Corellium

Video Webinar

Episode 3: *Mobile Vulnerabilities Exposed: Data in Transit*

Full Transcript

Brian Robison: Good morning. Good afternoon everybody. Thank you for coming in and taking the time out of your busy day to join us in our webinar today. My name is Brian Robinson. I am the Chief Evangelist here at Corellium and I am joined by Steven Smiley, one of our esteemed Corellium researchers, and we are in part two of a series talking about mobile vulnerabilities with data in transit. Good morning, Steven.

Steven Smiley: Morning, Brian.

Brian Robison: Awesome. Steven's going to come on in a few minutes and take the bulk of the conversation today and talk about mobile app pen testing and how to do some stuff in data in transit. And then I will also come on towards the end assuming we have some time and show some live demos and things like that. So it's going to be a lively conversation today for you and hopefully you're going to get some great information about some of the things that we do and what we basically empower you to do as well with the Corellium platform.

(00:01:14)
So a little bit of housekeeping before we get started. As you know, you are on mute, so all attendees are on mute, but we do encourage engagement, so please use the zoom Q&A tool to ask us questions. Please do that throughout the webinar. We'll both keep an eye on it and try to address those questions as they come up so that you get as much information out of this webinar as you possibly can. That's the goal. The session is being recorded, so if you've come in after we started, the session is being recorded and we will be making it available to you very shortly. And again, as I said before, this presentation consists of some slides, discussions, and then live demos as well. So there's a good mix of content and information available for you. So I would like to take a few moments and talk about how, as I mentioned early on, this is part two of a series of webinars.

(00:02:12)
We started in September with a Corellium 101, kind of an overview of the platform, and then Steven did part one of the Vulnerabilities Exposed: Data at Rest. So last month we looked at data at rest and the vulnerabilities there. Those two webinars are on demand. You can watch

them at any time. Just go to the website. You see here, our webinar series is at Corellium.com/events and you're able to watch those essentially right now. The third one over on the right hand side is essentially today's webinar that we're watching now live and it is the data in transit. And then right before Christmas break here, we're going to do another one, which is part three of the series, which is going to talk about reverse engineering tactics and techniques. So we do look forward to seeing you on that webinar as well, and it will be listed once we get it all set up. It'll be listed on that events page as well for you.

(00:03:15)
The other thing I'd like to talk about—and sorry for the long URL at the top, again, this will be available to you afterwards—but basically we have a new set of trainings that are available if you are interested in learning, taking your pen testing knowledge or your vulnerability research knowledge and moving that from physical devices into the Corellium platform. We do have trainings available and they're on our website. If you go under *platform* to *support and training*, you'll get to see things like the mobile app pen testing course, which is offered virtually as well as on site with and without labs, all those kinds of fun things like that. And you can get more information. You can review the syllabi that are available and there's training available for pen testing as well as iOS vulnerability research if you're interested in doing that level of things with the Corellium platform.

(00:04:15)
So new trainings are available. We also have on that site what we call our Quick Start Training, which is offered monthly. And basically it is a quick crash course, two and a half hours maybe, of the platform. A quick overview of the platform, how the different tools work, all the different things that you have available as well as a crash course into vulnerability research in iOS and mobile app pen testing. Kind of an overview there as well. Also API. So that's a course that repeats every single month. It's the same content, so you don't need to come each month, but check out that support and training page and you're able to attend those trainings. Those are actually free trainings that are available from us. So please check those two things out. So why is—and I just wanted to cover this very, very quickly before we move into the actual content—but why is this thing we call mobile security research so difficult?

(00:05:16)
The reason it's difficult to do is because there are three major limitations on what you can do and how you have to work with it. The first thing is devices, right? We are limited on the types of devices we have access to that we can get. If you're doing vulnerability research on the latest, greatest version of iOS, just go buy the device and off you go. But if you're doing mobile app pen testing specifically, you're looking for specific devices, and the reason you're looking for those specific devices is because you're looking for specific OS versions. And now why would you look for specific OS versions? Well, because to do mobile app pen testing and even vulnerability research correctly, we have to have root-level access to these devices so that we can do things like investigate data at rest files that aren't encrypted data, that's not encrypted, things like that.

(00:06:16)
And to do that, we need jailbroken operating systems. We need rooted operating systems. So that relegates us to finding older devices that haven't been upgraded yet, trolling the third party used markets on Facebook Marketplace or Craigslist or something and meeting shady characters in mall parking lots to exchange these devices—kind of feels like drug deals almost. And so those are the two major reasons why this is hard, because it's hard to get access to the proper level of access that we need on these devices to do the testing that we need to do. The third element is time. Now that we have these devices and maybe we have a small stack of them, if you're only one person doing this type of work, that's usually not too bad. But if a global team—or if you're got people in multiple locations—now you're shipping these devices around, you're doing all this kind of stuff, and with physical devices, especially iOS, the jail breaks aren't necessarily 100 percent reliable every time you boot it up.

(00:07:21)
If it's a tethered jailbreak, you've got to redo the jailbreak; you've got to redo all this stuff every single time. So you end up sacrificing time that you would normally spend testing to just managing devices and dealing with shipping and dealing with losing devices, and all these kinds of things begin to kind of weigh on you during your testing. And there's a cost to that and really it's a cost of distraction in my opinion. You're being distracted from doing your job, which is testing applications or searching for vulnerabilities. Now, what does Corellium offer? So we basically solve these three issues by giving you an actual virtualized platform for iOS and Android. So there are emulators and there are simulators and there are other things that are out there, but we are a true virtualized platform. So we don't run these operating systems on x86 chips.

(00:08:23)
We run them directly on Arm-based chips. So there's no translation layers or anything out there. And we essentially have the only Type 1 hypervisor for the Arm platform. So if you think of an x86 analogy, we are VMware, but we are VMware for Arm and we're not doing Arm operating systems like Linux or Windows or things like that. We're doing mobile device modeling. So the virtual machine that gets built is actually a model of a specific device including sensors and telemetry and all these. And there's multiple elements that get modeled into these, and we run directly on Arm servers. We're available in the cloud as well as on-prem. And a lot of our customers actually choose the on-prem appliances mainly because they're doing research that needs to be all done offline and things like that, or tested with an internal enterprise grade network and things like that.

(00:09:30)
So basically we allow you to virtualize iOS and Android devices, run them simultaneously, do testing. Probably the most important thing that saves time is now that these things are virtual machines with the ability to snapshot them and restore them nearly instantly back to a known good snapshot. All of the devices are already jailbroken because we actually are a virtualization

platform. We don't rely on a vulnerability to be in the operating system for us to apply jailbreak. We control that boot process. We actually insert what we need into that boot process. So all of the iOS versions are jailbroken, including the latest and greatest 16s that are out there right now. We provide instant jailbreak access to all of those devices no matter what version you're trying to run or what type of device you're modeling, even all the way up to the latest 14 Pro Max.

(00:10:32)
So you're able to save those snapshots, restore them—you can even clone them and share the snapshots with other users in the system. So really it's that virtual platform that is the major time saver. So we're going to see the platform today. Again, we're not going to do a full on demonstration of all the different tools that are in the platform. If you want to see that go back to the 101 demonstration. But with that, I'm going to go ahead and stop the sharing and hand it over to Steven who's going to take over and walk us through today's content on data in transit. Steven, please go ahead and proceed.

(00:11:15)
Steven Smiley: Awesome, thanks Brian. I will go ahead and share my screen now so everyone can see the content. OK, so as Brian said, we're going to talk about mobile data in transit today. So one thing we're not going to talk about is we're not going to get into depth with the actual vulnerabilities. You should be testing for mobile. There are tons of great resources out there that can kind of help you if you're kind of looking for that information around. If you go look at the mobile security testing guide or the OAS guides for API as well, tons of great resources when you get into the actual vulnerabilities and looking for that sort of stuff. What we're going to do today and what we're going to talk about is data transit in general. We're going to talk about how that is such a key component within a mobile pen test.

(00:12:04)
We'll look at things like data leakage and we'll go over kind of a lot of details around that—what it is, what are some of the common components of that—and we're going to look at some of the best practices when it comes to data in transit for your mobile apps, what you should be looking for, what the best practices are from a dev side for your teams. We'll look at various security controls that can be added to kind of protect some of this data. And you guys probably heard of some of these certificate pinning and things like that. We'll look at a couple different options for bypassing those security controls from a pen test or AppSec side of things that may be necessary. And we'll kind talk about that more in depth, why that's necessary to at least understand. And then we're going to talk about app transport security and Android Network Security Config, some of the defaults that can be implemented right away, and some things to look for in there as well to see how your data is being protected.

(00:13:06)
We'll move on to generally talking about pen testing, mobile applications, how that works with all the things we previously talked about, the security controls, and all that sort of content. And then

we have a couple demos as Brian mentioned. We're going to go through the platform as well and how the Corellium platform can help you along if you want to move on to third party tools as well with your virtualized device, how easy that can be, and how you can capture some of that data. And as Brian said, feel free to use the Q&A option throughout this. Ask some questions. We do have some extra time I think in this, so feel free to ask questions. We'll stop, we'll answer them, and we'll make sure everyone has a good time.

(00:13:50)
So mobile data leakage—what is it? Well, essentially, if internal or sensitive data is made accessible to people that are not authorized to see it, this can obviously lead to greater attacks and things like identity theft, reputational risk, and potentially advanced attacks depending on what data is being leaked. Now there are a large number of mobile applications that still are leaking data, and that could be anything from just responding to too much data in a request. If you're sending requests, especially in a financial application, maybe the application is responding with data that the user doesn't need to see that is providing another attack vector potentially relating to that identity theft. Credit card numbers, maybe they're not being masked correctly. You have things like URLs as well. If you're putting parameters in your query strings, your URL query strings, you're going to see data that way potentially if you're intercepting traffic, which we'll obviously talk about as we get through this.

(00:14:55)
But there's a handful of ways that data is coming out there, and it can be very simple data—like we're talking about credit cards and things like that, financial data. But this could be as simple as somebody sending a malformed request where the servers respond with the framework versions, any sort of backend systems, and the versions they're using. And that sort of stuff can lead to more advanced attacks where attackers now know what infrastructure you're using, can take that information, and build more advanced attacks and more targeted attacks on your systems. So it all kind of plays together. So how can you tell your application is protecting your data? Well, there's a handful of different ways, and this is kind of up to every pen tester and everyone is a little bit different and every application is a little bit different in terms of what data is being transmitted, what needs to be looked at, and things like that.

(00:15:49)
But essentially, you obviously need to be reviewing the traffic being sent from the client to the backend server and vice versa: what's coming from the server back to the client. All that traffic should be reviewed. Again, for things like data leakage and for potential vulnerabilities, you need to look through traffic being sent to third parties. There were a ton of cases out there, and you can search this as well, where either too much data was being sent to third parties or analytic companies. The data wasn't anonymized, it was actually sending personal information, maybe location information, things like that. So you need to figure out what data is being transmitted out there as more and more applications start to use these third party SDKs and these analytic services. You need to check for data encryption. I've seen a few applications, this doesn't happen quite often, but in the actual request itself where they're encrypting data, that is

a possibility that's more advanced. You'll see it in some potential financial applications where they're really trying to protect user data, but for the majority, you're kind of looking at the responses and you're looking at the data coming back from the server to see if things are being masked correctly. Is encryption being used when it's needed, and what level of encryption is being used? Obviously, if you use something like MD5 or you're encoding stuff, that's not going to protect you on the same level as using a SHA-256 or something like that.

(00:17:20)
Session tokens for one: obviously using them, making sure that all the data and all the requests being sent are associated with a user. You have those timeouts, you have those tokens that are using industry best practices, you're using JWTs, JWEs, so that data is kind of being protected. You're not allowing additional attack vectors, IDOR attacks or submitting requests to get data that they shouldn't be getting. Things like that. And then obviously as we start to talk through this kind of lesson or webinar we have, we're going to talk about the security controls, but analyzing those security controls that are implemented for your application. How are they implemented? Are they easily bypassed?

(00:18:05)
Is that protecting your data to the next level when it comes to attackers? The more security controls you have, even if they can be bypassed, they're still protecting you to a certain level. And unless someone is really targeting your application, someone would much rather go for an application that doesn't have the security controls. So they do bring your level of protection up. And then as I just briefly mentioned, the URL query strings, some people in mobile applications are still sending data in that URL as part of the request, and those things can be easily captured as part of a man-in-the-middle attack or just easily seen and potentially cached in the actual local data directories if you're using web views and things like that. So there were potential ways to find that data a little bit easier than putting it inside like a post request. We did get a question as well. Can we wait to answer it or, Brian, do you want to take a look at it? In the meantime, I'll move on to best practices and I will come back to that question.

(00:19:12)
Brian Robison:
Yeah, we can answer that question in a little bit.

Steven Smiley:
Yeah, sure. Yeah, we'll hold onto that. No problem. So we're going to talk about the best practices. So a CTP is really the foundation of all data exchanges on the web. I think everyone is kind of familiar with that now. Everything over HTTP is plain text, but a lot of mobile applications, because they believe they're a little bit more protected the way they're implemented, a lot of them still use HTTP. And it's not to say that you shouldn't be using it. There are use cases where that is very valuable. If you have a marketing application, you want to pull down some images, even in potentially some financial applications. There are certain use cases where not all traffic will be HTTP of course, because you do need to protect that data, but

there are cases where you're pulling down images or things like that.

(00:20:02)
So there can be a mix in your application, but it's understanding what data is being transmitted over that. Obviously HTTPS uses TLS to encrypt the information exchange in order to prevent unintended exposure. So, #1: obviously financial data should be any sort of sensitive information. Logging information should be sent over HTTPS. There's not a ton of applications that are still really sending, at least that I've seen, potentially sensitive information over HTTP, but it is a case, right? So, #1: that is just a best practice. You should always be using HTTPS for those sensitive transmissions of data. Now is that enough to protect your data? Well, it really depends on your application and nobody can really tell you “yes” or “no” to that question. But there are other options available that can kind of take your network level security to the next level. And things like certificate validations, certificate pinning, certificate transparency, and then the two built-in ones for iOS and Android would be App Transport Security and Android Network Security Configuration, making sure you're utilizing those to the best of your ability and you are protecting your data in the best way possible.

(00:21:25)
So now we're going to talk about those more in depth just so people can kind of get an understanding. I think these terms get tossed around a lot in the mobile community and maybe not everybody fully understands what it is and what is being protected and what the risks are associated with that. So, certificate validation: it's really the process of validating the contents of the certificate that is used when making a network connection without proper validation. It obviously leaves your application vulnerable to man-in-the-middle attacks. Now with certificate validation, it's slightly different than certificate pinning, which we will explain next. Certificate validation focuses on the user level instead of a trusted or root certificate, which certificate pinning uses. So in Android—they actually maintained a list of preselected trusted CAs. So by default, after Android 7 when this was actually implemented, if you're not in that list, a connection will actually be rejected.

(00:22:25)
So if you actually create your own cert and you put it in the user store, not the trusted store for Android, it will actually typically block you. But now, obviously, if you go to a trusted CA and you get a certificate and you're using that one to intercept traffic, you'll kind of be allowed to do that. Now, user level certs are much easier to add to a device than trusted. If anybody has done any pen testing on the later versions of Android, it can be difficult to get a cert and put it in there and then move it to the trusted store. There are some packages that can kind of do that, or you can remount the system and kind of move them. It's not the easiest process, it's not the quickest, but if you're adding a user level cert, it is very easy to do.

(00:23:18)
You just need someone to basically click it and accept it so they download it. You can send it in an email, you can send it as an attachment, a link, someone could download it and get pushed

to the device. So it's kind of very easy to be tricked into doing that, but it is an extra level. Somebody will have to get that cert in there. So it may not be the best protection, but it is a good one that's kind of implemented fairly well already through Android and iOS. And so it’s added, and then you could do additional development to add this in for different certs and things like that to protect any user that's being installed. Now for iOS, they just evaluate the certificate trust based on their policy. So it's basically the same as Android in terms of the preselected CAs and the certificates coming in. Now for iOS, for the certificates getting installed that do require user interaction: If you've ever done it, you go download a cert, you basically go into settings, and then there's a prompt that says, do you want to install this cert? Do you accept it? That will install it as a user cert, and then you can go into an additional menu and then give it root permissions as well if you want to get to that level. But there is that user interaction as well.

(00:24:33)
Certificate pinning. So certificate pinning is a process of associating a host with their expected X.509 certificate or public key. The certificate, as the name suggests, gets pinned to the application, gets put somewhere in the application, and only trusted certificates will be accepted. When you're making those connections for a man-in-middle-attack or something like that, all of the other connections will be dropped. I'm kind of curious—certificate pinning is a very big conversation point in mobile. Some people believe it's a vulnerability, some people do not, and it kind of has a lot of controversy. Brian, we have a poll that we're going to do. Just quickly, if you guys don't mind—I just want to kind of see where everyone kind of stands in terms of certificate pinning. Do you believe it is a vulnerability in itself? And then if you do, what severity would you consider it to be? I'm just kind of curious because the industry is kind of divided on certificate pinning right now. So I just kind of wanted to see what that is.

(00:25:43)
Brian Robison: We have a nice interesting mix coming in. When it's done here, I'll share it with everybody.

Steven Smiley: I was going to say, is there a way I can see? I can only think I can see the result.

Brian Robison: No. When everybody is done putting in their votes, then I'll end it and it'll show everybody the results so we can discuss them.

(00:26:01)
Steven Smiley: Great. Yeah, so we'll give everyone a minute there. Yeah, this is a really interesting topic. So, I really wanted to do this poll.

Brian Robison: Alright, and it looks like it's pretty divided. I'm going to go ahead and end the poll now and we'll share the results.

Steven Smiley: Oh, perfect. OK. We're literally divided right in the middle pretty much.

Brian Robison (00:26:23): Pretty much.

Steven Smiley: Yeah. Wow. OK, certificate pinning. So we're going to kind of talk about the logistics of this as we get into this, but just on the poll itself, do you consider certificate pinning to be a vulnerability? Now there's no right or wrong answer by the way. Nobody's wrong for saying “yes” or “no.” In this scenario, it was to drive a conversation. With certificate pinning, what we've seen a lot of times, depending on how it's implemented, there are a lot of risks associated with it. So there's a lot of companies and a lot of applications, a lot of public applications that do not choose to implement certificate pinning because they don't believe it provides enough protection. Now there are other cases we're going to talk about after this slide. We're going to talk about certificate transparency as well and how Google is trying to protect you as well to a certain level.

(00:27:16)
But it's interesting to see where people fit in this. I do think, just personally, certificate pinning should be implemented for certain applications. Obviously, financial applications or sensitive applications, I think it's a good added security control. It's a good protection. Typically when it comes to the vulnerability side of things, it's kind of difficult to say. It really depends on the industry and the company regarding how they rank those severities. But a lot of times you will come in on the informational or the low side of things just because—and we'll talk about this in a couple slides—when it comes to certificate pinning and even these other controls, they're fairly easy to bypass. It's not impossible. You should never consider this as an end-all it's going to work, it's going to solve our problems, our data's going to be protected. The data is still at risk, there is still potential to do that, but obviously it gets you to the next level where your average attacker might not be able to perform an attack. It's going to take somebody a little bit more advanced. So, it is kind of interesting to see, and I hope maybe that poll gave people some time to think about where they stand with it and where other people stand about it. And then maybe we could start a conversation if anybody has questions on certificate pinning, we can kind of talk about those as well.

(00:28:37)
So now we'll move on to just the certificate pinning logistics and risks. So there's basically three certificate options when it comes to pinning. So, you can pin the leaf certificate, which is obviously the most secure option. The issue with this is that the app needs to be updated every time the search expires. That can cause downtime. If some of your customers aren't updating their apps that often, they might lose access and need to go re-update. That might actually anger some people and might have them not use the applications. So there's all sorts of concerns that people have around that. Now you can get to the next level where you're pinning the intermediate cert, which is basically the next level. So any coming from the same intermediate is valid as well. So, what you'll see a lot of people do—I actually see a lot of questions come in. We'll kind of talk about those right off the side.

Brian Robison: Yeah, they're kind of a continuation, but basically what are some good ways to essentially bypass certificate pinning?

(00:29:40)
Steven Smiley:
Oh yeah, great question, because in two slides—and I don't want to spoil it—but in two slides we're going to talk about those bypasses as well because people do implement them and they don't necessarily make it easy for people to pen test those apps. So we'll talk about that for sure. But on the pinning, the intermediate search—so what a lot of people are starting to do is they're pinning the intermediate and the leaf certificate so that if your certificate needs to be updated quickly or you're coming up with a new cert, as long as it's coming from the same intermediate, you don't need them to update the application right away. So there is that kind of potential as well. And then obviously the least secure option is pinning the root certificate, but it will trust any certificate signed by that root authority. So it is still an added level of protection, but anybody could, depending on who the root authority is, just get a certificate from them as well, and then essentially they've bypassed your certificate pinning. S,o that's a really basic way to do that, to get past that. But there are other ways to bypass, so we'll kind of talk about that.

(00:30:56)
OK, certificate transparency: I briefly mentioned it on the last slide when we were talking about the poll. So it was started by Google basically after multiple attacks on certificate authorities. So there was a handful of attacks. You can look these up where there were CAs that got compromised, rogue certificates got issued. There were probably 500-plus, 550, something like that. And then those were found to be used in various men-in-the-middle attacks. And these were I think Iran and some various countries were seen doing this and then using those rogue certificates to perform malicious man-in-the-middle attacks. So this is kind of a different approach from certificate pinning. So a certificate is issued by the CA, a timestamp is then added to the certificate and it's uploaded to a public distributed network of log servers. And those servers use these Merkle trees, which are kind of interesting concept.

(00:31:49)
You can go look it up, but they make these logs publicly available, verifiable, and tamper-proof. So anybody can kind of go look up a certificate and you'll start to see, if you start to search this up and look it up if you haven't, it's kind of very interesting because a lot of companies, like Google and Chrome is starting to do this—they're taking all their certificates and they're uploading them to these trees and making them verifiable so that if any malicious certificate comes in, they're kind of going to be blocked. So these can be added into your mobile applications to kind of protect you. So it's basically saying that any verifiable cert, any added cert in here that can be verified is OK, but anybody else is not. So it's a public way to protect you versus certificate pinning where you're just pinning to an exact cert or a root CA or something like that.

(00:32:45)

This is kind of more trusted in the public and what everybody else trusts as well. So it is an option that people are going with. Aside from certificate pinning, people might choose to use certificate transparency to kind of protect them and then obviously make sure that their data as well is as protected as can be, right? You should never use any of these security controls as an end-all for your application. You should be looking at that data in transit and we'll kind of talk about that with Brian as well about how you can use Burp and our Network Monitor and things like that to be able to actually look at your traffic to find those additional vulnerabilities because that should be protected as well.

(00:33:27)
OK, so this was one of the questions: bypassing certificate pinning. When it comes to bypassing security controls, there really is not one answer. I wish I could just say go run this and it's going to work. That's not the case. I wish it was, it'd be super easy for all of us, but this can be an essential skill, especially when it comes to advanced pen testing. When it comes to pen testing, everyone is kind of different. Maybe we should have done a poll on this actually, but everyone is a bit different. Some companies provide you multiple builds where they're removing security controls if they added them of course, and they're allowing you to kind of pen test on an open build that doesn't have security controls and they're giving you a second one where you can kind of test those.

(00:34:12)
So that is kind of a use case, but there are potentials if you get into maybe consulting or some companies that want to test those security controls as well. There's a couple of use cases where you need to test those security controls or they give you a build with them enabled and they still want you to pen test the app. So you don't want to spend a week trying to bypass these security controls. You want to know the quickest way to do it. So we'll talk about a few of those and it's not an exhaustive list. There's a bunch popping up, there's new projects popping up, there's new repos like Cydia repos and things like that popping up, new Frida scripts coming out. So there's all sorts of options. So always keep looking. This stuff's kind of changing all the time, but when it comes to it, there's basically two options.

(00:34:53)
So we'll look at dynamic bypasses, which is the first one, things like Frida. So we can actually kind of look at the Frida console if you've never seen Corellium and the actual platform itself, we actually, for all those virtualized devices, we have Frida built in. So you can actually upload scripts, you can write scripts right in there and attempt very quickly to do things like this—bypass security controls or additional advanced analysis. So we can show that as well. But Frida in general, you're looking at scripts. Those could be custom scripts that you're writing to hook different methods that you found from reverse engineering. You could use the Frida CodeShare, which is available if you go online. Frida CodeShare,if you've never seen it, it's basically hundreds of scripts were put out there by the public people who do this stuff on a day-to-day basis. Not every script is going to work. Some of them are pretty customized, but it could give you a good starting point if you're trying to get into that scripting or trying to take a

look at a very specific use case for yourself, you could use that as a starting point and build off of it.

(00:35:58)
Objection—great tool built off of Frida. You could do a lot of different pen test techniques in there, but one thing they are good at as well is adding things like certificate pinning, bypass. They also have root detection bypasses, things like that as well. Keychain dumping, a handful of pen test utilities. But in terms of the data in transit, you could run any of those two scripts, Android SSL pinning disable or iOS SSL pinning disable obviously depending on what OS you have. And that will attempt to bypass some certificate pinning. Now I know that project is being updated right now. There's some people working on it to add additional measures and additional updates to those scripts, the pinning ones and the jailbreak bypass ones. So you will see some new stuff as well. So they're trying to keep as up to date as possible on some of that stuff, which is great.

(00:36:57)
It won't bypass everything, but it does a pretty good job. SSL Kill Switch for iOS, there's the GitHub project, so you guys can look it up if you've never used it, it's just a .deb file. You could just install it on your phone, pretty easy to install it just if you have a jailbroken phone. Of course this can actually be done on a virtualized device as well. But basically you just go into your settings, there's an option, you select the application, you enable the bypass, and it will attempt to bypass it. And for the majority I think it probably bypasses, I'm going to say 80 to 90 percent of applications that implement certificate pinning. So it's kind of a really good option for iOS if you don't want to have to use custom scripts and things like that. There is the Xposed Framework for Android, some scripts in there, it's not as maintained as often anymore. Kind of hard to install depending on your Android version or your device. So it's not really the go-to for Android, but it is an option. And like I said, there's just new options every day. There's new GitHub projects coming out on this stuff, new versions coming out, new scripts coming out. So always take a look at that.

(00:38:09)
So that's the dynamic side of things. We talk about the static side of things. These are mostly for Android. There are some iOS things you can do, but it's not as easy. This can kind of be a lot of trial error. It's very similar to if you've ever patched out an application, especially on Android, where you'd actually go in, look at the code, figure out what you want to either delete or change, you'd modify that and repackage the application and rerun it on your device. Now, sometimes the app will crash, sometimes there'll be other dependencies, other things calling it. So there is a lot of trial and error. It's just how it is. But here's just a couple of different options and look at. So the Android Network Security Config, which we're going to talk about after these slides, you can actually add your own certificate if one is specified.

(00:38:59)
So in the Security Config—there's an option which I'll actually show on a future slide, but you

can actually change it if it's pointing to a specific cert that they decide to pin that's locally stored in the binary. You can actually go put your cert in there, change the path and make that work with your certificate instead of theirs. You can remove pin digest, which is inside that security config as well, which could be looking for a hash. You can actually just remove that or change the hash, or you can use a lower version of Android. So the Android Network Security Config came in Android 7. If you have a device that's Android 6, you could potentially use that if your application supports it, and that would actually just bypass that implementation. If you have OKHTTP implemented, you can actually replace the certificate hash with your own. So you just grab the hash of your certificate, drop it in there, and you'll be good to go. TrustManager—they actually store the cert in this res/raw folder within the binaries. You can actually go in there and either delete that cert or add your own cert in there and it will start using that.

(00:40:26)
OK, so this is talking about searching the binaries. These are some great commands. You guys can look at this on demand and kind of grab these or take a screenshot of this if you want to. These are some great commands that you should be using as part of a pen test certificate hashes for one, it's kind of looking for all the hashes within the binary. And then if they are using one, if they're using a certificate hash instead of an actual file, you can go in there and obviously swap that out similar to what we showed in the last slide with your hash certificate files. Same thing, right? Go look through the binary. You're going to look for the .crtfiles, CRT, CER. You're going to look through the whole binary for that. If they're available, you can swap them out. And same with the Trust store Files.

(00:41:09)
So these are a couple ways to just search the binary and just replace it with your own right. It's not the same as doing dynamic. There's a lot of dynamic options out there, but sometimes those don't cut it and sometimes they just won't work. And if that's the case, potentially you could go look statically and try to find how they're doing it. You could also use things like JADX on Android and kind of just decompile the application, take a look at how they're implementing and potentially change the code if you have some knowledge of Smali code and things like that to be able to do that as well. But those are super customized because it really depends on the application, their obfuscation levels, things like that. So there's just so much to it.

(00:41:51)
iOS app, transport security. So ATS was added with the release of iOS 9. It’s a feature that works on the network layer to protect the data transmission between the client and server. So what does that really mean? Because it kind of sounds confusing, but essentially it means if enabled, which is by default after iOS 9, all HTTP connections will be forced to use HTTPS unless an exception is made in the info.plist. So always be checking the info.plist list to see are they using exceptions, are they actually disabling ATS? And a lot of applications will actually disable it or provide exceptions because maybe they need to pull resources from an HTTP site, they're doing some communication over HTTP. Whatever the case is, you want to take a look at that because that in itself is a vulnerability and then potentially could lead to a bigger

vulnerability when it comes to data in transit and looking at some of that unencrypted traffic. So the NSAllowArbitraryLoad is set to “True” is kind of what you're looking for for the majority. If there's exceptions, there's a handful of things you can look for and you'll see those exception domains as well. If they have internal domains that they want to use or certain sites that they want to run over HTTP, those will all be in there. So, you can kind of take a look at that.

(00:43:13)
So Android security config, it's very similar to ATS from iOS. The Network Security Config is just an XML file in which developers customize network security settings for Android. So we can take a look at some of the flags that can be set within that file and you'll see some of the stuff that's similar to what we kind of already talked about. One, cleartextTrafficPermitted—by default in Android 9 this was set to “false”. If not specifically set, the app is used on lower versions, is enforced as “true.” So, if you actually are running on an older version and that is not specifically set, it will actually be set to “true” and you will allow HTTP traffic within your application. So kind of a key thing to look at. This file can be just pulled from the binary. You can go take a look, see what's in there.

(00:44:03)
“trust-anchors”—they can define if your application will allow user or system certs. So you'll actually see in there if they're setting the trust anchor, it'll literally say user or system, and from that information you could basically tell: are they doing specific validation, are they potentially doing certificate pinning? If they're trusting system certs, are they looking at a specific cert? So they could actually be calling out a specific cert and a file within there, and then you can quickly go look and go see the cert and pull it out and change it to your own if you want to intercept anything like that.

(00:44:43)
So now just generally talking about pentesting data in transit in mobile, is it necessary to test data between the client and server? And the answer really is absolutely. Now there are some cases where applications are strictly marketing or very simple applications. There might not be much there, but you still should be looking at the data. Mobile has got to a place now you start to see more and more people get into mobile security. We have the mobile security testing guide, we have the MASTG. We have these standards out there that are really bringing up the level of mobile security. So in a lot of applications you might not see as many vulnerabilities on the static side or in local storage, things like that. That stuff kind of is being more and more protected lately. So where those vulnerabilities are starting to come from more and more in mobile is in the data in transit and people are starting to believe that just some of these security controls are going to protect them. They're the end-all and they're not. I showed a few ways to bypass them. They are pretty easy in a lot of cases to get past that. And it's like trusting your firewall for your network. You need to protect your entire network. You need to protect your entire application and the data that is being transmitted, not just put up a wall and say, OK, that's going to protect us. Nobody can get over that.

(00:46:08)
Corellium Network Monitor. It is a great tool for quick analysis. We actually by default—and I'm actually going to show a demo of this right after the slide—but you can actually bypass certificate pinning implementation on a large majority of applications and you can review your application traffic quickly obviously if it's 80 or 443. So, it's really quick analysis. You're not going to be able to modify the request and response like you can with a Burp or a man-in-the-middle proxy. But you will be able to see the request, the responses and the headers. So for quick validation, maybe data leakage, things like that, you can kind of look very quickly. And then we do have the ability to use third party tools as well really quickly with the virtual devices to get to that next level of testing. We can bypass the security controls.

(00:46:56)
Obviously we have the Frida integration as well. With the virtual devices you can use Objection pretty quickly on your own machine. So there's lots of ways to bypass those security controls. This should be done for pen testing. Like I said, not every time will people give you a bill that they're are disabled in. So you should always be testing and then again, request multiple bills. When pen testing mobile applications, it's always a good opportunity if you can, to get multiple bills. So you can not only test the security controls in one, but also fully test the data on the additional applications, not just trust that certificate pinning or certificate transparency or whatever they plan to use.

(00:47:39)
So before I do the demo, we had some questions, so I want to answer that quickly. I think we have time, right Brian? I can answer these quickly.

Brian Robison: Absolutely.

Steven Smiley: OK. How can your application prevent certificate pinning bypass? The truth of the matter is you can't at the end of the day—and actually pretty much every training says this as well, all the standards say this—is that if somebody wants to bypass a security control you've implemented, they will do it. Now, there are some advanced third party implementations that do it. There's more advanced ways to implement it. Don't use, obviously the common practices, the easy ways that I kind of mentioned to bypass. So there are a little bit more advanced ways to kind of do that. If you use heavy obfuscation and stuff, maybe that'll bring it to the next level. But at the end of the day, people have access to your binary for an infinite amount of time.

(00:48:40)
They can download your app, they have your binaries, and an advanced attacker is eventually going to be able to…the goal of adding these security controls is not necessarily to be OK at the end of the day, no one's ever going to bypass. This is going to protect us. This is the end-all. It is to get your application to the next level, to up your security to a point where your average attacker can't go do it. You don't have somebody just going to download your app and trying to intercept or look for data leakage and things like that. What you want to happen is that someone

downloads your application, they try to intercept the traffic, they get errors, they try to simple bypass, it doesn't work, and they move on to another application because they don't want to waste their time. So you just want to limit the risk of your application.

(00:49:25)
I hope I answered that question. There's one for how to avoid certificate pinning by detecting the Frida process. So beyond just these controls, you can add additional controls as well, which might not necessarily relate to data in transit. And we'll probably talk about some of these in the next session. But there are additional ways where you can do runtime detection for applications where you can detect the Frida process running and kind of block that in an application. So there are some applications, especially financial applications that are doing this already, where they are trying to detect Frida running, detect these runtime tools so that potentially some of these scripts get eliminated and people can't do it. It's another control you need to bypass, and it's all about limiting the risk of your application and getting you to the next level. Hopefully I answered that.

And there's one more: "How do you guys test Android and iOS autocomplete caches for the respective keyboards and Corellium common cache locations?" We can follow up with that question because what we'll have to do is we'll actually take a look at that directory. Brian, can you see if we can get maybe his email address or something? Or maybe he can send us an email where we can kind of follow up on that question because that's one kind of more for data storage. So we can kind of take a look at replicating that and seeing why it doesn't come up as well.

(00:51:10)
Brian Robison: So for the person who asked that, it was submitted anonymously, so I don't have that information. So we're going to give our contact info out on the last slide. So please do just reach out to either Steven or myself, just drop us an email and we'll get you an answer to that.

Steven Smiley: Yeah, absolutely. And just the last question, more a statement: there are ways to bypass Frida detection as well. Absolutely. Any control you basically implement, there are ways to bypass it. So we're not going to say at any point there's no way to bypass it, but again, it brings you to the next level. If somebody needs to come and bypass multiple security controls, they're not easy. They need to write custom scripts, they're probably moving on to the next guy. So it's just bringing your application to the next level and protecting your data as much as you can. Oh, it was Kyle. OK, that's fine.

Brian Robison: Excellent. Thanks for clearing that up. We'll get back to you.

(00:51:59)
Steven Smiley: Gotcha. No problem. So I'm just going to quickly look at a demo. And Brian's got a quick one too. Should be able to wrap them up in a couple minutes. But this is a virtualized

device, iPhone. If nobody has seen one, you could select these jailbroken all the way up to the latest version, iOS 16.2 right now. As the new versions release, you can spin up these devices as well. But what we're going to kind of show—we have the DVIA application for iOS just to kind of show…I actually have data there. It's still monitoring, but what we can show here…I'm just going to put a bunch of numbers in here for a fake credit card, CVV. So you'll see there's HTTP, you can see the data came in. HTTPS data came in. Really cool. One is the certificate pinning. You can see the data came in as well.

(00:52:51)
So within these, you can obviously see the request and the response. This is just going to example.com for this test. So you won't actually see a response, but you can see all the data there and you can see the headers. So this is just a really quick way, if you're looking for data leakage, if you're looking for certain really quick vulnerabilities, headers, missing data in the response, how the requests look, things like that, this can all be done super easy within here. Especially if you have certificate pinning implemented, you should be able to see a lot of your traffic in here and kind of take a look before you move to the next level. So, I'll let Brian show that because I think he has it set up already for doing more advanced analysis. You can't edit anything in here, you can't edit any of these requests, so you're going to want to use a Burp or man-in-the-middle proxy. So we'll just show how easy that is. So I'm just going to stop sharing. And Brian, I will let you go ahead.

(00:53:49)
Brian Robison: Great, thank you. And Steven, if you take a look at that last question there that just came in while I switch over to my screen sharing.

Steven Smiley: Absolutely.

Brian Robison : All right. OK. So as Steven said, we offer kind of the basic built-in Network Monitor, but in a lot of situations you're going to want to use a more complex tool like Burp Suite or Charles Proxy or something like that. And we make that really easy actually, which is something that you don't get from other cloud-based emulators or anything like that. It's actually quite simple. So when you look at one of our virtual devices, you have to connect to it essentially, and the best way to connect to it is to essentially VPN to the device, to the network where the device is located in the cloud. So as you can see here in the bottom right-hand corner, you can see the device's IP address and the services that are configured to listen on that device. So for example, if you want to get into the shell on that device via remote, you can easily get there.

(00:54:58)
And I'm using Tunnelblick as my VPN connection into this based on the OVPN. So you can use any of the compliant OVPN, VPN tools to get in there, but basically once you're part of that network, you can then go ahead and prepare Burp. In Burp, what we need to do is basically start her up and configure a proxy. In the proxy, we basically want to use the specific IP address

that puts us into that network range that we got when we VPN-d into our project into Corellium. And so we have basically this IP address, we're going to start that listener up on 80 80, and then we can simply move this out of the way for a moment. Now on the device, normally the traffic will be just sent directly out to the internet. So on this device, which is similar to the one that Steven was showing.

(00:56:06)
Steven Smiley: Sorry, just to confirm, we did have a question come in. If you are trying to show Burp, it's not actually available on your screen, just letting you know. We can only see your browser.

(00:56:16)
Brian Robison : Yeah, I thought I was sharing the whole screen, but standby.

Steven Smiley: Sorry. Thank you.

(00:56:21)
Brian Robison: Yep. Here we go. I think this will be a bit better. So basically what you can see on this device right here, we get through this ad stuff, is that this device is coming out of Columbus, Ohio, which is our AWS Graviton Center out of Columbus. And to make it go through my local Burp, what I do is I go into the iOS configuration, into the wifi settings and configure it to use that proxy IP address on Port 80 80. And now we're going to start seeing traffic coming through Burp now. And so if we actually go back into what is my IP and we refresh the page—oh no, they changed this recently and it's kind of ticking me off.

(00:57:34)
They used to show all this on the front page where you could get your information. So you see here, my IP address changed, and that's because I'm actually browsing directly out of my local machine through the Burp proxy that is running here. So now this allows me to go in with Burp and do some of that more advanced stuff that I want to do by potentially repeating traffic, modifying traffic and looking at all of the information coming back. So this is kind of a unique advantage that we have on the Corellium platform is that I can actually attach to local tools as if I were running the device physically on my network locally. And if you have other tools that actually need USB connection, we actually offer this USBFlux tool, which actually allows your environment, your virtual environment devices to run as if they were locally connected via USB cables.

(00:58:34)
So you can run local tools like Xcode and actually churn these devices into developer devices where they actually would show up as a physically connected device. And I've got Xcode spinning wheel here.

Steven Smiley: Sorry Brian, can you just show your proxy settings on Burp? We got a question for it, so I just thought it'd…

Brian Robison: Yep. So you can see here, we'll pair this device because it hasn't been set up before and it's going to ask me for the pin code on the device. Alright, so the proxy settings and Burp. Yep. So I forgot, I did not share that. But basically when you go in here to the proxy listener, you want to set it up, either you can set it up for all interfaces, which is kind of a catchall, or you can set it for the specific IP address. After you connect through to the VPN, you're going to basically have an IP address that's on that same network as your devices.

(00:59:32)
Steven Smiley: Yeah, and that's just for the cloud, right? For on-prem you just have a direct IP, which would be easier. And we did get a question: "do you have USBFlux in Linux?" And we do have the source available for that. So you can actually do it in Linux as well.

(00:59:44)
Brian Robison: Yep. The Linux source is right here. The Mac version is here. We do not yet have a Windows version, but we do have Mac and Linux, so you can get that very quickly there. So we are out of time, although I just want to just describe the Android method a little bit. As Steven mentioned earlier, you have to install that certificate, the Burp CA, and we make that super easy because we have the file browser built in. You can just go into the SD card downloads and then quickly upload your Burp CA so that you can install this on the device. And then essentially you're going to go into settings, into your security settings and install that CA. Once you do that on the Android side, we actually use a cell connection, basically baseband connection. See what you do to set the proxy is you go into the baseband connection and you go into the APN, the access point and you set the proxy and the port directly here in the APN for the device itself. And that's where you set it on the Android devices to proxy that traffic locally through your bird connection.

(01:01:06)
Alrighty, I think we've answered most of the questions. If not, we're going to get back to you. Please let us know. Our contact information is here on the screen, so please feel free to drop us an email if we don't get back to you. Ask us anything you want regarding the platform or different types of techniques and tools. We're all available to answer your questions for you. If you're interested in trying out the platform, we do offer free trials. Please check us out on the web@corellium.com. And from Steven and I, we both humbly thank you for taking the time out of your day and attending our webinar. The recording of this will be available shortly. You should get a follow-up email very quickly after this with that link to the recording. So thank you all very much. I hope everybody has a great day, and take care and be safe. Thank you.

####