kind of simple datagram traffic is nice, is we're using it right now. This is the way I'm connected to Leo and vice versa, VoIP, Voice over IP. The idea here is that we're not so concerned with the absolute perfection of the transmission of data as we are that it stays timely. That is, if a packet gets lost, it represents some few milliseconds of sound, and we're just going to skip it. We'll just ignore it. There'll be a little bit of a blurch or maybe a little perturbation in the audio, but the message still gets through.

The TCP protocol was designed for things like file transfers, like email, when we're sending a file between two points, or the web. HTTP is a protocol that runs on top of TCP, the transmission control protocol, which guarantees delivery of the data as it was sent. And that's the difference is TCP is a guaranteed delivery system, which means that a certain sequence of data is sent, and exactly that sequence of data will be received.

Now, we've already talked about all the things that can go wrong. You can have packets disappear. It's also possible, since the network sort of lets every packet fend for itself, one packet might go off on a slower route and end up coming in after a packet that was sent after it. That is to say that the packet ordering is not guaranteed on the Internet. So the TCP protocol solves all those problems. It, at the protocol level, takes responsibility for holding onto the data which has been launched out onto the Internet until the far end acknowledges that it's been received, at which point the sender can let go of it. The point is, at the application level above TCP, the application just says "send this," and the other side receives it, and all of the troubles that might occur along the way get resolved and handled by TCP. That's its brilliance.

But there's some problems with that. And that is, if a packet gets dropped, then what the protocol does is stop until that packet gets received again. That is, because TCP is guaranteed, packets need to be retransmitted. And so the sender will stop sending at some point until it gets acknowledgment that the packets have been received. So there can be a stall. You can also retransmit.

We also know that TCP always operates trying to find the maximum rate at which it can send packets. And the way it does that is it keeps sending them faster and faster, until they actually are lost somewhere. That is, it finds the problem, it finds the maximum speed that the network between its two points can accept data, by going past it. When you go past that speed, some router along the way gets overflowed and starts dropping packets. And so when that happens, TCP the protocol recognizes packet loss and backs off its speed. But then, because it might be a transient bandwidth problem on the network, it again creeps back up again. So you sort of have a sawtooth shape where it goes up and then drops and then up and then drops and then up and then drops. The point is, it's deliberately pushing the limits and stalling all the time.

So that's the reason, for example, that Voice over IP, like we're using right now, does not use the TCP protocol. It's because the TCP protocol doesn't fit the application. So this is the brilliance of the Internet is that we have a collection of protocols that we can choose among, depending upon what kind of service we want from the 'Net. Do we care about real-time service, like with gaming, where if one particular move got dropped between you and the gaming server, it might not be critical, but if a byte got dropped in a file that was being transferred, that would be critical. So we have different kinds of services.

Okay. So we already know that websites can be connected to securely using what's known as SSL or, more recently, TLS. SSL was Secure Sockets Layer, the original name of this protocol. More recently it's been named TLS, Transport Layer Security. SSL runs on top of TCP. So again, here we have sort of this stack, as it's called, a protocol stack, or layers in a hierarchy, where we have the packets at the lowest level, the IP protocol, then the TCP protocol that provides guaranteed delivery of packets in order. And then on top of that is SSL, which is a protocol that runs above TCP. That is to say that the entire SSL protocol assumes reliable, in-order delivery of packets. The SSL protocol or TLS protocol running on TCP is oblivious to packet loss and retransmission and out-of-order delivery and all of the problems that can happen because it relies on the TCP protocol layer below it to deal with that, to solve all those problems.

So what happened was, back about six years ago, in '06, a couple security researchers, and this is a tough name to pronounce, Nagendra Modadugu, who at the time was at Stanford, today he's at Google, and Eric Rescorla. Eric Rescorla we've run across before. He's a long-term security researcher, and I do get a kick out of the name that he set up for his company, it's RTFM, Inc., and at RTFM.com. And we all know that RTFM is an acronym for Read the F***ing Manual. Anyway, they decided to address the problem of needing security for UDP protocol in the same way that we need security for TCP.

Now, you might say, okay, wait a minute, why do we care about security on UDP? Well, we care about security for TCP, for example, because we want both authentication, and we want privacy. We want to know that we're talking to BankofAmerica.com, and we've endlessly discussed the problems of certificates and certificate authorities and signing and all that, which are all services that TLS, or SSL, provides the users of that protocol. So that gives us authentication, that we know who we're talking to. And we also want privacy for whatever reason. It may be between two anonymous parties. We use SSL just because we don't want to be eavesdropped on.

Well, there are a growing number of applications where authentication would be useful,

and privacy would be useful, even though we don't need TCP. The problem with TCP, for example, is not only is there the startup handshaking, the SYN packet, the SYN ACK and the ACK and so forth, that establishes the TCP connection, you get that set up first. Then you establish the SSL connection within the TCP connection. But mostly it's the issue of real-time. TCP, for the reasons we just looked at, is just no good for real-time communications. By its nature, that's not what it does. It guarantees all of the data gets to the destination exactly as it was sent. It also tries to do it as fast as possible, but in doing so it's inherently going to be stalling and having problems, in addition to just the inherent problems that the Internet provides or creates.

So what these guys did was they said, okay, UDP is seeing increasing use. The SIP protocol, SIP, Session Initiation Protocol, is what VoIP uses. It's one of the standards that VoIP uses for initiating Internet-based audio connections. And what's happened is, because there has not been a standard traditionally, other providers, like Skype for example, we know that our Skype communication is encrypted. It's not encrypted using any standard protocol, so it's difficult to extend. It's difficult to create proxies and basically to interact with it in a number of ways that might give us more freedom.

And now that I have said it, I was saying earlier it wasn't clear to me why Microsoft had added this to Windows 7. Well, maybe I've answered my own question since they're now also the owners of Skype. And it's always nice to be able to build these, to extend standards rather than have to just create some ad hoc solution.

So these guys were saying, look, there's voice communications, and also increasingly gaming. It would be very nice to use UDP, which is what gaming protocols typically use because they need real-time response. They can tolerate losing some detail, but they need to keep tied in real-time, and TCP doesn't do it. UDP does that beautifully. So they said, okay, how do we add good security, like TLS-type security, which was designed to run on top of a secure underlying protocol of TCP, how do we do that on UDP?

Well, the result was done right. These guys have a depth of understanding and experience and knowledge with security. So they said, look, one of the things we know is we do not want to reinvent anything we don't have to. And it wasn't at all that they were lazy. It was that, because they understand security, they understand that anything that they would do that would be new, good as they are, as thoroughly as they could imagine thinking it through, would be inherently troublesome. Even SSL and TLS has, years after it's been deployed, they're still finding little edge cases in it. And we've talked about those as they have come to light, where it's like, whoops, here's something we never thought of before. And it's a subtle little problem, but we always know that these things start small, and they get bigger. As Bruce Schneier has famously said, "Security problems never get smaller, they only get bigger."

So what these guys decided they would do is they would start with the existing SSL/TLS library from the OpenSSL project. And they would reuse as much of that as they could, so that they would get the benefit of everything that's been learned about how TLS operates on top of TCP. The OpenSSL project is, believe it or not, just short of a quarter million lines of code, about 240,000 lines of code. They were able to add UDP support. It's now part of OpenSSL.

And in fact I should mention that this DTLS protocol is now in OpenSSL. It's in another package, CyaSSL. It's in GnuTLS and SChannel, which is Microsoft's library in Windows. So it's there and also in libsystools. So this now exists in all of the commonly used packages and is available to anyone using those packages who would like to get the benefits of authentication and privacy a la TLS, but also have the real-time connectivity that UDP perfectly uses.

So what these guys did was they looked at what UDP limitations were and essentially made the smallest possible changes they could to a variant of TLS to allow it to operate over an insecure channel. The trickiest part was the SSL handshake, which occurs at the beginning. And we did a whole podcast on how SSL works some time ago. If anyone hasn't heard it or wants to refresh, it's available from us [SN-195], both Leo's and the archives at GRC.com. The idea is, quickly, that the client that wants to connect to a server sends a so-called "client hello" message containing a list of all the protocols that it knows about that it's able to support. The server receives that; and from its own list of the things that it supports, it chooses the best of those that are supported by the client and responds with its own hello message which returns to the client the one protocol that the server has chosen from among those that the client offered, essentially. So that allows them to negotiate the best connection possible.

The client has also sent a pseudorandom number, a nonce, which the server signs using its private key. That gets returned to the client. It's able to verify that using the server's certificate public key. So they're able to establish some secret data and verify that they each are each other, or at least that the server is who it says it is. So this handshake goes back and forth a few times.

Well, this was a problem to do over UDP because packets might get lost. Also, some of these messages being sent are much larger than a UDP packet. This is not a problem when you're on top of TCP because TCP doesn't have the concept of packets. Packets are at a level, and here again we're talking about a hierarchy, at a layer underneath TCP. TCP just sees a stream of data that's guaranteed to go in each direction and get to where it's going; whereas the UDP protocol is all about individual packets. There's no notion of a connection. There is just, oh, I sent a packet at that direction. Oh, and I just sent another one. Oh, and I got one back. But there's nothing else, there's no sort of overriding semantics on top of packets.

So what these guys decided was, again, they just did not want to reinvent the wheel. TLS was solid. It's been made really, really bulletproof. So they added a couple fields to the normal TLS records which is used just during the handshaking phase in order to handle the problem of essentially fragmentation. That is, if a large message needed to be carried within several smaller UDP packets, then technically that message would be fragmented into pieces, each in its own UDP packet. Those are serialized in order to handle out-of-order arrival and also missing packets. And so there's like a little tiny sort of a mini TCP which doesn't - it's not fancy. They kept it as simple as possible.

And in fact, to give you some sense of this, whereas the OpenSSL library is 240,000 lines of code, they were able to implement this DTLS protocol only adding an additional 7,000 lines of code, so 7,000 versus 240,000. And of that 7,000, about 60 percent of that was code they copied and pasted out of OpenSSL into their own region and then edited that a little bit for their own purposes. So even 60 percent of that was mostly from OpenSSL, but they couldn't change OpenSSL. They created their own little subset of it. So their code addition was very small in order to add essentially UDP compatibility to the existing SSL protocol.

So there's a need to negotiate between the endpoints in order to exchange certificates, verify in the same way that we do with a website, verify the identity of one side or the other, and in some cases both, it certainly supports that, and to establish the cryptographic keying under which all subsequent packets will be encrypted. And then they just made a few other little tweaks to the protocol, essentially taking up a little bit of space in each UDP packet for some context maintenance so that, once the endpoints agreed, they would be able to maintain some context for the connection so that they're able to disambiguate other similar streams running between the endpoints. And that's it.

So essentially the story is a couple guys who really understood security, with a broad, deep background in how to do this right, made minimal changes to the existing OpenSSL package which is used, for example, Apache runs on top of OpenSSL for all of its HTTPS security connections. And it is regarded as the open standard expression of how to do transport layer security correctly. These guys took that. They understood that they did not want to recreate the wheel. No reinvention here. They carefully held back from adding features. And in their paper where they talk about the design of this, they say, look, there were things we could have added that would have been kind of fun. But all of our experience is that it'll just get us into trouble. All we want to do is take a little bit of space from each packet in a UDP packet for the overhead needed. Once we establish a connection, we will use the existing TLS handshake, adding to it only enough to sort of create a little bit of a mini TCP because the handshake has to have in-sequence and guarantee, otherwise we leave this alone.

And that's the DTLS protocol. It exists. It started off as an RFC 4347. And that was in '06. And it was replaced at the beginning of this year when TLS went to v1.2 in order to deal with a problem that we discussed at the time, which was a problem with the CBC mode in which packets were being encrypted. The context from one packet was being carried over into the next, and some clever attackers realized they could use that as a wedge to get in. So v1.2 of TLS fixes that. So at the beginning of this year the RFC covering DTLS was updated to RFC 6347, specifically to - and actually they haven't. They call it DTLS v1.2, not because there were any - there was never any 1.1, but they wanted to keep it in sync with TLS v1.2.

And we now have another protocol that I wouldn't be surprised if we're hearing about more in the future. It's one of the reasons I wanted to give this a podcast was so that our listeners had some foundation and understanding about this because my sense is we'll be hearing that World of Warcraft is now running over DTLS, and it's like, whoa, you know, what's that? Well, now we know. It is datagram-based transport layer security, which is a variation of the existing transport layer security that was originally designed to run over TCP. This one runs over datagrams and doesn't care if some get lost, doesn't care if they arrive out of order. It still gives you privacy from encryption and authentication thanks to that startup handshake. And that's the story.

Leo: That's the story, morning glory. DTLS. Cool, very cool.

Steve: Yup. Another protocol in the bag.

Leo: [Laughing] Eventually you have to do a whole series of Schoolhouse Rock episodes explaining all of this so that people like me can understand it. Thank you, Steve Gibson. Steve's notes are available online at this website, GRC.com. That's also, by the way, where you'll find 16Kb versions of this show for the bandwidth-challenged. You'll also find transcripts, for those who like to read as they listen. And all of the great stuff Steve offers, including SpinRite, the world's best hard drive maintenance and recovery utility. Lots of freebies there, too. GRC.com. Steve's on Twitter, @SGgrc.

Steve: And before long, Leo, you're going to be able to say, and you'll be able to pick up plans for Steve's acoustic dog training project.

Leo: There are a number of people in the chatroom who say, "I could really use that."

Steve: Leo, the problem is amazing. I mean, it's one of the reasons I feel glad that I'm getting back to this. I felt a little badly that I dropped it. There were people saying, hey, you know, I want one of those. And so one way or another I want to make that possible, make that happen.

Leo: Excellent. Well, you can watch this show 11:00 a.m. Pacific, 2:00 p.m. Eastern time. That would be 1900 UTC on TWiT.tv. That's when we do it live. But you can also get on-demand audio and video after the fact from us. So you get the small audio from Steve, the larger high-quality audio or video from us at TWiT.tv.

Steve: And transcripts from me, too, thanks to Elaine.

Leo: Yeah, nice job. All right. Thank you, Steve.

Steve: Okay, my friend. Thank you, and we'll talk next week.

Leo: On Security Now!.