Corellium

Video Webinar

Episode 5: Mobile Vulnerabilities Exposed (Getting Hands Dirty 1)

Full Transcript

(00:08)
Brian Robison: Good morning, good afternoon. Thank you all for taking the time out of your busy day to come to our Corellium webinar. This is our first webinar of 2023 and we are very happy to have you with us. For those of you who've been with us for the past couple of months or few months, we have been kind of working through a series of vulnerabilities around mobile app vulnerabilities and we've talked about data at rest, data in motion, reverse engineering. And today Steven is going to actually bring you some really cool things that are happening. We have built our own vulnerable app called Corellium Cafe. It is a coffee shop-type of application that has vulnerabilities in it that you can use your skills and techniques to discover the vulnerabilities. So that's why we titled the webinar “Getting our Hands Dirty.” So that's exactly what we're going to be doing today.

(01:07)
I am Brian Robison, Chief Evangelist here at Corellium, and I am joined once again by my esteemed colleague, Steven Smiley, who is our Corellium mobile app pen testing research expert. And he is going to be running through the vast majority of today's webinar. This webinar is being recorded. It is being recorded and will be sent out to those folks who attended as well as be available for viewing at a later time on demand through the Corellium website. So this is a webinar format. The attendees are muted, but please, please keep engaged and participate. There may be some polls, there might be some surprises and definitely feel Frida ask us questions. We will keep an eye on the Q&A widget within the Zoom tool, so please feel Frida engage with us, ask us anything you want, even things like, “Hey, how do I have Corellium logos on these old Macs behind me?”

(02:06)
Something like that. Just whatever. We love to engage with all of you. So again, it's being recorded and it'll be available for everybody. You'll be able to share it with your teammates who maybe didn't make it or view it offline. And this presentation does consist of a few slides, but mostly live demonstrations. And then sometimes with live demonstrations we can see some interesting artifacts come up. Sometimes we have screen sizing issues and font issues that

make it difficult to see kind of what's going on, but please bear in mind that we will explain to you what's going on where we've practiced and we've tried to make things as legible as possible. If there is something to difficult to see, Steven will obviously point it out and talk about what he's doing and why he's doing it. And then of course you could zoom in on the on-demand one when that one comes out.

(03:00)
Alright, so I do want to plug our contributor program. I did introduce this in December, but I do want to plug our contributor program. In fact, we are actually getting ready to post our first community contributed blog and pay that person a significant amount of money for creating the content. This blog should be coming out here in the next week or two. So this is definitely a way for those of you who want to participate and have a voice in the mobile security realm, whether it be pen testing or vulnerability research or even just threats to mobile devices and users. We're looking for all kinds of content. So please feel Frida fill out that form. It does come to me personally and I'm more than happy to engage with you to see what we can come up with and see if you can't make a couple extra bucks in the meantime creating content.

(03:54)
In December, we also launched our Mobile Malware and Threat Research Solutions brief. And so this is kind of tangential I guess, to mobile app pen testing. A lot of the tools and techniques are actually the same. Reverse engineering is the same procedure. We look for different information like hard-coded URLs and things like that in some cases, but a lot of the techniques are relatively similar. So in March, our March webinar, a month from today, we're probably going to—at least tentatively planned right now—we're going to look through doing dynamic analysis on mobile malware utilizing the Corellium platform. So I wanted to brief on this one really quick while we still had everybody kind of on the line. We're going to be talking about mobile malware and threat research and even some things like on how to put your virtual device completely in a different country, both GPS and network, using VPN technologies and things like that.

(05:00)
So it should be an interesting little webinar that I'll drive. OK, so just a very brief kind of overview about what the Corellium platform that you're looking at today in some cases is all about. And then I'll hand it over to Steven very quickly here, but basically we are all in the mobile security space, whether we're doing pen testing or vulnerability research or whatever, but it is difficult to do this in today's world. It's not just because the operating systems are designed to be sandboxed and isolated where apps can't affect other apps. And this is why we have what I call fake AV solutions on mobile devices because we actually can't stop users from installing apps and running them. We can only detect behavior afterward, that kind of thing. But really it's difficult in the world because we need access to devices, but more importantly we need access to specific operating systems.

(06:02)

And what is it about specific operating systems that is so important? Well, from the security research side, you must have devices that you can jailbreak or root because to do real security testing and especially when you're looking for evidence in data at rest data that is stored on the drives essentially of these devices, you have to have root access to these file systems to gain access to that evidence. You can make all the assumptions you want about encryption and all these other kinds of things, but if you can't actually validate that that data is actually stored encrypted on disk, even when the device is compromised, then you do run the risk of leaking data. Just like back here in September of last year when there was an article on BleepingComputer about thousands of iOS apps leaking hard-coded AWS creds. It wasn't a malicious attempt, it was just, oh, we forgot to obfuscate or we relied only on device encryption or something like that.

(07:10)
And as soon as those single layers get defeated, then all of a sudden data gets exposed and our companies can become compromised. So we need access to very specific operating systems on very specific devices. And as you all know on the iOS side especially, it is difficult to get modern devices with jailbreakable iOS versions. It's also difficult to go back to unsigned versions from Apple if you're flashing devices back and forth and things like that. So those are the two major reasons why virtualized devices make a heck of a lot of sense. The third is the time. So if you're out there dumpster diving and you're flashing devices and you're jailbreaking and you're doing these things, you're spending a lot of time that is not security testing or security research, you're spending a lot of time doing menial tasks and things like this. So again, a virtualization platform like Corellium helps you resolve these time issues simply because you've got the ability to create VMs, fire 'em up, shut 'em down, destroy them, fire up a device with a different iOS version jailbroken or not, even the latest 16.3 is jailbroken with Corellium because we're a virtualization platform—don't have to rely on a public vulnerability and an exploit to do the jailbreak.

(08:42)
And then so again, our platform is virtualization. It is not simulation or emulation. We run Arm-based operating systems on Arm hardware, whether you use the cloud environment, which is what Steven's going to show you today, which is hosted out of AWS Graviton hardware or you purchase our on-premises appliances. These are Arm-based servers, running Arm-based OSs like iOS and Android. So there's a lot of benefits that you get from that virtualization like the ability to power up any device with any OS and take snapshots and restore snapshots and share snapshots with other users, all those kinds of good things. So with that, I'm going to end my portion of the presentation. I'm actually going to hand it over to Steven who's going to take over and get into the real meat of what we're doing today. So thank you again all for coming and hanging out. Steven, this show is all yours.

(09:46)
Steven Smiley: Thank you, Brian. Good morning everybody. Hope everybody's having a good start to the day. I'm going to go ahead and share my screen here so we can get started. So as

Brian mentioned, we've developed this new application and we call it just the Corellium Cafe, which is a fictitious coffee shop application that obviously allows users to purchase beverages. I mean, the goal of the application is to be vulnerable in a handful of ways, and that can include things like insecure data storage, communication, various security misconfigurations on Android and iOS bypassing security controls. So there are security control implements it within the application. So trying to bypass those runtime level vulnerabilities, things that you can interact with Frida and try to bypass, manipulate, things like that. And then reverse engineering where you're actually reversing the application to look in the code to find those features that you would need to write a free script for or to find some hardcoded values, things like that. Was there a question, Brian? Sorry, Brian, there's a question. Is the vulnerable app going to be public? Do we have an answer on that at this point? I don't know that we do.

(11:05)
Brian Robison: Yeah, we haven't fully decided on that yet. I mean absolutely we would like it to be. What you're going to see today is kind of the very first iteration that actually runs and we can do some work with it, so we're close to making it available. We definitely want it to be out there as part of the community, so bear with us a little bit longer while we figure all of this out. But yes, it will be available for people who want to play with it. Maybe initially it'll be available in like a Corellium trial, but then hopefully we do plan on releasing the APK and the IPA publicly. But again, we're working through those issues right now.

(11:53)
Steven Smiley: Definitely. Thanks Brian. Yeah, this is all super new. We just developed this. We're trying to obviously improve the community, help out, give some good suggestions on vulnerabilities and the ability to kind of test those. So there's still work to be done. We'll obviously see more information about this. And on the question for the walkthrough, we're going to actually walk through a lot of the vulnerabilities that are in the app today, so you'll see that. So you guys will get kind of a good idea for it. And obviously this is a recording, but in the future, yeah, we will, if we launch it, there's going to be walkthroughs and stuff like that.

(12:26)
Brian Robison: But Steven's not going to show you all the vulnerabilities, so we're not going to give you all of the answers in today's webinar,

Steven Smiley: Correct. Yeah, so let's go ahead. We're going to start with an app walkthrough just to kind of see the application, see what's there. So I will go ahead. This is my Corellium Cloud instance to be able to see that I have an Android 12 device here running. So I'm going to show you the Android app. So here it is.

(12:56)
I'm going to go ahead and launch that. Maybe. One sec.

Brian Robison: I have this issue occur. Oh, it looks like you lost networking.

Steven Smiley: Yeah, I just restarted it. I'm going to go ahead and restart it. I did a bunch of demos this morning and then kind of left it. So this application has been run a handful of times. I think it's just got stuck there, so that's totally fine. This will just take a second to restart and then we'll kind of go through that. In the meantime, I'm just going to skip ahead on the slide, we'll kind of talk about the insecure data storage and then I'll come back and do both and we'll do it that way. So for one insecure data storage. So as you probably know if you guys joined the webinar late last year, we talked about insecure data storage. We got a couple of blog posts on it as well, so we're not going to talk too detailed about it. Obviously there is the Android data directory, which is /data/data on the device, and then the app.package.name with inside that you can find things like application databases, shared preferences, cache files. There's a handful of files. Depending on your application, every application being different, you are not always going to see the same thing, but that's a good location to look. And then you have your static files within the Android, APK. So obviously if you're decompiling the APK, you have things to look for, things like strings, xml, potentially configuration files, you have potential hardcoded values. If you start using maybe tools like JADX or things like that where you're going to go look to see if anything is hardcoded, you got a few different possibilities within there. So here's our device here. So we're going to go ahead, I kind of look through this here. Any of my applications are running right now. Give me one second.

(15:06)
Brian Robison: Go back to your devices and then come back in. I've seen screeners issues.

Steven Smiley: There we go. OK, so this is the application Corellium Cafe. So you have a login screen here and by default you don't know the creds, you don't actually know how to log in, so you can actually continue as a guest. Now you got a couple options. You got a coffee, latte, cup of tea, you have your order screen. You also have this blog screen as well. So we do have the Corellium blog, which is launching within a WebView within the application, so you can see that as well. And then yeah, you can go ahead and just add any of these. So you go to a cup of coffee, you can add the item to your cart, you can add a handful of things to your card if you want. You can go and you can scroll through all these items. Whatever you have here, you can go ahead and check out. And then it's going to prompt you obviously for some input, some various pieces of information so we can go test. It doesn't really matter, I'm just going to put some fake digits in there. Continue to payment. It's going to prompt you for your payment information. Now obviously this is a test application. Don't put real credit card numbers in here just as the FYI. So you're going to put that information in there. You're going to put an expiry, CSV, and then you're going to put your payment zip code.

(16:40)
You're going to go ahead and review the order. You'll be able to scroll down and see all your items that you have here. If you have additional items with your total, you got a promo code. If you were given a promo code, you can apply that to try to get a discount or try to get it free. And then you can go ahead and submit order and then it will submit the order and say thank you,

and then you'll be back to the main screen. So a fairly simple application, but there are a lot of vulnerabilities in here. There's a lot of stuff that you're going to see in various public applications that you'd be kind of aware of. You'll see these in other merchant applications, even potentially banking applications. Any sort of application should have some of the vulnerabilities that we kind of have built into here as well.

(17:23)
So on the data storage stuff, which we kind of briefly talked about already, we'll go over and we'll actually do just a quick demo of that just to see what is there. So if you go to the files tab, obviously within Corellium it makes this super easy. You can go to the files tab /data/data like I mentioned, and then go to the package name, which in this case is com.Corellium.cafe. So you can see there's a bunch of stuff in here, things like shared preferences, which has a thing called customer data, which sounds bad. So you can actually view this right inside. So you can actually go, look, there's nothing too crazy in here just from a security perspective. There's first name, last name, and phone number. You probably don't want to be hard coding the phone number or putting it somewhere there. So it is a risk, maybe not the highest of risk, but it is a vulnerability. Nonetheless, there are some other things in here. You can have databases, customer payment, which has, if you go into that, it has the credit card data in there, which we can open as well.

(18:34)
So won't actually, this is just opening in a text editor. If you open it with a database view, you'll be able to see this. But the credit card number is there, it's just tables and with all the credit card numbers that have been entered since the application was installed. So if you actually delete the application, all that gets cleared out so you won't actually see any of that data. Obviously having an unencrypted database in plain text is a bad thing. So things like that. There's tons of stuff in here, cache files, WebView files, you've got random files, promocode.text, there's just lots of stuff in here to kind of go through. So using Corellium obviously a big benefit to kind of go through some of these files a little bit quicker than maybe doing it manually with a physical device, but all sorts of vulnerabilities in here relating to that local storage.

(19:25)
Now this is just obviously in the data directory. Now we did talk additionally about the APK and finding files within there. So I'll just switch over here. I do have the Corellium Cafe APK here. So what you can do is if you guys have never done this, APKTool just decompiles the APK and makes everything readable. Really good tool if you've never used it. So you can use APKTool, it's going to decompile the application. You'll see it here. It's going to create a folder called Corellium_Cafe with all the files. So you'll see a couple things in that folder. So if you actually, I can show you here, you can see everything's in here. The whole directory, your Android manifest, all your assets, all your code, everything you need is in there to go through. One thing that I will go through is, sorry, one sec.

(20:19)

If you go to /res for resources and then values, you could find a bunch of files. But this one: strings.xml is very important in most Android applications. It is where you store a lot of hard-coded strings for an application. In a lot of cases that could be nonsensitive data, things like error messages, various little screens and popups that are used within your application that are being called. There's a whole bunch of things that could be stored in there, but when used incorrectly, they could be storing hard-coded things like API keys, credentials, various things like that which you can find. And that becomes an issue. So if you actually look at the strings.xml file within here, you can find things like this which has a test email and password, which is actually the login creds for the application. So using stuff like that, you'll be able to find that and actually try to log in. So we'll come back to that in some later stuff that is actually used to trigger another vulnerability.

(21:23)
We'll come back to that in a minute. OK, insecure communication. Corellium Network Monitor. If you've never used it, it's a really good tool to quickly observe network traffic from an application. HTTP, HTTPS, even if you have certificate pinning for most cases, it's going to bypass it and you're going to be able to see that traffic. Now, it's not an intercepting proxy, so you're not going to be able to do all your API analysis or all your backend testing that you would do in a full pen test, but it is very quick to look at maybe the endpoints that are being used from your application, what traffic is being sent to who, is it being sent to a third party, is there are too much data being sent, what headers are being used in both the request and response. Potential information leakage: is the application responding with too much data in some situations that it doesn't need to, things like that.

So in this case of this application, it doesn't. We just have one network call that you can actually capture. So if you actually go ahead and try to purchase—I did turn on the Network Monitor by the way, but if you go ahead and try to purchase something like a latte here, for example, I'm going to go check out and I'm just going to put in some random details here. I think I use the same phone number, but that's fine. So we're just going to enter in some random details here and try to enter in a credit card. So when you actually submit that order, it sends a post request to Corellium.com with the credit card number in here.

Now, in a lot of cases, it’s not necessarily a vulnerability to have the data sent in the body of a request. That's actually kind of common practice. Obviously they can encrypt it or they can do something additionally to protect that data. In this case, our goal here was just to kind of show that there is the potential to quickly look at some of the data that's being sent, both the requests and the response and see the data that's in there. And then you guys can decide for yourself, obviously is it an issue? Do I need to use Burp or another tool to do more analysis? What do I need to do from here? And I think we had a question there, Brian. The certificate pending that Corellium does is on iOS. Is that on by default or part of the implementation? Is there a way to turn it off temporarily? Brian, do you know from the product side, there's no way to turn it off currently. I don't know if there were plans to.

(24:17)
Brian Robison: Yeah, there is. It's available through the API interface. So if you're using the CLI, you can disable and re-enable a certificate pinning through the CLI while you're doing a test.

Steven Smiley: OK, so there you go. It is possible to do that as well. Obviously if you go outside as well and you go into Burp or something like that, you're not going to have the certificate pinning bypass, so you also have it off. So there's a couple ways to do it depending on obviously your configuration, if you're using the CLI, you're using Burp, you're using a different tool, what all you're kind of using as well. OK, so common misconfigurations.

(25:02)
Steven Smiley:
Things like AllowBackup, which in itself might not—it depends on the application—if that's a vulnerability, AllowBackup just means the Android operating system will allow a backup of the app data directory and that could either be stored in local storage or in the Cloud. This obviously becomes a bigger issue when sensitive data is stored. Now if you look at this application, we're storing credit cards in a database. That's a problem. That's a big problem. If you're allowing backups of that data, that's an even bigger problem. Now you have potential credit card data being leaked to a database and being stored in a backup somewhere. Those backups can get anywhere. So there's all sorts of issues. That in itself is a vulnerability. You can obviously look for that just in the Android manifest and see that. Another one that people might think about but might not is when you back up an application on Android by default, that actually takes a screenshot and stores that locally.

(26:03)
If you use the flag_secure option in development, that will actually stop that. It will stop screenshots on Android. And you might see that if you've ever been in some financial applications and you try to take a screenshot, they might actually stop you. And if they do, it's kind of to protect from this, right? So if you back up your application, no sensitive data is being stored. So if you look on the login screen here, you can unmask the password. So if someone backs up the application when doing that or the credit card data and backs up the application, that screenshot is going to be stored locally and the /data/system_ce/0 directory, there's snapshots in there and it's all going to be stored. So those screenshots get saved. There is potential for obviously information leakage. It's not a crazy vulnerability, but still one of those misconfigurations that might get overlooked, it's kind of easy to check for.

(26:58)
Another one is debugging code. So debug code is left over in a lot of applications. This happens quite often, especially applications that need a lot of troubleshooting as they're going through, they're turning on the debug logs, they're checking for everything, they're going through everything they need to, and then they end up leaving that on when they go to PROD and they finally get stuff fixed and working. So this happens quite often and using tools like Logcat for one

to just see what is actually being logged by the application, is anything sensitive being logged in any way, username, password, any sort of information that all can be looked at. Obviously from a reverse engineering perspective, you can obviously look at the code and see if any debug stuff has been left on as well or any functionality enabled based on it being debug, things like that. So there are a handful of things you could do as well. But Logcat obviously being the biggest one when looking for data. And we'll kind of look at it in this situation as well.

(28:01)
So if I look, I'm actually going to…and you can actually specify Logcat to a process ID if you want. In this case I'm just going to run it. It looks like a lot, but it slows down. So now what I'm doing is I'm actually logging everything from Android right now. So if you actually go through and try to add a cup of coffee, w
We're going to go through this one more time. I'm just going to enter some random data here and you won't see anything in the logs yet, but what you will see is if I do this…one second. OK, so there we go. So I'm actually going to stop this.

(29:04)
If you look right here—and that might be small to see versus the other stuff, but what you can see there is it's actually logging the credit card. So as soon as the credit card gets entered in, it gets sent to the logs as well. This was in an average application, there might be a situation where something wasn't working correctly and they decided to submit it to the logs to see where something happened. Were they missing something, was there an error happening either before or after some certain functionality, and then they just forget to take it out. So that's usually the case in a lot of applications. It's good to look for and obviously in this case, they decided to log the credit card number, which is a big issue.

(29:53)
Android activities. Maybe something that not everybody thinks about, they see a lot of activities in an Android app. Sorry, there's another question as well.

(30:10)
Since Drozer is dead, have you guys got a solid way of automating content providers, receivers services, checks with Frida? We're going to look at actually an activity right now and fixing that. Drozer, it was a good tool. It is definitely outdated. It is not being really used anymore when it comes to a lot of this stuff. It's more manual honestly, at the end of the day, looking for things like what I'm going to go over now for activities, content providers, custom URL schemes, just kind of looking in the code and seeing how it's being utilized. Is there a way you can use ADB to potentially do anything malicious? Can you use broadcast receivers in any sort of malicious way? How is the code utilizing them? Are they using it to, you know, one common practice is using it to go between two different applications. If you use maybe a second factor of authentication, they're using a token or something in another app, sometimes you can capture those or create another malicious app that uses the same name and kind of captures that data.

So it's a lot more manual effort since kind of Drozer gone, but definitely still possible to do some of that stuff.

(31:27)
So for this one, If you look at the Android manifest, you can check for exported activities and then use a ADB to interact with those activities. Now in some applications this could be hidden admin pages, extra functionality, it could be various things, all hidden behind activities. Now you should also obviously be checking for broadcast receivers, intense custom URL schemes, potentially dangerous permissions, all things should have registered typically in the Android manifest. And then you can obviously use reverse engineering to look at the code as well to find additional use cases or how they're using it and things like that. So let's take a look here. I actually have a screenshot here so you can see a bunch of activity and this is actually the Android manifest. I can actually bring this up as well.

(32:13)
Let me launch this here. This is the Android manifest from Corellium Cafe. So you can see here, I'm going to try to make this bigger maybe. There we go. OK, so you'll be able to see it. So we have a bunch of activities here. If you notice almost the entire application is built based on activities. So you can see order, cart, WebViews, descriptions, and most of these are false for exporting. So they're not actually, you can't do much with them, you got to kind of figure that out. So there's two that are true—one for the MainActivity and one for the SecretActivity. And if I was to guess just a random guess, I would probably think the secret activity is the one you'd want to target. So for one, let's just kind of copy this so that I have the name and I don't need to do that again. I'm actually going to close out of this and go over here.

(33:10)
So if I'm in the application, I'm actually just going to submit that order so that I can just get to the main screen. If I'm here, I can run things. So typically, most people, if they've used a physical device, they'd be used to using ADB first and then running these commands. But obviously we're right on the shell, so I'm just going to go ahead and do it. But if you do am start, which is the Activity Manager, you can specify the app com.corellium.cafe. And then, so what I did was com.corellium.cafe and then I did / and then I did com, what that secret activity was. And I'm just going to go ahead and run that. And then I'm missing something.

(33:58)
I meant to do -n there. I totally forgot that. So what it'll say is Starting: Intent, and if you look at the screen, there's this credit card history. So you can see a history of all the credit cards, and this is the same one that's in the database. So you can see all the credit cards and the CSV numbers of everything within the application, all hidden behind this secret activity. And you can actually delete the database as well, which has all the data in it. So that's actually a possibility and it'll clear out the database. So these are the kind of things that need to be tested for. You need to look at the activities, you need to look at the broadcast receivers, the content providers, anything within an application, Number One in the manifest. And then you can use reverse

engineering. If there's custom URL schemes, you can look in the code to see how it's being utilized and potentially use things like ADB to be able to manipulate that and to do something. Either load certain content that shouldn't be loaded, maybe inject certain things like URLs that shouldn't be and try some malicious attack that way. Bypassing security controls.

(35:10)
So obviously security controls could include anything like certificate pinning, root or jailbreak detection, tamper proofing, runtime detection, a number of things. And there's various ways to do that. There's really no correct way to tell people to do this. You can use Cydia packages, SSLKillSwitch for iOS. When it comes to certificate pinning, there's things like Liberty which are built in for iOS. For jailbreak detection, there's Frida scripts, there's manually changing the code to try to bypass that, either delete the functionality or edit it out in the small code and then repackage the app, which is kind of one option. So there's really a ton of different ways to kind of bypass a lot of these. And every implementation is different. Some are super easy, some are really hard.

(36:00)
So there's really just no correct answer here. So for one, I also want to share just the Frida codeshare if you haven't been to it. And if you don't have a lot of experience with Frida, this is a really good starting place. There's tons of scripts in here that work out of the box, some don't, depending on the functionality, but some obviously do. So you could use these, pull 'em off and then use them as well within Corellium or within your own devices. So that's a good option as well. But let's look at this application for a second. I'm actually just going to go ahead to kill this application and launch it again.

(36:42)
And then we're going to go ahead and try to log in. I'm going to go ahead and try to log in. And what's going to happen is it's actually going to tell me that root level permissions have been detected. So there is root detection on it when you try to log in. And yes, to answer your question Paul, sorry, it is being recorded. So it will be sent out after this webinar is finished. So there is root detection on it. Only if you log in, not if you're a guest. So you can actually click okay, it's going to crash the application and then you can use tools like Frida to kind of bypass that. I can actually show you and there's no correct answer. There's a bunch of Frida scripts that you could use, you could use some other stuff too. You could patch out the application. Every one is a little bit different when they decide to kind of do these sorts of things. So what I'm actually going to do is I'm actually going to—Brian, can you just mark that question as completed?

(37:55)
Thank you. So what I'm going to do here is what I did is I actually hooked the process, the cafe process. You can see here it's actually hooked. I got a prompt there. So what I'm going to do is I'm actually just going to do this root detection. I think it's this one. I'm going to go ahead and execute this one. Just want to make sure that's the right one. And then I'm actually going to try

to log in and then what's going to happen is it's going to try to prompt here and then you're going to log in again. So what is happening is it's actually looking for various things within your application. And I can show you the script just to see what it's looking for. What'll happen is they'll actually log in. So you can actually just click wait and it'll work, and then it'll fully bypass that root detection. So now I'm actually logged in. You won't notice any different functionality within the application. It's just mainly to show that root detection can be kind of bypassed fairly easily. And if you actually look at this one, I won't go through the entire script, but what it's looking for is various root packages that are quite common. Things like chainfire, there's a whole bunch. Hidemyroot is quite common. Xposed framework. There's a bunch of stuff, a bunch of root binaries: su, busybox, supersu. A lot of the common stuff that you use is basically looking for those within the application and returning false—those aren't there when the application kind of looks.

(39:24)
OK, so that's that. We're on the last section now. Lemme just check here. So last section, which is the custom Frida scripts. So there's two vulnerable spots within the Cafe application. We're going to launch a poll question here because I'm actually curious, as you guys have seen the functionality of this application, can anyone actually just guess which functionality we should target with Frida? There's actually two spots. I mean feel free to guess two or just one if you want to. But we're going to throw up a poll question just to see where everyone's mind is at in terms of where do we go from here to attack with Frida, where do you see potential malicious spots?

(40:18)
There we go. I think it's just launched now. So I'll give everyone a second to kind of answer that. While everyone's answering. We will talk about how do we go about writing the scripts for Frida. There's a couple different options. One, you are going to go through reverse engineering. You're going to look at the code, you're going to identify areas that you feel are vulnerable for whatever reason, and then you could try to write scripts around that. A lot of this is kind of trial and error. You're going to look at it, you're going to try to write scripts, various scripts are going to fail, you're going to get errors. There is kind of a lot of work to put in when you're doing this sort of attack. Obviously some various tools like JADX can be very helpful on Android, Frida-trace depending on what you're looking for or just the Frida-CLI. You're going to write the scripts and you're going to see what errors come out and things like that. How much time are we giving on this question, Brian?

(41:16)
Brian Robison: Just a few more seconds. There was one suggestion I saw that was also mentioned in the question.

Steven Smiley: Yeah, I just saw that.

(41:23)

Brian Robison: All right, I'm going to go and end the poll.

Steven Smiley: Do we get to see the answers or do you get to see the answers?

Brian Robison: You should be able to click on view details.

Steven Smiley: I cannot.

(41:43)
Brian Robison: I see promo codes, fingerprint bypass, any input functionality, passing of variables, certificate pinning, root activities.

Steven Smiley: Got it. Yeah, no, I mean good answers for sure. The promo code was a big one because it's a good point. In any input field you do want to obviously target input fields. If they're coded incorrectly, you have a lot of opportunity to use Frida to manipulate the data that's going in there. So promo code was a huge one and it is actually vulnerable, so we'll actually go over that. So yeah, it's good to see obviously the security controls, we did that with the root detection, so that was kind of fun. So yeah, I mean there's no wrong answer. When you go through a test you're going to go through a lot of different functionality. But yeah, I was just kind of curious what people thought about that. So let's go ahead and look at some things to see if we can solve this. So for one, the promo code.

(42:39)
So let's go ahead and start here. One, if we go to files, this thing called promocode.txt. Now I'm actually just going to view this in here. It's easier. Oh it can't because it's actually encrypted. So if you actually launch this, you see this and that means absolutely zero to everybody or should mean absolutely zero to everybody. So it is encrypted. You're not going to be able to get that from the promocode.txt, although it did kind of look fancy and a really good opportunity to just kind of pull it out of there. So you're not going to be able to. So one thing we need to look at is start looking at the code. So obviously a great opportunity to use JADX and we can—lemme just go to this tab—so if you've never used it, JADX is a fantastic tool.

(43:29)
It's just jadx.gui and then the APK file. So this will just launch JADX, you can see all the code. It's going to obviously take a look at that really quick so you can see all the codes. This is the code for this application. There's not a ton there. It is a smaller application as you kind of saw from our demos. So if you go to com.corellium.cafe, you can kind of see this is where everything kind of starts and there's a bunch of stuff in here. Now for this one, ui.activities. So there's a bunch of stuff in here because what you saw initially in the Android manifest was that there was a bunch of activities, order activity, WebViews, review orders, there's a whole bunch of stuff in there. So everything is kind of done through these activities.

(44:15)

Obviously the main activity being the start of everything, you could see the root detection was in there and you could see a handful of things. But looking through all these, and this takes some searching and I'm going to obviously skip just in terms of time and everything like that—but typically you would use JADX to be able to search for some key values, search for what you're looking for, promo code, things like that and kind of take a look. So OrderReviewActivity is the one I believe we want to look for. So there's a bunch of things in here. So if you look through the code—I'm not going to go through everything line by line—but you can see they're using cryptoManager, they are doing some sort of encryption. As you kind of scroll through, you're going to get more and more information about what they're doing.

(45:02)
You can see here promocode.txt. So like I said, initially you're going to be running some checks, you're going to be doing some searches and things like that. Things like promo, if that's what you think. So you will be brought to this file as you do your searching. I just got here a little bit quicker. But yeah, you're going to search through, there's a whole bunch of things in here. There's a successful decryption you can kind of see here about what's going on. So this is the part that you really want to pay attention to. There's a decrypt function here that's using the cryptoManager and they're taking in two inputs to decode this. So what you can do with Frida now, one great ability, let me just go over here and I'm going to actually cancel out that. I have a Frida script here called Promo_Code.

(45:51)
So if you go to edit it, what's happening is I'm basically calling out com.corellium.cafe, ui.activities, OrderReviewActivities. That was what we were in. You're going to look at the decrypt function and you're going to pass at two variables, same as what basically the code was asking. The code's asking for two variables, file and file. They're looking for two files. So you're going to send it to, that's going to do everything that decrypt function wants and then you're just going to log it out. So I'm actually going to go ahead and I'm going to add this coffee to my cart. I'm going to go to my cart and we're going to go through and see what we can do here.

(46:51)
We're going to continue to payment and we're going to enter some false payment information just to see. It should just take 2 seconds. We've been through this a few times. So once we're here, what we're going to do is we're going to actually hook the cafe application again, you're going to see it's hooked. That's all good. We're going to go ahead and execute this Promo_Code script. It's going to execute here. So it's actually running. And then we're going to put anything in the promo code. It doesn't matter what you put in. Put a number in there, it doesn't matter. You're going to hit apply and it's going to give you the correct promo code. Even though the app says the 444 was invalid, you could put the correct one in 39 4210064.

(47:59)
So you're going to see that and you're going to apply that and you're going to see the total go to free. And then you could submit your order. Now this promo code, it is super dynamic. It

changes every time. If I relaunch this application, that promo code will no longer be valid. This changes all the time. So you need to use Frida to hook that and grab that. So this isn't a hard coded value, it's not something that you could just quickly look at the code and grab the promo code and figure out what it is and put it in there and it's good for forever. This more resembles an actual application. With it being dynamic, it's not going to be the same value all the time. You're going to have to hook it in order to get that. You're not going to be able to solve this another way.

(48:44)
But yeah, that's just a really cool opportunity about just kind of reviewing the code, looking for a script and looking for some various possibilities there. So I actually have one more Frida one. Do we have time? Yeah, it's 10:51. So we'll look at one more. So another functionality that I don't think anybody mentioned, but I think we only briefly looked at was we have this blog post here. So if you go to this blog page, it actually launches in a WebView. So WebViews tend to be super malicious. There's a ton of different ways to attack WebViews because people open them up. They usually enable JavaScript, they enable all this functionality within the application just for ease of use. So that's one thing that we can kind of target as well. I'm actually just going to relaunch this application after we already used a rescript just to make sure that everything is all good. And then we'll go look at the code. So let's go look at the code. So there's a few different activities here.

(49:46)
There's a thing called WebViewActivity. So if you actually kind of look here, when we start looking at the WebView and realizing that the blog is obviously in a WebView, we start driving our attention to more of the WebView content within here. And we're going to start looking through. So there's kind of a bunch of things to look for if you start scrolling down, but there is one that becomes super interesting, which is this setup WebView, which is taking a string value. And if you look at the bottom here, they're loading a URL, which is the string. So we can use that information to be able to build out a simple script that's going to give it a URL so that we can kind of change the WebView in a malicious way. So let me just restart the device.

(50:34)
I want to make sure everything was all good from everything else we've done. So we'll just give that two seconds and we will continue to write that. And by the way, if you're looking for ways to search this, obviously within JADX, if you look here, if you notice in the app, it's going to launch the Corellium blog. So you can look for HTTP, HTTPS, things like that within this code to find some areas that might be calling that URL to potentially figure out that this is the class that you need and this is the functionality you need in order to modify that and change it to something else. So let's go ahead here. The device is up, everything should be all good.

(51:18)
Now we're going to launch the Corellium Cafe. We're going to continue as guest so I don't have to log in. I'm going to launch this blog which launches the Corellium blog. So what we're going to

do, if we're actually going to go look at the script first, but if you look at the script, it's very similar to the last one. In a way we're going to use this vulnerable WebView. We're going to use the Java, we're going to call out the WebViewActivity, which we kind of just found within JADX. We're going to give it one variable, which is that string value that it was looking for—the URL instead of two values, which you saw in the promo code. And then we're just going to set up WebView, we're going to provide it a URL, and then we're going to go from there. So we're going to quickly hook this application. We're going to attach it there, you're going to see it in the console. Everything is all good.

(52:15)
We're going to execute this WebView, it's going to load, we're going to click the Corellium blog and it is going to play a YouTube video instead of the Corellium blog. So that's just a really cool way to be able to maliciously change a WebView to something else. And you'll see WebViews in a lot of applications. They're used quite commonly and a lot of times they're not protected. There are some more advanced tasks you can do with WebViews using maybe the API or using some sort of man-in-the-middle attack. But this is just a really good way to be able to have a script that can run, that can obviously redirect. And to the user, it's no different. They launch the application, they click the WebView functionality and it's going to load a page. So if you're trying to be really malicious, you're going to load another login page or something third party. Obviously when it's in a WebView, you're not going to be able to see HTTP, HTTPS. So you could send them to an HTTP site that looks like a login page. They're not going to be able to know they might enter their creds and now you've stolen those creds. So there's just a ton of use cases in terms of WebViews. But yeah, this is just a cool example of just how you can write a simple Frida script just based on some of the code that you looked at.

(53:35)
And that's really it. We're coming up to time. So we made good time on that. Was there any questions? I'll leave it open to anybody else who has any questions remaining. We got a couple of minutes; we can answer any of those. But yeah, in terms of the vulnerabilities in the app, that's our quick walkthrough.

(53:53)
Brian Robison:
Excellent. Thank you Steven. And like Steven said, if there's any questions, feel free to ask them now. Keep an eye out for the announcement of the next webinar that will be coming up in March, probably about this time in March, third week or so that we're going to talk about mobile malware and mobile threats. So if there are no further questions, it doesn't look like anybody has any questions. Great turnout today. Thank you to all of you for coming and attending today. We'll go ahead and end this webinar today. Thank you, Steven, again for your time today and all of you for attending. We really appreciate it and keep an eye out for Corellium out there in the news and we look forward to working with you some more. So everybody have a fantastic day. Thank you very much. And take it easy.