"THE SAVAGE CURTAIN"

(Mobile SSL Failure

Stardate: 92492.76

Who are these guys?

Penetration Testers at LinkedIn

Tony Trummer - Staff Security Engineer aka “SecBro”

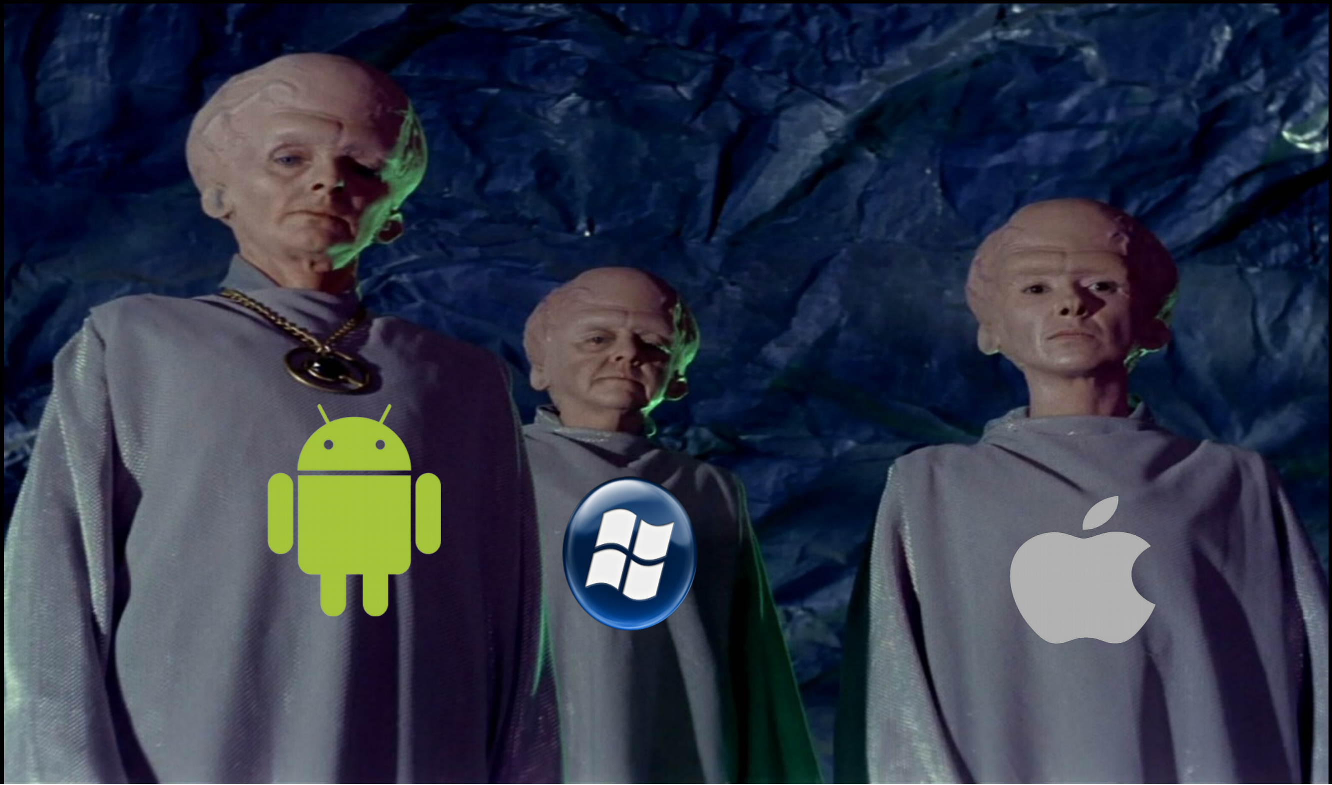
Tushar Dalvi - Sr. Security Engineer & Pool Hustler

A Private Little War

Our employer generally does not have prior knowledge of, condone, support or otherwise endorse our research

The Menagerie

- Apps are mash-ups of native and web code
- Java, Objective C, Swift, etc.
- Developers control app security settings



BUT WHEN WE DO, ITS
GEORG LUKAS
memegenerator.net

Basics

This is probably not the site you are looking for!

TLS provides several security features

- Encryption
- Authenticity
- Integrity

In apps, unlike browsers, whether you see a certificate warning is up to the app developer.

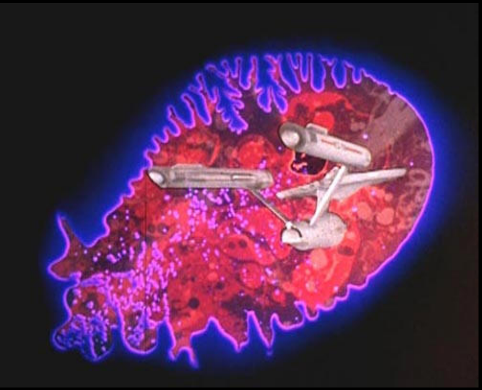
Wolf in the Fold

- TLS is really the **ONLY** protection against Man-in-the middle (MitM) attacks
- MitM is significantly easier to exploit against mobile devices

Tomorrow Is Yesterday

Before dismissing the idea of large-scale or supply-chain attacks...

- Recent reports of pre-installed trojans on low-end Android devices
- In 2013, Nokia was found to be performing MitM on customer traffic, reportedly for performance reasons
- In 2013, reports surfaced claiming that the NSA and GCHQ ("Flying Pig") were actually performing real-world MitM attacks



Immunity Syndrome

Infosec folks often roll their eyes when they read statements on sites or in apps that tout TLS use and how big their keys are

Journey to Babel

One night, after a few drinks, we decided to test some apps, starting with proxying their web requests

Into Darkness

Damn it, Jim!

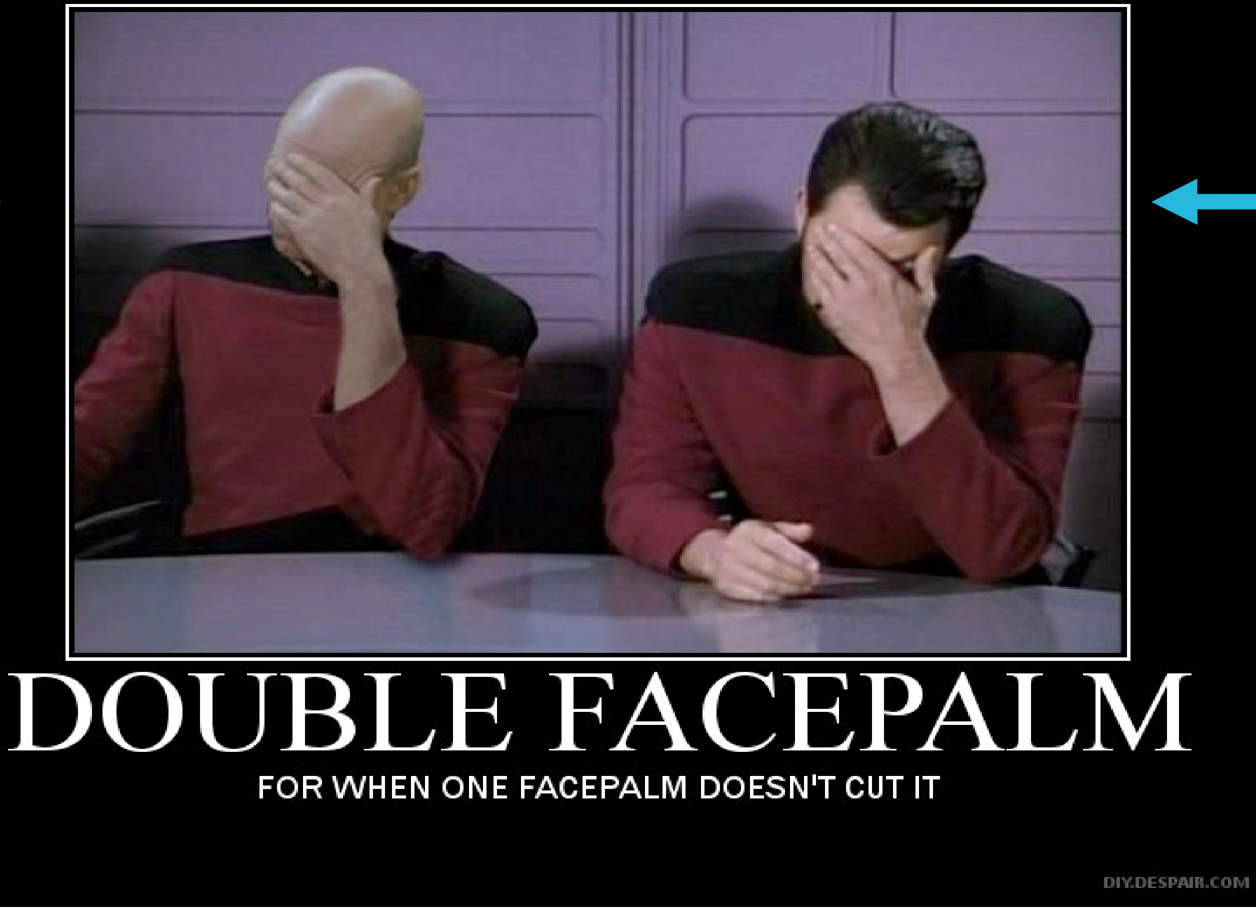
First aspect of certificate validation

The app or OS must verify the certificate is cryptographically signed by the private key of a Certificate Authority that is pre-trusted

Forget Something?

The app developers had disabled Certificate <u>Authority</u> validation

Tony → ← Tushar



aste of Armageddon

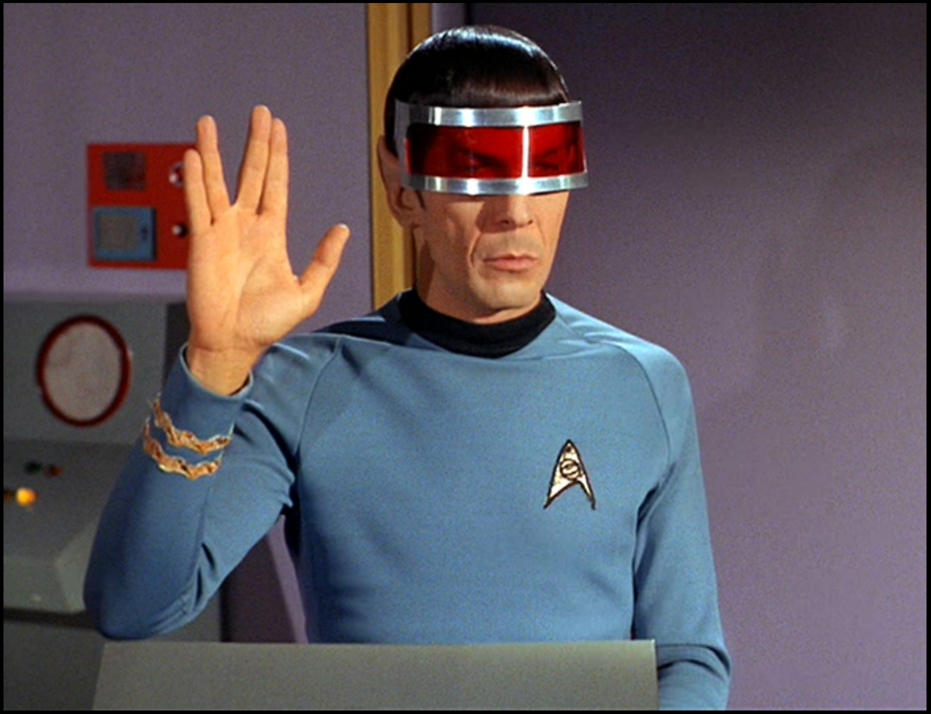
The Trouble with Tribbles

Testing for CA validation

- Configure device to use proxy
- Configure BurpSuite's proxy listener to “Generate a CA-signed per-host certificate”
- DO NOT install the proxy's CA certificate on the test device
- Verify you see a certificate warning in the native mobile browser
- Step through each section of the app
- If you see HTTPS traffic, in Burpsuite, the app failed

Second aspect of validation

Does the Subject Common or Alternative name match the hostname of the site you're visiting?



By any other na

Testing for proper hostname validation

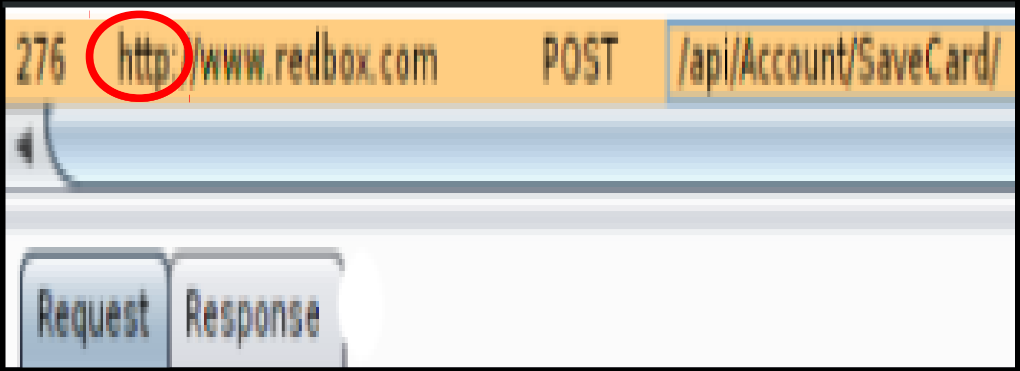
- Obtain a valid certificate for any domain other than the target, signed by a CA the device already trusts
- Configure your device to use a proxy
- Configure proxy listener settings to "Use a custom certificate"
- Verify you see a certificate warning in the native mobile browser
- Step through each section of the mobile app
- If you see HTTPS traffic, the app failed

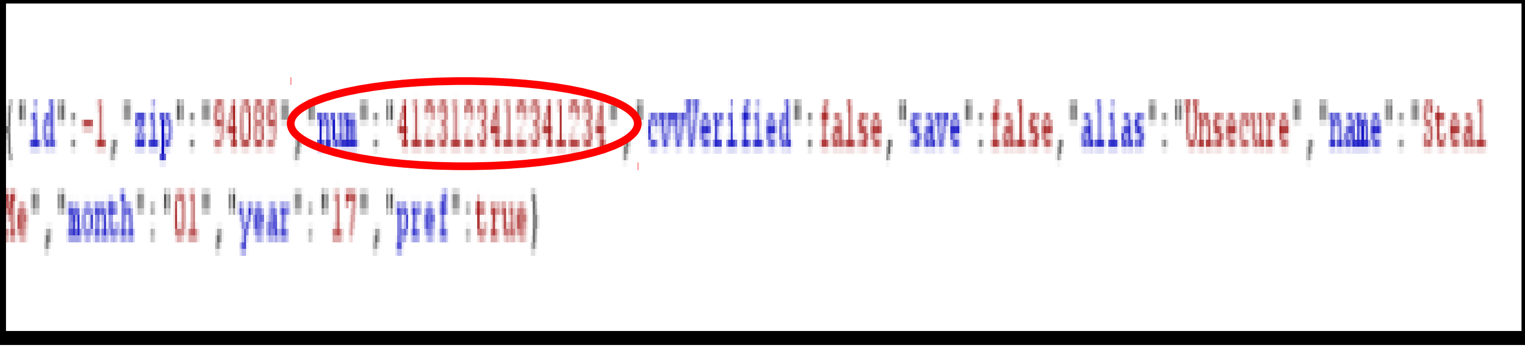
The Naked Time

- Credit card numbers (RockBot)
- Passwords, session cookies, etc.

Dagger of the mi

- Unencrypted credit card information
- Tier 1 PCI merchant
- 10 million+ installations


```
276   http://www.redbox.com   POST   /api/Account/SaveCard/
```


```
{"id":-1,"zip":"94089","num":"4123123412341234","cvvVerified":false,"save":false,"alias":"Unsecure","name":"Steal
Me","month":"01","year":"17","pref":true}
```

Court Martial

FTC vs. Fandango & Credit Karma

- One of the major flaws cited in the suit was failure to validate SSL certificates on mobile applications
- Agreed to “establish comprehensive security programs”
- Agreed to “undergo independent security assessments every other year for 20 years”
- Scolded publicly for not keeping “their privacy promises to consumers”

But wait!
There's more!
ICANHASCHEEZBURGER.COM

SSL session caching

- During the initial handshake the certificate is validated
- Subsequent client requests re-use the previous handshake and do not re-validate the certificate

The Enemy Within

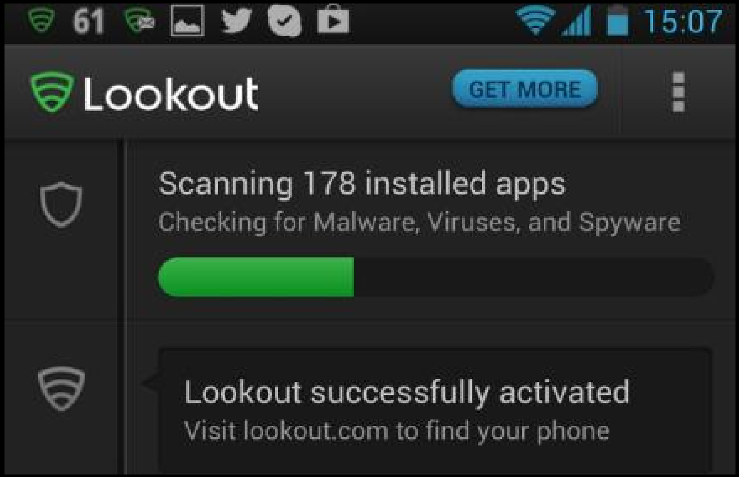
How would a bad guy get my phone?

Why is it more likely on mobile?

Patterns of Force

If I have physical access, couldn't I just...

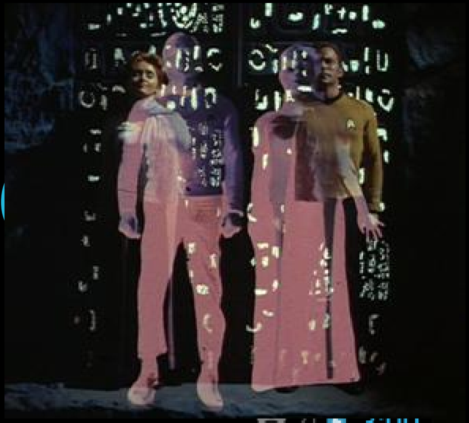
- Install malicious app
- Access your data



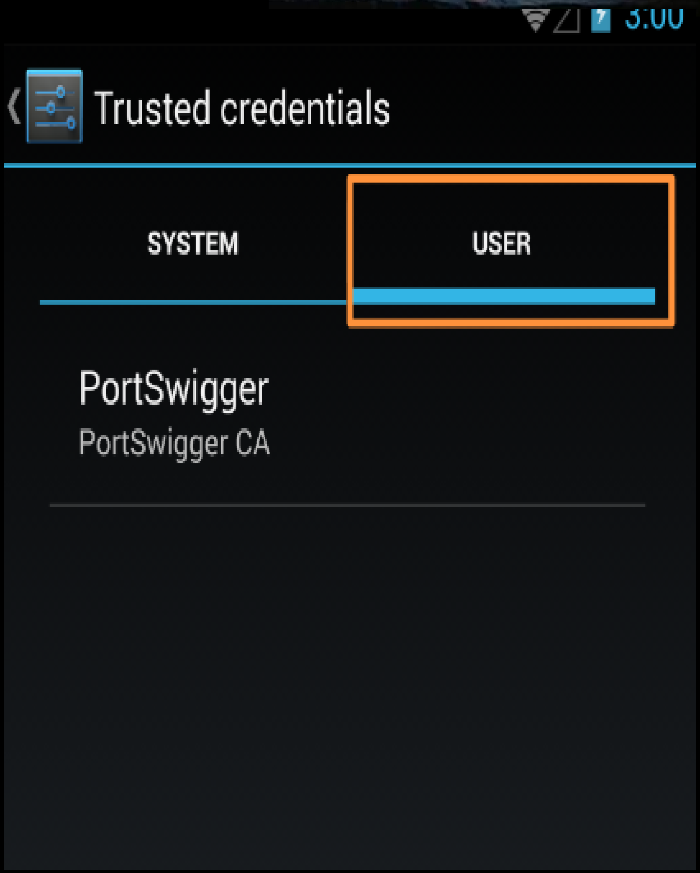
Scanning 178 installed apps
Checking for Malware, Viruses, and Spyware

Lookout successfully activated
Visit lookout.com to find your phone

Turnabout Intrud



Since SSL session caching only checks the certificate once, you only need it on the device for as long it takes you to make the first connection, after which you can delete it



PortSwigger
PortSwigger CA

The City on the Edge of Forever

- Server decides how long to accept the cached session
- In other words, the bad guy gets to decide how long to accept the cached session...
- We refer to this *feature* as "EverPWN"

TREK YOURSELF
BEFORE YOU WRECK YOURSELF

Shields Up!

{ Review your code
{ Implement policy
{ Test pre-release
{ Train developers

Shields Up!

{ Review your code

In Android, investigate these:

{ TrustManager
{ SSLSocket
{ SSLSocketFactory getInsecure
{ HostNameVerifier

In iOS, investigate these areas:

{ _AFNETWORKING_ALLOW_INVALID_SSL_CERTIFICATES_
{ SetAllowsAnyHTTPSCertificate
{ kCFStreamSSLAllowsAnyRoot

Shields Up!

- { Certificate Pinning
- { Dev and prod signing certificates are **required** to be different in both iOS and Android
- { Build a trust manager that only allows certificate validation to be disabled in dev builds.

Live Long and Prosper

Contact and testing instructions:

http://www.secbro.com

Tony Trummer:

http://www.linkedin.com/in/tonytrummer

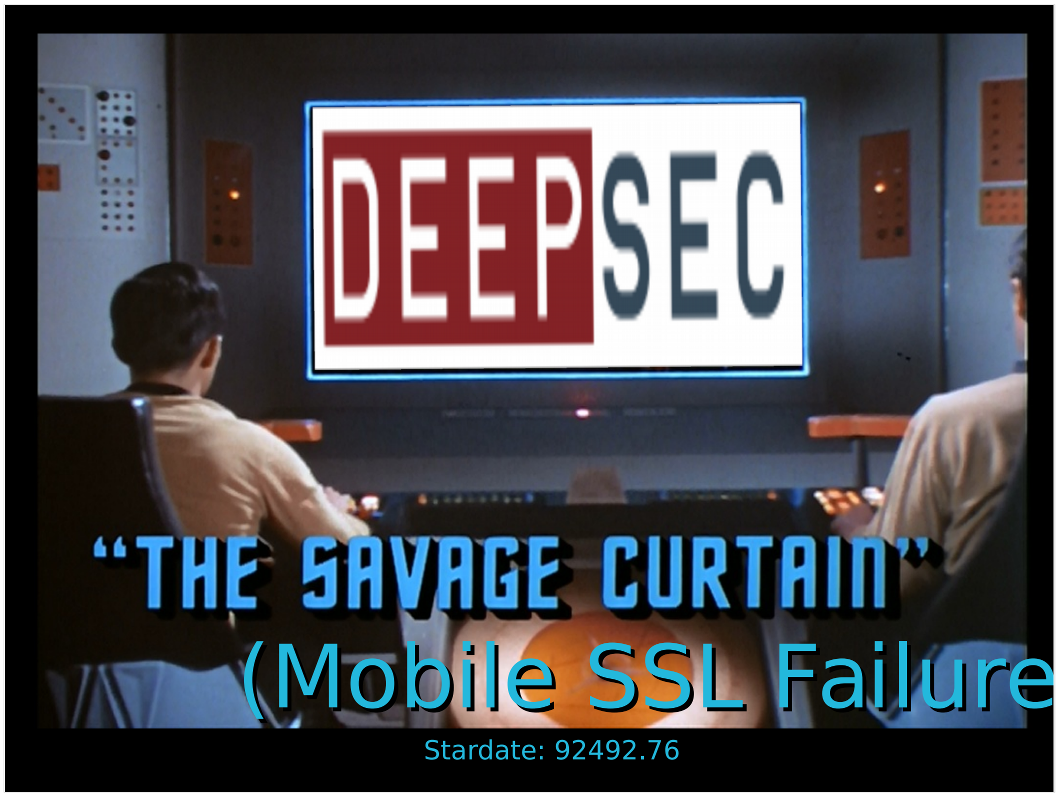
@SecBro1

Tushar Dalvi:

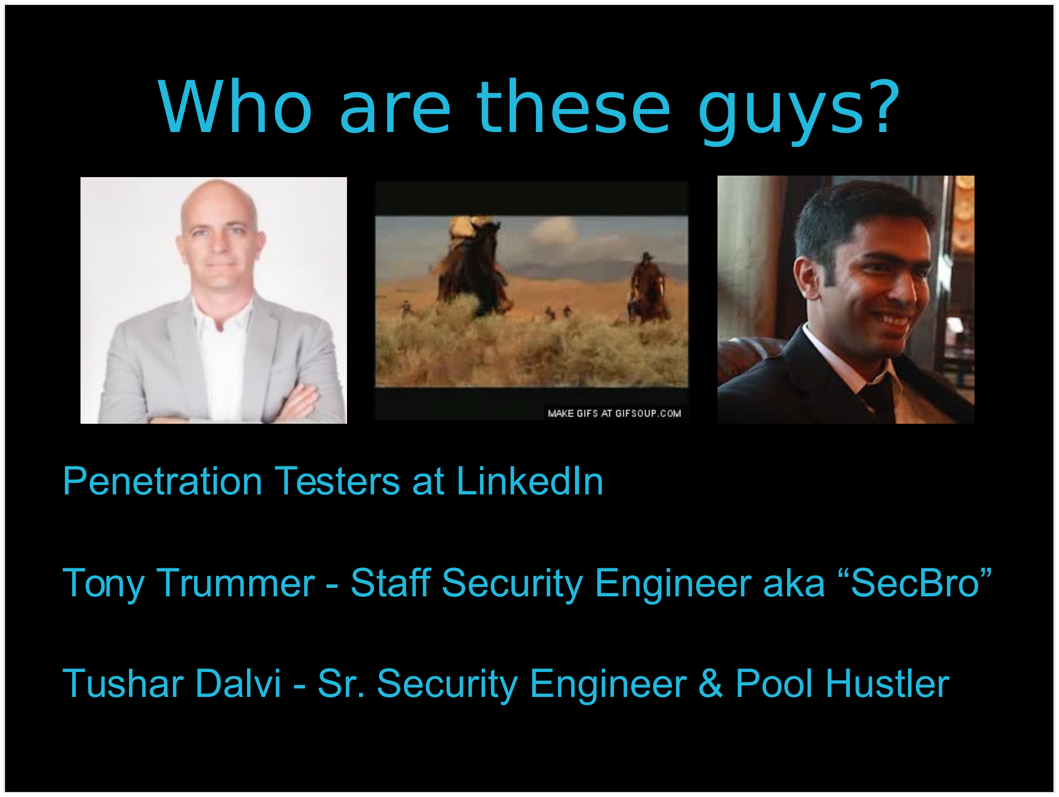
http://www.linkedin.com/in/tdalvi

@TusharDalvi

R.I.P Reggie Destin



All right people, our presentation is on the topic of mobile SSL failures. We are really excited to be here at DeepSec and this is my second talk ever at a security conference. Our plan, is to showcase, in the next 45 mins or so, some of the systemic issues we found in popular mobile applications and operating systems. Also, we will be presenting a lesser known technique of achieving almost undetectable and persistent man-in-the-middle capabilities in certain iOS and Android applications during this presentation.



Who are these guys?

Penetration Testers at LinkedIn

Tony Trummer - Staff Security Engineer aka “SecBro”

Tushar Dalvi - Sr. Security Engineer & Pool Hustler

So a little bit about our background...I am Tushar Dalvi, I have with me, Tony Trummer, we are both Security Engineers at LinkedIn, responsible for penetration testing and vulnerability research, but like most security folks, we spend most of our time herding cats.

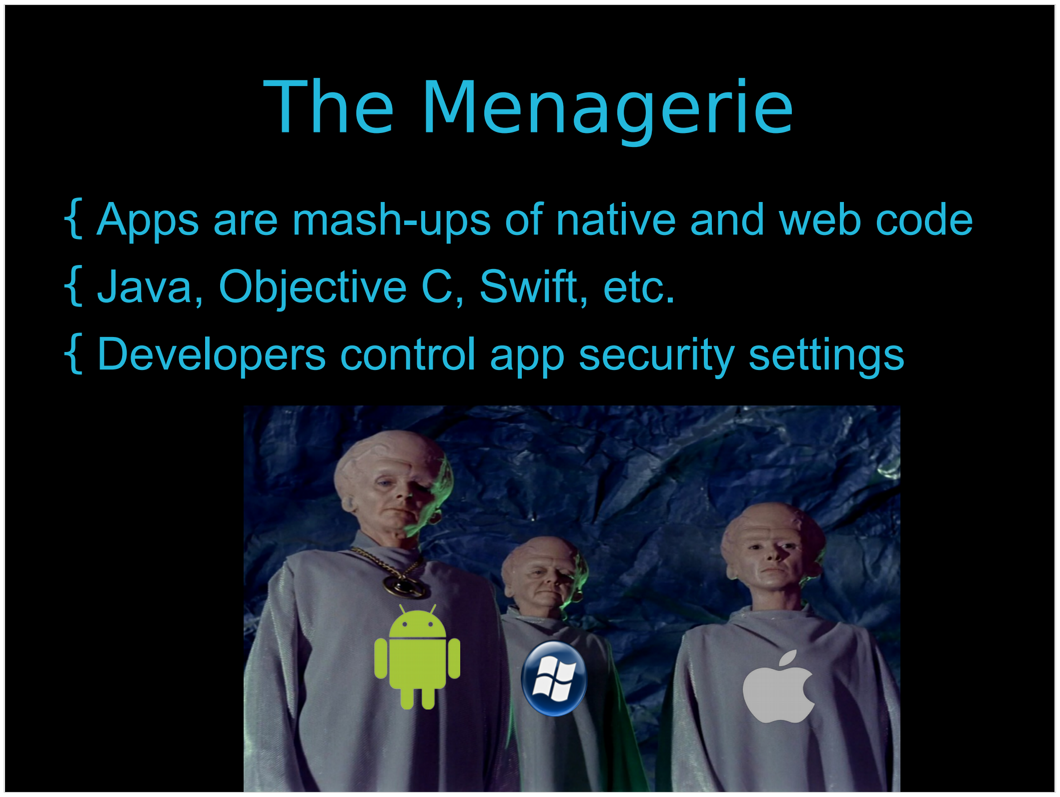
Previously, I worked at McAfee as a Penetration tester, after getting my Masters Degree from John Hopkins in Security Informatics.

Tony, is a serial drop-out, comes from a military and networking background and has nearly 20 years experience in IT.

A Private Little War

Our employer generally does not have prior knowledge of, condone, support or otherwise endorse our research

Just to be clear, this presentation is purely part of our personal research work, and our opinion does not necessarily reflect the views of our employer.



The Menagerie

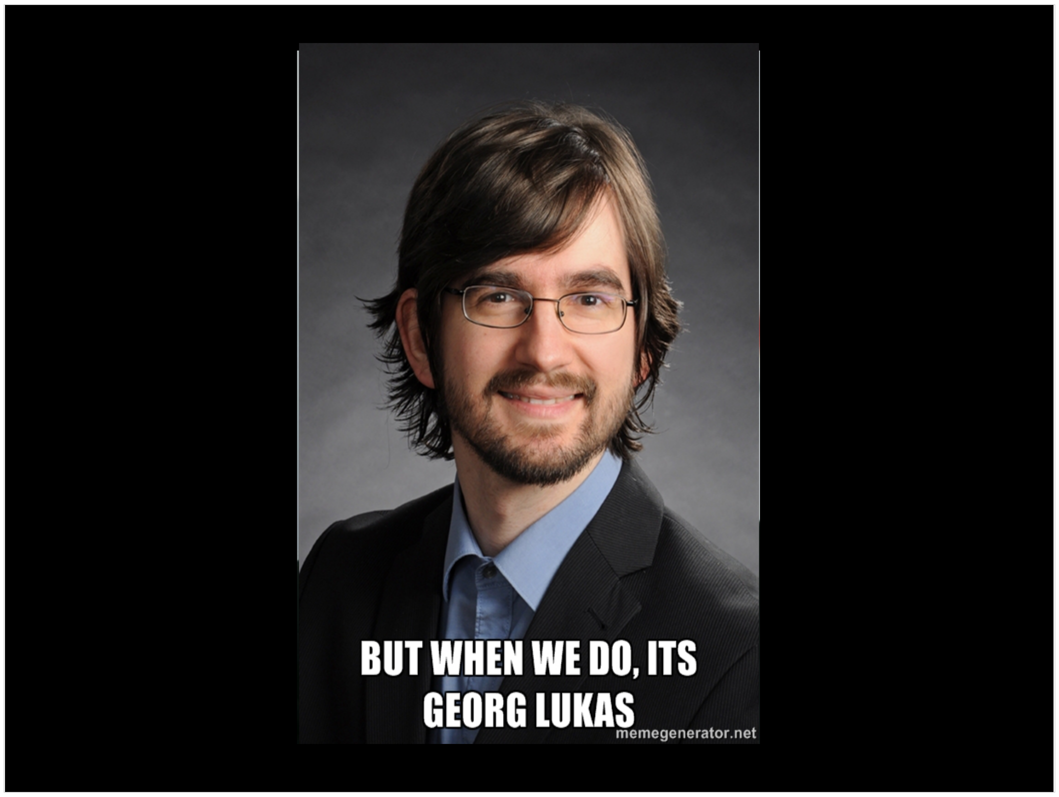
- Apps are mash-ups of native and web code
- Java, Objective C, Swift, etc.
- Developers control app security settings

Alright, the premise of this presentation, is that mobile applications have come a long way and offer so much more control to the developer to define their behavior.

In most cases, an application would be written for several different platforms; You have iOS, Android, Windows Phone and so on. Of the problem apps we'll show, most were Android, there were a few for iOS, but surprisingly, we didn't find any on the Windows Phone.

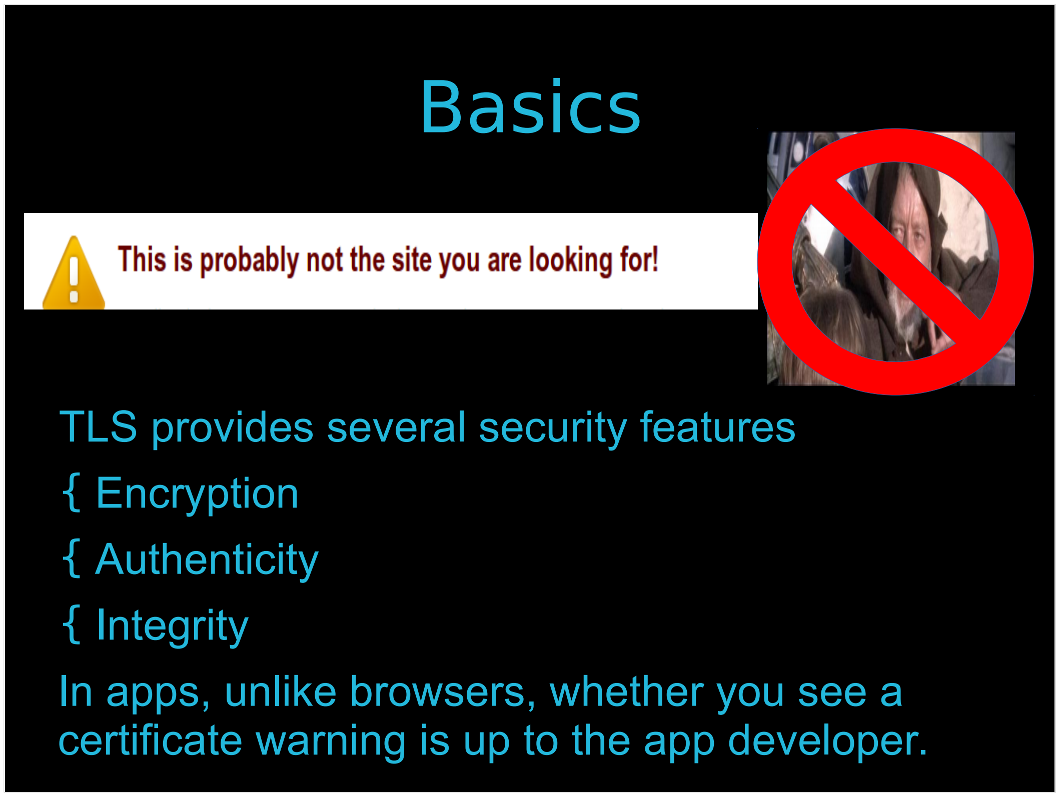
Since each one of these platforms have differences in the way they can do the exact same thing, the developers need to have a clear understanding of the implications that come with this flexibility, when trying to implement features across different platforms.

This is especially important, when it comes to security controls, such as TLS...



We don't normally mention George Lucas in a Star Trek presentation...but when we do it's Georg Lukas **(CLICK),** who presented just yesterday at DeepSec and how many of the issues he was warning about, actually have already manifested themselves as vulnerabilities in popular mobile applications.

Since many of you may have attended that talk, or are otherwise already familiar with SSL, we'll just quickly touch on the basics..



Basics

This is probably not the site you are looking for!

TLS provides several security features
- Encryption
- Authenticity
- Integrity

In apps, unlike browsers, whether you see a certificate warning is up to the app developer.

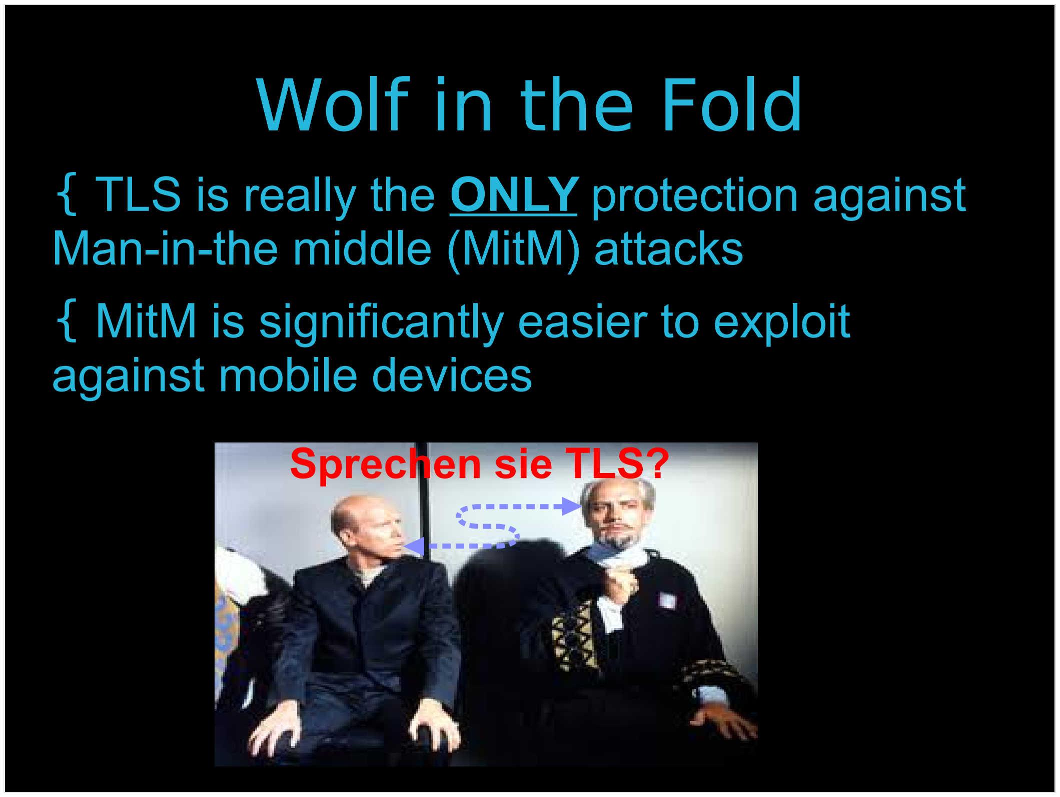
So, TSL provides several key benefits:

1) Secrecy or encryption, and

2) Authenticity, which guarantees the identity of the entity you are trying to connect to

3) Integrity – knowing nothing has changed in transit

Typically, in the event of a handshake failure **(CLICK)**, browsers show a certificate warning before allowing you to continue, but in mobile applications, this safeguard is at the mercy of the developers.



Wolf in the Fold

{ TLS is really the **ONLY** protection against Man-in-the middle (MitM) attacks

{ MitM is significantly easier to exploit against mobile devices

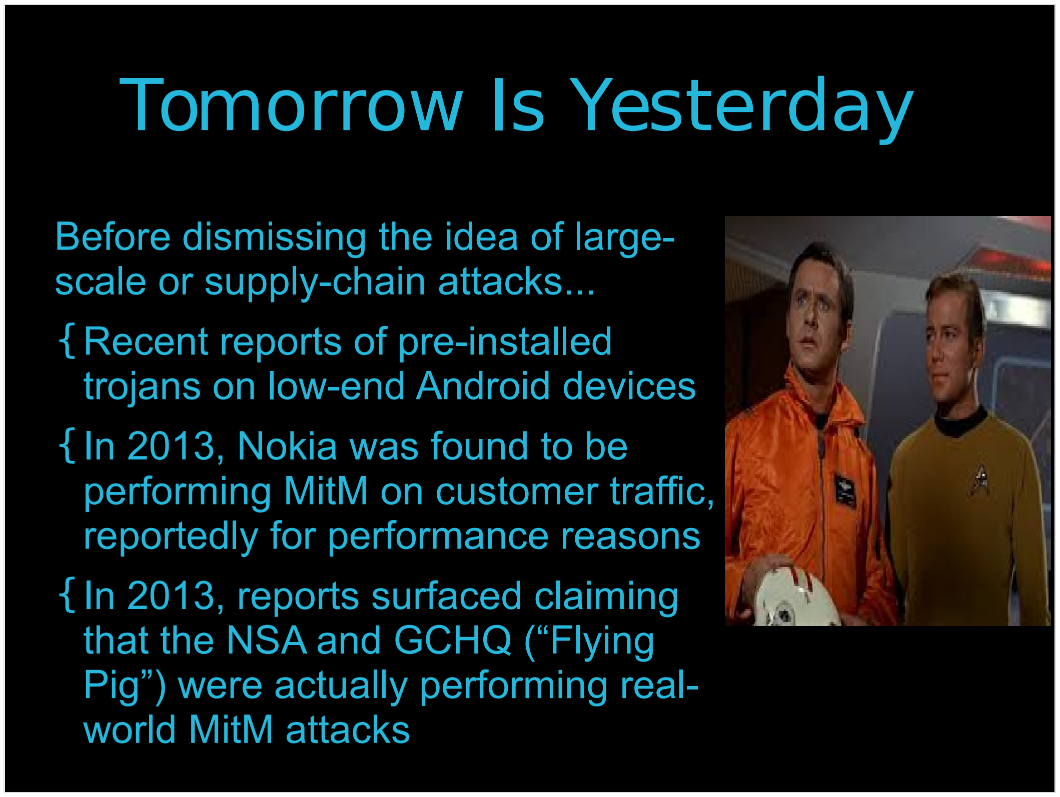
TLS is the only real protection against MITM attacks.

MitM attacks are are significantly easier on mobile devices, for multiple reasons, like:

The fact that most users' cellphone data plans are not unlimited and the connection speed is slower over cellular networks, so they often opt to use open WiFi networks whenever possible.

Cell phone providers, actually encourage this to save bandwidth. That is why, for example, your iPhone automatically connects to any SSID named ATTWIFI.

Some previous research has shown that even without you trying to actively connect to a WiFi network, mobile devices will automatically try to connect to previously known or pre-configured SSIDs, such as NETGEAR, LINKSYS, etc. Also, the Snoopy framework is designed specifically to look for WiFi probes from devices and imitate whatever SSID they were looking for.



Tomorrow Is Yesterday

Before dismissing the idea of large-scale or supply-chain attacks...

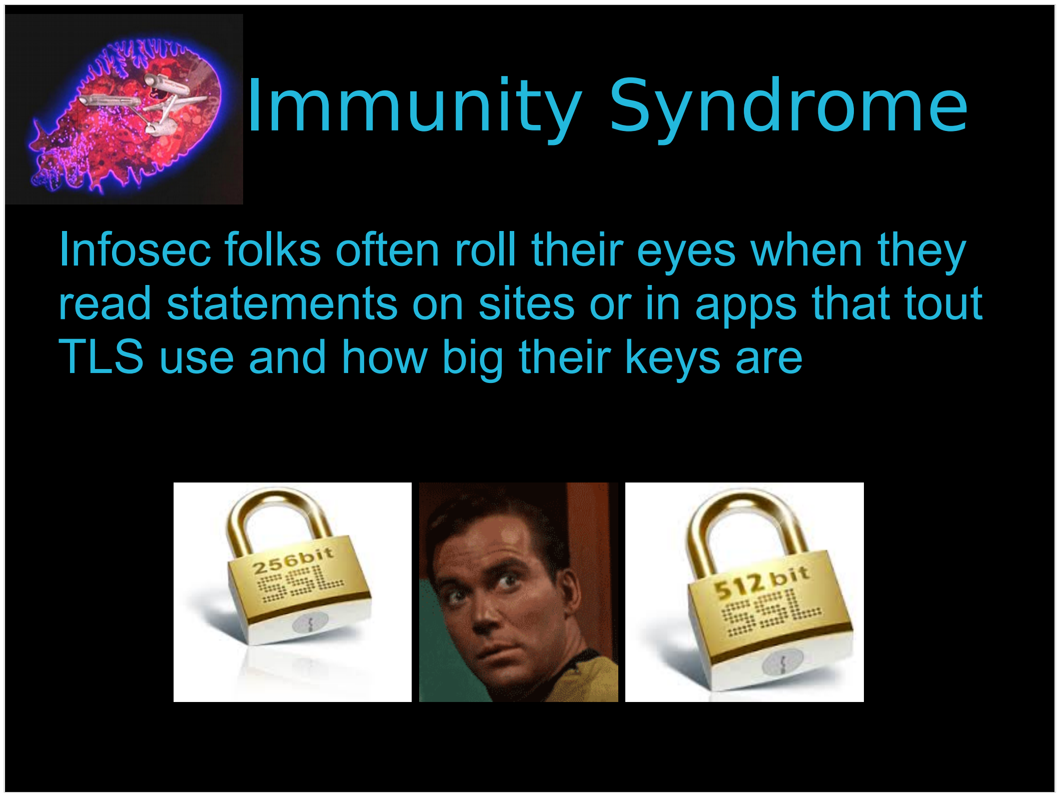
- Recent reports of pre-installed trojans on low-end Android devices
- In 2013, Nokia was found to be performing MitM on customer traffic, reportedly for performance reasons
- In 2013, reports surfaced claiming that the NSA and GCHQ ("Flying Pig") were actually performing real-world MitM attacks

Many of you may be thinking, "what are the chances of a large scale supply chain attacks taking place"?

Well, there have been recent reports regarding pre-installed trojans on low-end Android devices and as far as MitM specifically, Nokia was found to be doing exactly this in 2013 . They reportedly did this for performance reasons, but think about that for a second...

One of the leading device manufacturers, was actively intercepting and decrypting their customer traffic...

Also, let's not forget, that according to allegedly leaked documents, the NSA and GCHQ were both actually found to be using similar MitM techniques to snoop on SSL traffic



Immunity Syndrome

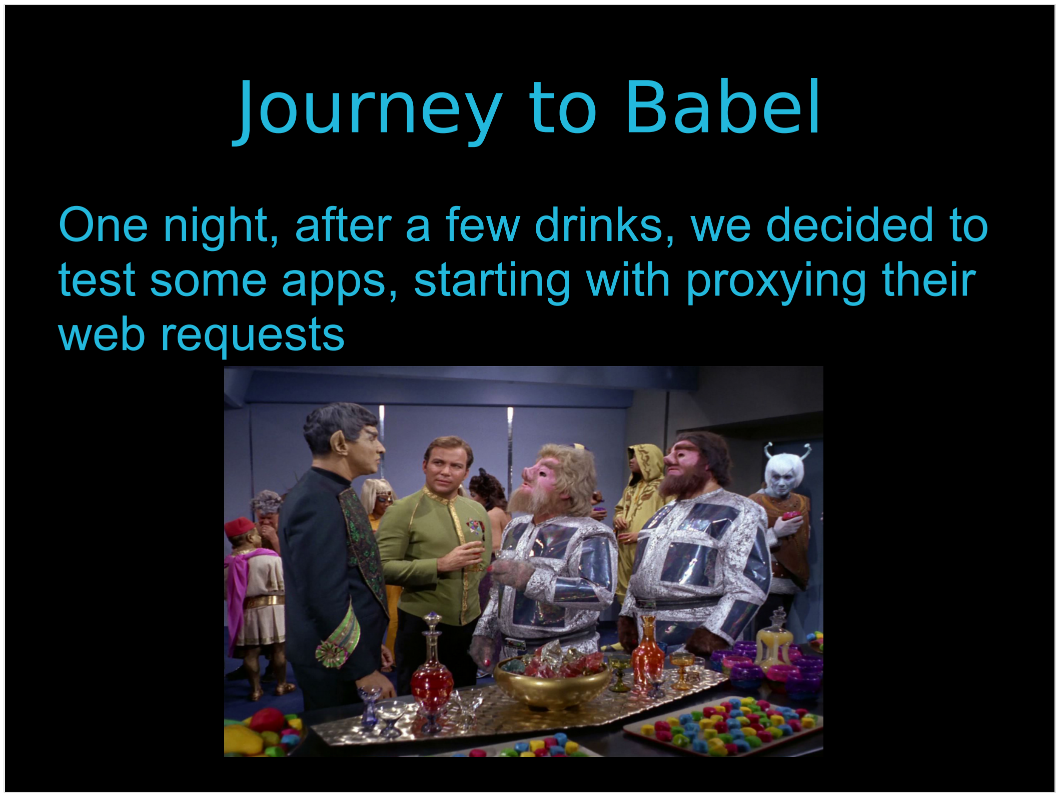
Infosec folks often roll their eyes when they read statements on sites or in apps that tout TLS use and how big their keys are

256bit SSL
512 bit SSL

So, how do companies show their customers that they care about SSL? Many, just put up these logos on their website, without considering what they are actually promising. Naive consumers, have a reasonable expectation, that these signify their data is immune to eavesdropping.

However, as we know, encryption is just a part of security, not the end of it.

We're not lawyers, but it turns out, that stating that you use TLS on your communications, in this way, or with other disclaimers, is actually a legally binding contract, which we will discuss later.



Journey to Babel

One night, after a few drinks, we decided to test some apps, starting with proxying their web requests

So, one night, after a few drinks, we decided to start hacking some apps.

As usual, we started by examining
 their web requests, by using an HTTP proxy

Then we noticed some strange behavior

Into Darkness

Tushar: Tony will now discuss the details of what we found

DO NOT CLICK

Tony: Thanks Tushar. Hello everyone! Hope you're having a good time at DeepSec, so far.

Damn it, Jim!

Before getting into exactly what we found, I wanted to give a little historical context around our testing.

We started this in February and based on our results, we somewhat presumptively declared this was the year that SSL would die.

After GoToFail, Heartbleed and now POODLE, it turns out we were more correct than we could have possibly known...

First aspect of certificate validation

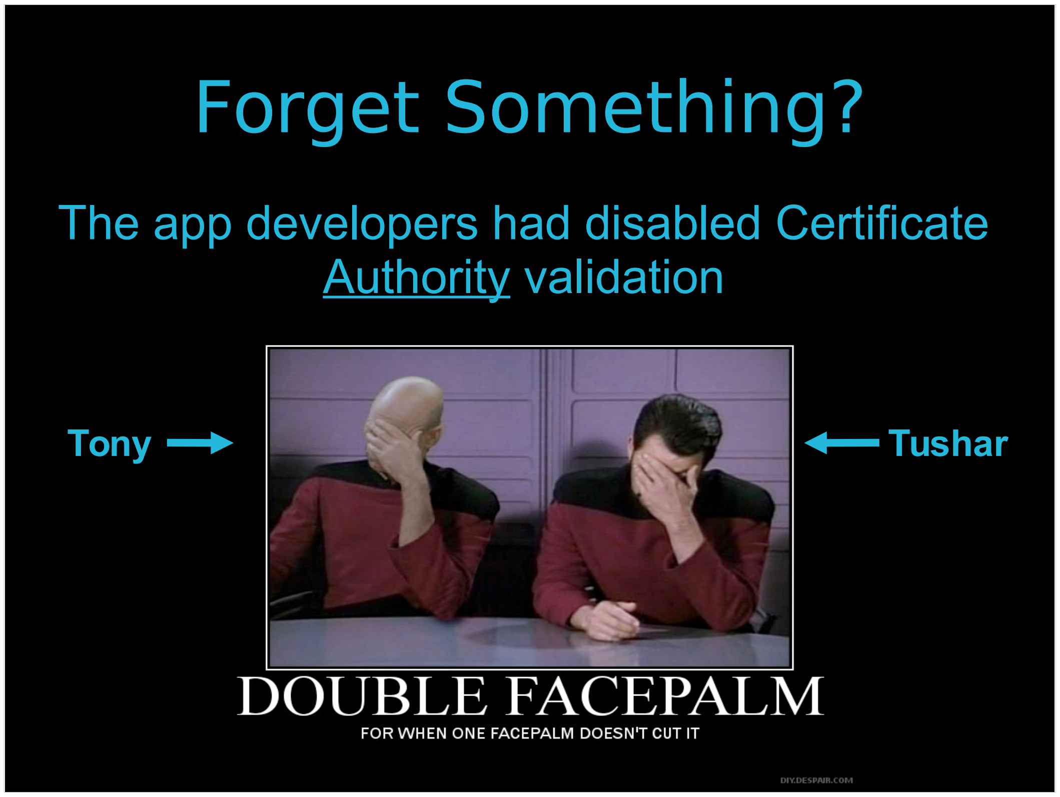
The app or OS must verify the certificate is cryptographically signed by the private key of a Certificate Authority that is pre-trusted

In order for TLS to validate the identity of the remote side of the conversation, it checks that the certificate received is cryptographically signed with the private key of a Certificate Authority, who's public key corresponds with one in the app or OS's store of trusted authorities. Companies like Verisign, Digicert or GoDaddy are generally included automatically in these stores, so nothing needs to be done to accept certificates signed by them.

However, when proxing mobile app TLS requests, you need to install the HTTP proxy's CA certificate on the device, since it is generally not already trusted. For Burpsuite, which is what we use to test, this is the Portswigger CA.

The entire trust model falls apart, if for example, an attacker could create their own CA, then generate a certificate for www.google.com and have the victim's browser or app accept it

Having established these facts, what we initially found, was that we were still seeing traffic in our proxy, from many apps, despite not having installed the Portswigger CA certificate on the device, which should not be the case.



Forget Something?

The app developers had disabled Certificate <u>Authority</u> validation

Tony
Tushar
DOUBLE FACEPALM
FOR WHEN ONE FACEPALM DOESN'T CUT IT
DIY.DESPAIR.COM

It turns out, the reason we were still seeing TLS traffic, was that sometime during the development lifecycle, the validation of CAs was disable. This may have been due to the developer not understanding what they were doing or simply because they disabled validation, so they could proxy requests for testing purposes, but then forgot to re-enabled it prior to releasing to production. Since you can install CA certs on devices or emulators for testing, there is no good reason to do it in code.

Giving the developer the benefit of the doubt that they just forgot, it still underscores the fact that just because you can do something in code, doesn't mean you should...

So everyone is clear, at this point an individual attacker could have gone down to the local coffee shop or any place with public WiFi, joined the network, gained MitM position (using DNS poisoning or ARP spoofing) or alternatively, gone

aste of Armageddon

Disclaimer: Since there are fake apps in the Play Store and some companies failed to respond or confirm our findings, we'll generally stick to showing the icons of the apps, rather than mentioning the companies by name. If you have any of these on your device, make sure you update them.

We obviously tried to focus on the top apps, by download count, that led to interception of passwords, significant session tokens, credit cards and/or sensitive PII, but trying not to get sued, unless we specifically state what could be pilfered, we're just asserting there was an issue. The impact varied between apps and we don't have time to discuss every finding in detail. It turned out that about 10% of the top Android and handful of iOS apps we tested were vulnerable in a meaningful way and without warning the user. Some apps were not included, because the impact wasn't apparently significant or they at least, gave the user a warning message.

Again, we focused on the top apps and this was just a “taste”...

The Trouble with Tribbles

amazonkindle
Zappos.com
Capital One
tc
newegg
BEST BUY
us
OTP
Mobile Connect
Google cloud messaging
UBER
ADP
cisco
SafeNet
CNN Money
YAHOO!
SplashID

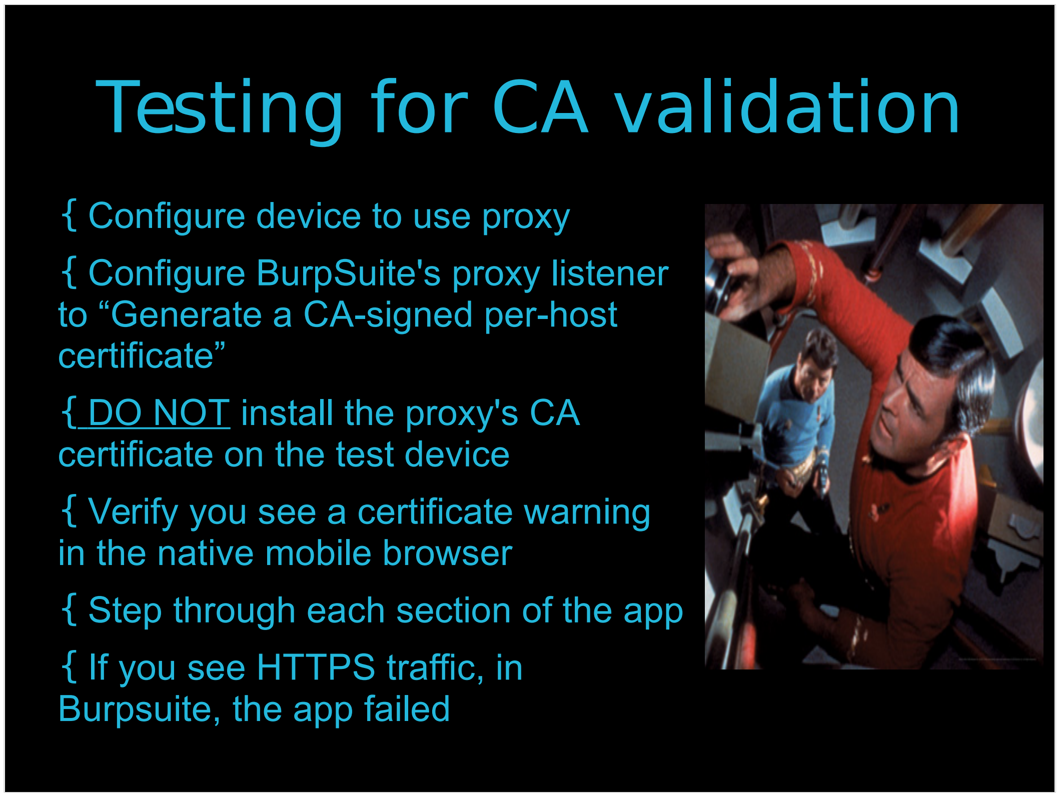
unately, like Tribbles, bad coding practices just seem to propagate
trollably...

hile the apps written for Windows phone may have faired well, Microsoft's app
 other platforms...not so much...

get me wrong, I'm not a Microsoft basher, but we did find more of their apps w
issues than any other company.

he Google Cloud Messaging icon represents the service used by nearly every
id app and I'm told, Chrome extension, to register with Google for receiving
notifications, but is only half of the equation, since they need to also register w
p creator's messaging server. Google recommends not using GCM for sensiti
ation, so this vulnerability's impact would vary on an app by app basis,
ding on whether that warning was heeded and we were unable to develop an
 scenario for this ourselves, but thought it was noteworthy none-the-less.

 a look at the other apps we found, you shouldn't have anything to worry abou



Testing for CA validation

- { Configure device to use proxy
- { Configure BurpSuite's proxy listener to “Generate a CA-signed per-host certificate”
- { DO NOT install the proxy's CA certificate on the test device
- { Verify you see a certificate warning in the native mobile browser
- { Step through each section of the app
- { If you see HTTPS traffic, in Burpsuite, the app failed

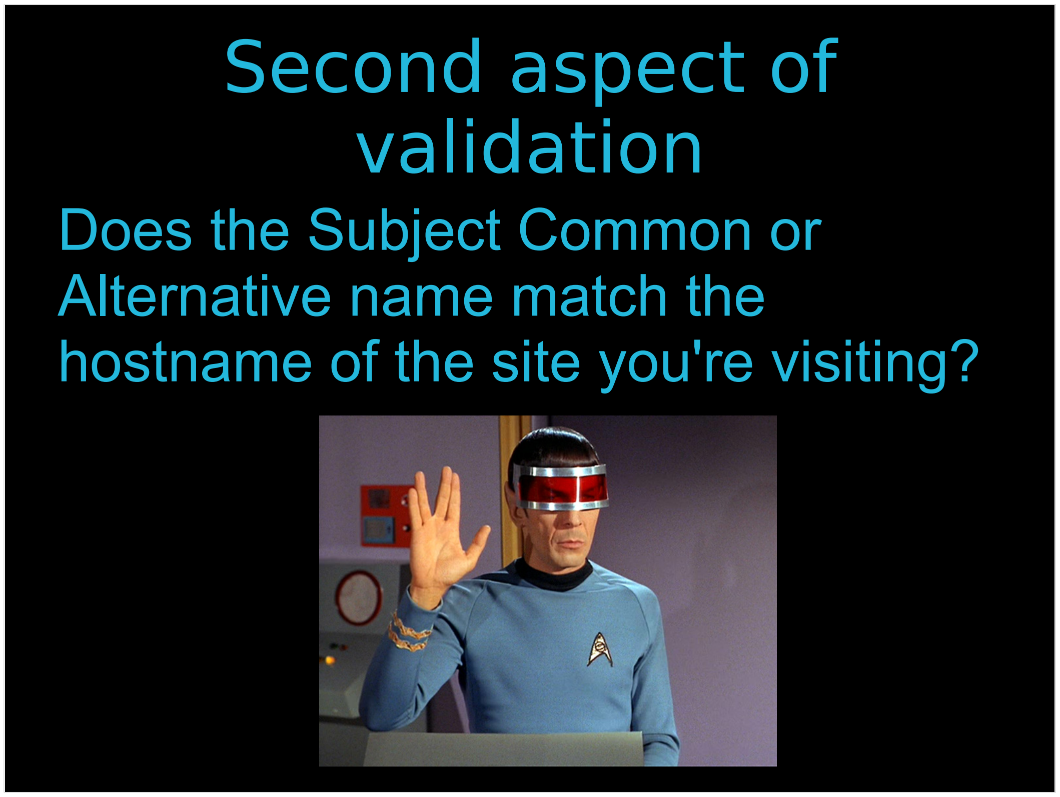
We briefly wanted to walk through how you test for this.

This information will be our site, secbro.com, so don't worry if you can't write it all down now.

Basically, you can use Burpsuite's default settings for the proxy listener, just don't install the CA certificate on the device first.

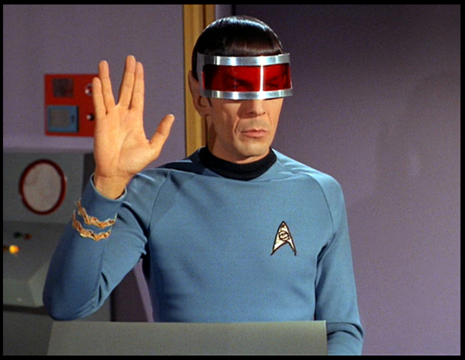
You should verify that you see a certificate warning when you visit a secure site in the mobile browser, before testing apps, just to verify you are hitting the proxy and that your CA cert is NOT trusted already, especially if you previously had the same Portswigger CA cert installed on the device

After doing this, you'll want to step through the different functionalities of the app and if you still see HTTPS traffic in your proxy, the app failed



Second aspect of validation

Does the Subject Common or Alternative name match the hostname of the site you're visiting?



Again, so that we're all on the same page, another part of the authenticity aspect of SSL is that we verify the certificate we receive is actually for the site you are trying to visit.

This is done by matching the hostname of the remote side to either the Subject: Common or Alternative Name(s) on the certificate

Again, the entire trust model falls apart if an attacker could register their malicious domain, get a certificate for it and use this to intercept any app's SSL traffic, regardless of the destination domain.

For example, if the certificate for www.nsa.gov was accepted when trying to connect to www.torproject.org,someone is going to have a bad day.

By any other na

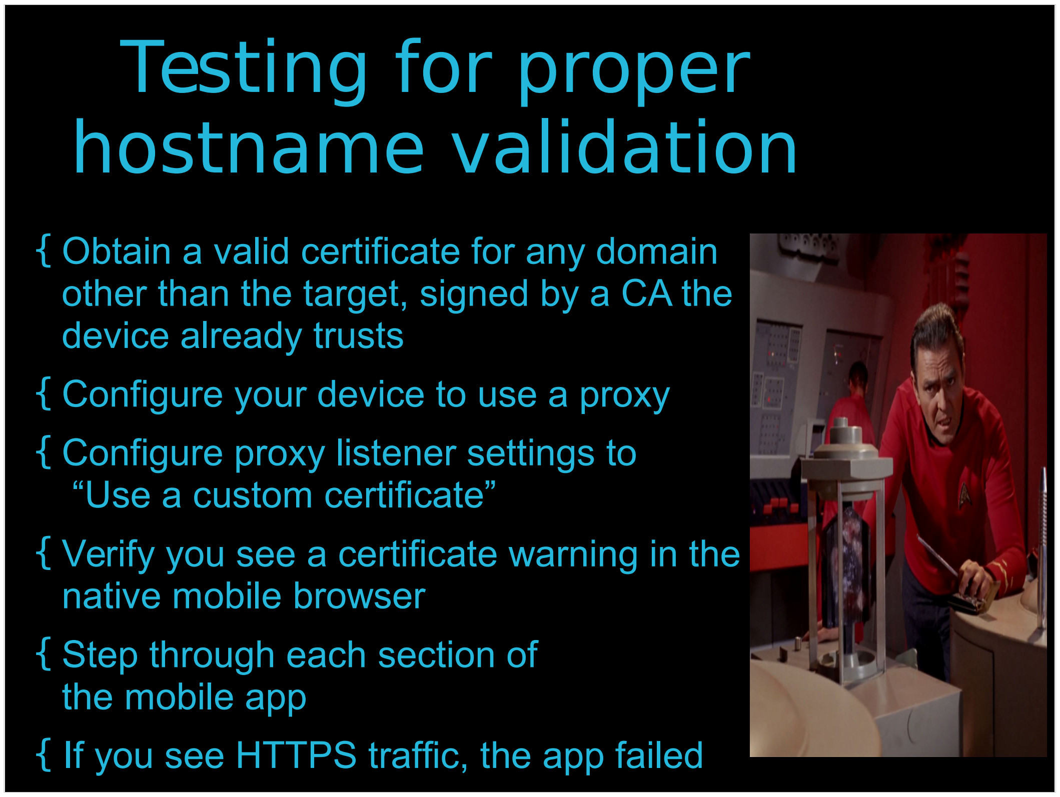
While most of these apps correctly validated the CA, none of them, verified the certificate's CN or AN, so we were able to intercept and decrypt the vulnerable SSL traffic, for all of them, using a certificate issued for one of our domains.

To be clear, the cert we used for testing, was signed by StartSSL.com's CA, which is trusted by Android and iOS, out of the box, so we didn't need to add it to the device first.

Again, we see several significant financial applications, tax software, the leading blog software's admin app, a domain registrar and SSL certificate issuer app ironically, an ISP, security software, a cable TV co., travel sites, one of the biggest Chinese Internet companies, the California DMV, who, if your a CA resident, has all your PII and again, multiple Microsoft applications.

Note: if you see an app on multiple slides, it is because it failed in multiple ways.

Disclaimer #2: Oracle wanted us to let you know they, opted for the



Testing for proper hostname validation

- Obtain a valid certificate for any domain other than the target, signed by a CA the device already trusts
- Configure your device to use a proxy
- Configure proxy listener settings to "Use a custom certificate"
- Verify you see a certificate warning in the native mobile browser
- Step through each section of the mobile app
- If you see HTTPS traffic, the app failed

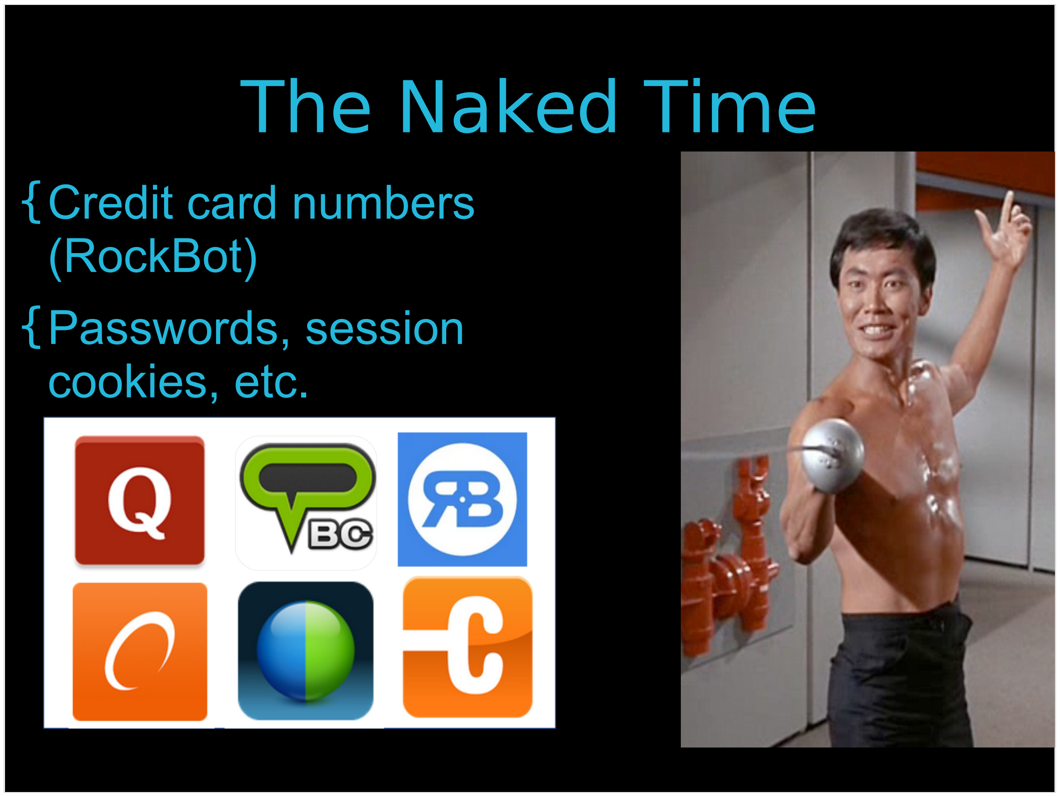
Again, these instructions will be on secbro.com, but to quickly review how to test for this.

First, get a valid certificate for any domain.

Startssl.com is a good source for free ones and it is already in the trust store of most devices.

The main difference here is you need to configure Burpsuite's proxy listener to the "Use a custom certificate" setting. If you see HTTPS traffic while stepping through the app, it failed

Again, we asked ourselves, what else could be wrong and we suspected some applications or devices may also accept revoked and/or expired certificates, but didn't pursue these further, because this was about the time iOS's gotofail bug was made public, so we needed to wait for that to be fixed and there was an Arstechnica article indicating others may have been performing similar research.



The Naked Time

- Credit card numbers (RockBot)
- Passwords, session cookies, etc.

That was by not using it at all...well at least, not in some important HTTP requests

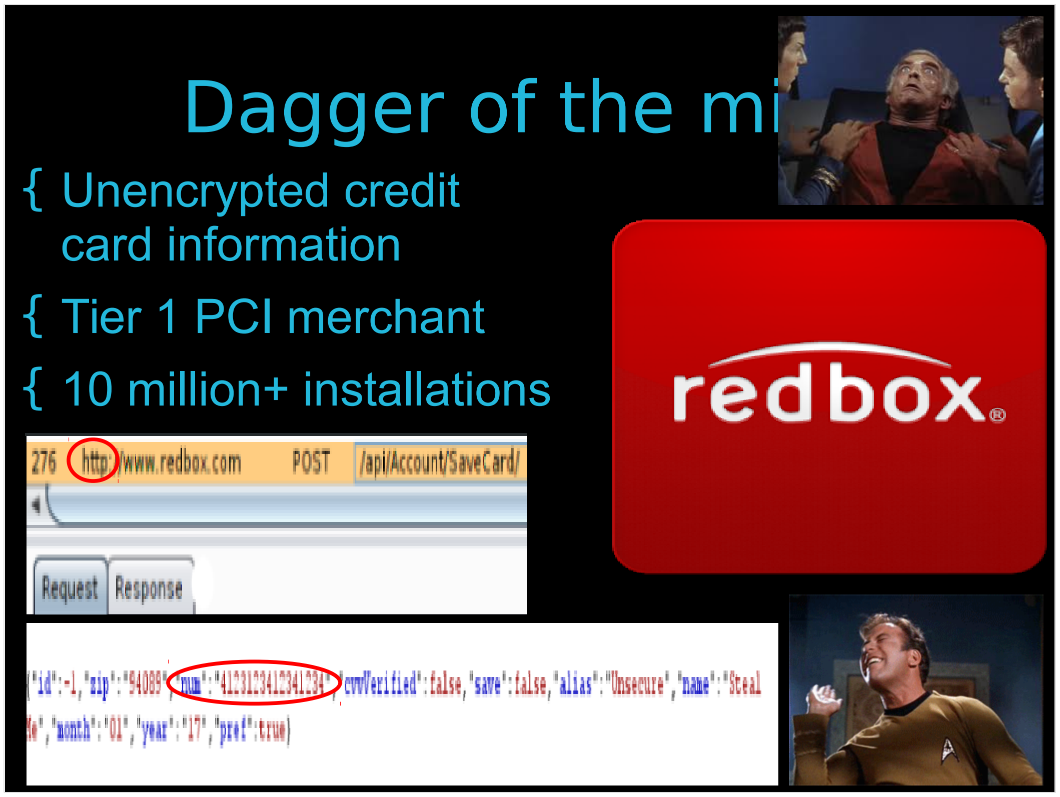
This included leaking session cookies from Quora, a popular information sharing application.

The entire registration process for Cisco's WebEx, including password creation, was not being encrypted.

Usernames and passwords not being encrypted for the Angie's List Business Center application

Rockbot, which is basically a modern-day jukebox, that accepts credit cards in the application, but failed to encrypt them in transit.

Then there was one that we found particularly interesting and



Dagger of the mi

- Unencrypted credit card information
- Tier 1 PCI merchant
- 10 million+ installations

```
276 http://www.redbox.com     POST     /api/Account/SaveCard/
```

```
{"id":-1,"zip":"94089","num":"4123123412341234","cvvVerified":false,"save":false,"alias":"Unsecure","name":"Steal
Me","month":"01","year":"17","pref":true}
```

Now, we really hope you enjoy the animations on this slide,we turned down thousands of dollars to show it to you... so we wanted to ensure we got our money's worth

To fail epically, you obviously would need to send unencrypted credit cards as well, but the key difference here **(CLICK)** is that they are a Tier 1 PCI merchant.

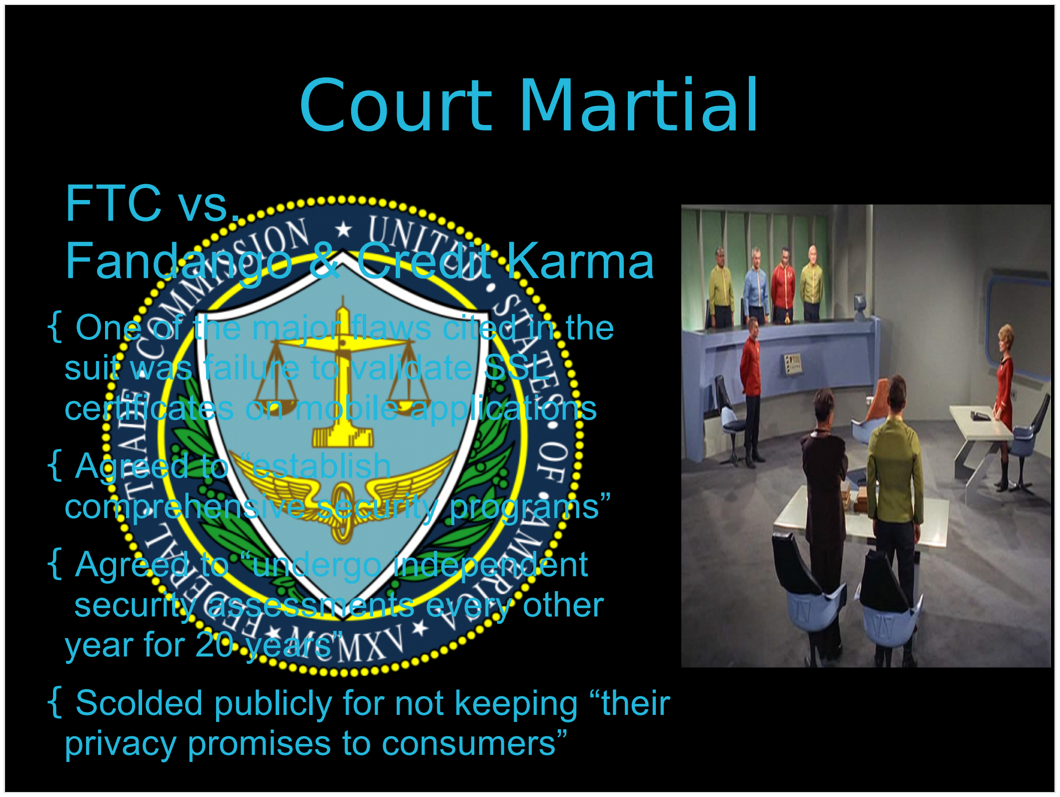
Now, I'm pretty sure PCI has a checkbox somewhere for using SSL.

Unfortunately, that Tier 1 status means they process **a lot** of credit card transactions. I mean to have a truly epic fail, you need scale. (**CLICK)** Their app had over 10 million installations..

So, who forgot that HTTPS, ends with an S?

(CLICK) *REDBOX* ***PAUSE (CLICK)***

If you can't see the screen shot, it clearly shows the POST to save



Court Martial

FTC vs.
Fandango & Credit Karma

- One of the major flaws cited in the suit was failure to validate SSL certificates on mobile applications
- Agreed to "establish comprehensive security programs"
- Agreed to "undergo independent security assessments every other year for 20 years"
- Scolded publicly for not keeping "their privacy promises to consumers"

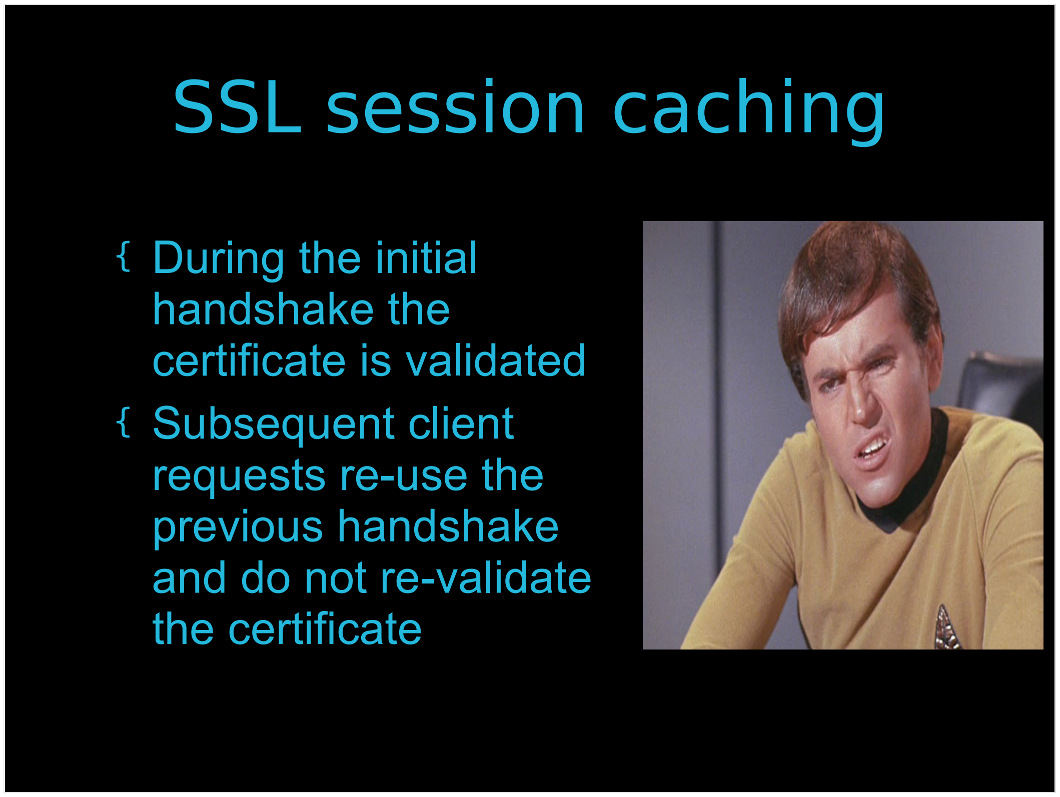
Getting back to those SSL usage disclaimers made by many sites and apps that Tushar previously mentioned...

During our research, we found out that there were previously FTC suits against Credit Karma and Fandango, in which the SSL usage claims, combined with the failure to perform proper certificate validation, were cited as violations central to the suits

The companies wound up settling with the FTC and in addition to being publicly scolded for breaking security promises to their customers, were compelled to institute "comprehensive" security programs, which they probably should have had in the first place and are now subjected to additional security audits for **20 years**...! This usually also means that any further infractions during this time period will result in even harsher penalties, since it wouldn't be there "first offense"

If this isn't your first security conference, you had to know this slide was coming

Sorry Bones, but we can't leave without giving the tin-foil hat crowd something to think about...



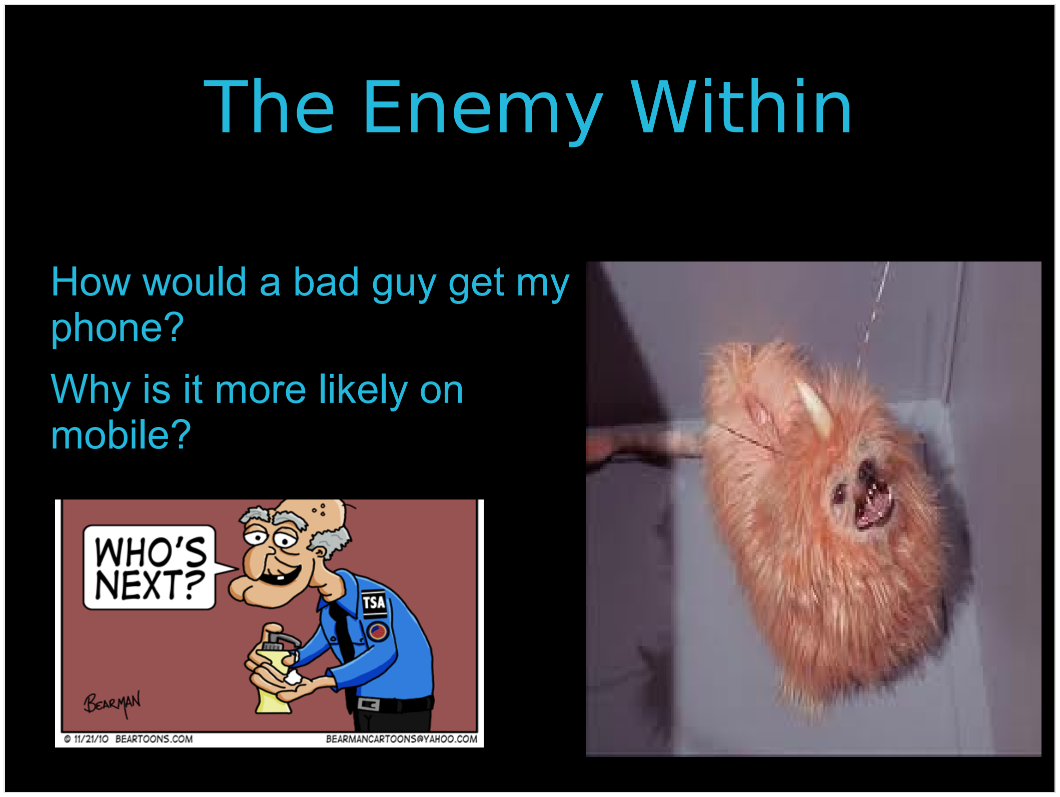
SSL session caching

- During the initial handshake the certificate is validated
- Subsequent client requests re-use the previous handshake and do not re-validate the certificate

So, in an effort to make this SSL thing a bit more efficient and faster, someone decided that it was a good idea to skip certificate validation on all but the first handshake request

This allows the apps that use this feature, Google Maps being an example of one, to store a session identifier that allows for resumption of previously validated SSL sessions

We thought to ourselves, what would happen if a bad guy made that first connection? Basically, why trust on first use isn't always a great idea.



The Enemy Within

How would a bad guy get my phone?

Why is it more likely on mobile?

You may think it is unlikely that Romulans would ever have physical to your device...**(CLICK)**

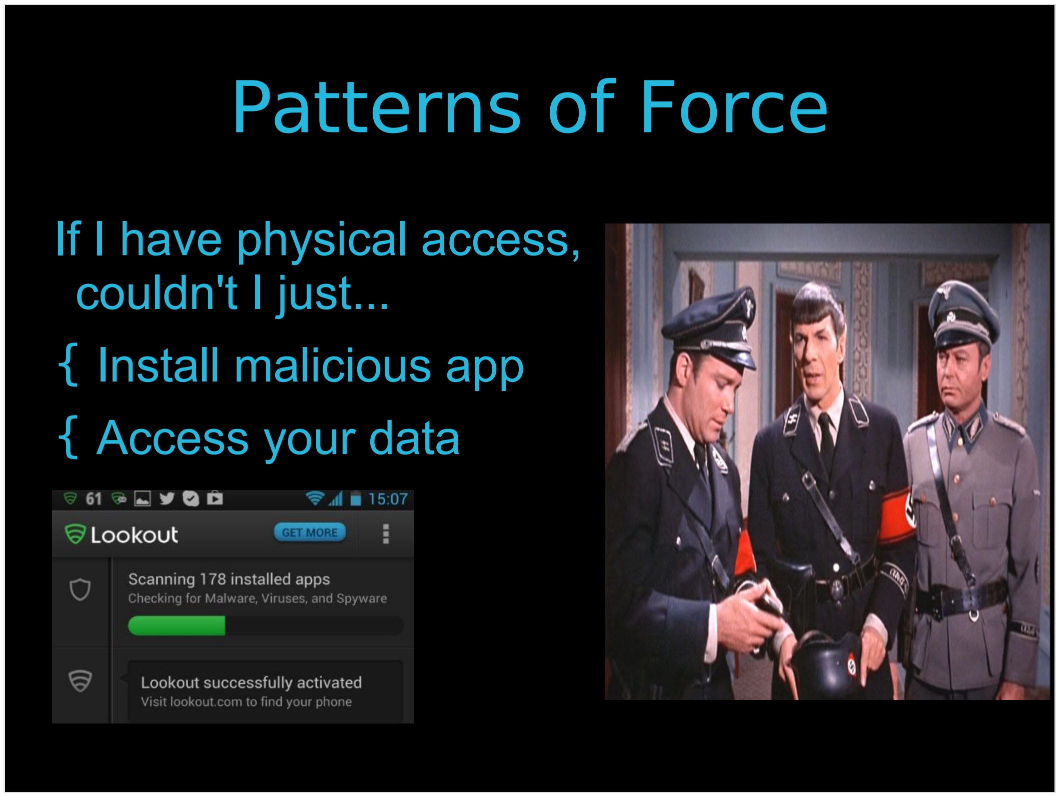
That's “logical”, unless you travel and have to get by the TSA or customs, get detained by law enforcement, have a jealous girlfriend, got your phone from, or have to give your phone to your IT department, remember that the tech at the cell phone store already had access and so did numerous folks in the supply chain. Remember, Nokia did this before and you can bet someone will do it again in the future.

You could also drink too much this weekend and “misplace your phone”. I'm sure your hacker buddies wouldn't mess with it, right?

I expect the skeptical amongst you may say, I would double-check everything on the device, before I used a phone that I “lost”, but in a moment I'll show why that isn't necessarily going to be of any help.

(CLICK)

The reason this is more problematic on mobile devices is because



Patterns of Force

If I have physical access, couldn't I just...

{ Install malicious app

{ Access your data

61 15:07

Lookout GET MORE

Scanning 178 installed apps
Checking for Malware, Viruses, and Spyware

Lookout successfully activated
Visit lookout.com to find your phone

So, you might say, well if the space Nazis, or any attacker gets access to my device,

Couldn't they just install malware, etc?

- Yes, but malware can be detected by security software and a malicious app is probably not going to unnoticed for long

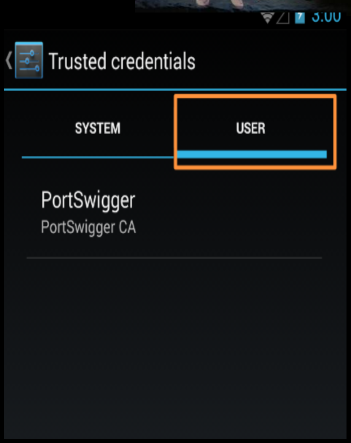
You might also ask, couldn't they just get the data they wanted off the device directly?

If the apps are not foolish enough to store data on the SDCARD, this should require a rooted device. If the device isn't already rooted, rooting it, might require wiping the device. If you're device is already running rooted, well, you're already screwed...

Also, what if the attacker wanted to access data that isn't on the device yet?

Turnabout Intrud

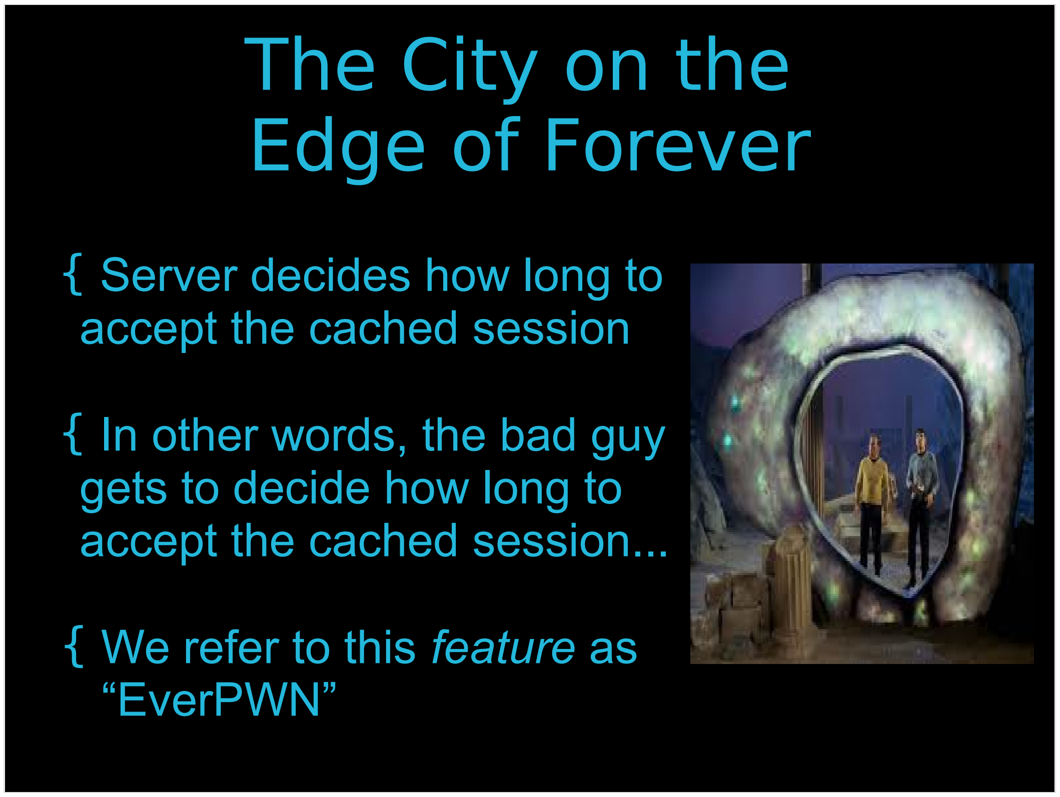
Since SSL session caching only checks the certificate once, you only need it on the device for as long it takes you to make the first connection, after which you can delete it



For both iOS and Android, you can delete the MitM certificate as soon as the first connection is made and there is virtually no way for anyone to know it was ever there...**(CLICK)**

Even that "Network may be monitored" warning that shows up on modern Android versions, to let you know when a User certificate is installed, goes away.

On iOS you'll still be able to intercept traffic until the device reboots, but interestingly on Android, this behavior is slightly different...as we'll explain shortly



The City on the Edge of Forever

- Server decides how long to accept the cached session
- In other words, the bad guy gets to decide how long to accept the cached session...
- We refer to this *feature* as "EverPWN"

A feature of session caching seems to be that the server gets to choose how long they to accept that cached session...

We've verified sessions can be maintained in excess of 24 hours, but unless there is some limit we've missed, I'm pretty sure the bad guy would choose to be able to intercept your traffic forever, although 2 years is probably the same on mobile devices, since that's about how long anyone actually keeps them.

So getting back to the Android behavior, unfortunately, the session cache files are persistent and survive reboots!

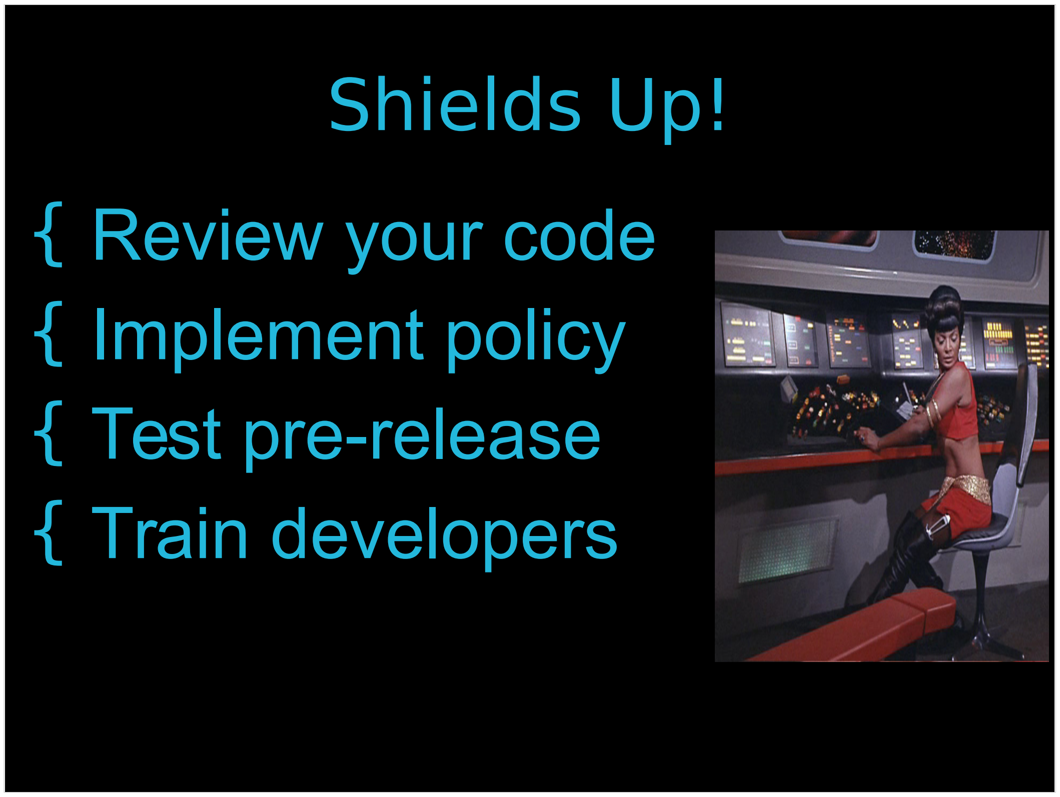
The only evidence of this state, is the certificate information that appears to be in the cache file, but reading it requires root on the device.

So, for as long as you can maintain the MitM postion, you

We thought we'd leave you with tips to take back to your organizations on how to protect your apps from these problems. Initially we thought to recommend that you should incorporate this testing into your SDLC, ensure your policies clearly prevent disabling validation, that your developers are trained properly and know how to install certs on devices and emulators and your code is reviewed for this issue, which is a simple grep away in most cases. **(CLICK)**For example, you'll want to look for X509TrustManager or HostNameVerifier interfaces that always return a true-ish result on Android and the slide shows some places to investigate on iOS as well.

Common thought is to use certificate pinning, which we agree with, but it can be a pain for testing and development, depending on how it is implemented.

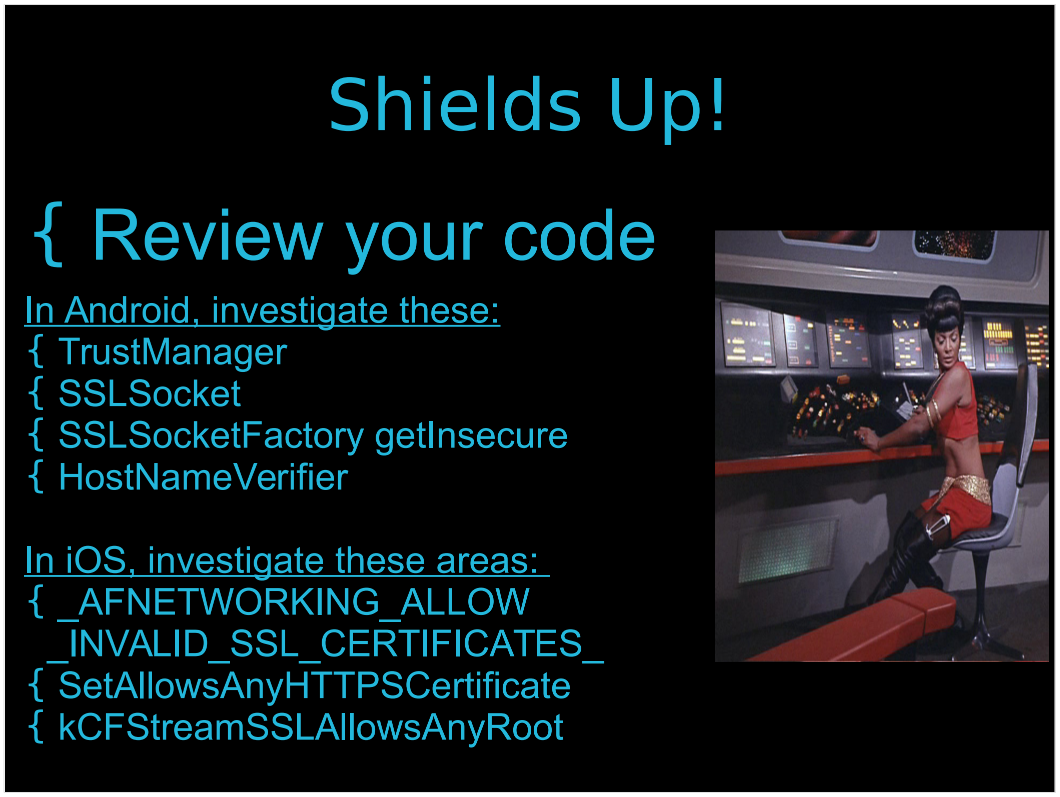
Regardless of whether the app pins or not, we're big fans of self-defending code, that eliminates the chance of human



Shields Up!

- Review your code
- Implement policy
- Test pre-release
- Train developers

We thought we'd leave you with tips to take back to your organizations on how to protect your apps from these problems. Initially we thought to recommend that you should incorporate this testing into your SDLC, ensure your policies clearly prevent disabling validation, that your developers are trained properly and know how to install certs on devices and emulators and your code is reviewed for this issue, which is a simple grep away in most cases. **(CLICK)**



Shields Up!

{ Review your code

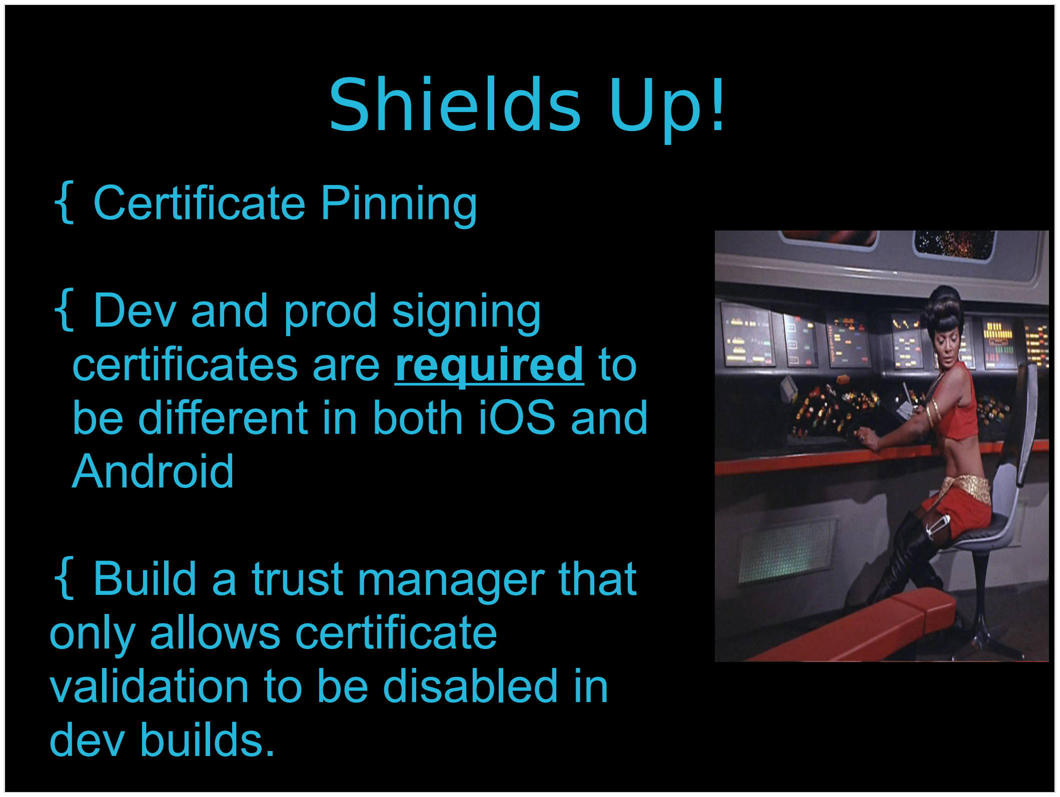
In Android, investigate these:

{ TrustManager
{ SSLSocket
{ SSLSocketFactory getInsecure
{ HostNameVerifier

In iOS, investigate these areas:

{ _AFNETWORKING_ALLOW
  _INVALID_SSL_CERTIFICATES_
{ SetAllowsAnyHTTPSCertificate
{ kCFStreamSSLAllowsAnyRoot

For example, you'll want to look for TrustManagers that don't actually do anything, HostNameVerifier interfaces that essentially fail open or use of SSLSocket where there is no call to getDefaultHostnameVerifier with the hostname specified and/or unchecheckedd results from HostnameVerifier.verify() on Android and the slide shows some places to investigate on iOS as well. **(CLICK)**

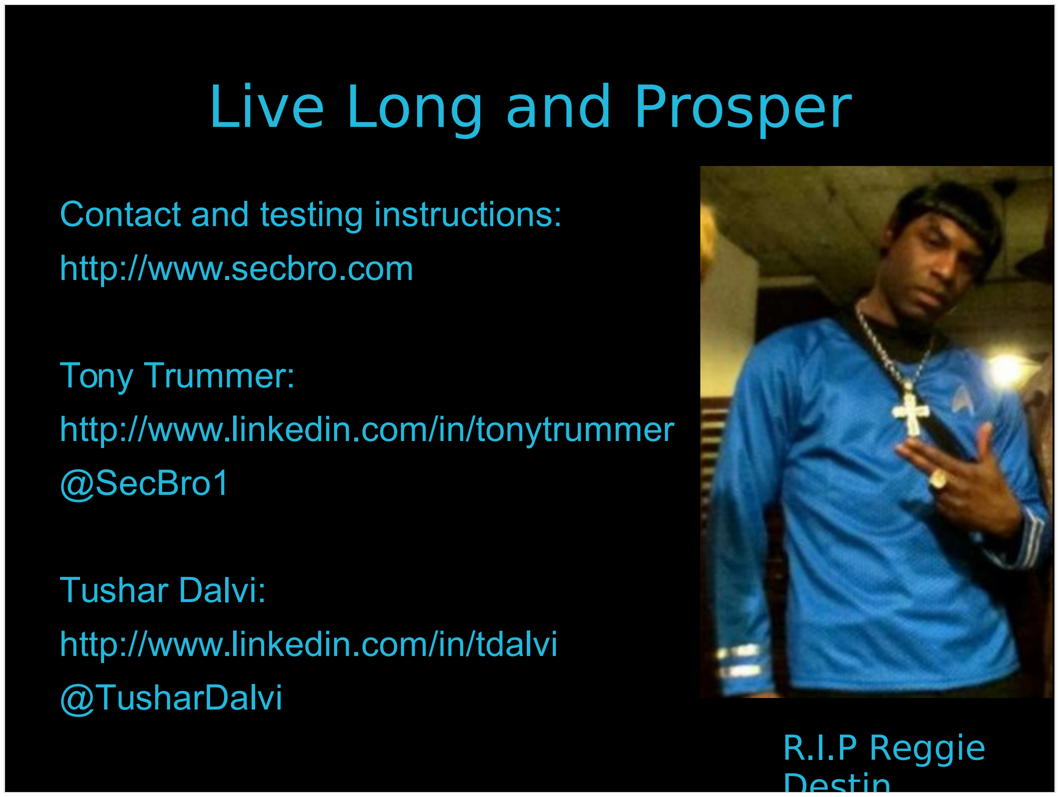


Shields Up!

- { Certificate Pinning
- { Dev and prod signing certificates are **required** to be different in both iOS and Android
- { Build a trust manager that only allows certificate validation to be disabled in dev builds.

Conventional wisdom says you should use certificate pinning and we don't disagree with that. Regardless of whether the app pins or not, we're big fans of self-defending code, that eliminates the chance of human error. Luckily, the IDEs for Android and iOS both use “development” signing certificates which CANNOT be used to sign production releases of APKs and IPAs, that are released in the stores. So why not have the code check which certificate it was signed with and create a fool-proof toggle between dev and prod settings and use this to set whether or not your application would validate CAs and CN/SAN on certificates, **if** this is needed in your environment?

We honestly feel this is a simple and straightforward solution that should work for everyone and unlikely to meet any realistic objections. **(CLICK)**



Live Long and Prosper

Contact and testing instructions:
http://www.secbro.com

Tony Trummer:
http://www.linkedin.com/in/tonytrummer
@SecBro1

Tushar Dalvi:
http://www.linkedin.com/in/tdalvi
@TusharDalvi

R.I.P Reggie
Destin

Well, that's all we have. Thank you for your time and we hope you enjoyed the presentation! We will be available afterwards for any questions/comments.

Feel free to contact us directly anytime using the information above.

Again, more detailed instructions available on secbro.com and the slides should be there early next week.

Live Long and Prosper!