
# The Enigma Machine

**Description:** Leo and I first discuss a surprisingly busy week of security news; then, we take a careful walk through the history (it's not what you may think) and the detailed operation of "The Enigma Machine" which Germany used to encrypt their sensitive radio traffic during the Second World War.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-490.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-490-lq.mp3

---

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We'll review all the big security news of the week; and, yes, there was quite a bit. But we'll also talk about something we both loved, the movie "The Imitation Game." We'll talk about the historical inaccuracies. But then Steve's going to go a deep dive on how the German Enigma Machine actually worked. It's kind of a fascinating story. It's all coming up next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 490, recorded January 13th, 2015: The Enigma.

It's time for Security Now! It's a winter, a cold winter Tuesday. But not where we are because we're in California. Steve Gibson is here from GRC.com. He's the guy in charge. He's the author of SpinRite, the world's best hard drive maintenance and recovery utility. Our security guru joins us each week to talk about security. And this week, security from the '40s, 1940s.

**Steve Gibson:** Yes. Yes, we're going to - this is the Enigma Machine episode. And, boy, the feedback I received after announcing that that's what we would do this week, last week, was really significant. I think everyone...

**Leo:** Good. Very excited. I'm excited.

**Steve:** ...is excited to learn how it works. I was thinking...

**Leo:** So I saw "The Imitation Game." I said I would.

**Steve:** Oh, yeah, yeah, yeah. Yup.

**Leo:** I said I would watch it, and I loved it. And Benedict Cumberbatch, it's going to be a tough Oscar field, but he'll get a nomination, I'm sure. He deserves it. He's great in this, really brings Alan Turing to life. Now, I don't know how accurate it is; but, boy, it's got heart, it's got soul, it's a beautiful performance. And I think, as far as I can tell, right, historically accurate.

**Steve:** A little less so than many people believe.

**Leo:** Oh, okay.

**Steve:** For example, he didn't build the first Enigma decrypting machine. The Polish cryptographers did that. Anyway, we're going to talk about the history of Enigma before getting into the exact mechanism and mechanics of how it works. So we will correct the historical record. That doesn't take away at all from the fact that it is - just it's a great movie.

**Leo:** Oh, yeah.

**Steve:** And I would say that it's loosely based on history. But that doesn't diminish it as a movie at all. I think it was just wonderful. And I loved, for example, how sharp the MI6 guy was. Wasn't he fun?

**Leo:** Loved him. Loved him.

**Steve:** He was perfect. I just wanted him to be that good, and he was just that good.

**Leo:** Yeah. And I shared my concerns, and it was not from watching the movie, but from a review, I think a specific review I'd read that said, why do we always have to portray our mathematical geniuses as somehow weirdos? And yes, Turing was a little odd. As played by Cumberbatch, probably Asperger's, at least, right, a little bit on the spectrum. Whether that's true or not, I don't think history reveals.

**Steve:** Although really the interesting criticism I saw was like, come on. He doesn't understand what a joke is? He doesn't understand he's being invited to lunch? I mean, it was like over the top for, like, okay, how spaced out is this guy?

**Leo:** And so that is the kind of criticism, is why is it that you can't be a normal person and brilliant at math because clearly there are - the vast majority of brilliant mathematicians are not weird. They understand a joke. And why does Hollywood kind of insist on portraying geeks in general as weirdos? But setting that aside, what a great movie. What a great movie, yeah.

**Steve:** Great movie. Great movie.

**Leo:** So we're going to talk Enigma.

**Steve:** We are. We're going to - it's funny, for a while I was thinking maybe we would do two episodes. One was how it encrypts, and the second one is how it was cracked. But when I got deep into, and I'm talking about weeds, into how the "bombe," as they were called, these things that we saw in the movie with their spinning disks, I can explain briefly how it works. But there is no way on a podcast to accurately convey the actual operation. I will for the Enigma machine. It's interesting how simple that is compared to what Turing had to design in order to decrypt it. And that really was a work of engineering genius.

**Leo:** Truly neat, yeah.

**Steve:** But we also had a bunch of news. Oh, anyway, so my point is I'm not going to do a second episode on how the cracking machine works.

**Leo:** Touch on the highlights. Touch on the highlights.

**Steve:** We'll wrap this one up with sort of the concept of how it works. And I think that's probably all we need anyway because how this thing, this mechanical device that was, like, clunky, was able to create a cipher this good is really an interesting story. And again, as I realized when I brought myself up to speed, it's like, oh, I can do this in a podcast. So we're about to see that happen.

But we also had a whole bunch of news. We want to follow up from last week on - we talked about CryptoWall and AppLocker and SPDY, which was faster, we found out, than HTTPS, sort of the next-generation protocol. Some news about ISPs having to have their behavior regulated under Title II of the Communications Act. SOHO Linux router is being taken over and turned into botnets. Windows 7 support changes starting today. Notepad++, which is a very, like a favorite editor for everybody over on Windows, site was hacked after the Paris attacks. Google annoys Microsoft yet again by predisclosing details of an unpatched Windows vulnerability. U.S. CENTCOM's Twitter and YouTube accounts were hacked. And unfortunately we're going to wrap up the week's wrap-up news by talking about the unfortunate position that our British Prime Minister David Cameron has taken.

**Leo:** Oh. Oh. Oh.

**Steve:** So, lots to talk about.

**Leo:** You know, I saw the CENTCOM hack, and I thought, oh, crud, we're not going to get to Enigma. But I think we can touch on all of this and still do Enigma.

**Steve:** Yeah.

**Leo:** All right. We'll save the commercial. We're going to do a nice shaving commercial. So we'll save that for you after the SpinRite, before we get to Enigma.

**Steve:** Perfect. Okay, so on CryptoWall a friend of the show, Christian Alexandrov, he's our friend in, I hope it's Bulgaria, who has the dentist whose...

**Leo:** Oh, yeah, yeah, yeah.

**Steve:** ...dental work he has paid for by fixing the guy's computers with SpinRite.

**Leo:** Right.

**Steve:** Christian has sent a number of testimonials in as he's wandered through the streets, fixing, like, the restaurants. He was with his girlfriend, and the restaurant computer wouldn't work, so he fixed that one. And then his dentist's computer wouldn't work, and he fixed that one. So basically he walks around with a copy of SpinRite and gets free stuff in return for fixing everybody's computers. Anyway, he did some experimenting with CryptoWall, which we talked about last week as the successor to CryptoLocker. And what he discovered I wanted to share with our listeners because it is potentially good news. And that is a non-admin account can be recovered.

So in a series of four tweets, Christian wrote: "If system protection and system restore run on all drives, including shadow copy services, on non-admin account there is hope." That's the first tweet. Second tweet was: "Under non-admin account, CryptoWall cannot see, delete, or alter any shadow copies used by system restore and system protection." Tweet No. 3: "Remove malware, then login as admin, use the shadow explorer utility to restore previous versions, before infection." And finally, "Access to read, modify, or delete shadow copies requires full disk access privilege, which non-admin accounts do not have, but admin does." So thank you, Christian.

Essentially, the short of that is, if users are running as everyone knows they should be, rather than as an administrator, as a non-admin account, if you get hit by CryptoWall, you're able to restore yourself using the shadow copy system because CryptoWall will only have the privileges of the logged-in user when you get yourself infected. And so if you disinfect yourself, even though your files are all still scrambled, you can then restore from backup shadow copies. So good to know for anybody to whom that happens and who was properly running as a non-admin account.

We've also been talking a couple times now about the notion of whitelisting apps. And it turned out, I got a tweet from Nathan Lamonski, who responded after we were talking about it last week, again through Twitter, saying you do not need MS Active Directory Group Policy. For a workgroup machine you can use the local policy editor, explained - and then he gives a link - here: HowToGeek.com. And the HowToGeek link, I imagine if you just put it into Google, it'll find it: "Block users from using certain applications with AppLocker."

So we know that that's not available on all versions of Windows, but it is on Ultimate and

Enterprise. So if you've got Ultimate, even if you're in a non-enterprise environment, you don't have to have all of the encumbrance, which is really daunting, of Active Directory and everything that comes with it. You can use just your local policy editor. So I imagine that's what I will be doing when I move myself to Windows 7. Oh, and I replied with thanks, and he said back, "I work IT in K12 education and use AppLocker. Love it. It helps with the malware situation big time." So when it's available, I think we're going to end up in a whitelisting land as we move forward.

And I did want to mention, to just sort of close this issue, that HowToGeek.com also notes that there is something called Windows Family Safety. And that's available in all versions of Windows. But it's really sort of - it has this weird, literally parent/child paradigm, where you're a child using the computer. And when a note pops up, it says, "You have to ask your parent for permission to do this." And so it's like, it's a little unfortunate. But apparently it's their family-friendly interface to AppLocker.

**Leo:** I guess better than master/slave. It's not intended for kids, right, it's just that's how you think of it.

**Steve:** Well, it's part of their, what do they call it, the Windows Live Essentials package. So you download...

**Leo:** So it is for parents, then. It is a parental control.

**Steve:** Right, right, right. Yeah, it is in that mode. It's built into 8, but it's installable under Windows 7. So if you don't have Ultimate, then that's a solution, too, although you do have to put up with being told that you've got to ask Daddy if you want to do something.

**Leo:** Papa Sysadmin.

**Steve:** So IE and SPDY, we talked last week about how Firefox and Chrome were currently SPDY-enabled. And someone shot me a note, noting that Internet Explorer 11 was. And I thought, huh? I don't quite remember that. But it turns out it's sort of conditionally so. It offers support - IE11 doesn't offer it in the Windows 7 version, but it does under Windows 8.1 with some problems. Apparently, if you use Google, you get Page Not Found errors. But then, if you reload the page, it's happy. So I found a posting saying one fix for this is to disable SPDY in Internet Options > Advanced. So, like, eh, okay, they don't sound like they quite have it right. And in any event, IE will be dropping support for SPDY because it'll be adopting the HTTP/2 standard, which we'll certainly be talking about at great length as that finally begins to happen. It's continuing to march along. It's in the standards bodies now.

And just to remind people, SPDY - and basically, SPDY got a lot of traction. SPDY is a win, and we owe our thanks to Google for their experiments with that because it uses a binary protocol rather than the verbose textual protocol that HTTP uses. It allows full multiplex communications over a single connection. Right now, because HTTP is non-multiplexed, meaning that it is a query response, query response, query response, where you can't go query query query query query and then get things back as they come, essentially a single connection is blocked by waiting for the response, which is why many

browsers now open multiple simultaneous parallel connections to the same server in order to have more going on.

The problem with that is, as we've talked about in the past, opening a TCP connection is a time-consuming process. TCP doesn't know how fast your connection is, so it does a slow start, and then it sort of waits until you start having packet loss before it backs off and, like, figures out how much bandwidth you have. So there are lots of reasons why - oh, and it also consumes lots of resources, both at your end and at the other end, before all these connections. So it makes way more sense if you do that just once, and you're able to just send stuff, send all your requests off and simultaneously be receiving whatever of what you asked for the server wants. But even more than that, the server can anticipate requests and send them ahead. So that's another big win, in addition to compression.

So lots of stuff that Google pioneered, and all of that is in the HTTP/2 specification. And by the way, they said they're not going to do like a 2.0 because then everyone, I guess, got confused by 1.0 and 1.1 that we've been living with now. So they're just going to do HTTP/2 as the official name for the next one. But again, we'll deal with that in more detail when it starts to happen.

Also, I got a kick out of the way this was characterized in the press in at least a couple places. During CES, during a one-on-one discussion with the president of the Consumer Electronics Association, that's Gary Shapiro, he was chatting with FCC Chairman Tom Wheeler. Tom implied that Title II, which we were discussing, I guess you were on your Sunday TWiT show.

**Leo:** Oh, yeah, talked about it a lot.

**Steve:** You had Brett on.

**Leo:** Yeah.

**Steve:** That Title II of the Communications Act will be the basis for the new Net Neutrality rules governing the broadband industry. And of course Title II allows the FCC to regulate telecommunications providers as common carriers. And apparently Obama has urged the Commission to use Title II to impose Net Neutrality rules, to ban blocking and throttling and paid prioritization and so forth. So anyway, the way the press characterized this was that Verizon's great lawsuit backfired because of course they sued and won their suit, but it looks like they won the battle but are maybe losing the war.

**Leo:** I wouldn't declare victory just yet. I think we'll see a lot more back and forth.

**Steve:** Yeah. But still, it's good that the - I'm happy to see that the government is taking a strong position from which then we can negotiate maybe from that position down, rather than just saying, okay, well, you guys do whatever you want to, and we'll trust you.

Brian Krebs reported that the Lizard Squad - we've talked about Lizard Squad in the last couple weeks. They came to the fore over the Christmas holidays because they brought

down both Xbox Live and the Sony PlayStation network over the holidays, like claiming that they did it to show that the security was weak on those networks, when in fact what they did was blast them with a ton of bandwidth, which is a fundamental sad fact of the Internet is that links have a limited amount of bandwidth. There are unfortunately many computers on the Internet which are not secure, which can be taken over and have historically been taken over by botnets and used to all send traffic to a single point of focus, which ends up collapsing the hardware, I mean, the routers on the way in. As the traffic aggregates from one router to the next, finally you get to a point where either the link can no longer carry that amount of traffic, or the router can't route that many packets per second.

What Brian Krebs reported, though, was really interesting. It turns out that Lizard Squad's botnet is largely powered by our little blue box consumer routers. We talked also recently about...

**Leo:** Right, you warned of this, yup.

**Steve:** Yup, exactly. There are known backdoors which have not been patched. I mean, in some cases patches are available. In other cases, the backdoor has been modified to make it much more difficult. You have to knock on the door essentially locally, which precludes a remote Internet attack. But routers are not perfect. And their public, their WAN interface is out there. And this is a little Linux machine running in these boxes. It turns out that they have created a worm so that compromised routers scan the 'Net for other compromised routers and are able to take them over. And the Christmas attacks were powered by consumer routers that are running their botnet in the background.

So I imagine that listeners to this podcast probably have replaced the default firmware with Tomato or one of the alternative better packages. Or at least they're keeping themselves current with what the manufactures provided. But we know that there are millions of these little routers that are sitting in a dusty closet, and they're doing their job, but they're also doing more than their job.

**Leo:** Well, the other problem is that router manufacturers consider these commodity devices and often don't update. Do we know what devices were suspect? It was huge. It was everything; right? It was a huge list.

**Steve:** Exactly. It was a huge list. It was cross-manufacturer and just a vast array of routers.

**Leo:** It was, what, reference code in the chipset that everybody implemented? Or I'm trying to remember the detail.

**Steve:** Yeah.

**Leo:** But you know. I don't need to. Anyway, is there a fix or a way to know if you're vulnerable?

**Steve:** There really isn't, as you're not going to see it on the LAN side. If you suspect, I mean, it's like, if you had lights on your cable modem or your fiber interface separate from the router, and they're going like crazy, but you're not doing anything, that is, your router is doing something, that would be a tipoff. Mostly just you want to not use default usernames and passwords. Brian noted that as a major reason that these routers have been exploited is that people just never, you know, they left the manufacturers' well-known defaults that way so it was very easy for bad guys to get into them. So you want to change username and password. Turn off remote management features. Almost no one needs them, but generally they're unfortunately all too often on, enabled by default. And keep the router's firmware current. If there is a firmware update, there's a reason. So as you state, none of these router manufacturers are excited about updating firmware for commodity devices. So if something has driven them to do so, there's a reason. So you want that firmware update.

Simon Zerafa, another friend of our show, sent me - he tweeted, and I was able to capture the screen of Notepad-plus-plus.org while it was defaced. The only thing that was really interesting about this was that, for some reason, well, I guess to show solidarity, the Notepad++ guys did a "Je suis Charlie" version of Notepad++. It's v6.7.4. And you can get it from them. And someone tweeted, Jon M, whose handle is @Liquidretro, he tweeted that he downloaded that version just coincidentally the night before, and it typed out a "Je suis Charlie" message the first time he ran it. So anyway, as a consequence of - I guess people over on the other side were unimpressed by Notepad++'s 6.7.4 version. And unfortunately their web server had a way in, so they suffered a defacement. Which last I looked, I guess it is back up. There was one point when they - it took them a while to notice it. Then they had nothing up there except sort of a naked directory listing. And then they got their site back up. So, but apparently they're still offering that version for anyone who cares.

Today - in my notes I said tomorrow, but it's actually today, the 13th - Windows 7 support status changes. Windows 7 is getting old. Probably old enough for me to start using it. In fact, what's happened is that, as of today, mainstream support ends, and extended support continues. Now, the good news is we really don't need mainstream support. Extended support is what Microsoft calls it where they keep sending out patches, which is all we really want. And we get five more years of that. Extended support for Windows 7 runs through January 14 of 2020, so five more years.

And so what's being lost with mainstream support is the no-charge incident support handling, warranty claims, any design changes and feature requests - not that they weren't ignoring them before, but now they're officially ignoring them. And any nonsecurity-related hotfixes, we don't get any of that anymore. But there's still available paid support, security updates, and of course all of the various online support resources will be kept available for another five years. So it does sound like it's maybe about time for me to switch to 7.

Okay, now, this was fun. We've seen this happen before, and it just happened again, which is that Microsoft is unhappy with Google over Google's pre-disclosure of a bug in Windows that Microsoft has not yet gotten around to patching.

**Leo:** This is the second one; right? This isn't the one that...

**Steve:** Yes, yes.

**Leo:** Because the first one they waited 90 days. It was not a major flaw and required physical access. And Microsoft said, all right, we're going to patch it. And that's kind of normal. That's normal; right? You wait. You tell them, tell the company. Ninety days later, if they don't do anything, I think there's a reasonable...

**Steve:** Yeah, I mean, so, yes. This second flaw is similar. It's a little more severe. It still looks like local access only and elevation of privilege. So you can get rights that you're not supposed to have. Okay. So looking at the Google page, reading about this, it also includes a proof-of-concept batch file. So it's like, okay, everyone can do this. And down at the bottom of the page it says: "This bug is subject to a 90-day disclosure deadline. If 90 days elapse without a broadly available patch, then the bug report will automatically become visible to the public."

Now, I don't - it's not clear to me whether, if a patch were available, then they wouldn't make this page available because at that point they sort of could. It'd be benign. But what I got a kick out of what was there was also a little bit of dialogue in the thread conversation among those who have privileged access to this. Back on the 11th of November, so, what, two months ago, in this dialogue thread it says "Microsoft confirmed that they are on target to provide fixes for these issues in February 2015. They asked if this would cause a problem with the 90-day deadline." And the response was "Microsoft were informed that the 90-day deadline is fixed for all vendors and bug classes so cannot be extended. Further, they were informed that the 90-day deadline for this issue expires on the 11th of January 2015." Now, of course today is Patch Tuesday. It's the 13th. So this 90 days expired two days before Patch Tuesday.

Okay, then a month later, on the 11th of December, we see in this dialogue, in this thread, Microsoft confirmed that they anticipate to provide fixes for these issues in January of 2015. And then, a month later, on January - oh, in my notes I have 2014, I meant 2015 - that is, two days ago, it says "Deadline exceeded - automatically derestricting." Now, Microsoft didn't like this at all. And there is a lengthy annoyed blog posting by the guy who's in charge of the security response center titled, "A Call for Better Coordinated Vulnerability Disclosure." So I can, I mean, the problem with the 90 days is that, due to Microsoft's fixed 30-day calendar, that is, that Microsoft has committed to only releasing on the second Tuesday of every month, this sort of has a 60/90/120 effect, meaning that Microsoft might have had the patch ready two weeks ago but couldn't patch until today. So if Microsoft weren't committed to only releasing on the calendar month, then they could be more responsive. If they had it ready on the 61st day, missing the 60th day, then they'd be okay.

Anyway, so just sort of interesting to see this going on. And here again, Google is just sticking to it, saying we think 90 days is enough. Now, of course, Google is asynchronously updating Chrome on a continuing basis. Google is updating all of their web apps on a continuing basis. So they have no notion of sort of a lockstep calendar-based patch cycle. Thus they really get 90 days. Whereas Microsoft arguably has a bigger problem with that.

But at the same time, also in the news, is Google's abandonment of pre-v4.4 Android updates, which surprised a lot of people. I think it's yesterday Metasploit, the hacking kit, was updated to include 11 new exploits which are effective against the default WebView-based browser in Android versions prior to 4.4, which is at the moment 61% of the Android install base, which Google has clearly stated it does not intend to fix. Now, the whole story is a little bit more subtle than that. So the press has gone crazy over this, about the idea that more than 60%, 61% of pre-4.4 version Android web browsers, that

is, the default browser for Android, now have 11 known and hacking kit-level, turnkey, script kiddie-able exploits that Google has said, eh, no, it's old, we're not fixing those.

It turns out that, in fact, in a dialogue that the Metasploit guys, the Rapid7 guys had with Google, Google said, well, and quoting from the conversation: "If the affected version of WebView is before 4.4" - this is Google speaking - "we generally do not develop the patches ourselves, but welcome patches which accompany the report for consideration. Other than notifying OEMs, we will not be able to take action on any report that is affecting versions before 4.4 that are not accompanied by a patch." Which everyone sort of thought, well, okay, but that's not the way it works. Normally we explain the problem and give you a proof of concept to demonstrate the exploit, and then you patch it. And so I guess Google is taking the position, well, hey, it's open source. If you want to fix it, show us the problem, give us the patch, and then we'll consider doing something with it. Which again is a different approach than we've seen before. So I don't know.

**Leo:** Isn't that what security people do, though, is set a deadline? I mean, it's not unusual to notify the company and say, hey, you know. Google cited the user's right to know, and of course the issue is always that hackers are eventually going to find it. So is 90 days typical?

**Steve:** Well, I think the problem is the particular demographic to which a very sophisticated device is being sold. That is, all of these old Android phones are still in use by people who don't know about alternative browsers or that they're in trouble. And so we have, I mean, I saw the number, it's like 900-some million Android devices that now have browsers that are in use on the 'Net that are riddled with holes. So essentially we've given innocent users very powerful Android-based smartphone devices, that are connected and now very vulnerable, that are never going to get fixed.

**Leo:** I'm sorry. I was talking about the Windows flaw. But the Android phone might be difficult to patch. I'm sure Google has the, I mean, Google could push a patch to the open source repository. But remember, these phones are generally patched by the carrier, not by Google.

**Steve:** Right, right. And so essentially the carrier - and so, I mean, we have a fundamental problem, which is, in the same way that we were talking about the manufacturers of routers, who are like, we sold it, we got your money, it's a blue plastic box, good luck to you, similarly the carriers are like, okay, you know, get a new phone.

**Leo:** They don't want to update, yeah, in many cases.

**Steve:** Yeah. So we have a problem.

**Leo:** I guess Google could, now, it's just - it's in the browser; right?

**Steve:** Correct.

**Leo:** So Google could, well, Google does offer an alternative browser. The problem is getting the word out, I guess; right?

**Steve:** Right. It's the built-in Android WebView platform that has the problem. Google has switched now to a Chrome-based browser in Android. And so that's what they're loving and caring for and feeding. It's like, yeah, that stinky old browser, we're not going to deal with that.

**Leo:** Well, I wouldn't assume that they don't have a patch for it. I would just - I don't know how they get it out to the millions of people who have old versions of Android.

**Steve:** Well, and so, yeah. So I guess the part that's...

**Leo:** It sounds like you'd have to patch the whole operating system to fix this.

**Steve:** Yeah. It's the browser built into the OS.

**Leo:** Not an app update, yeah.

**Steve:** Yeah. So, I mean, but conceptually we're now entering into an interesting, as-yet-unexplored territory where nearly a billion users have a sophisticated vulnerable phone which probably won't get fixed.

**Leo:** Isn't it analogous, though, to I buy a machine, I mean, Google, I'm sure all the phones that Google sold directly, the Nexus phones, will be patched because Google patches those directly. But it isn't...

**Steve:** Not older phones. Not before KitKat, as I understand it.

**Leo:** Oh, okay. So if I bought a Galaxy Nexus from Samsung, there's no patch available for it.

**Steve:** Right.

**Leo:** There's no way to fix it.

**Steve:** Right. And so the Metasploit guys said, hey, here's all these things that we've just published. And Google said, yeah, well, we don't, you know, sorry, it's too old. We're not going to do anything about it.

**Leo:** It's not as old as XP.

**Steve:** No, except that, well, no, that's a good point. Though, again, the demographic is different. We have...

**Leo:** It's not, really. I mean, a lot of dumb Windows users, too. I mean, I think it's just users.

**Steve:** Yeah, somehow the idea of just people not appreciating the power that they have in a cell phone and its ability to, I mean, its fundamental ability to be compromised. If we are to make a prediction on the podcast, it's that we'll be hearing - we'll be coming back to this topic, the idea that there are a billion unpatched vulnerable browsers in old Android-based cell phones.

**Leo:** Yeah. I don't know. But, see, that's the thing is I don't know - this is true of Windows XP, as well. I don't know - how far is this company responsible for fixing it? I mean, most of these are manufactured by companies other than Google, that they used the free open source system on it. It's like saying, I mean, it's a tough one. I don't think it's immediately obvious what to do.

**Steve:** No, I'm not suggesting a solution. I'm just saying we have a problem.

**Leo:** Yeah. It's a problem. I agree, yeah.

**Steve:** Yeah. As an industry, we're entering a new experiment here which is what happens with a billion cell phones with known vulnerabilities and exploit kits? We're going to find out.

**Leo:** Yeah. Yikes.

**Steve:** Yeah. In the show notes I have a screen capture of the tweet that was posted to CENTCOM's Twitter account.

**Leo:** Oh, this is terrible.

**Steve:** "In the name of Allah, the Most Gracious, the Most Merciful, the CyberCaliphate continues its CyberJihad." A little embarrassing for U.S. Central Command, to have their account hacked. And it's worth noting, again, the press got a lot of mileage out of this. It's worth noting that this doesn't mean anything bad about the integrity of Central Command's network. This is Twitter that got hacked. And Twitter's getting hacked all the time.

Now, multiple people probably have access to it. Who knows what the back story is. Bad

password, weak password, somebody got their computer infected who was logging in, I mean, good, we just don't know. It took them a while, though. And then they tweeted about a day ago, U.S. Central Command is back. They said, "We're back! CENTCOM temporarily suspended its Twitter account after an act of cybervandalism. Read more." And then there's a link there in their tweet which basically says, yeah, so this wasn't us. This was our social networks, both Twitter and YouTube. I guess a bunch of videos were posted in CENTCOM's name to YouTube.

**Leo:** I'd like to point out that both services have second-factor authentication, which obviously CENTCOM is not using.

**Steve:** Now, Leo, that's too tricky, you know, that's for experts.

**Leo:** But on the other hand, you made the excellent point, which is a lot of people probably post to these social accounts, so it's very difficult to control that.

**Steve:** Right. They wrote from Tampa, Florida in their note, they said: "Earlier today, U.S. Central Command's Twitter and YouTube sites were compromised for approximately 30 minutes. These sites reside on commercial, non-Defense Department servers, and both sites have been temporarily taken offline while we look into the incident further. CENTCOM's operational military networks were not compromised, and there was no operational impact to U.S. Central Command. CENTCOM will restore service to its Twitter and YouTube accounts as quickly as possible. We are viewing this purely as a case of cybervandalism." Which I guess is the term now. That's what Obama called the Sony hack; right?

**Leo:** Yes, yes.

**Steve:** He used the term "cybervandalism" in his pre-Christmas, final news conference of the year last year. So it's cybervandalism. Okay. Finally, I think I'd like you to play this 60-second video into the podcast, if that's possible.

**Leo:** All right. Absolutely. The lovely and talented David Cameron.

**Steve:** Mm-hmm.

**Leo:** Let me just turn on my audio so you can see it. It says a brain - what is this, a brain living within its means? Let me go back to the beginning.

[Begin clip]

DAVID CAMERON: In our country, do we want to allow a means of communication [Leo: Oh, Britain.] between people which even in extremis, with a signed warrant from the Home Secretary personally, that we cannot read? [Leo: Yes.] Now, up until now, governments of this country have said, no, we must not have such a means of

communication. That is why, in extremis, it's been possible to read someone's letter. That is why, in extremis, it's been possible to listen in to someone's telephone call. That is why the same applies with mobile communications. Let me stress again, this cannot happen unless the Home Secretary personally signs a warrant. We have a better system for safeguarding this very intrusive power than probably any other country I can think of. But the question remains, are we going to allow a means of communication where it simply isn't possible to do that? And my answer to that question is no, we must not. The first duty of any government is to keep our country and our people safe.

[End clip]

**Leo:** Huh.

**Steve:** So Cameron said, "We must not allow terrorists safe space to communicate with each other." And so he's promised a "comprehensive piece of legislation" to close the "safe spaces," as he puts it, "used by suspected terrorists to communicate online with each other."

**Leo:** By the way, we should point out that the British government does in fact have secure communications, and nobody's proposing there be a backdoor in that.

**Steve:** Correct. And he said he recognized such powers were, quote, "very intrusive," but he believed that they were justified to counter the growing threat to the U.K. as long as proper legal safeguards were in place. And essentially what he's saying is we're not going to permit encrypted communications that we are not able to eavesdrop on, meaning that Trust No One (TNO) connections like WhatsApp offers, like iMessage and FaceTime both offer, would be or could be banned under new surveillance plans. And the press has picked it up, saying in the wake of the Paris attacks, "Prime Minister wants to ban encryption that government can't read in extreme situations." And of course this whole notion of "in extremis" that he keeps talking about, well, that just means there's a way. It doesn't matter, it doesn't have to be really important. It just, you know, you either can or you can't. So, oh, Leo. I mean, I really - I'm worried about the future.

**Leo:** I don't worry at all. First of all...

**Steve:** Really.

**Leo:** Yeah. Here's a couple reasons. First of all, we don't know if he's actually going to try to get this enacted into law, and you can see already there'll be a lot of opposition if he does. Let's say he does.

**Steve:** Yes.

**Leo:** Remember the U.S. government forbade strong encryption export. Didn't stop it. It'd be very difficult for the British government unilaterally to prevent people from using strong encryption. I just don't think that they can do it. It's not enforceable. Are they going to throw everybody in jail who does?

**Steve:** They can outlaw commercial products which provide strong encryption.

**Leo:** They should because those aren't really strong encryption. And by the way, they won't outlaw those because they just go to Microsoft and others and say, hey, put a backdoor in, and it's done. It's the open source, non-commercial software that, as usual, is going to be very hard for them to stop.

**Steve:** Okay. I guess we're sort of talking at cross-purposes. I mean, you know...

**Leo:** Well, WhatsApp, you know, WhatsApp may be dead. But who cares?

**Steve:** Okay, what about iMessage?

**Leo:** TextSecure - who cares?

**Steve:** So iMessage will be...

**Leo:** TextSecure is not going to be reengineered and will be available in Great Britain, no matter what they do.

**Steve:** Okay. So - oh, okay.

**Leo:** See what I'm saying? So, yeah, maybe they will - maybe. But, frankly, do you trust iMessage now?

**Steve:** Well, no. We know that it's possible for...

**Leo:** So what, then?

**Steve:** ...Apple to provide the keys; right.

**Leo:** Right. Commercial companies, we have, I think, said pretty consistently you cannot be sure that they aren't already putting in a backdoor, regardless of the law.

**Steve:** Okay. So...

**Leo:** In fact, I would submit that the U.S. law does already have this feature. That's why you stopped doing your crypto solution, because they can write...

**Steve:** No, I anticipated that we were going to have a law come out that said...

**Leo:** But we already do. It's called the Patriot Act. They can go to anybody and say you need to put a backdoor in this. Already, right, in the U.S.? Don't they? With an NSL?

**Steve:** No. No. They're able to say give us what you have. And so the companies are now saying, "We're happy to because we no longer hold the keys." So that's what Apple says; that's what Google says. We no longer - we have reengineered our systems over the course of 2014 so that we no longer have the keys. So you're welcome to this blob of noise. Good luck, because it won't help you.

**Leo:** Good. And I'm not sure I fully credit that. So if you want to be secure, and if you're a bad guy, you don't use iMessage.

**Steve:** Right.

**Leo:** You'd be nuts to use iMessage, or Messages, as they call it. But so the real question is, okay, so the British government may do this, making explicit what's probably been going on behind the scenes all along. But does this thwart open source solutions like TextSecure?

**Steve:** It outlaws them so that we can't...

**Leo:** I would say unenforceably.

**Steve:** Right. But still, it outlaws them. So potentially you're then committing a crime if you employ strong encryption for your own non-terrorist activities because the government has said strong encryption is no longer legal. You cannot communicate in a way that we're unable to intercept. To me, that's a profound change in the world.

**Leo:** Yeah, it's not good. I'm not saying that. But I think it's unenforceable.

**Steve:** I don't care. It's profound.

**Leo:** It's wrong.

**Steve:** I mean, you're a criminal if you employ strong encryption. That's profound. I mean, you know...

**Leo:** Well, I'd like to see them arrest 10% of the British population. That's the other thing. You can make a law. If it's widely ignored, what are you going to do?

**Steve:** Yeah, I do notice a lot of people on the freeway going faster than 55.

**Leo:** There's a lot of laws in the world.

**Steve:** Or 65.

**Leo:** Yeah. If they're widely ignored, you can't arrest your population. I would anticipate some strong civil disobedience in the U.K. if this were to be passed. I think Cameron knows that, too. But we'll see. You know, maybe not.

**Steve:** Well, and the U.K. doesn't affect me because I'm in the U.S. And I wouldn't be at all surprised.

**Leo:** Because, of course, if WhatsApp is illegal in the U.K., it makes it difficult to, you know, I mean, that can spread.

**Steve:** Right. Well, yeah, exactly.

**Leo:** And somebody in the chatroom says, "What about HTTPS?" Jeff is in London, so he has an interest in this. Do you ban HTTPS? Is SSL banned? And which vendor do you go to to get the backdoor?

**Steve:** Yeah.

**Leo:** I think it's hard. I think it's very hard to make this happen. But maybe - but you're right, we should call attention to it, yeah.

**Steve:** We'll be watching, yeah. So I do have good news, before we start talking about Enigma. Something that is not an enigma is SpinRite. I got a nice note on the 8th of January from someone who signed it JS Cube. I just know that his email was - he's in Shaw. Shaw is his ISP in Canada. And he said: "Hello, GRC peeps. This is not a query, but a testimonial. I wanted to leave one on your site, but I couldn't find a link to do so." That's true, we really don't have one. He said: "You may not care, but I'd like to leave a testimonial anyway." And of course I'm more than happy to have a testimonial from JS Cube. He said: "I had an old but working hard drive that had bad sectors in it, which made it fail any previous attempts to clone it. Your product was my last resort, and it saved me a LOT [in all caps] of time reloading everything into my new SSD. I was able to

clone my old hard drive, no problem. I'm sure you hear this all the time. Thanks!!!" With three exclamation points. And he says: "I love your product. It's f@@n' awesome."

**Leo:** Freaking awesome.

**Steve:** Freaking awesome. "Best Regards, JSC." So thank you, JS Cube. I don't think you're a podcast watcher, but I'm happy that you're a SpinRite user.

**Leo:** Oh, so he didn't know about Security Now!.

**Steve:** I don't think so. He sent this to the GRC peeps, and it came to me via Sue.

**Leo:** Can there be anybody who does not know about Security Now!? How is that possible? All right. By the way, one of our chatters sent me a link to a Guardian article about "The Imitation Game" in which they go through, step by step, the historical errors or mistakes. And there's quite a few.

**Steve:** I know.

**Leo:** So I'm going to take it back that it's historically accurate. In fact, in some respects it's appalling inaccurate, and in some insultingly inaccurate. However, a great movie. And you know what, movies have to be movies, not life. And so that's what happens.

**Steve:** Right, right. As I said, it's a great movie, and I would say it's based on historically accurate characters. And it would be nice to have really had a really good sense for who Alan was, you know, exactly what was his demeanor.

**Leo:** Well, they give us an example that - and I don't want to do a spoiler. But Turing at one point was blackmailed by a spy. And this never happened. And the Guardian says it paints Turing as a coward.

**Steve:** Correct, as a traitor, actually.

**Leo:** As a traitor and a coward. And it never happened. And so they should - I understand from the moviemaker's point of view, the purpose of that engagement was to show how scared and lonely and persecuted homosexuals were in the Britain of that era. But still, it didn't happen.

**Steve:** Well, and for example, I mentioned when we were talking about this a couple weeks ago, the other thing that annoyed me was this completely artificial midnight deadline, where all the work they had done would be scrapped on the stroke of midnight. It's like, uh, no, that's not true.

**Leo:** It seemed pretty stupid on the face of it, on the part of the commandant of Bletchley Park. But so, you know what, just search for "The Imitation Game" at the Guardian, "Inventing a new slander to insult Alan Turing." It's TheGuardian.com. And you can enjoy the movie, but it's a good idea to understand some of it is just completely ridiculous.

**Steve:** What I did read of his character was that, while he did sort of have, for whatever reason, some aloofness to him, whether just essentially a social disability, he did understand humor. And if he liked you, you sort of got moved into his inner circle, then he could be quite charming. So that's not quite the person that the movie painted.

**Leo:** Yeah.

**Steve:** So here's the way this whole thing actually - the history is really interesting, too. So we can have the story, which is not historically accurate. But the good news is the truth is every bit as interesting. So radio was the way that we were communicating, "we" everybody, in wars. World War I had radio. And the problem with radio is that one person sends it, and everybody can receive it. So eavesdropping is sort of what you do. I mean, and there were, like, listening stations all over the place, specifically for receiving what anybody else transmitted.

Toward the end of World War I, Germany believed, up until they found out otherwise, that they had an uncrackable crypto during World War I that was protecting them. And they were apparently quite annoyed to learn after the end of World War I that in fact their unbreakable crypto - which was not the Enigma machine. The Enigma machine was World War II. So the First World War, everybody had cracked the German uncrackable code and was merrily decrypting their messages. And so they had actually no protection at all, and they were convinced their code was uncrackable. So they essentially decided not to make the same mistake for World War II.

Now, what's interesting is that, shortly after the First World War, that is, well before the beginning of the Second World War, a German inventor by the name of Arthur Scherbius invented and developed and patented the Enigma machine.

**Leo:** Wow.

**Steve:** So this wasn't something that the German military or the German war machine created. This was a commercial product. Not many of them sold because the number that I saw in one relating that I read said that it was $30,000. Now, I assume that's translated into today's money. But he was trying, I mean, the point was that communications was, like, by courier. You would write a letter, and then you'd send it somewhere. So he was trying to sell it to companies, corporations that needed to keep their communications secret. But it wasn't very successful because the darn thing was very complicated. And, I mean, beautifully engineered, you know, so-called "German engineering." Just works of art, but way expensive.

So the other thing that is worth noting is that simultaneously, pretty much around the world, these so-called rotor-based enciphering machines came into existence. So there was co-invention, as often happens. It was just sort of time for these code wheel

machines to happen. And so Arthur named his the Enigma. It was the machine that the German military then adopted. But there were others that operated in similar but different fashion.

Sandwiched in between Russia on one side and Germany on the other is Poland. And they were nervous because, well, they had Germany on one side and Russia on the other. And so they had a very active "cipher bureau," as they called it, whose job it was to decrypt the messages that they were able to catch out of the air in order to know what the Russians were thinking about their border and what the Germans were thinking about their border. And the crypto bureau in Poland did something that, as far as we know, other code breakers hadn't yet. And that is they employed real mathematicians. They didn't sort of employ puzzle solvers. They employed a trio of very good Polish mathematicians who applied everything they knew about math to this problem. And in the space between World War I and World War II, they were keeping an eye on things, decrypting messages, and things were fine.

Then Germany began purchasing and using Arthur's Enigma machine. And the Polish mathematicians started having a problem. They started receiving a new type of cipher over the radio that they could not crack. And so this represented a problem because, again, they needed to know as best they could for the sake of their country what Germany was doing on the border that they shared and what plans Germany had. So there was a traitor who provided some documents which they were able to get a hold of. But mostly it was truly inspired reverse-engineering. We'll talk about the - you'll get a better sense for that when I explain how the Enigma machine works and what its pieces are.

But it turns out that, although they had almost no useful information, the Polish mathematicians were able to reverse-engineer the complete design of the Enigma machine based on the code, that is, looking at what it encrypted and figuring out what the plaintext was and then thus what the wiring had to be. And you'll have a much better sense for what an amazing piece of reverse-engineering that was when I talk about how this thing works. They built clones of the Enigma machine from their reverse-engineering. And they also built the first so-called "bombe." And there are conflicting stories about why it was named that. One report said that, well, it made a ticking sound, like a bomb that was going to go off. And of course bombs were on everyone's mind at that period of time with all of these hostilities mounting. The other was that it was the name, like one of the Polish mathematicians' favorite ice cream was named after that.

**Leo:** It's a dessert, yeah.

**Steve:** Yeah, so we're not sure where the name came from. But they were able to take advantage of a crucial flaw in the protocol, that is, the instructions that the Enigma operators used that I'll talk about in a second, in order to create a machine that was able to figure out, essentially crack Enigma. And again, we'll talk about what that is. But so that's the history of this. Then at some point what Germany was doing changed. And just before the outbreak of World War II, they met with British intelligence, I think the Brits and the French, and showed them what they had.

Now, both French and English cryptographers at that point were completely mystified, completely befuddled. This is before Alan Turing's time. So they had no idea what was going on. And when these three Polish mathematicians said, uh, here's what's going on, I mean, apparently, in one recounting of this, one of the British agents had a fit because he was so upset that there was this much, this trove of information about what Germany

was doing that the Poles had not until then chosen to share. And, I mean, it was crucial. It was Enigma machines they had reverse - working Enigma machines that they had figured out from decrypting. And, oh, look, here's this thing we built that's called a "bombe," and it decrypts the code. Or at least it did until shortly before then.

Okay. So now we switch to Bletchley Park and the effort to decrypt what Germany did. And I'll talk about what the change in protocol was. Essentially what happened was the Polish mathematicians were using a trick that Alan Turing and his group didn't think was reliable. That is, it was about a procedure in setting up the code that Turing and his group realized the Germans could too easily change. And while it made cracking easy, it was deemed too fragile. And in fact on May 1st of 1940, Germany did change their procedure, and the Polish approach for cracking Enigma completely collapsed. So it turned out in retrospect to have been exactly the right call, even though what Alan and his group ended up having to build was far more complex.

Okay. So what is this Enigma machine? We start with a keyboard with 26 keys, so just the alphabet, A through Z. No upper or lowercase, no numbers, punctuation, not even a space, the idea being that, if it's converted into your language, back into German, even though it'd be nice to have spaces, this is military, and you can figure out what the words are by looking at it closely. They did sometimes use the character X in order to indicate the end of a sentence, just because X was uncommon, and that would sort of give you a clue.

The way they handled numbers, I saw one report that said they were spelled out, and that's incorrect, actually. They would use the prefix character Y. And then above the top row of keys, QWERT, it actually wasn't QWERTY, it was QWERTZ, a slightly changed key arrangement, but basically the typewriter that we're familiar with. Along the top row were nine keys, so those were numbered 1 through 9, and then P was down below. That was given the designation of zero. And so if you wanted to send a number like 1940, you'd send a Y and then a series of alphabetic characters, each corresponding to a digit. And that way somebody reading it would see a Y and then something that doesn't make any sense at all and realize, oh, this is a number.

So basically what we're sending is 26 symbols. And each one of those keys is connected to a wire that goes into what's called the "scrambler." And there's an intermediate stage I'll come back to in a second which is the plugboard which exists on the front of it. But let's sort of take this step by step. So you've got 26 keys, like buttons. And each one energizes a wire. So those 26 wires go over to a circular array of contacts. So we have a circular array of 26, I'm sure they were beautiful brass contacts. We then have a series of what were called "rotors." Think of the rotor like a hockey puck. And it's about four inches in diameter and about an inch thick. So, you know, kind of hockey puck-like.

And this hockey puck, this rotor, has a similar set of 26 contacts on one face and 26 spring-loaded pins on the other. So this hockey puck has connections on one side and pins for connecting to the next hockey puck in a stack. Inside this rotor, this hockey puck, one side of the 26 connections are cross-connected, like on one face, one face of these 26 connections are cross-connected, one for one, but in a completely haphazard fashion, to the other side. So all 26, there's, like, wires inside, and it's a maze of craziness inside as every wire from one face goes over to a different connector on the other face. So it's called a "scrambler" because it scrambles all of the - in fact, Leo, the picture down below, on the page below that you're showing right now is a slightly better one. Oh, are those my notes?

**Leo:** No, but...

**Steve:** No, okay. In my notes I have a better picture.

**Leo:** All right, good.

**Steve:** Yeah. Anyway, so the original machine had three of these rotors that would be inserted in the machine at any one time from a set of five. So there was a set of five that were designated with Roman numerals, you know, I, Roman numeral II, Roman numeral III, Roman numeral IV, and Roman numeral V. And so the part of the configuration of this Enigma machine was choosing from one of five for the leftmost rotor, one of the remaining four for the middle rotor, and one of the remaining three for the rightmost rotor. And there were 60 different ways, five times four times three, 60 different ways to set up, basically to choose the stack of rotors which would be used.

And so to make sure everyone understands this, what we have is we have from the keyboard, as you press a key, one of 26 wires is electrified. And that successively moves through each of these three rotors, going in one location, essentially emerging in a completely different location, where it then connects to the rotor to its left, going in that rotor and emerging in a different location, where it then connects to the leftmost rotor, going in it and coming out on the other side. So these are rotors because, as the name implies, they can rotate. And this is part of the magic of Enigma.

Now, we've got, as this comes out on the left-hand side of this stack, we now have 26 connections. What are we going to do with those? Well, what was very clever was what Arthur designed, but it was also one of the several fatal flaws in the design. And that is that the final stage to the left was called the "reflector." It, too, had 26 connections to match the 26 connections coming out of the left-hand side of the leftmost rotor. And its 26 connections were connected, again in a haphazard fashion, to others. So that is to say it had 13 wires with each end connecting a pair of the connectors on its face. Thus it reflected. The electricity, the current would go in one of its 26 connectors, having been well scrambled on its journey from right to left through these three rotors. Then it would emerge from a different one of the 26 and make its way back through the same stack of rotors back to the right. Whereupon one of 26 lights would be illuminated.

So now you get a sense for this. So you press a button, and electric current flows through one of 26 wires through these three, the stack of three rotors, which internally each are scrambly. And when it finally emerges, it is sent back, based on this reflector, through the scramble again, to emerge. And that current then lights one of 26 lights. So basically that is sort of a static picture of Enigma. Which means that, as you're pressing buttons, very difficult to predict lights are lighting up.

But there's one interesting characteristic of this design which I refer to as a "fatal flaw." And that is, you'll note that there's no way a letter can ever encode to itself. That is, if you press T, and one of those 26 wires happens, nothing that can happen in there is able to send a current back out the same wire it came in. Well, that was very clever because it allowed for a simplification of the design, but it was cryptographically a flaw because it gave a big clue to decrypting this. No letter could ever encode to itself.

Now, the other thing we have at this point is not something very complicated. That is, essentially, we've gone to a huge amount of effort to create a very simple substitution

cipher, which is, and we've talked about those before, like the encoder ring. You take the alphabet, A through Z. And then you could put below it randomly arranged characters A through Z. Now you look up the plaintext on the top row, and the corresponding ciphertext is on the row below. So that's a substitution cipher. We know that they're not difficult to crack because, if you know anything about the language which the plaintext is written in, a simple frequency analysis of enough ciphertext will quickly identify the letters that are most common, the letters that are least common. You can then start making guesses about the ones in between based on your knowledge of the language, and it's easy to crack.

So that's called an "alphabetic cipher." What we want is a polyalphabetic cipher, that is, one where the mapping between the plaintext and the ciphertext is continually changing. And that's the first brilliant part of the Enigma. And that is, when you press that key, whatever key you're wanting to encrypt, not only does an electric signal go to one of 26 contacts to begin making its way through, but the act of pressing the key rotates in sort of the manner of an odometer. It rotates the rightmost rotor to its next position. It steps it through one of 26 steps.

And there is a moveable ring on the left-hand side of that right-hand rotor. It's also known as the "fast rotor" because it turns the fastest in sort of an odometer sense. This repositionable ring can be also in any of 26 settings. And there's a notch in it which determines at which position of the fastest rightmost rotor a cam will drop, allowing the second, that is, the middle rotor, and it's called the "middle rotor," to then step. So you have a situation where, once every time around, with the fastest rotor, the second one will then engage, and it'll advance. And when that happens is set by a moveable ring on the face of that fastest rotor. And similarly, the middle rotor has a ring also which determines when the leftmost rotor rotates. And then there's one extra little kink which causes the machine not to operate exactly like an odometer, which is that some of these cams can multiply engage and rotate more than one wheel in some circumstances.

So it's a little trickier than just that. But the concept is, every single - and this is the key. Every single time you encrypt a character, the cipher changes. The rotor steps to its next position, completely changing the scrambling, that is, the mapping between the A through Z character set to a very different A through Z character set. And thanks to the fact that you've got multiple rotors, the mapping is complex, and it does not repeat. Now, the Germans understood that some information was leaking, and so they limited the length of the messages set with Enigma to about 250 characters. They understood that, if this thing ever did repeat, then that would create a cryptographic opportunity which they wanted to prevent.

So now there's one last part of this, and that is there was a plugboard on the front of the machine. So before I leave, we have the notion of, from a set of five rotors, you choose three, and you choose what order they're going to be in. Then you choose also the starting position of each of those. So you dial those to a starting position, where the message begins, as the settings. Then finally this plugboard, that intercepts the signal going between the keyboard and lights and the scrambler stack and performs a static swap. So with this plugboard you plug one into T and one into W, for example. Then if you didn't have that in place, and you pressed T or W, the signal would go directly through to the T or W wires at the beginning of the scrambler stack. But with the plugboard cable plugged into the T and the W holes, they were exchanged. So this was both the keyboard and the lights had their wires swapped, so T would become W and W would become T.

And it turns out that the Germans felt that this provided a ridiculously large number of connections. The Enigma machine had 10 wires that could be plugged into any of 20 of

the 26 available holes in this plugboard. Now, it's interesting, too, because you might think, well, wait a minute. Why not 13 cables because there are 26 holes? It turns out that's actually less secure. If you have 13 wires, there are fewer combinations than if you only have - actually the optimal is 11.

**Leo:** That's clever.

**Steve:** Yes, yeah. For some reason, the Germans chose 10 wires. Maybe one broke at the beginning, and they stayed with 10, or they thought 10 was an easier number. It's slightly better to use 11. But then 12 gets worse, and 13 actually is redundant because, if you have 12, there's only two left open. And so you know that those are going to get swapped because there's no other choice.

Okay. So we end up with lots of combinations. I talked about how we have 60 different arrangements of the rotors. Then we have the rings, which on the rightmost rotor and the middle rotor give 26 settings each. There actually was a ring on the third rotor because that third rotor might be in a different location, but it had no practical effect because there was nobody for its ring to control the rotation of. So that is 26 times 26, which is 676 possible combinations of two 26-position rings.

Then we have the initial rotor settings. Each rotor, A through Z, could have its initial settings set. So that's 26 times 26 times 26, which is 17,576 possible starting positions for the rotor stack. The plugboard, oh, my lord. That had by far the most complexity of all. On the order of, let's see, what is this number I'm looking at? It's 150,738,274,937,300. I have it here in the notes. But, I mean, a ridiculous number of possible ways of swapping letters. So altogether 60 times 17,576 times 676 times the plugboard combinations ends up with about one times 10^23 possible combinations. Which in English is 100 sextillion, or 100,000 billion billion combinations.

And, I mean, think about that. From a simple mechanical device back in World War II era. I mean, just incredible crypto. And very, very clever. Basically, a polyalphabetic cipher, which could be set to an initial condition with a vast number of combinations, and every single character you enter changes the alphabetic mapping from plaintext to ciphertext in a very complex way that never repeats during the course of your sending even a very long message. What, 26 times 26 times 26 - no, that's that 17,576 number. It was slightly less than that because you could have multiple rotors changing at the same time. So the rotor positions would come back to home. But still, Germany never sent more than 250 characters in a single message, so they prevented disclosing too much information at once.

So the way this operated in practice - so I should also mention that what this was, this was a system, it was a classic keyed cipher. We've talked about, obviously, keying ciphers. The concept of a keyed cipher is that, even if you know everything there is to know about the design of the cipher, that still doesn't weaken it. In this case, actually, it does a little bit. But the theory is that it's the key that matters. You could completely disclose the cipher mechanism itself, and your privacy of what you encrypt remains intact. And we know of stories through the course of World War II where U-boats would get sunk, and people would go down and recover the Enigma machine from it. Or they would pull the crew off the U-boat, and the scuttling charges hadn't gone off yet, or failed, and again they'd pull an Enigma off. So the Allies had Enigma machines. We weren't limited to the ones that the Polish mathematicians reverse-engineered. But having the machine didn't help us that much because it was a keyed cipher. What we needed was the key.

So what the Germans created was a codebook, not of the cipher, but of the key. And they would arrange to get the codebooks to everyone who had Enigma machines. The book had a long enough life; three months was what I encountered in my research. It had three months of codes, of Enigma machine settings, by day. So on a given day, by month and date, the Enigma operators would look up the rotor order. So this was settings for the day. So you would grab the No. 4 rotor and the No. 1 rotor and the No. 5 rotor in that sequence to build your stack, and then insert that and lock it into the Enigma machine. Oh, but before doing that, you would set the rings on the rotors, on two of the rotors, to 23 and 2. Some of the rotors were numbered 1 through 26. The other ones had A through Z on them. I think they felt that numbers would be clearer, just so as not to confuse the other instances where letters were being used.

So you'd set the rings on two of the rotors, insert and lock down the stack. Then you would - the codebook had letter pairings, for example, AR, KT, WM, LC, XD and so forth, 10 pairs of letters, which were the plugboard settings. So you would take your 10 plugboard cables and plug them into those letter pairs in order to perform the static swaps between the alphabetic keys and lights and the substitution cipher stack. And finally, the last piece of information in the code book was the starting positions of the rotors, like TXM. So you would dial those three numbers, or, sorry, letters of the alphabet so that they showed through little windows in the front of the machine, and now you were ready. So that would be the setting for the day.

Then the instructions to the operator were come up with three letters at random. That's the per day key. And we already know how many combinations there are of those, a ton. So now the operator comes up with three letters at random which the operator enters into the machine and gets three cryptographic letters out. Then he does the same thing a second time and is going to get a different three. Because, remember, every time you put a letter in, the cipher changes. It advances the rotors from the right one, which is turning fastest, on down to the middle and the left one. So you're going to get, even though you put the same three characters in a second time, you get three different outputs. That is used to prefix the message that is sent. And then the operator sets the wheels back to the random setting of the three that he chose, and then encodes the message. So essentially we have a day based on the code book, and then we have a per-message encryption based on three random letters which the operator prefixes the transmission to.

Now, one thing I didn't say, which may be obvious from the design, but this is the other very cool thing. Remember I said that that reflector represented a flaw in the design because no character could encode to itself. But the motivation for that was extremely cool. And that is, think about it, if you - say that you press T, and the wires run through in both directions and come out Q, and light up the Q light. Well, similarly, if without changing the rotors you pressed Q, the electricity is going to go in the Q wire and come out the T wire. Which is to say that the Enigma machine creates a mirror image. That is, all you do to decipher a message is set up the exact same conditions as was used to encipher the message because the ciphertext will generate the plaintext in a mirror image of the plaintext that generated the ciphertext.

So on the receiving side, they've got the same codebook. They initialize their machine in the morning with the same settings. And then over the radio they receive a message. They're going to receive those first three letters which are the transmitting operator's randomly chosen setting for the message. They'll decrypt it, and they'll receive the original three letters that the operator sent. Then they're going to get three different letters, which is the reencryption that second time of the same first three letters. And they should get the same three plaintext out. So that was a verification.

Unfortunately, it was also a huge weakness. And it was exactly that protocol, the fact that the same three randomly chosen letters were sent twice, that allowed the very skilled mathematicians in Poland to crack Enigma initially. Just knowing that, they were able to analyze enough messages in order to reverse-engineer the entire complement of rotors, all the scrambly stuff going on inside, and build themselves an Enigma, which frankly is an astonishing feat.

The problem is Germany stopped, on May 1st, 1940, stopped the practice of duplicating those first three letters. And they made some other changes, and that completely broke the Polish cryptographers' solution. And it was that characteristic that Turing and his group understood was too fragile, so they never based their decryption on the assumption that the Germans would always be doing something that could be so easily changed. And in fact, that was the right choice to make because Germany ended up no longer prefixing. Basically it was unnecessary for them to do it.

So that is how Enigma works. And the Enigma machine design did evolve over the course of World War II. We went from five rotors, from which three were chosen, to eight rotors, from which four were chosen. So we got more rotors and a fourth rotor in next-generation design. But otherwise, that was the system that functioned.

So in order to crack this, there were a number of interesting clues that came out of this. The fact that the Enigma machine could not encrypt a character to itself was crucial because what Turing and his group realized, and we did see this in the movie, was that, if they knew some of the text, that is, some of the plaintext that had been encrypted, that gave them a clue. And in classic cryptography, this is known today as a known plaintext attack. They couldn't choose the plaintext to encrypt, but they knew what it was. For example, the German messages were often signed, essentially, or ended in a "Heil Hitler."

Or one of the most reliable first-thing-in-the-day broadcasts was the weather. The weather from certain locations, certain stations, was broadcast. And that broadcast had a heavily scripted, repeated format. So they were able to, in fact, it began with the phrase "weather report." So they knew the German for weather report at the beginning of the broadcast. They knew when it was going to be sent. And the Germans, if nothing, were very punctual. So the punctuality of that, because they also had the time and date that was in the message, so that gave them a chunk of stuff that they knew. So they would take that plaintext and write it out and then lay out the ciphertext of the weather report.

Now, one thing that they knew was that, by nature of the design of the Enigma machine, no letter could encode to itself. Well, if you have enough plaintext and enough ciphertext, and only an alphabet of 26, it turns out there are lots of collisions, that is, lots of places where somewhere in either of those strings, the same character will be above and below. Well, you know, then, that that's the wrong alignment of the plaintext with the ciphertext. So you shift it over one. Now three other places collide. Whoops, that's impossible. You shift it over again, oh, there's - nope, two places collide now. Okay, that can't be it. So you shift it again. So basically they would shift it until they would find a setting, that is, just an alignment of ciphertext and plaintext that didn't break that one simple rule, that the text could not encipher to itself.

Now, if you think about the design of the machine, it didn't have to have a reflector. The reflector was convenient. But if instead those 26 wires came out of the left-hand side and went to light bulbs, then you would have encryption without reflection, and letters could encrypt to themselves, and the No. 1 most useful hint would have been lost. It's true that then decrypting wouldn't have been a symmetric operation as it was the way the machine was cleverly designed. You would have to arrange to swap the wires, the input

and output wires, so that you were basically sending in on the left and coming out on the right, rather than always - rather than doing both over on the right-hand side. But that could have been done and would have made a vastly stronger, though slightly more complex and less easy to use, system. As it turns out, that was the Achilles heel, the fact that they were able to eliminate so many possible ciphertext and plaintext mappings, simply because the same character could not encrypt to itself. That created a ciphertext and plaintext test.

Now, the way this crazy machine, the bombes, operated, that Turing and his group designed and built, was essentially it found and rejected many other impossibilities. What the machine was trying to do was it was trying to determine the day's settings from one or more captured pieces of crypto for which they could guess the plaintext. And in looking at the design, I decided, okay, I'm not going to be able to explain it because it is really, I mean, this is why Turing was a genius. This thing - and to have built this in the technology of the era, basically with wires and rotating cams and contact, I mean, like, no electronics. Basic electricity.

And he was able to come up with a system which would test and reject possible settings of the Enigma machine because it turns out that, the more you think about it, when you think about this network of wires, not only can a character not encrypt to itself, that's the most obvious and easy observation, turns out when you sit down with a pencil and a lot of paper, many other impossibilities fall out. And so it was possible to automate the process, which is what Turing did, of rejecting almost all daily settings or potential settings for the day. And then when this thing would stop there would be, based on the limited input, it would be a possible setting for a Enigma machine. They would then run over, see if it made, you know, impose that setting, see if it made sense for a different piece of that message or some other message. And if not, they would keep going and wait till it found another possibility. And that's how Enigma works and how Enigma was cracked.

**Leo:** Yay. It's no longer an enigma. This is all very cool. And while you've been talking I've found a number of interesting Enigma emulators.

**Steve:** Oh, yes. The 'Net has a bunch. There are both hardware and web pages. And one is written in Java, and the guy wants to recode it in JavaScript for browsers but has not had a chance to do it yet. But, yeah, there was a Kickstarter project. There are beautiful, there's like a thousand-dollar, like in a wooden box, the whole deal. You're able to - basically they are 100% faithful reproductions of the exact Enigma technology, such that if you had the settings and German ciphertext, and you spoke German, you could figure out what they said.

**Leo:** Someone named Louise Dade, who provided the graphics that we were using, has got one online. And then this is one for Windows, and I'm very tempted to download...

**Steve:** Yes.

**Leo:** ...that beautiful graphic. This is...

**Steve:** It's just gorgeous.

**Leo:** It's from the Dutch coder. But it requires Windows, any version of Windows, including the most modern. So this looks kind of fun. I'm going to put this on my Windows machine. I don't happen to have one here, so I couldn't run it. But isn't that nice? People love this. It's a great subject.

**Steve:** You know, and I think one of the things that's nice is it is accessible crypto. I mean, I just described how it works. And it's not that hard. It's clever. And, boy, when you think about what they did at the time, like in World War II era technology. Basically just all they had was buttons and lights.

**Leo:** And radio.

**Steve:** And wires.

**Leo:** And radio. And is this - I guess this is symmetric key crypto because you have to have, at some point, exchanged these sheets that tell you what the settings are. Otherwise you can't decode. And that's of course the flaw of ciphers.

**Steve:** Well, yeah. It's keyed. It's keyed crypto. And it is sort of, I mean, it is symmetric. It's sort of like an XOR where you put in...

**Leo:** Well, symmetric in the sense that you have to exchange a key with a recipient before you guys can talk.

**Steve:** Yes, exactly. So you have to have the code. In fact, one of the things that would happen is, since these code books were issued every 90 days, sometimes the Allies would capture a code book shortly after issue, and they would get free settings for the balance of 90 days. If they caught it shortly after issue, they'd get almost three months' worth of settings. If not, they would get however much time was left on the code book before it was reissued. But so those were windfalls, when they would get the codebook. But what was cool was, because it's a keyed cipher, even having a machine didn't help you. We had machines. But we still had to build crazy bombes in order to decrypt them.

**Leo:** And you mentioned this, but in a very primitive way, like having a Captain Midnight decoder ring, Enigma being the ring, a very fancy version of the ring. But you still had to, ahead of time, exchange the settings. Otherwise you couldn't decipher each other. And that's why public key crypto is so remarkable, because that eliminates that exchange and eliminates one of the big holes in crypto. Oh, and we now have - John has brought me a sign so that, if I ever lose your audio, I have something to hold up. Really, really interesting stuff. Fascinating, actually.

And I hope people get to see - I've seen an Enigma machine in person. Many museums have them. There are quite a few extant still, probably a dozen. But I'd

love to go to Bletchley Park someday. And they have quite a nice online site describing a lot of this. Very different from the movie, I'm sad to say. Now that I've seen all the discrepancies in the movie, it takes some of the joy I felt in watching it. It's a wonderful movie, but not a piece of history, unfortunately.

**Steve:** Yeah, yeah.

**Leo:** Thank you, Steve. I appreciate it.

**Steve:** My pleasure. And we've got a nice podcast, and we've got a Q&A next week.

**Leo:** So go to GRC.com/feedback. That's where you can ask those questions. While you're there, get SpinRite, world's finest hard drive maintenance utility, and all the freebies Steve gives away all the time. And of course 16Kb versions of the audio of this show, plus transcriptions. I saw that Elaine Farris, your transcriptionist, has put out an EPUB book of our year-ender, or actually it was the New Year's, first show of the New Year, wasn't it. Or was it the year-ender? No, it was the year-ender that we did right before our New Year's Eve show, where you had all of the exploits in the year.

**Steve:** Ah, okay.

**Leo:** Did you know she did that?

**Steve:** I did see that, yeah.

**Leo:** Yeah, so that was nice. And I don't know, I presume it's free, I don't know. Anyway, I've tweeted it.

**Steve:** Oh, I'm sure, yeah.

**Leo:** And but of course all this stuff is free from Steve's site, GRC.com. We have high-quality audio and video at our site, TWiT.tv/sn, and of course on all the podcatchers. Security Now!, one of the oldest podcasts in the world, so it's easy to find on iTunes and everywhere else. And I'm happy to say that Mark Lane has released a new version of TWiT Pro for Android that has v3.0. So we have - and by the way, these are all independently designed and developed by our, you know, volunteers who just do this for fun. But TWiT Pro just got updated. And of course Craig Mullaney does a great job with the TWiT apps on iOS. He also did a Roku app for us. So lots of ways to watch, even on Windows Phone.

So please do watch and participate and subscribe and be back here next Tuesday, 1:00 p.m. Pacific, 4:00 p.m. Eastern time, 2100 UTC for a great Security Now!. We'll

see you later, Steve.

**Steve:** Thanks, Leo.