Corellium

Video Webinar

Episode #7: *Mobile Vulnerabilities Exposed: Getting Our Hands Dirty Part 2*

Full Transcript

(00:03)
Brian Robison : Good morning. Good afternoon, everybody. Thank you for taking time out of your day to join our webinar today. We got a pretty exciting topic for you today. Back in February we did a “Getting our Hands Dirty: Part One,” which we introduced to the world the Android Corellium Cafe app, and this one is going to be Part Two. We're going to introduce you to the iOS version. And I am joined today by our esteemed Corellium Researcher Steven Smiley, who's actually going to run through doing a pen test on our Corellium Cafe application. Now, just for those of you who are relatively new to this, we did an entire series of webinars back in October, November, December, talking about the different elements of application pen testing. And then we actually brought out and are going to be releasing very soon—they're in beta right now. We're going to be releasing our own Corellium Cafe applications, which are essentially a coffee shop type application that has vulnerabilities intentionally created in it so that you can find them and hunt for them. Some of them are relatively easy and some of them are a little more complex to discover, but it is an interesting toolset to utilize and we encourage those of you who are doing Corellium trials to use these applications during your trial to see how many of the vulnerabilities you guys can actually discover. And then we can make the answer keys available to you at any time as well. So we are working on figuring out how we're going to release these to the general public shortly. Obviously you're not going to find them on app stores, but we're working on the procedure of how to do that very shortly. So I know there'll probably be plenty of questions asking about when you guys can get your hands on, but we're working on that right now.

(02:07)
So just a little housekeeping before we start today's webinar, all of you are on mute and there's a good chunk of you guys in here and people are still coming in, so we keep everybody on mute to keep the interruptions to a minimum, but we really do encourage conversation. So please ask questions using the Q&A tool in the Zoom platform. While I'm speaking, Steven is available to answer questions and while he's speaking I'm going to be available to answer questions, and I will interrupt him from time to time if there's a good question or whatever and ask him that

question. So please do engage with us. Last month's webinar on mobile malware and threat research that I did was really cool. We had a ton of questions coming in and it was really engaging and it made the hour a heck of a lot more fun even for me.

(02:59)
So please do that. As you probably heard before my audio kicked in, the webinar is being recorded and it will be made available via our website for on-demand view very, very shortly. And again, as always our presentations are some slides as well as live demos. Sometimes the tools that we are using do not allow us to scale the screen properly and things like that and may be too small to see very clearly. So we will try to talk you through any of those issues that we see. Hopefully we've kind of narrowed those down so far and hopefully we don't have too many of those, but just keep that in mind. You'll always have access to the recording to view it again, zoom in, those kinds of things after the session. So before I hand it over to Steven, I just wanted to plug our technical write-ups that have been coming out lately.

(03:54)
We have been producing a ton of these as you can see over here on the right hand side. Steven, he also works on a lot of these himself. But as you know, for the past couple of months we have been playing up and touting our contributor program where we actually pay people to write articles. And David over here on the two left ones here, he's gotten a whole bunch of these done and there's even more coming very, very soon. Some really interesting stuff coming up shortly about how to actually get true SMS messages into virtual iOS devices into the Corellium platform from an outside phone number. So pay attention to some of these technical write-ups that are coming out very soon. And if you wish to write up any of these like this and be featured on our website, we do pay. We do payout and you can find all that information on our Corellium website on our contributor program.

(04:54)
I think it's under the resources and it's called “Write for Us.” Check that out. But please do check out these cool technical write-ups that both Steven and David do pretty much all the time for us. And for those of you just maybe a little bit newer to the Corellium platform—really the Corellium platform is something that is trying to make mobile security research easier to do and more efficient to do. And the real reason that this is a problem in the market today is because we spend an inordinate amount of time hunting down specific devices that we need to get for pen testing, for mobile threat research, for malware research or whatever, and we create these libraries of physical devices that take time away from what we're supposed to be doing to obtain them and manage them. And potentially you're meeting shady characters in a Walmart parking lot or something exchanging their old iPhone for cash.

(05:58)
But the real reason we're looking for these older devices is because we need very specific versions of operating systems and that's usually tied especially on the iOS side, to getting a version of iOS with a device that you can actually jailbreak because we need to get that

jailbreak access so that we can access the devices at their root level. So even with app pen testing, we need to be able to see what data is being stored on the device in its storage. We need to be able to see as we run through applications and they store data, we need to be able to gain access to that data at the root level so that we can examine it for vulnerabilities. And that's exactly what Steven's going to talk about today a lot is about accessing the data storage, getting the databases off of the device that we need, getting that access to that device at that jailbroken level.

(06:52)
But to do that, it requires a lot of time if you're doing this in the physical world to find those specific devices to find those specific operating systems. And Corellium essentially makes that problem go away and we help save that time. That time is taken up by a lot of flashing devices, re-jailbreaking, doing those kinds of things. Whereas Corellium can provide a jailbroken version of every iOS version that is out there, even the latest and greatest 16.4, 16.4.1, even the beta 16.5 if you're an on-premises premium customer as well as Android, being able to have access to Android devices that are pre-rooted as well. And we do that because we are actually a virtualization platform. We're not an emulation. We're not a simulation. We actually run iOS and Android code, native Arm code directly on Arm hardware with our type 1 Hypervisor because we're actually bare metal hypervisor on the Arm hardware and we can run those operating systems in their natural native environments.

(08:02)
Now we don't rely on public exploits or vulnerabilities in the operating systems to route them or jailbreak them because we are virtualization. We can control that boot process. So the jailbreaks are perfect, there's nothing for you to have to do, no multiple keys to hold down, all those kind of things to try to get a jailbreak to work. It works every time. It works perfectly. It's not even something that you have to do. Just when you choose which OS you want on that iOS device, you choose jailbroken or not. And recently we have released the ability for non-rooted Android, just like you have the ability to do non-jailbroken iOS. You can actually now do non-rooted Android if you wish to test on that platform as well. So if you're interested in that, please check us out. All of these things are available to you in the Corellium platform. And so with that, I'm going to go ahead and stop talking and blabbering and turn this over to Steven who's going to actually do the more interesting content. Steven, go ahead and take it away.

(09:15)
Steven Smiley: Thanks, Brian. I'll go ahead and share my screen here for everybody to see and we'll get started.

(09:25)
OK, so as Brian mentioned, we have the Corellium Cafe application that we developed, which is a fictitious coffee shop that obviously allows people to fake purchase beverages and it is vulnerable in a handful of ways. Things like insecure data storage, network communications, security, misconfigurations in app development. It has some security controls that can be

bypassed potentially, and then some reverse engineering attacks. So there is kind of a lot in here just in relation to your average pen test, your average mobile assessment. You will find a lot of vulnerabilities that are pretty similar in that nature. I don't know if any of you were here for the Android one we did two months ago, but it'll be kind of similar. So we do have some similar vulnerabilities, but there are some differences with iOS, especially when it comes to reverse engineering—Frida use, things like that. Even data storage is essentially different. So we'll be looking at those various vulnerabilities from the iOS side today.

(10:29)
So first we'll just start with an app walkthrough. We'll just show you the application, go through, generate some data, and then from there we'll start working on the vulnerabilities. So I have my Corellium interface here. It's a cloud version right now. I got a device spun up, which is an iPhone 7 Plus running 15.7.1. If you actually look at the apps tab here. I've got the Corellium Cafe application installed, which is right here. So when you launch the application, there's an email and password. We actually don't know that. That is somewhere you would have to find, which we will as we go through today's webinar. So right now you can't actually log in. It's going to tell you that there's incorrect credentials and that the device is jailbroken. There's a couple checks there. So we'll need to bypass that and figure out how to get past each one of those. We do have the home button here, which has a couple options. One is the Corellium blog, which actually just loads to our Corellium blog window within the application as a WebView. So potentially some issues there. We have this XSS simulation just to give us that extra step of testing some of the backend functionality of an application, being able to use intercepting proxies, being able to actually manipulate some of those calls.

(11:48)
So you can actually click that. It'll actually load a vulnerable site where we can kind of take a look. So we'll take a look at that after. There was a question as well. Brian, should we go ahead and answer that? Where can I find the vulnerable app? We haven't decided yet exactly what's going to happen. Brian, you can add to this if you want to, but the end goal is that's something we're working on right now—putting the answer keys together, getting this vulnerable app in the public or at least to trial users. So hopefully soon we'll have an answer. I don't think we have one yet, do we Brian?

(12:20)
Brian Robison: Yeah, again, as I mentioned before I switched over to Steven these apps are in beta right now. They're basically almost finished and then we are going to figure out where we're going to be placing them. They will most likely be available the quickest through the Corellium Trial Program. So if you have a Corellium trial account, you'll be able to gain access to these applications to use during the trial. However, if you are a Corellium customer, they'll also be made available inside your accounts. We're just working with engineering on where to put them and the best places for that right now. So they will be available shortly. We have not decided whether or not we're going to make them publicly available. We do hope to, but right now we're working on getting them kind of internally released first.

(13:16)
Steven Smiley: Yeah, definitely one step at a time. So we'll definitely have some more updates as well and more blog posts related to some of this content as well. So we're going to go over a lot of the vulnerabilities here today, but it'll be good eventually for you guys to take these back to your teams and additional staff to take a look at it as well. So we have the order screen here as well. So we've got some fake products: coffee, latte, tea, hot chocolate. You can add them to the cart. You'll see the cart update here. You can actually go to the cart. You have a promo code option, which you can try and get some free coffee instead of having to pay for it, but if you do have to pay, you go to the checkout, you can enter some information here. So I'm just going to put in some information here.

(14:02)
You put a credit card, I'm just going to enter that in there. Enter a CVV code, a zip code, and then enter a phone number and then you can place your order and it will order. So obviously disclaimer: don't put a real credit card number in there. It's not actually processing it. There's no actual traffic that's being sent to, but the goal of it is just to replicate what you would see in the public with a generic application where you'd have the functionality of having the cart, having some stuff to buy, being able to add and modify that, having promo codes and then going through the full process of that purchase. So that's the application just as a whole. And then we'll probably start going through some of the vulnerabilities associated with this. So right off the top, everything looks good on the screen. I'm not seeing anything crazy, but obviously behind the scenes there are quite a few vulnerabilities here.

(15:02)
So we'll start with insecure data storage. One thing we kind of start with is the iOS Keychain, which the Keychain for anybody who doesn't know is just a storage container for sensitive data on iOS devices. Obviously insecure implementations can lead to vulnerabilities. That can be things like passwords, personal information, certificates, session IDs, things like that. Now you'll see a lot and depending on your development experience, depending on what you look at. That Keychain is the number one storage space for iOS. A lot of the docs will tell you to store all your sensitive data there, all these things, passwords, personal information, credit cards, anything like that. There's a couple of key things to note about that which don't get talked about enough, which is that if something is stored in the Keychain, you have two options. One, you should either encrypt it if it's sensitive data so that when someone does pull the Keychain, they won't be able to actually decrypt that and it will just look like random text.

(16:01)
The other option is to use these Keychain security controls and you see a screenshot here of one of them. So if you use this secure access control reference API, there are some various attributes you can use which determine when that data can be pulled. So do you need another fingerprint on your device in order to pull that data? Do you need the user to authenticate in some way the passcode? Is it accessible when the device is unlocked? Is it accessible when the

device is restarted? Things like that. So these security controls just help limit how people can access that Keychain and what data is there. So obviously sensitive data being stored when the device is unlocked or when the device is restarted becomes a little bit more of an issue.

(16:49)
So how can this data be exploited? Well, obviously if it's not encrypted and anything like that, you'll be able to access the Keychain. So we have two options really to access that Keychain. One Keychain-Dumper is a third party tool. You can get it off GitHub. You can install it on your device locally and it does work with the Corellium devices. And then you can just run the command from the command prompt to dump the Keychain and you'll see all the data there. Objection Framework, which I'll explain on the next slide—really easy way, just one command. Dump the Keychain and you'll be able to see what's being stored in there for your application.

(17:25)
So, Objection—what is it? So, Objection Runtime Exploration. They have a whole GitHub project with a lot of information on there, so I recommend going to check it out if you haven't used the tool at all because it is a really good tool. But it's a runtime mobile toolkit powered by Frida. So it really gives that front end to Frida, which can be a very difficult tool to use. So you can inspect and interact with container file and systems for iOS. For iOS and Android—specifically today we'll just talk about iOS—you could bypass security controls potentially. There are some built-in scripts within there to bypass geo detection certificate pinning. They don't work all the time, but they do a fairly good job. You can dump the Keychain, you can dump other storage within iOS, you can actually use hook application methods to get some data back. There's just a lot of things you could do. Hard to go over everything, but it's a really good project. So I recommend if you haven't used it before, go check it out, read the docs. There's probably a lot more that can be done within there.

(18:29)
So how is it performed? How do you do this? Well, if you have the Corellium Cloud, which I do for today, you connect to the VPN, which is actually a VPN profile. When you log into the device, you have the connect tab, you'll be able to see that as well. You can download the VPN profile. You ensure USBFlux is running, which is our tool to emulate the virtual device to look like it's plugged into your computer, right? So if you go into Xcode, you'll be able to actually see it. It just looks like the device is actually plugged in even though it's the virtualized device. And then you go ahead and hook the target application. This is the easiest command: objection -G, which is for gadget. Pick your application name or your package name and then do explore, and that'll give you that command prompt to be able to do a little bit more. There is some additional stuff as well. Startup commands if you want to run commands at startup instead of launching the application and then running. So there's a few things you can kind of do there.

(19:28)
So right now we'll kind of take a look at actually doing that. So I have a terminal window here and if anybody can't see it just let me know. I can try and increase the size or anything like that.

Brian Robison: Looks pretty good, Steven.

Steven Smiley: OK, perfect. I'm just going to move this screen sharing thing out of the way so I can actually see. If you don't know by the way what your package name is, you can look on here, you can find the package name here, which is a really good way to do it. But for iOS, if you don't know, you can also run Frida, which is what I just did. So I just ran Frida-ps to grab the processes. USB because I'm running USBFlux and then I'm listing all applications installed. So you can see the ones that are running right here, which is the cafe application. So if I just copy that and then if I run Objection, like I said in the slide, -G for gadget, the application, and then explore, what'll happen here is this will actually relaunch the application and then it hooks.

(20:31)
And like I said, there's just tons of commands, but if you do this command, which is, I always Keychain dump. You can actually dump the Keychain and you can see this is a security control that I talked about on the previous slide. So this is when unlocked. So if the device is unlocked, you're going to be able to see this data and this is the credit card number in plain text. So in a normal case, this should obviously either be encrypted or there should be some additional protection around this where they would request the fingerprint of the device in order to do that and to pull that data back. Ideally, the best way is just to encrypt this.

(21:12)
Any questions around the Keychain stuff? And there's Keychain-Dumper as well. It's on GitHub. I'm not going to go through how to do that, but it's another good project if you don't like using Objection. OK, so continuing on the same theme of the data storage, you have NSUserDefaults. So the iOS system actually allows an application to customize its behavior to match the user preferences, and a lot of this stuff is stored in NSUserDefaults, and that could be something as simple as different themes, light mode, dark mode, a unit of measurement, kilometers, miles, something like that, various application preferences. What do you want to see first when you log into the screen? Make it a more personable experience for the user when they install the application. Now most of this stuff completely harmless.

(22:07)
At the end of the day, if someone gets that data and sees it, no one's really going to care that you like light mode, but I like dark mode or you like to see this screen first and I like to see a different one. No one's really going to care and there's really no security concern there, but when it's used insecurely, sensitive data can actually be stored in there. And I've seen this in a handful of applications and you'll actually see it in this application where something for the application gets stored in there that shouldn't be. So how can you access it? Easiest way again is Objection. You can actually pull the data really easily. Alternatively, there's actually a preference file stored locally in your data directory, but you have to convert the file. It is kind of a pain, it's not the easiest way. There is some guides online to do it.

(22:51)
We should probably publish a guide as well, but Objection is really the easiest way to kind of pull that data. So here actually, I think we will take a look at that. So if you look here, we have this hooked already. So if we do iOS NSUser, not that one, NSUerDefaults and then get, it's going to pull all the user defaults. So some of these are generic from Apple and you'll see these with the keyboard versions, some of the versions as well, some of these ratings, this all comes from Apple directly, but what doesn't come from Apple is this. So if you see a username and password here, this is directly programmed in, developed in, for this vulnerable application. And now you can see that we do have the credentials that do work for the application. Now we do have some jailbreak detection as well, so it won't actually do anything, but you can actually coreadmin@corellium.com. So it'll still show that the device is jailbroken. It does say that it is logged in, but it does say the device is jailbroken, so you do need to bypass that as well, which we will kind of talk about later as an option. But that's the easy way to just pull the NSUserDefaults.

(24:36)
So iOS Application Database. It's kind of the last component for data storage. Now there's a handful of databases that can be used within an iOS application. They're all stored within the local data directory. They might be slightly different paths in just that final folder depending on what database they use and how they store it. Really only accessible via the application unless it's jailbroken. A jailbroken device is really the way to go for this sort of information. You get full access to the data directory, you can pull all the data you need to, not just the databases but potentially other files that are stored. Obviously we provide Corellium as a tool, it provides file system access to all iOS versions. Brian even touched on it—we have beta versions coming out for on-prem customers and they come out really quickly. The versions all come out really a couple of days to a week after they get launched, so we have them.

(25:34)
Obviously data stored in plain text is vulnerable to anyone with access to your device and really how is that performed? Well, we already kind of talked about it, but you just got to identify the path of the data directory for your application, search that local data directory and then you can find some database content or potentially other content beyond the databases and then just open those up and review them. Sorry, I meant to switch the slide, not the screen. So the iOS local data directory, if you don't know, if you haven't done a lot of work on iOS or if you're primarily Android is /varmobile/Containers/Data/Application and then this <ID>. So how do you get the ID? Well, there's two real ways for Corellium and there might be some other ways as well. These are the main ways that you would use in the Corellium file tabs.

(26:23)
If you actually go over here, you have this Last Modified, so if you just installed the application or you know when you installed the application, you can use that to identify when you see a list of±and we'll get to this, I'll show it in a second when we do the demo—but when you see a list of applications, you can sort it by Last Modified and know just when the application was installed

and have a good idea that that's the one you're looking for. The other way to be a hundred percent sure is just to use Objection and run this ENV command, which is going to give you the paths for your library directory, for the cash directory bundle and documents.

(27:03)
Brian, we got a question. Did you answer that or are you typing?

Brian Robison: OK. Yeah, I've got it.

Steven Smiley: Gotcha. OK, so first thing first is we'll just run ENV just to make it easier so that we actually know what the path is and then we can actually go take a look at this. So if we do var/mobile—sorry, clicked private—it's VAR/mobile/Containers/Data/Application 4BB. So if you, actually that was really quick, you might not have actually seen that, but if I clicked Last Modified, if I sort it going down from the most recent, I actually installed the Corellium Cafe application on this device yesterday I think. Yeah, yesterday. And so you'll see it's the top one. It's the easiest way to figure it out. Alternatively, I just also ran Objection to see what that path was and then you'll get access to this folder.

(28:03)
So you'll see a bunch of stuff in here. There is a documents folder, there's nothing in there for here. If you look in library and application support, you will see this SQLite file. So they're using a SQLite database. You can download it. Now this is one thing that I want to note about Corellium, just so that you don't get confused and it's really about how the databases work and we're actually going to be publishing an article on this maybe next week. We'll see about that one. But one thing to note right now, and I'll show it, if I download this SQLitefile and I try to open it, there shouldn't be any data in there. So you'll see CUSTOMERINFO, sorry, CUSTOMERINFO table, there's nothing there. So the credit card field is there. So what I'm actually going to do, I'm going to download all three of these.

(28:58)
So if you select multiple files, you can download as a zip archive that'll open up here and then you can actually open up the SQLite file and you'll see the data is in there. So one thing that we don't want to get confused about and because we give—lemme just go back here. OK, so you want to see the one file now, that's OK. So what databases have is they have this write-ahead log. So what happens is the data gets stored here first and then periodically after a certain amount of time that data will get written to the SQLite database. That doesn't happen right away. It depends on the development, depends on the configuration, but typically it's not very quickly. So if you're entering data in the application as part of a pen test or part of an assessment and the data is in this file but hasn't been pushed to the SQLite file, it's best to download all three, download the entire folder and then open the SQLite file.

(29:52)

Because what'll happen when you saw it the second time when I downloaded it as a zip archive and I opened it, it's basically reading this write-ahead log and seeing all the data's there. So it's actually writing the data itself at that time. Whereas if I just downloaded the SQLite file, you might not see the data. So that's something you really just need to be aware of because we give you the unique ability to be able to just go in here, navigate this entire file system and download what you want. I don't want to get confused about just downloading the SQLite file or downloading the database you see. Download the entire folder just in case the data hasn't been published over. It's a good way to make sure you don't miss anything. Potentially sensitive data being stored that obviously gets missed. We don't want that to happen. So at the end of the day, that's the best way to do it. If you saw in there, there is a credit card in there as well. So I don't want to get away from that. Plain text credit card in a database. There are a couple options here—obviously that database either probably needed to be encrypted or that data needed to be stored somewhere else and not in just a database for everyone to access.

(31:03)
So a couple different options there. OK, insecure communication. If you guys have been around for the other webinars, you'll know about the Corellium Network Monitor. If you've used Corellium a handful, our Network Monitor allows you to quickly observe network traffic from an application. Regardless of HTTP or HTTPS, you're going to be able to capture all the traffic. We actually bypass most certificate pinning implementations so you can actually see what's going on, what the app is communicating with, what servers, what endpoints, what requests are being sent. At the end of the day, it's not an intercepting proxy, it's not going to solve all your problems, it's not going to be the end-all when you're doing an assessment. You will still need to do some of that additional testing, but it really gives you a good starting point to understand what the application is doing, who it's communicating with, what data is being sent, and then obviously when you're looking for stuff, what's being sent to a third party. Anything?

(32:01)
Is it only talking to its own servers? What headers are being utilized in the request and response? Are they utilizing the proper headers? Are they missing stuff? Do they have misconfigurations around those headers? Potential information leak? Is just the application responding back more than it needs to? Potentially credit cards, account numbers, things like that. All sorts of stuff can be quickly analyzed just from the Network Monitor alone before you get into potentially an intercepting proxy. So we can just take a look if you actually, I can do a start monitoring here. If you actually order something in this application, you will note there is one request that gets sent…and I'm just going to…

(32:56)
Sorry, I'm just going to enter all this data, place order. See, we'll see here a request does get sent. It's just being sent to our server. So the goal of this was just to show that Network Monitor can be used. You can see the requests and responses. We're not going to get a response because obviously Corellium.com is not expecting this data, but you will see what kind of data is

being sent and in the case of a response you will see that as well. So for this, it's kind of going to a third party. Sorry, we had a question too.

Brian Robison: Yep. I was just going to mention, there's a pretty good question in there if you wanted to take that one, Steven.

Steven Smiley: Alright, so imagine that the credit card number is saved encrypted in a SQLite database or Keychain as you mentioned. Where should the key be stored securely and decrypted? Client side or server side?

(33:49)
I mean I guess it depends on the key. Everyone does it a little bit differently. You can do it client side. The issue with doing it client side is depending on where you're storing it, eventually someone is going to find it. At the end of the day those applications are going to be out in the public, especially ones that are in the App Store, ones that are being given to a bunch of people. There's infinite time. Anybody who has those applications has them forever and eventually they could find those keys. So server side is really the best way to do it, but it really depends on the applications. I'm not saying that every application is going to do it server side. Still client side is better in having some sort of encryption than storing in plain text. But server side at the end of the day is the way we want to go.

(34:40)
So yeah, if you look here, this is the only request you get in this application. It was really just to show off the Network Monitor and its quick ability to be able to find what the application is doing, who it's communicating with, what the request and responses look like. But at the end of the day, it's not an intercepting proxy. So there's more that can be done and some of that needs to be done through an intercepting proxy. So great. That's a great segue into the next slide, which is about the intercepting proxies itself. Typically Burp Suite is the go-to/ Everybody's a little bit different. Some people might use man-in-the-middle proxy, Charles Proxy, something like that. Burp Suite typically tends to be the go-to, but the configuration in setup is the same. It won't be very different depending on the tool you use. Very easy to configure on a Corellium device, Android or iOS. For iOS, you actually don't need to add the certificate. Only for Android, you need to. Or for non jailbroken iOS you would need to add it. And then once you configure that, you can get all your traffic from the application into Burp. There's free and paid versions. I'm sure everyone is aware of that. The paid just offers additional scanning, safe projects, a little bit more utilities, things like that. But the free version can do a substantial amount as well.

(36:01):
So I actually have Burp running somewhere on a screen. So I have it running here. It's running on my VPN IP that I got from Corellium. So what I'm going to do here is I'm actually going to turn off the Network Monitor so I don't have to see all that, but if you just go in here and configure your proxy, you can change it to manual. I have the memory of a goldfish, so I can't remember these things, but 10, 11, 3, 2 or 80 80. So same configuration you see in Burp, you're going to

save that and then all the traffic should be coming through now. So if you launch this Cafe application and go to, let's say the Corellium blog. Yeah, there you go. It's going to bring all this stuff here. So there's the Corellium site and there's a bunch of other stuff as well obviously being loaded when you go to a major site like that. But what I wanted to show was they have this XSS simulation. You don't actually need to necessarily use Burp for this, but within here they have this text field and then you could subscribe. So it's like an email list subscription. So you can put test@test.com or something like that, subscribe to that, thank you for your subscriptions. So you'll be able to see that right here. And I've been added to the mailing list. Oh, yay.

(37:35)
So part of this—so there's two real vulnerabilities here. One is with the WebVew. So if you actually look at the blog, you can actually swap out that site for something else. They're not doing any checking. So you can actually take that. So right now we go to the Corellium blog site. So you can actually intercept that and change it to any site you want. If I actually go intercept on—so you're going to see a lot of switching between screens here. So it is going to Corellium.com with the blog. So you can actually change this to something else if you have another site in mind and try to swap that out. Sorry, there's another question.

Brian Robison: Yeah, there's a question. A continuation on the previous one.

(38:28)
Steven Smiley: Sorry guys, I'm just reading the question. I apologize. Yeah, totally correct, right? If you're going to store a client side, I mean you can store it somewhere else. I mean iOS is going to tell you to store it in the Keychain as well. Obviously if you're using newer encryption methods, it's going to be a little bit more difficult for people to actually still decrypt that. Obviously using very weak methods that use one key or something like that is obviously going to be an issue. Server side is really just the way to go beyond that. It just really depends on your application and the data you're storing, right? At the end of the day, if you look at a banking app or something like that, credit cards and personal information, you really need to be using the best security going server side. But if it's something basic like someone's email address, you're just trying to protect at the end of the day, those are really fairly public. So the encryption probably doesn't need to be the same level of banking information or other people's personal information.

(39:33)
So you can swap this out, this URL out, there's actually a way to do it with Frida as well, which I'll leave to other people. We don't want to give everything away. And there are some other webinars and stuff we have upcoming that are going to kind of talk about some of that stuff. So we'll leave that. But yeah, for the XSS simulation, you can do this through Burp as well. You can actually enter it in Burp if you want, but if you write…they actually are vulnerable to XSS. Let me make sure I type everything right. I can't spell. It might've been my spelling. But you can also do this in…I might just do it in here to make it easier.

(40:57)
And there you go. So if you can intercept it, you can modify the field to perform something like XSS. I mean the goal of this was really just to show two things. One, Corellium is really quick at being able to just set up Burp and be able to configure that and run various attacks. This is a very simple attack, but at the end of the day, still an attack nonetheless. And the other is just making sure that we are using Burp and we're trying to get everything we can out of this application. We're not just testing the client side of the application, we are testing the backend. We are testing every component, every field, and doing a full kind of test of this application.

(41:46)
OK, so common misconfigurations. How much time do we have? We have 16 minutes. OK, perfect. So common misconfigurations in iOS. You have the info.plist, which I'm sure most people are aware of if they've done any sort of iOS assessment or reviewed anything for iOS, especially when it comes to installs and configurations, but their iOS configuration files used to provide essential information about the application. That could be things as simple as package name, application version, supported device models, iPhone, iPad, things like that. There are some more complicated things you see in there sometimes, which are potentially API keys or third parties, app transport, security settings and various misconfigurations. So when it comes to the app, transport security, you obviously want to ensure your application and network communications are secure and utilize modern supported technologies. So allowing disabling ATS can allow people to downgrade TLS 1.0, use HTTP, whatever they want to communicate.

(42:48)
So if you're disabling that globally, that's a bigger issue. So how do you go about accessing the info.plist? Well, the easiest way is just to unzip that .ipa binary, navigate into the Payload folder and then just open it. So I'll just show that really quick. I actually have it here. So if you have your Cafe application or any application, you can just unzip that you're going to get this Payload folder. You go into the Payload folder, you can do show package contents and then you can just review everything that's in here. So you have info.plist for example, and there could be other PLIST files in here, not just to specify the info. Info's always going to be there because it's part of the configuration, but there's potentially other ones, right? If they're using third parties or different APIs, they could have a PLIST as well, which could have potentially hard coded data. But yeah, if you look in here, there's not a whole lot in here, but if you do see the App Transport Security, they are allowing everything from local networking to arbitrary loads to arbitrary loads in web content—literally allowing everything. So that is a misconfiguration, any sort of network communication is really at risk. Then people are going to be downgrading that to HTTP using insecure protocols, things like that. So any protections that you've already implemented to kind help your traffic are kind of gone out the window by allowing all of this.

(44:13)
OK, debugging code, which is probably a bigger issue on Android is if any of you guys have done Android, ADB Logcat is pretty big and you can find a decent amount of data in there. Sometimes if developers forget to turn it off or the logging username and password count

numbers, things like that, that happens quite commonly in iOS. It really doesn't happen all that often, or at least it's not looked at all that often, but it can be done with Xcode. So because I'm in the Cloud product, I have this USBFlux running, which is, like I said, it's just making it seem like my device is actually plugged in even though it is virtual. So if I launch Xcode, which will take a second because Xcode is not the fastest tool in the world, but once it loads here, if you go to devices and simulators, apparently there it's OK.

(45:08)
So that took a second, but you will see the Corellium device here as well. Load it up. So what I'm going to do is…so when you get to the cart, there is this promo code field that we kind of talked about. So there's basically three ways to get this promo code. 1): being this way through the debug logs because it's actually recorded in there. 2): through reverse engineering, depending on whatever tool you want to use, Hopper, Ghidra, something like that, you can get it through there, it's hard coded. 3): you could potentially hook the method and get it through Frida. So I'm going to show two today. I'm going to show this one and we'll just go over quickly the reverse engineering side of things to look at it. But if you open the console here, you can actually start streaming data here for this iPhone. So you will see here, everything's going to be coming in. If I try to apply a discount, it doesn't matter what I write in here, I can just try to apply discount and then—let me pause this for a second just to see if I can find it in here.

(46:24)
OK, so you'll see here there's a code here, which obviously does not match with everything else. So that's when you kind of know it was logged as an error as well. So that's one way to look at it. But you will see here, CorelliumizAwesome and you can actually type that in. It will actually work. Oh, is there a capital? Oh, there is. And your total goes to $0. That's one way to find it through the debug logs. But obviously if that gets turned off or this goes to fraud and potentially it's removed, you might not find that easy way to grab it. So you'd have to go with reverse engineering, which we'll take a look at as well.

(47:27)
Oh, sorry, two more things. 1): Bypassing security controls. Security controls obviously can include certificate pinning, jailbreak detection, tamper proofing, runtime detection. There are a couple ways to do it by default that are really just easy. One,Cydia packages or SSLKillSwitch for iOS. Really easy to install. Package that works really well for certificate pinning. That can all be installed through Cydia through your Corellium device. Frida, it's a little bit more customizable. You can write scripts, you can edit scripts. You do have the full Frida console within Corellium as well. So you can actually run these and there is the Frida codeshare, which you can grab scripts from. So I won't actually bypass it because I already showed that it's there. But if you go into Frida here, you do have the console where you can hook a process here, like this Cafe application, if I spell it right, you can hook that and you can have a full access and then you can just run a script.

(48:24)

So I have this iOS jailbreak bypass. If you actually execute this and then run that script, if I try to log in now it's actually going to bypass it and you'll see the debug log here and then you'll be able to log in. There's no difference to the application itself. The log in was really just to show that jailbreak detection and things like that were there and that it was fairly easy to bypass. So I'm not going to go ahead and run that script. I want to be able to show the reverse engineering stuff, which is kind of a prelude to what we have upcoming.

(48:59)
So, reverse engineering. So for iOS tools, there's a handful. There's no default answer to say this one's the best because really it just depends on the user. Hopper is good, it has a free edition. It's only a short period of time you have to look at a watermark, but it does work. Ghidra is a really good tool that a lot of people have been using lately. It was built by the NSA. It's actually freely available. Really good tool for reverse engineering. You can use Radare2 or Rabin as part of it to search for strings or various values.

(49:34)
There's obviously more. IDA Pro and things like that, but they get kind of pricey depending on if you do this all the time or what you're actually doing. But at the end of the day iOS reverse engineering can be quite a bit more difficult than Android. You search for methods classes, try to gain more understanding around what the application is doing. So in this case, in this demo, we got a couple minutes, they're actually going to look for hardcoded values. That's step one when it comes to reverse engineering. Is there anything hard coded? Potentially email addresses, usernames, passwords, API, keys, things like that? What is actually hardcoded in the application? That can be pulled out and give you more value, potentially a vulnerability in itself and that can potentially give you more information for a further attack. What you want to do after that is use what you've found to build...

(50:26):
If you don't have a lot of hardcoded stuff or you have some methods you want to look at, you can use that information to build out some free to scripts and perform some more advanced attacks. So we actually are going to be doing a webinar. Correct me if I'm wrong, Brian, Android is May and iOS is June, right?

Brian Robison: Yep. I think, yeah, I believe we're doing Android reverse engineering first. Yeah, so next month.

Steven Smiley: Yeah, so we have two more webinars coming up where we're going to perform a little bit more of an advanced attack. We're not just going to look for hardcoded values using reverse engineering. We're actually going to hopefully take some of the information that we've gathered through. In Android it is going to be like JADX and then an iOS, we're going to use Hopper or Ghidra, gather some information about the application, build out a rescript and run that. That's going to be a really cool session as well if you want to do something a little bit more advanced. But for now, we'll just look at the hardcoded values here.

(51:22)
What I'm going to do is I'm going to hop over here, we'll just load Hopper. We can do this with any tool. So I'm just going to load the Cafe application in here and I'll talk for a second while it just analyzes this application. But yes, we've got some really good webinars coming up. If you're into reverse engineering, feel free to register for those. Those are going to be kind of exciting. We're going to build some Frida scripts and actually kind of walk through that. So that should be kind of exciting. But yeah, in the meantime we're just going to look for hard-coded values. So for this application we already know what the value is. So if you know what a value is, you can look for it.

(52:08)
It's a little slow because it's still analyzing. But you will see CorelliumizAwesome here, so you'll be able to find it. That's not the way that I would necessarily go about it because you don't really know what that was. We only know because of the debug logs that it's actually there. You can also, if you're searching for emails or something, you can also use the @ symbol and start searching for stuff. You can get the username through there, like CoreAdmin, you can find Corellium.com because it is a username. But in terms of the promo code, there's a couple of things you want to look at. What's a promo code? What's in the cart? What are some ways you can look at that? So typically you're going to start walking through the code and start figuring out, but you really need to know where to start.

(52:53)
So in general, the best way is just looking at where in the application it is. Are we looking at the cart for example, there's a few here, the cart view, cart card. So you can see a few things here. The cart controller. You see some references to promo code here as you scroll through. You can see a handful of information. Now you see this thing called your cart is empty. So you can see this and then just below you can find a hardcoded value. So they're using this CorelliumizAwe. It's short form for awesome there. That's going to be hard to search for. It's not just called promo code. You can find it here by looking towards the cart. The other thing I would kind of always look for is what else happens? Looking at the promo codes, you start looking at promo code, you see a promo code here, you see a couple methods you can actually scroll through. There's a few different options.

(53:57)
And then this one we're really kind of in the code here looking at it. So if you actually scroll around, you can look, there's the credit card number, it's pulling. You can see the reference to Corellium blog, which by the way is a good opportunity to kind of take this information to actually try to swap out this using a Frida script. But if you actually scroll up, you'll be able to find CorelliumizAwesome. Now, this is obviously just the beginning of reverse engineering. You can search for anything. In this case we're searching for obviously promo codes that we're trying to find something of value. In a general application, you could be looking for things like an API key,

anything hard coded with an API key. You could be looking for a password, an email address, things like that. So there's a lot of different methods you got to walk through.

(54:46)
You got to search some stuff, you got to gain some information and do that. There's some other tools as well that you could run prior to this to be able to gain information about what methods are being used, what can actually be hooked, what can be ran. We actually wrote an article not that long ago. I think we're writing another iOS reverse engineering one, but there's an intro one out you can actually take a look at where we talk about running some tools like strings and things like that Otool to get more information about the application and then use that as you move into tools like Hopper and Ghidra and stuff like that. But like I said, we have some new articles coming out as well and I'll just bring it to the thank you screen. But yeah, we have some new articles coming out on reverse engineering, some more advanced stuff, talking about the scripts. We have the new webinars coming out in May and June. So if you're into reverse engineering, this was just really an intro to talk about our vulnerable app that we built, some of the ways that it works with Corellium, some of the ways you can walk through some of the pen test techniques to be able to find some of those vulnerabilities. But there is some more advanced stuff coming. So I definitely register for those and keep an eye out.

(56:03)
Brian Robison:
Well thank you very much, Steven. And again, like Steven said, May and June are going to be much, much deeper dives into the tactics and techniques of reverse engineering, especially around why you're looking for certain things and how to do that. So we look forward to seeing you guys next month. If we see a couple of questions that are coming in right now, we're going to go ahead and we'll address those with you folks offline. Our contact information is there on the screen. Thank you Steven. Really, really appreciate everybody.

(56:34)
Steven Smiley: I can answer the one if you want, Brian.

Brian Robison: Go ahead.

Steven Smiley: The one was just a thank you. So thanks, Carlos. If you have any questions, feel free to reach out. The other question was just if we developed it natively, which it is. We did not do a hybrid application by any means. We didn't use Flutter, we were going to use Flutter, we decided against it. So we are using SwiftUI for iOS and Kotlin for Android. So it's all done natively.

(57:03)
Brian Robison:

Awesome. Thank you very much and thanks everybody. We look forward to seeing you in May and June at the next set of webinars. Steven, thank you once again and I hope everybody has a great rest of their day and rest of the week. Thank you all. Take care.