HTTPS REMOVAL TECHNIQUES FOR NON-BROWSER APPLICATIONS

Jacob Thompson

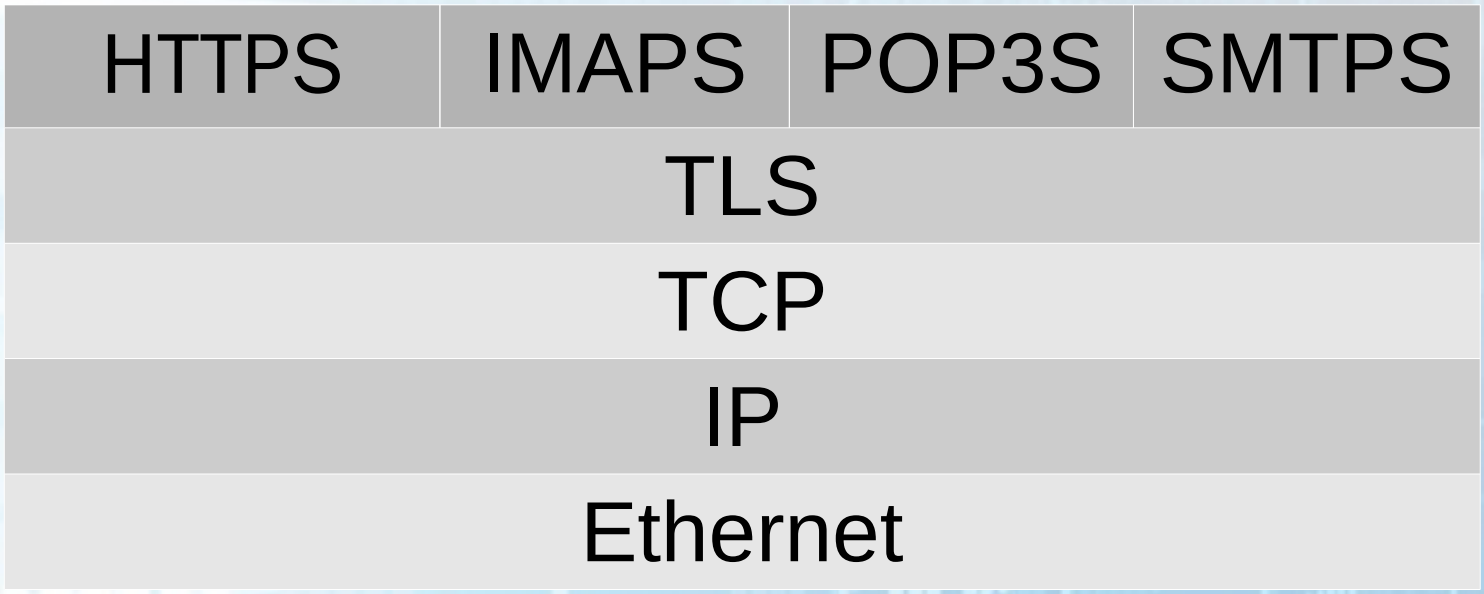
October 6, 2016

About ISE

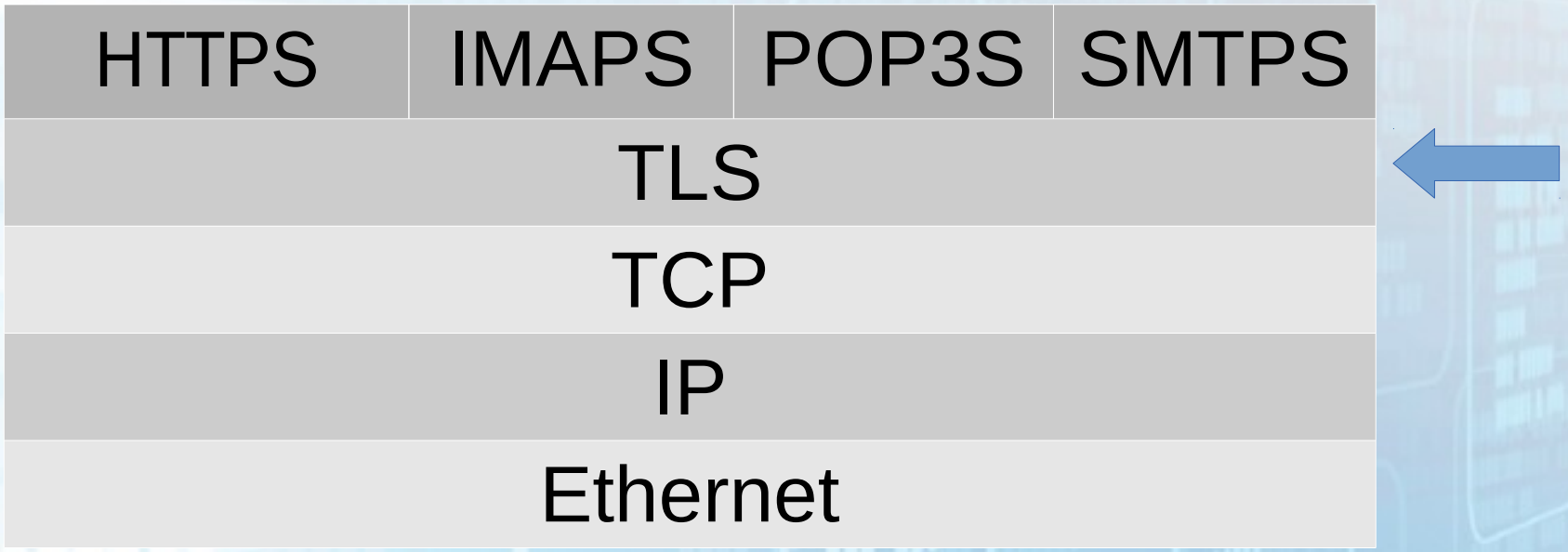
- We are:
  - Computer Scientists
  - Academics
  - Ethical Hackers
- Our customers are:
  - Fortune 500 enterprises
  - Entertainment, software security, healthcare
- Our perspective is:
  - White box

HTTPS vs. TLS

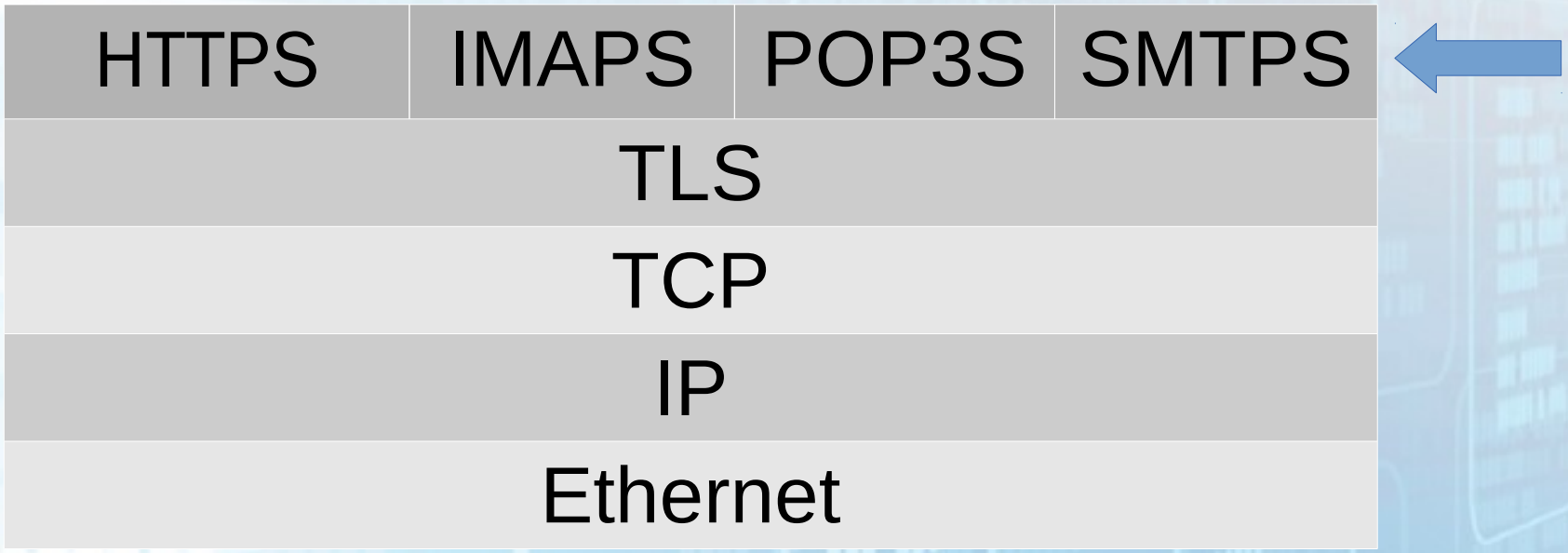
HTTPS/TLS in the Networking Stack



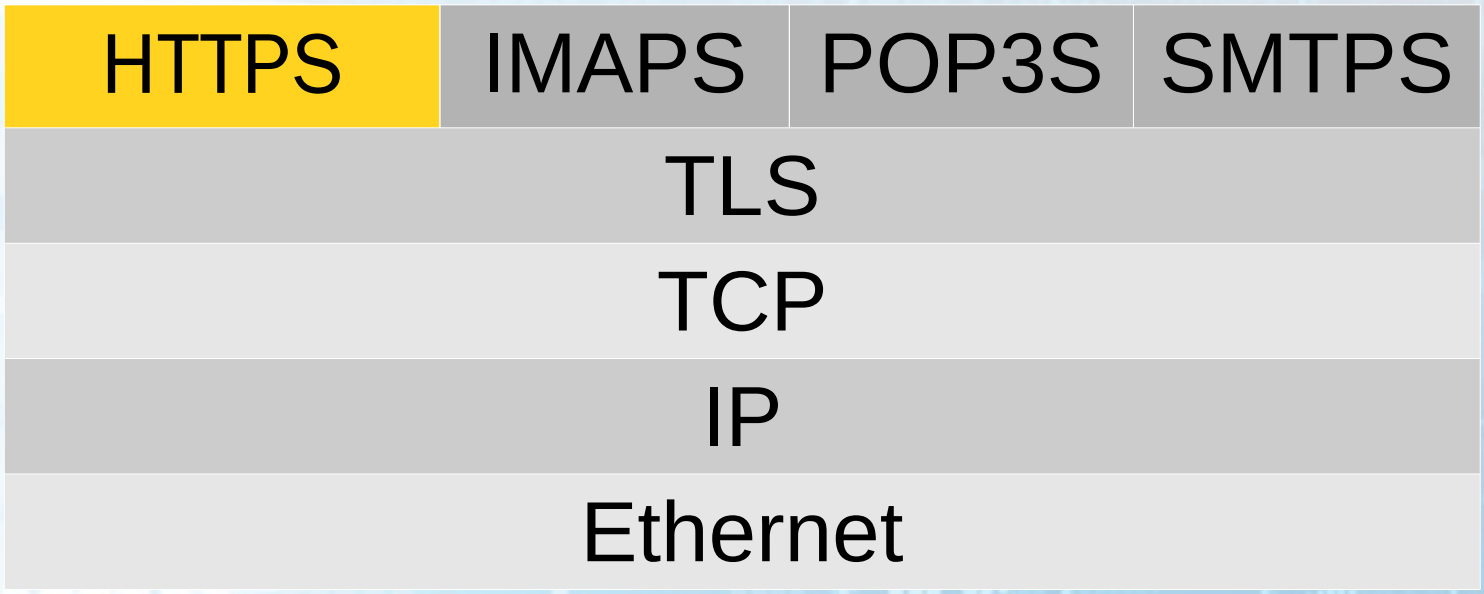
HTTPS/TLS in the Networking Stack



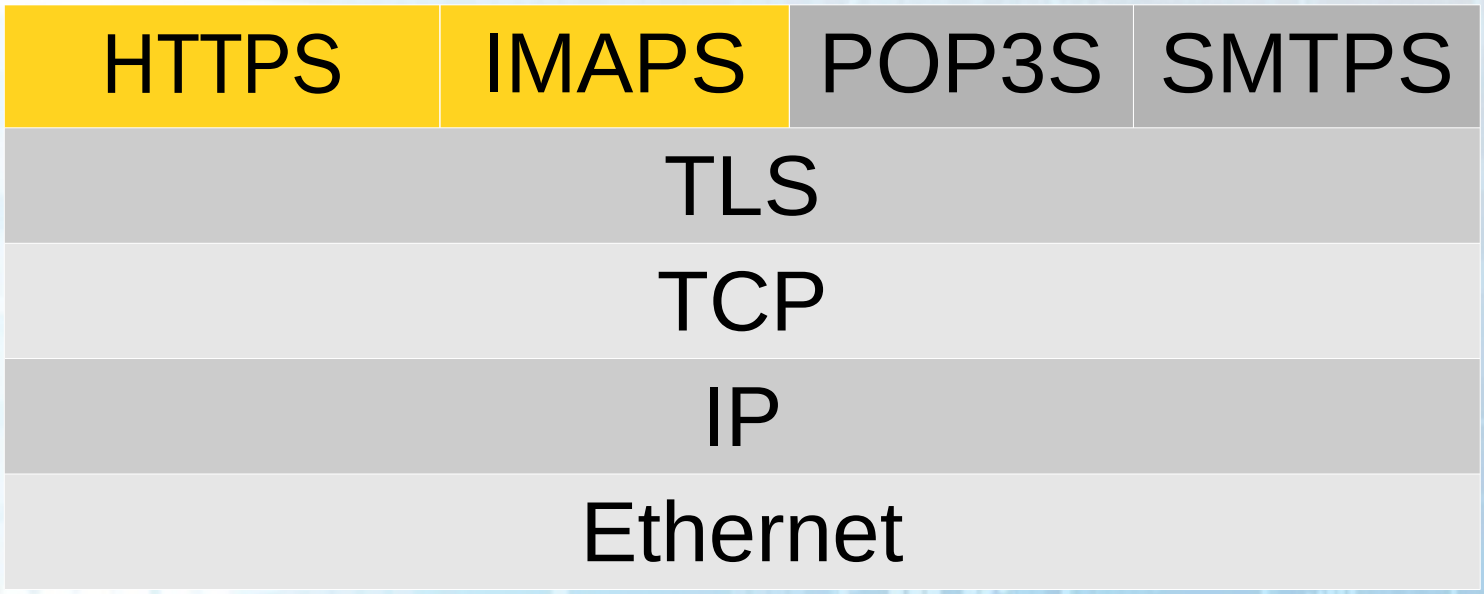
HTTPS/TLS in the Networking Stack



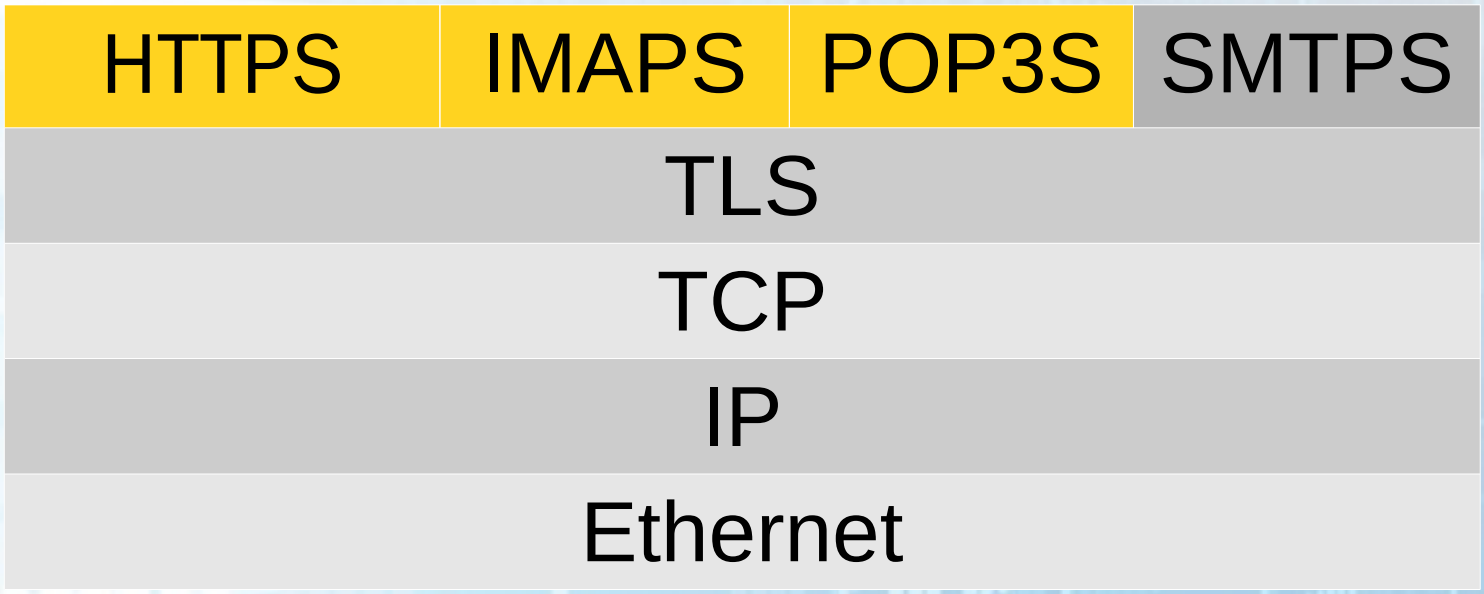
HTTPS/TLS in the Networking Stack



HTTPS/TLS in the Networking Stack



HTTPS/TLS in the Networking Stack



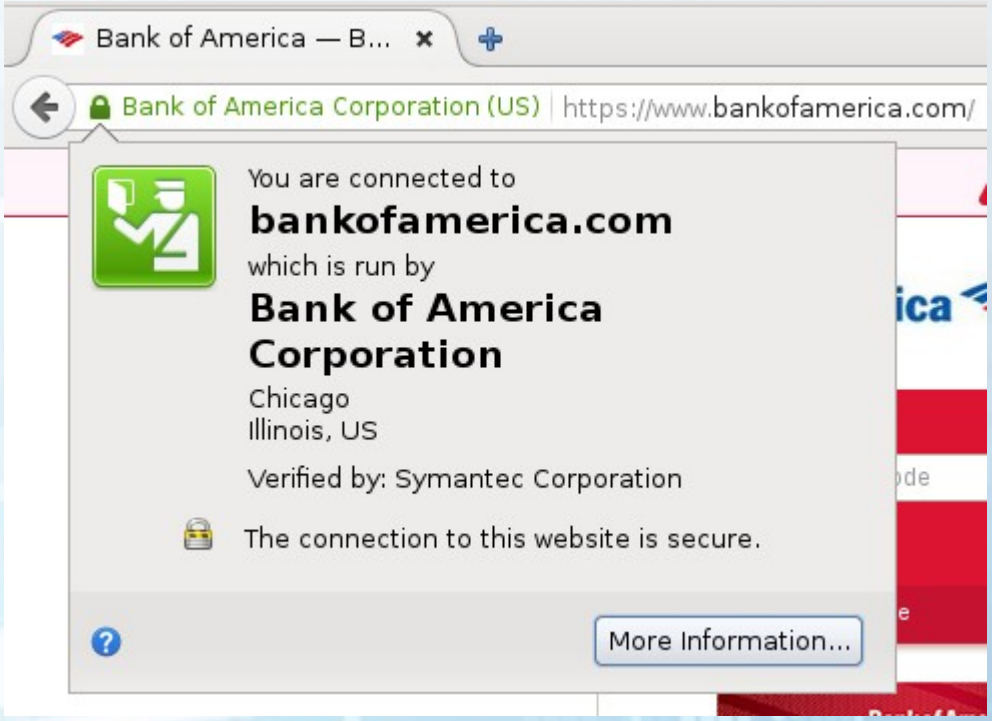
HTTPS/TLS in the Networking Stack

HTTPS	IMAPS	POP3S	SMTPS
TLS			
TCP			
IP			
Ethernet			

HTTPS/TLS in the Networking Stack

?
TLS
TCP
IP
Ethernet

HTTPS



HTTPS = HTTP over TLS

CA/Browser Forum vs. IETF

Fundamentals of TLS

Fundamentals of TLS

1. Message Confidentiality

Fundamentals of TLS

1. Message Confidentiality
2. Message Integrity

Fundamentals of TLS

1. Message Confidentiality
2. Message Integrity
3. Server Authentication

Fundamentals of TLS

1. Message Confidentiality
2. Message Integrity
3. Server Authentication

Preview: Defeat server authentication and we defeat TLS

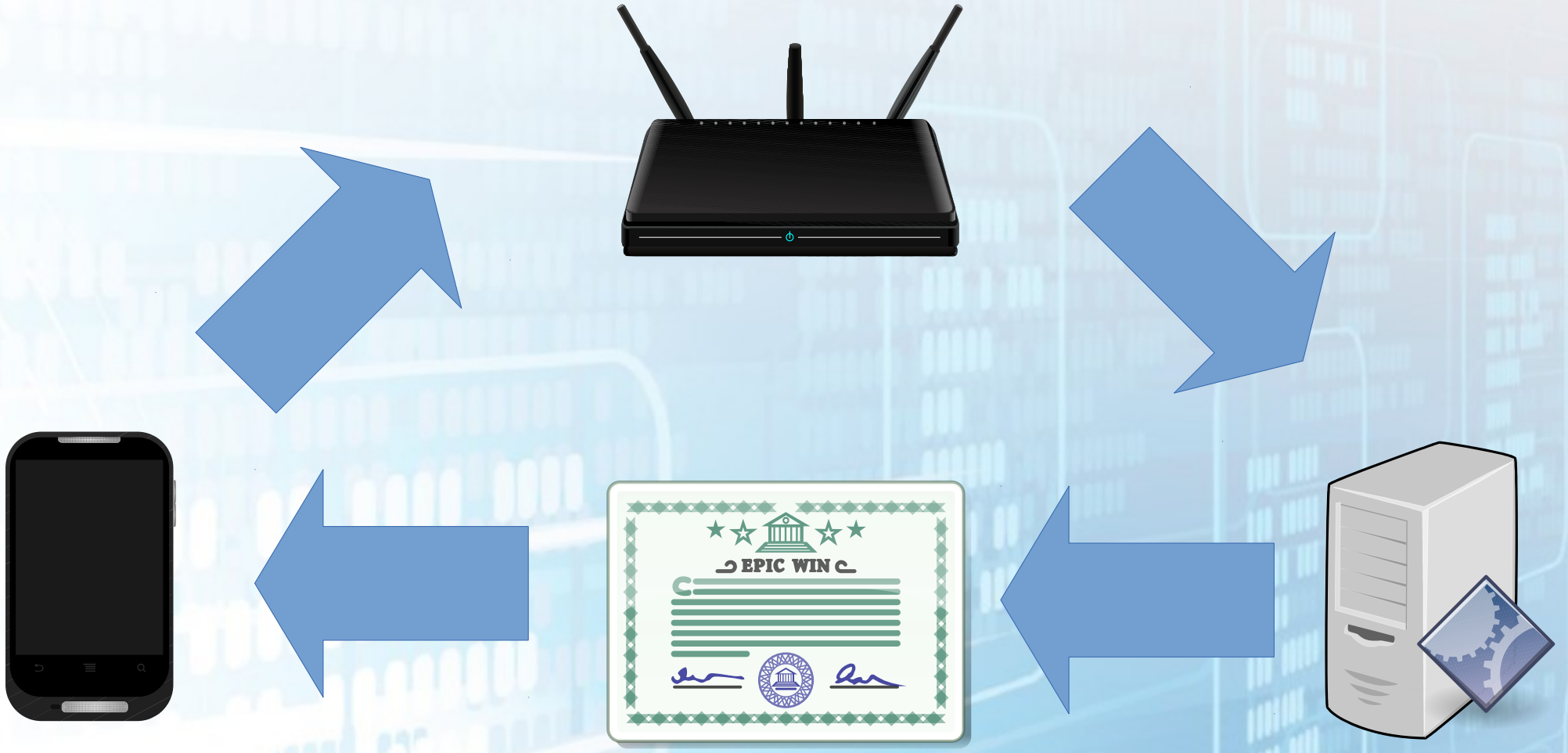
TLS and Security Testing

TLS and Security Testing

TLS and Security Testing

TLS and Security Testing

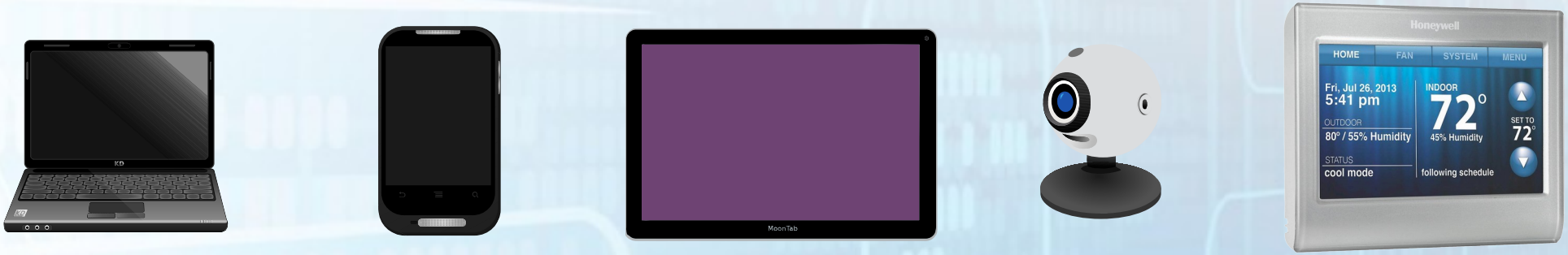
TLS and Security Testing



TLS and Security Testing

TLS and Security Testing

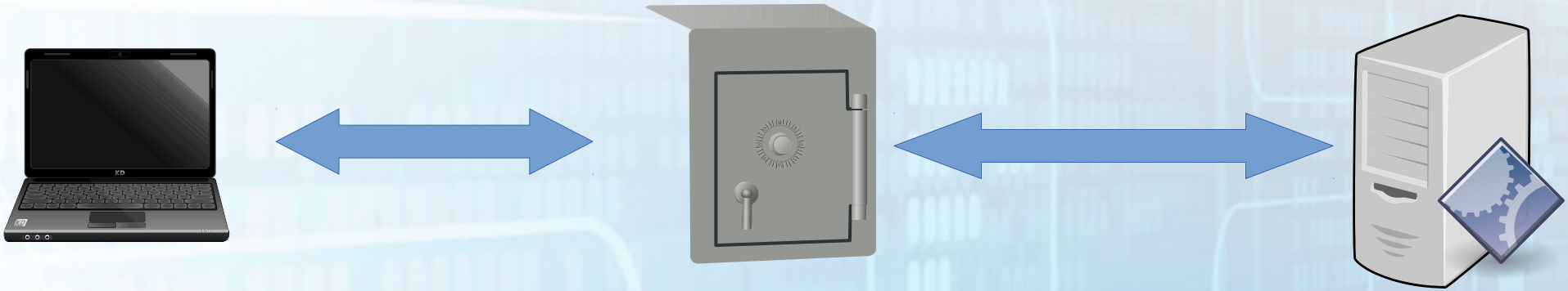
Bypassing TLS



Defeating certificate verification is the focus of the remainder of the talk

How TLS Works

How TLS Works



Asymmetric Cryptography

Asymmetric Cryptography

Asymmetric Cryptography

Asymmetric Cryptography

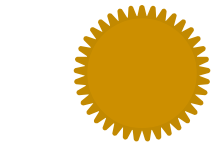
Certificates

Certificate

I hereby certify that the public key attached hereto belongs to:

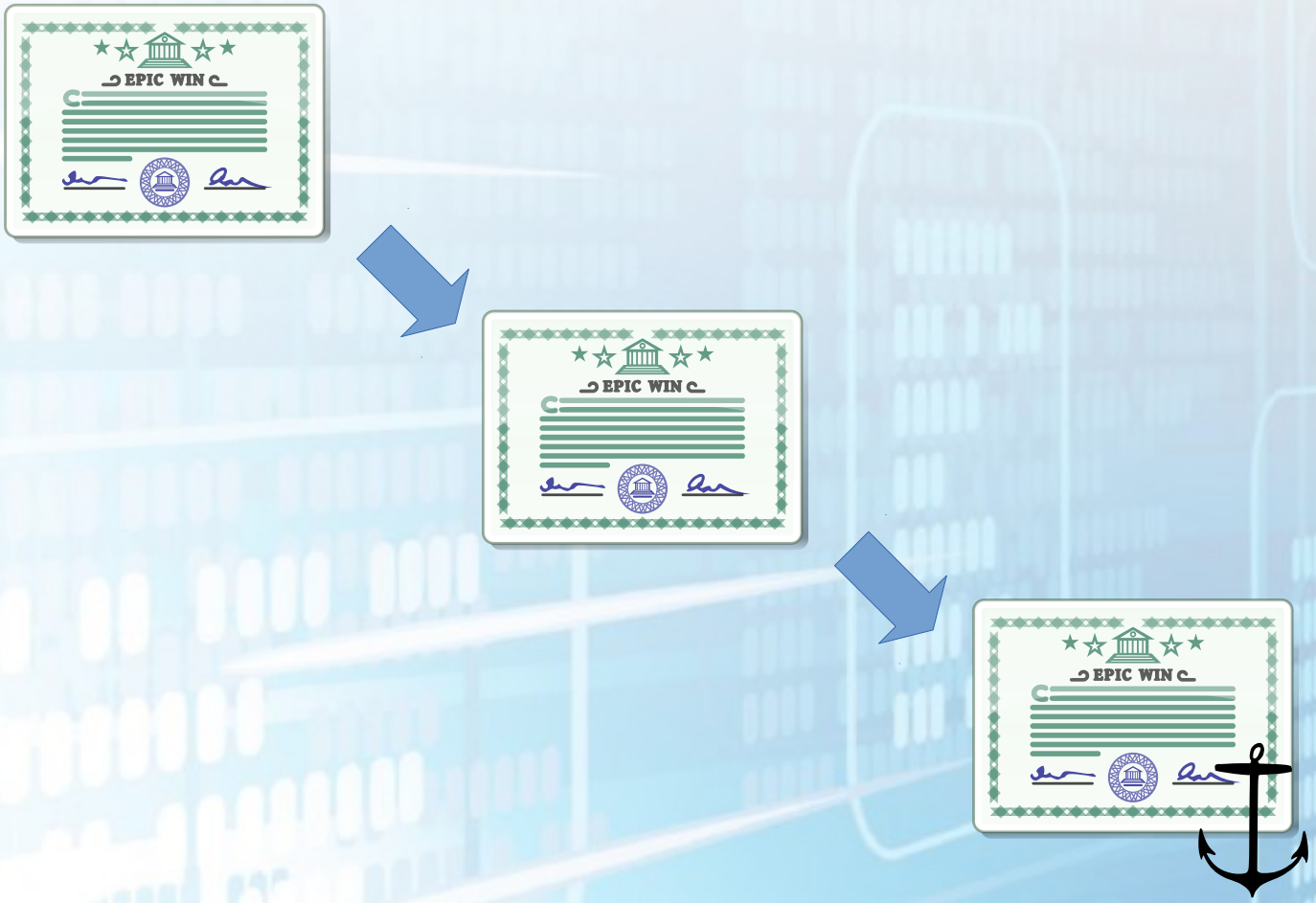
www.example.com

Valid from:
Jan. 1, 2016
To:
Jan. 1, 2017



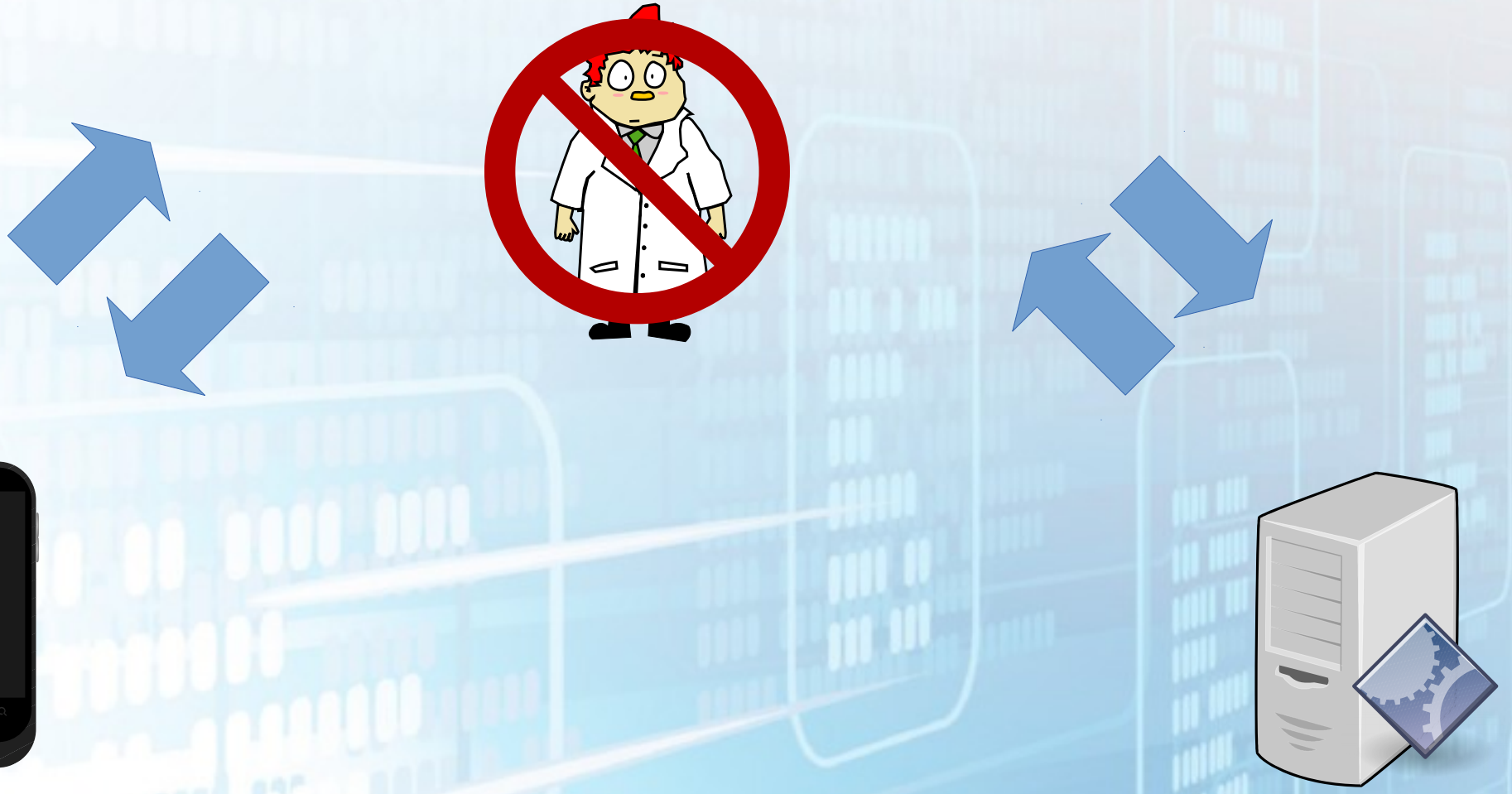
Certificate Authority

Certificate Chain

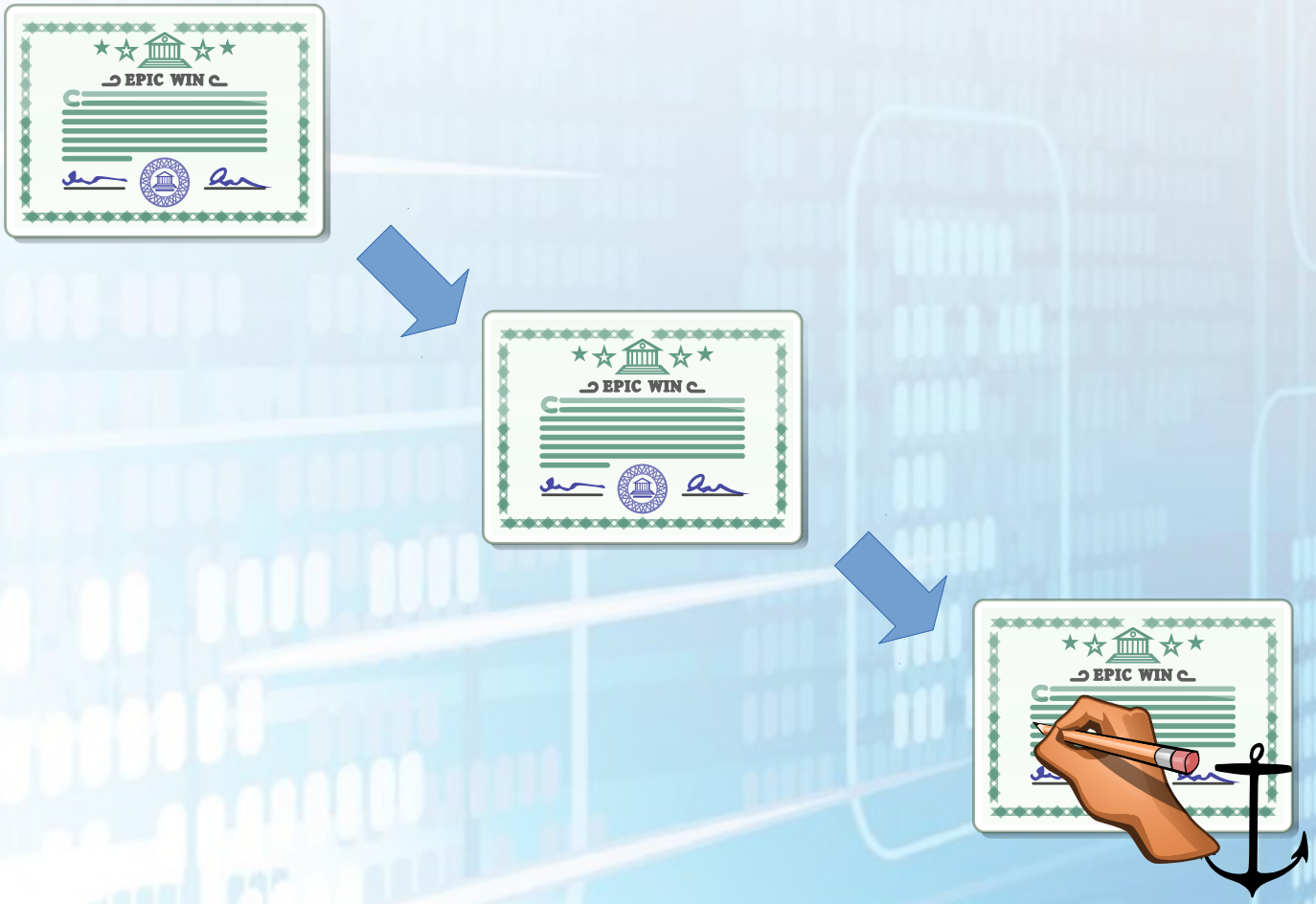


TLS and Real World Attack

TLS and Security Research

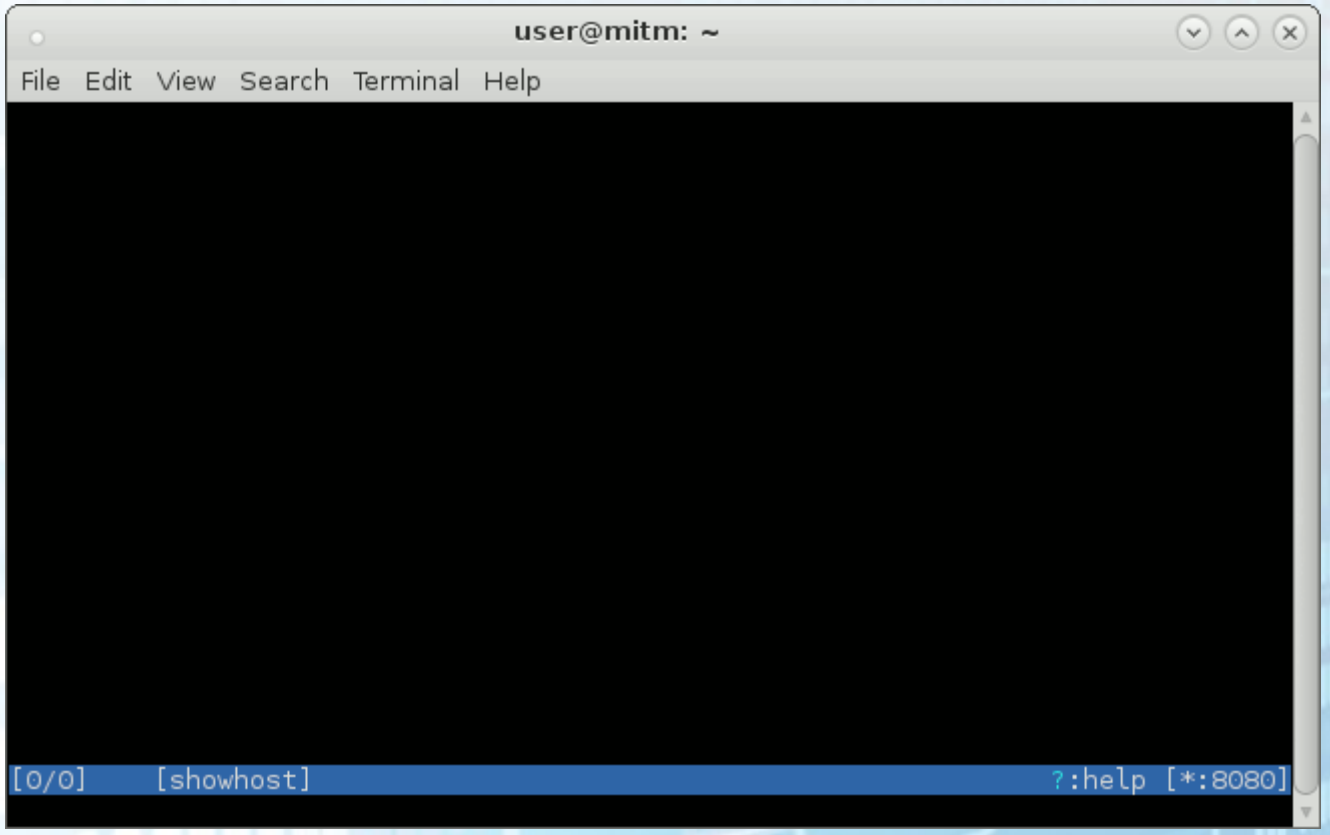


Custom Certificate Authority



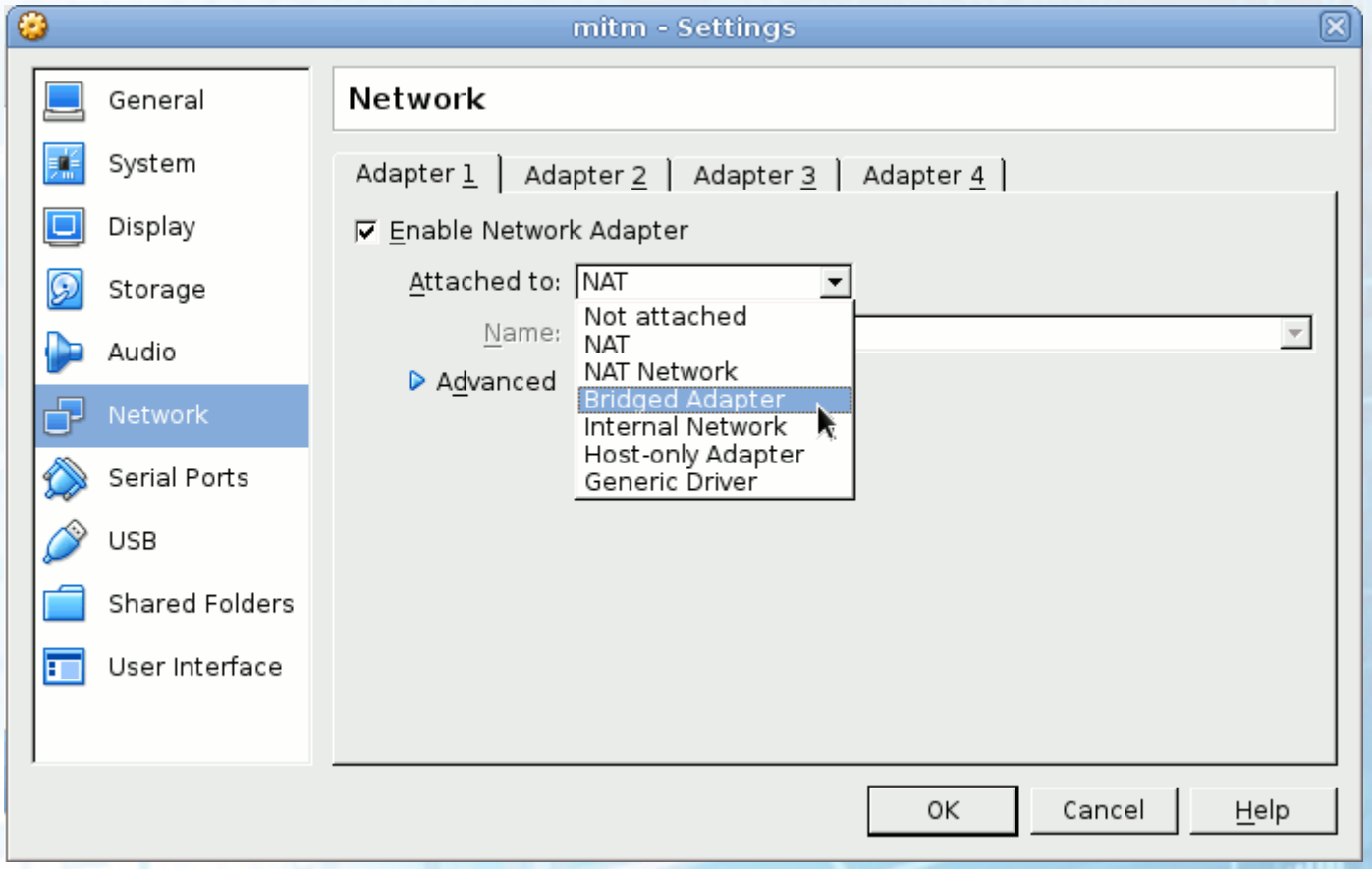
Mitmproxy

Mitmproxy


user@mitm: ~
File Edit View Search Terminal Help
[0/0] [showhost] ?:help [*:8080]

Virtual Machine Setup

Virtual Machine Bridged Mode



Virtual Machine Firewall

```
SUBNET=10.42.16.0/24

for i in /proc/sys/net/ipv4/conf/*/send_redirects; do echo 0 > $i; done

iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p icmp -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -s $SUBNET -j ACCEPT
iptables -A INPUT -d $SUBNET -j ACCEPT
iptables -A INPUT -p tcp -j REJECT --reject-with tcp-reset
iptables -P INPUT DROP

iptables -A FORWARD -s $SUBNET -j ACCEPT
iptables -A FORWARD -d $SUBNET -j ACCEPT
iptables -P FORWARD DROP

iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8080
iptables -t nat -A POSTROUTING -j MASQUERADE

sysctl -w net.ipv4.ip_forward=1
```

Virtual Machine Firewall

```
SUBNET=10.42.16.0/24

for i in /proc/sys/net/ipv4/conf/*/send_redirects; do echo 0 > $i; done

iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p icmp -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -s $SUBNET -j ACCEPT
iptables -A INPUT -d $SUBNET -j ACCEPT
iptables -A INPUT -p tcp -j REJECT --reject-with tcp-reset
iptables -P INPUT DROP

iptables -A FORWARD -s $SUBNET -j ACCEPT
iptables -A FORWARD -d $SUBNET -j ACCEPT
iptables -P FORWARD DROP

iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8080
iptables -t nat -A POSTROUTING -j MASQUERADE

sysctl -w net.ipv4.ip_forward=1
```

iptables rules

Launching Mitmproxy

```
mitmproxy --host
```
(HTTP proxy mode)

```
mitmproxy -T --host
```
(Transparent mode)

Mitmproxy's ability to run as a transparent proxy is what allows us to study the traffic of non-proxy-enabled devices or software- so long as we can control certificate verification!

Configuring the Client Device

Configuring the Client Device

- Proxy-Enabled
  - Browsers
  - Most desktop applications
  - Most iOS applications
- Non-Proxy-Enabled
  - Most Android applications
  - Many standalone devices

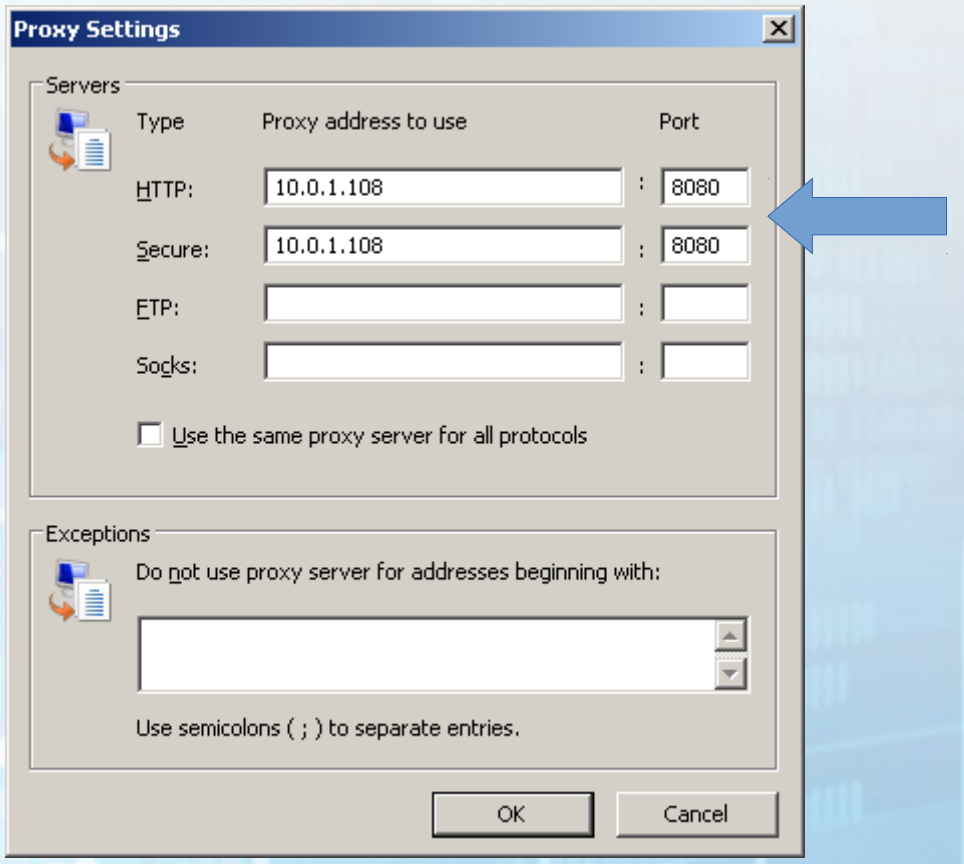
General Technique

- Route traffic through Mitmproxy VM
  - Explicit proxy if available
  - Default gateway configuration if not
- Add Mitmproxy CA to trusted CAs
- View traffic

Windows Applications

Windows Applications

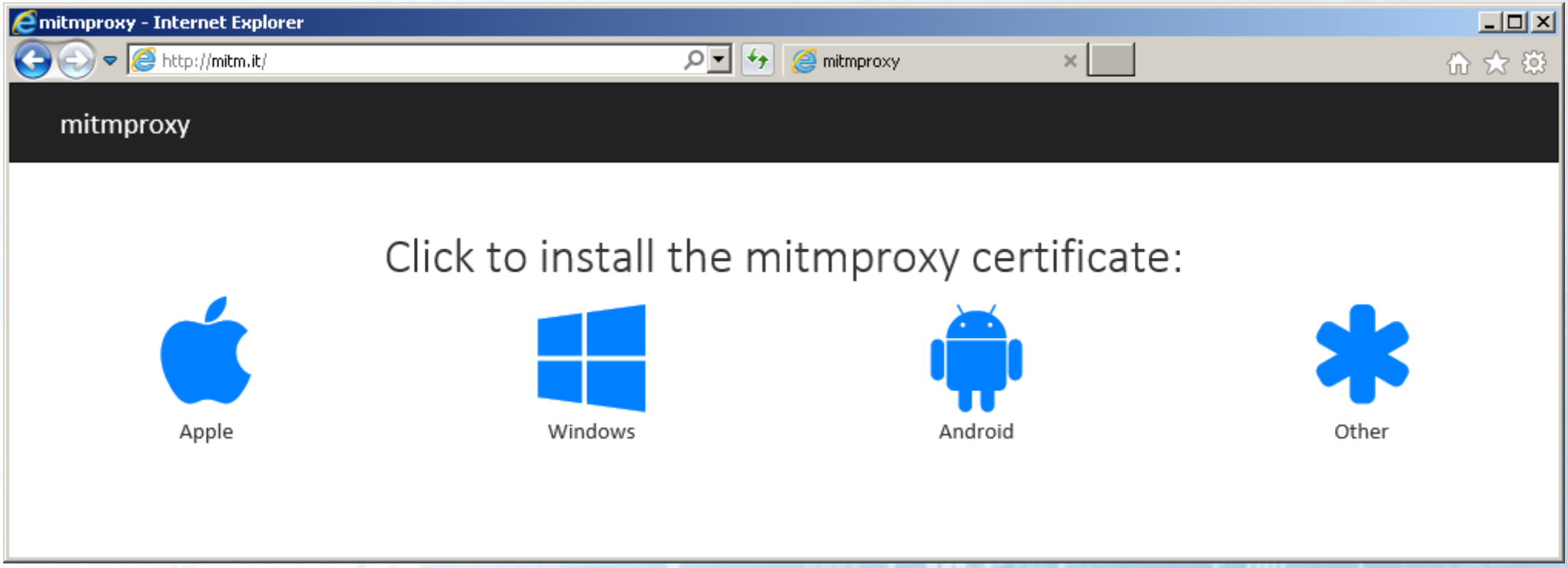
IE Proxy Configuration



Tools → Internet Options → Connections → LAN settings → Advanced

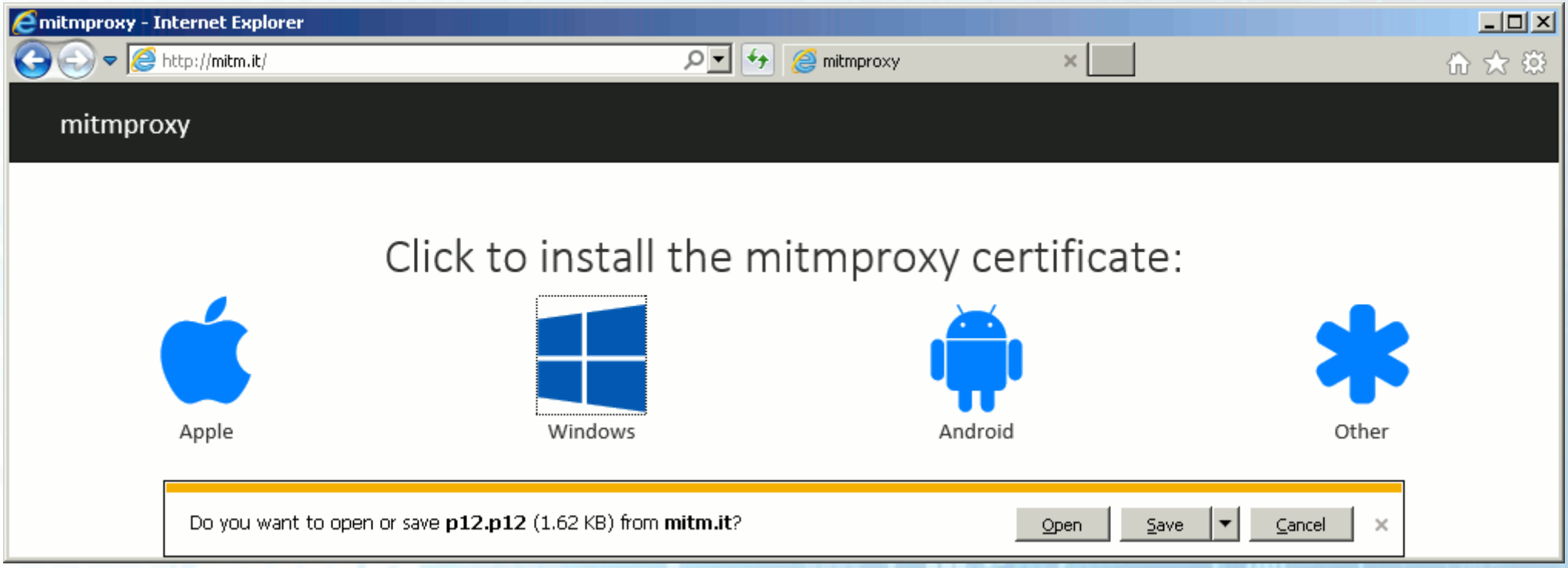
Verify that Mitmproxy is in HTTP proxy mode!

Mitmproxy CA Download Page

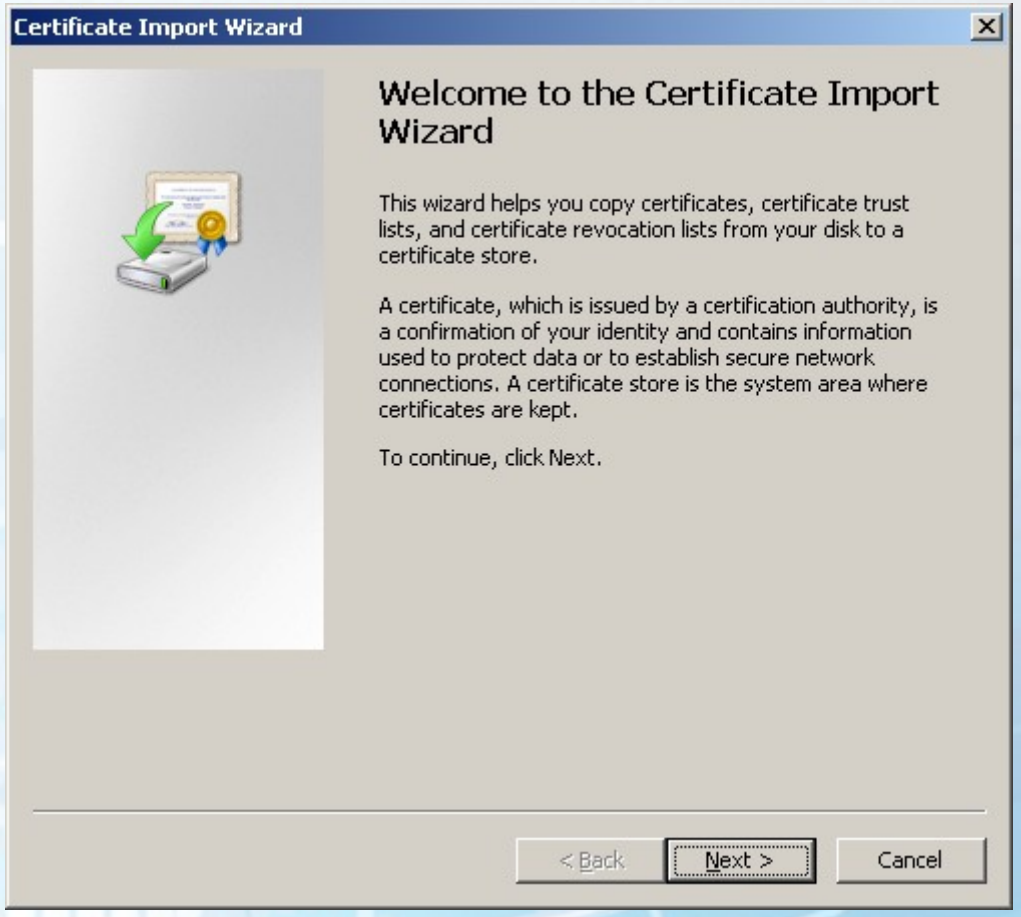


http://mitm.it/

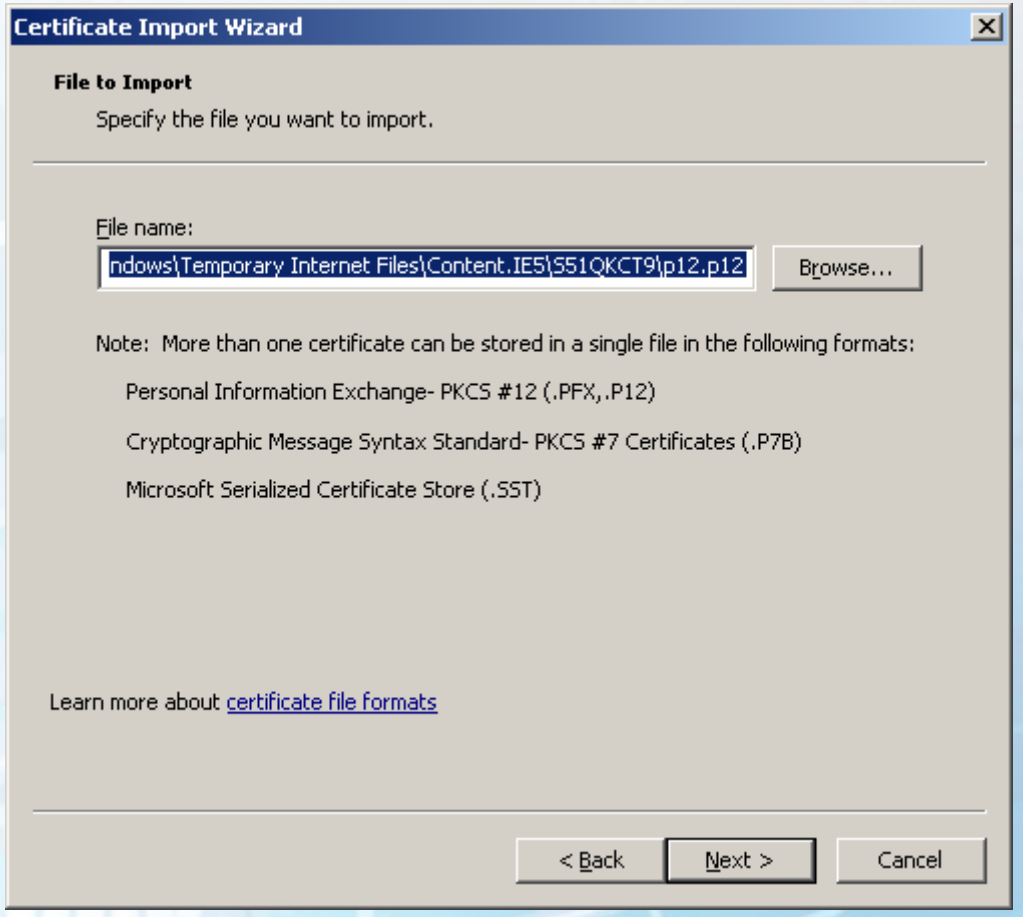
Mitmproxy CA Download



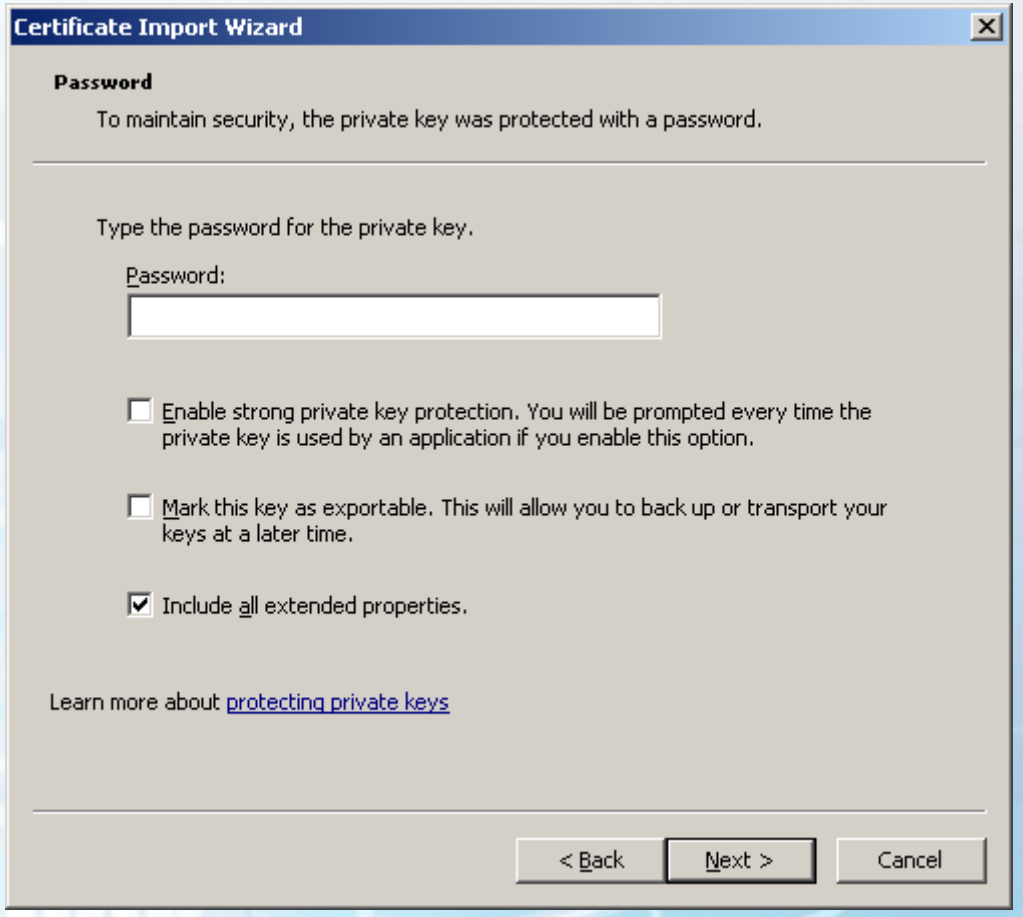
Importing the Mitmproxy CA



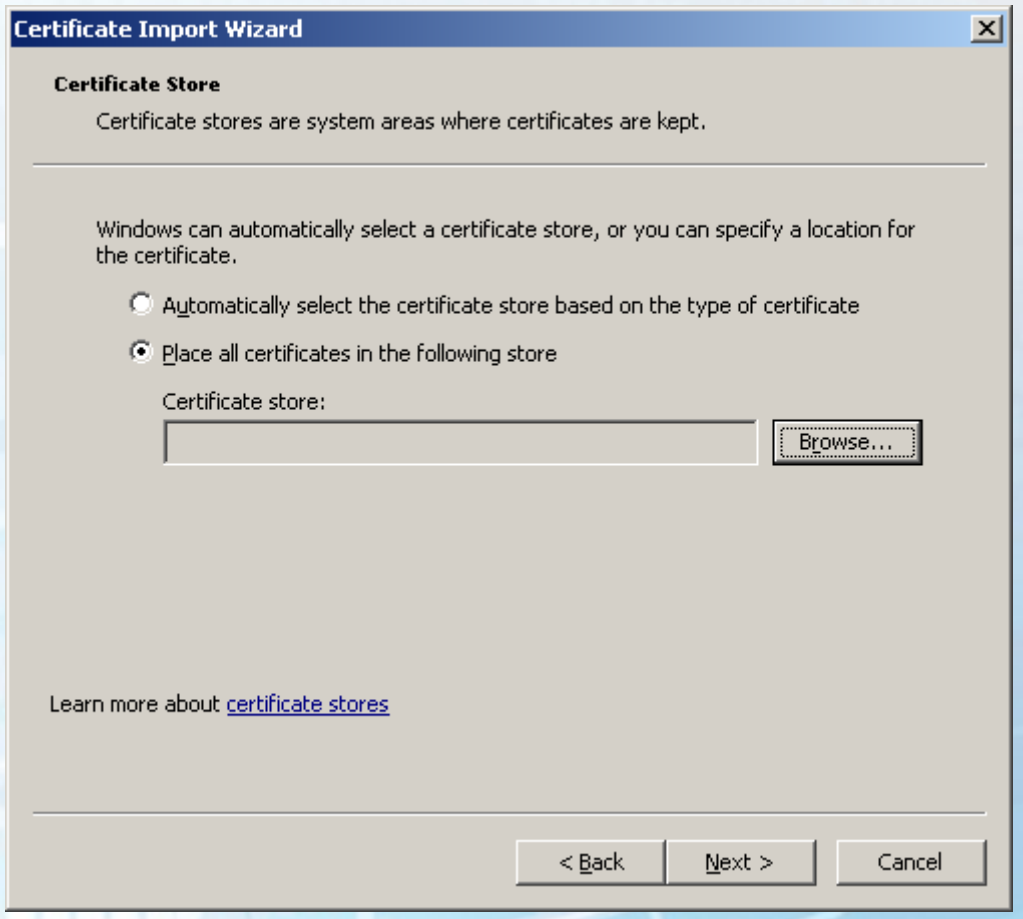
Importing the Mitmproxy CA



Importing the Mitmproxy CA

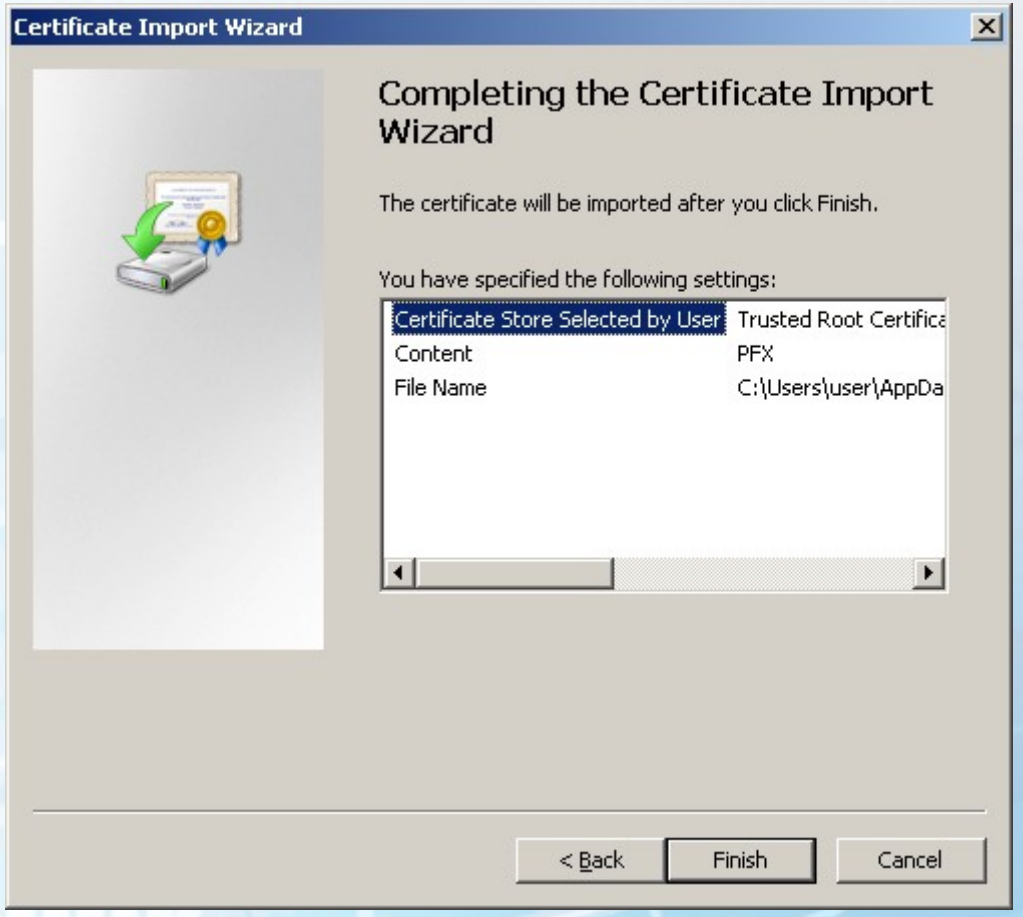


Importing the Mitmproxy CA

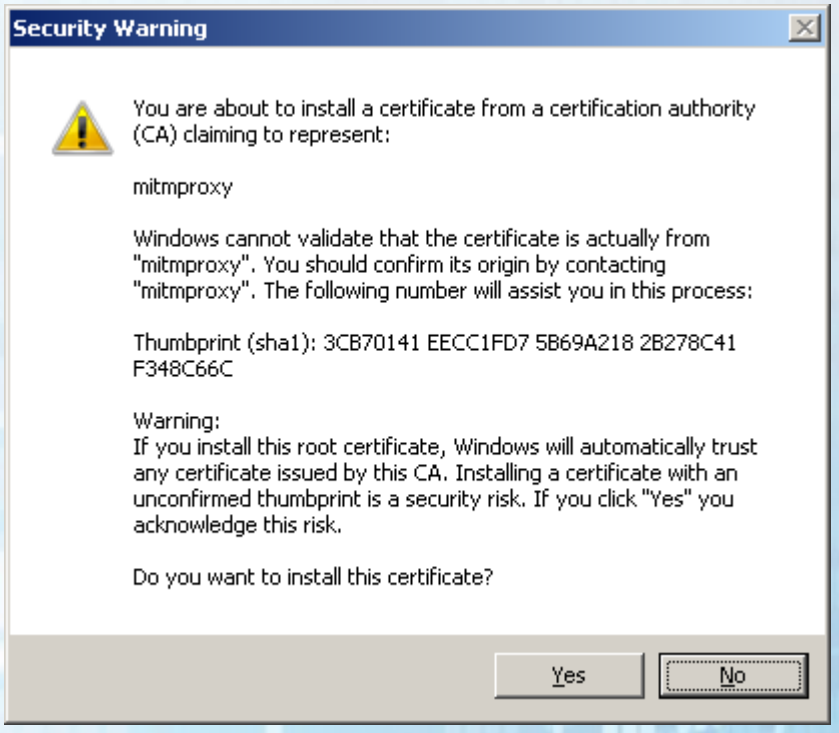


Importing the Mitmproxy CA

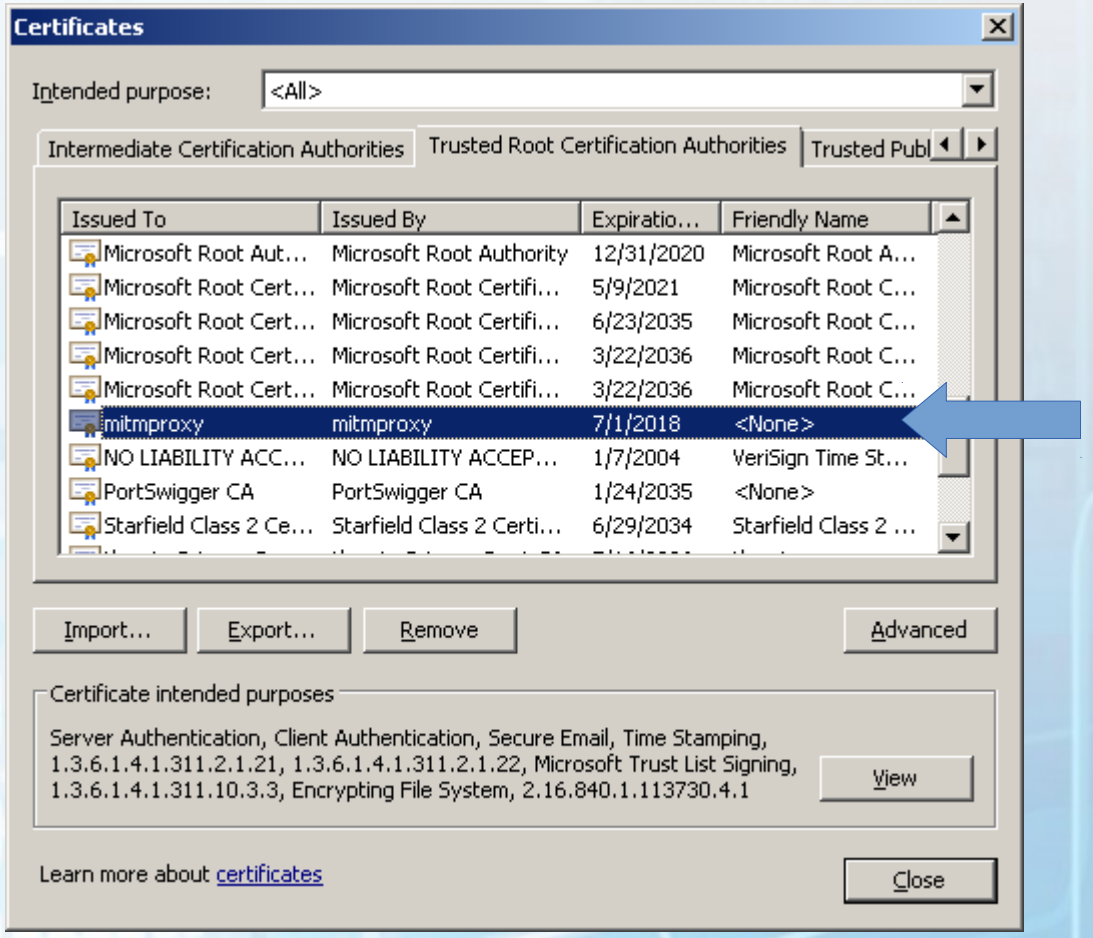
Importing the Mitmproxy CA



Importing the Mitmproxy CA

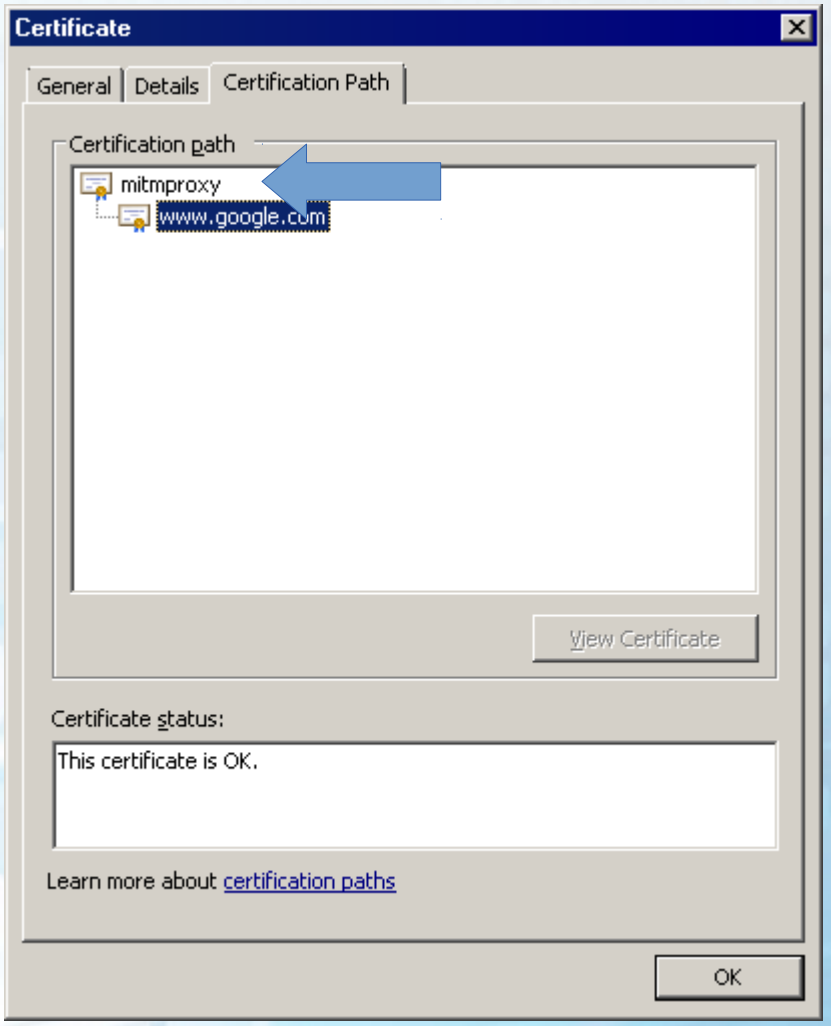


Verify CA Installation

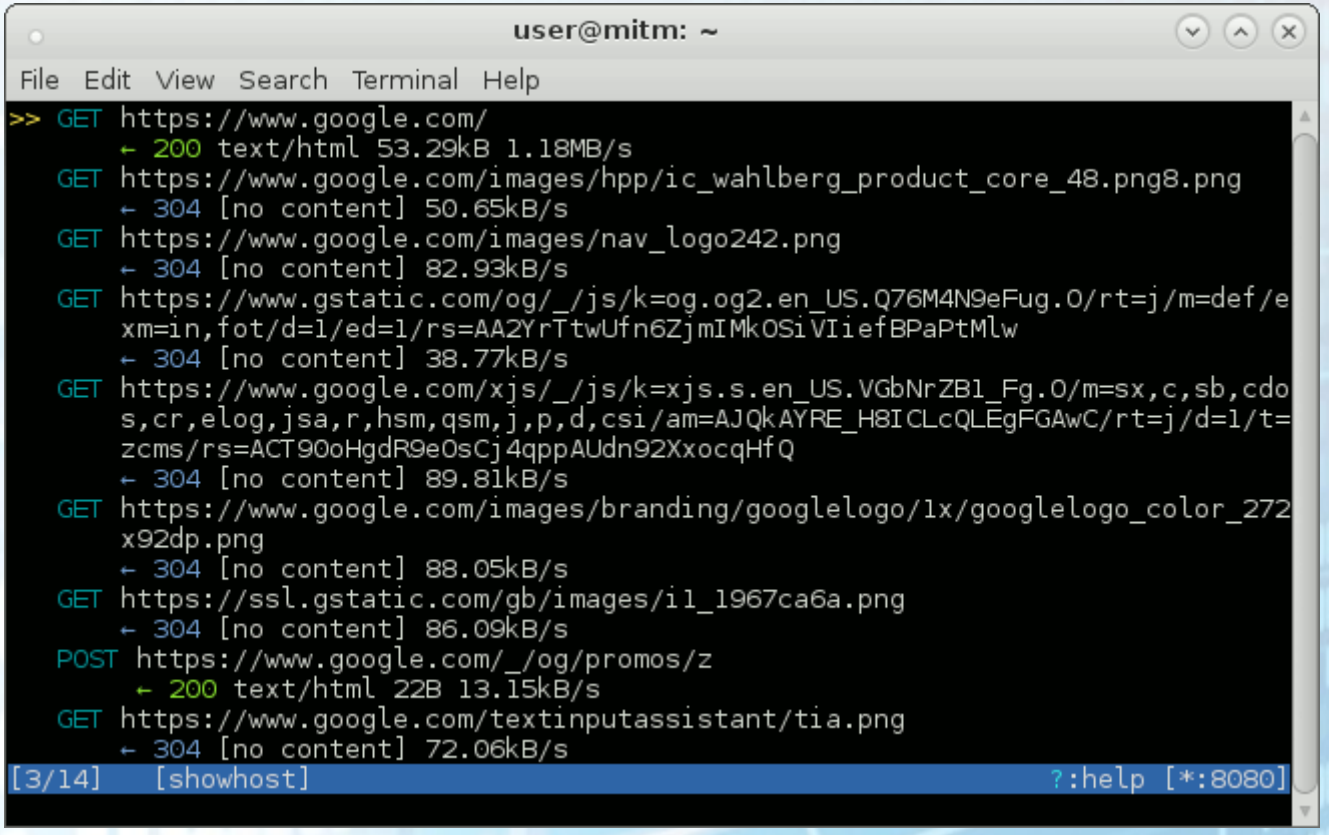


Tools → Internet Options → Content → Certificates → Trusted Root

Visiting an HTTPS Site



Visiting an HTTPS Site



Troubleshooting

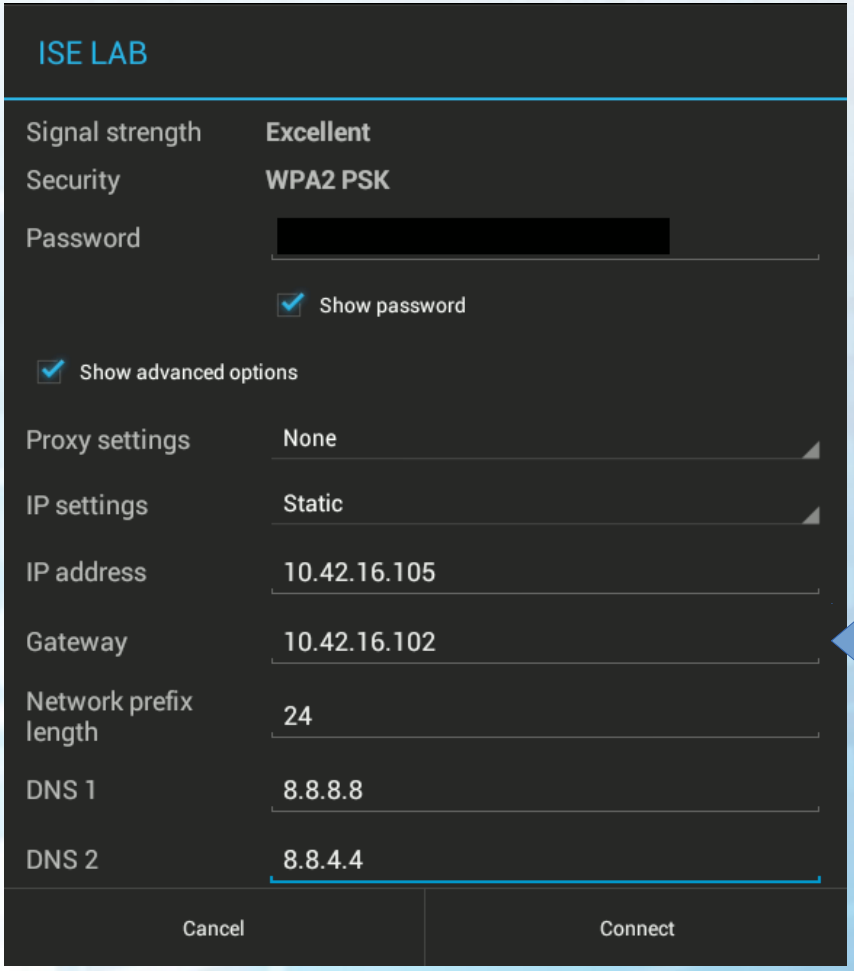
- Applications that don't follow IE proxy settings
  - Use default gateway technique
  - As on Android (up next)

Troubleshooting

- Applications that don't follow IE CA settings
  - Mozilla Firefox
  - Most applications with embedded TLS libraries
    - OpenSSL
    - NSS
  - Java applications
    - More to come later

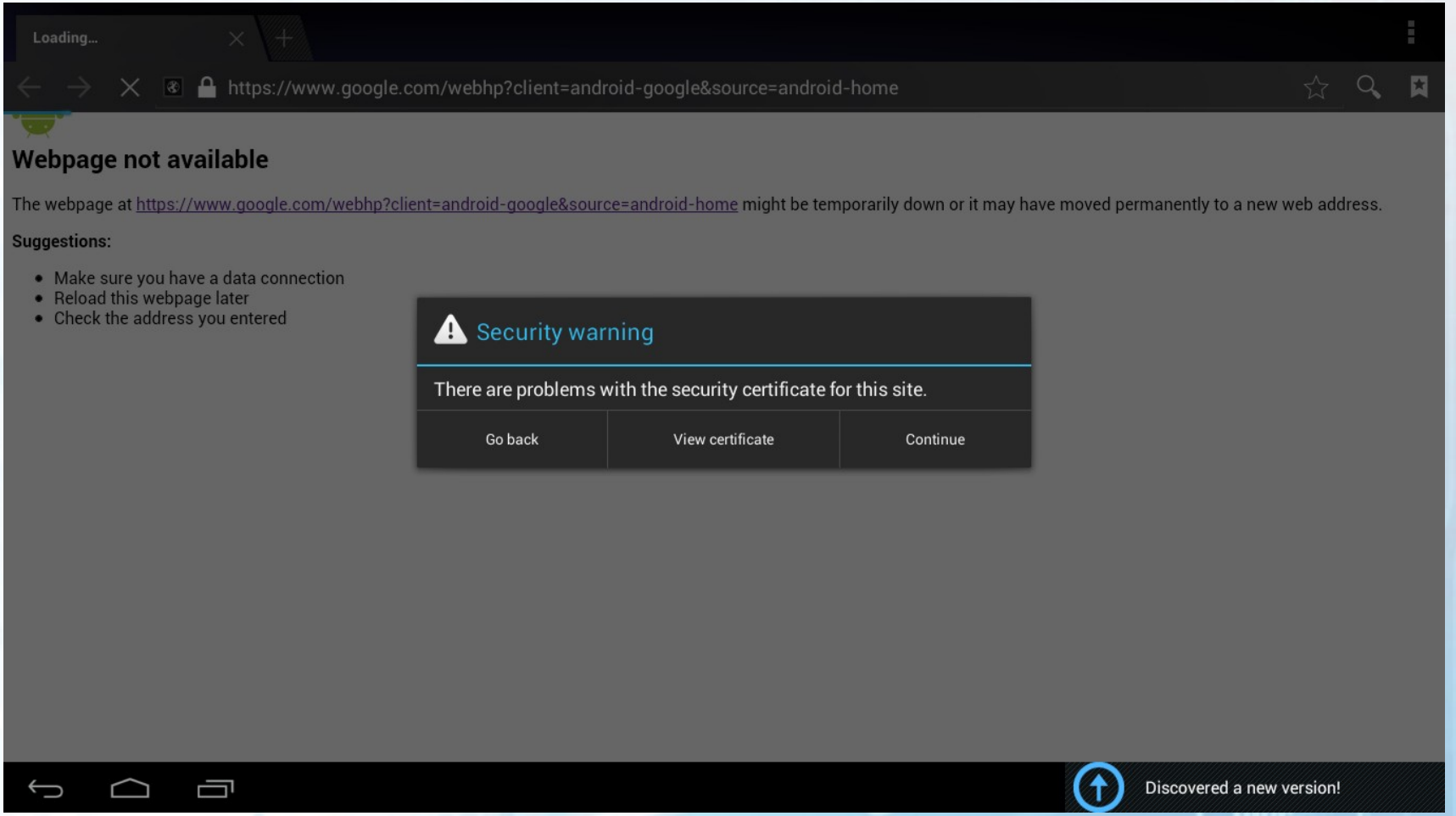
Android Applications

Android Network Settings

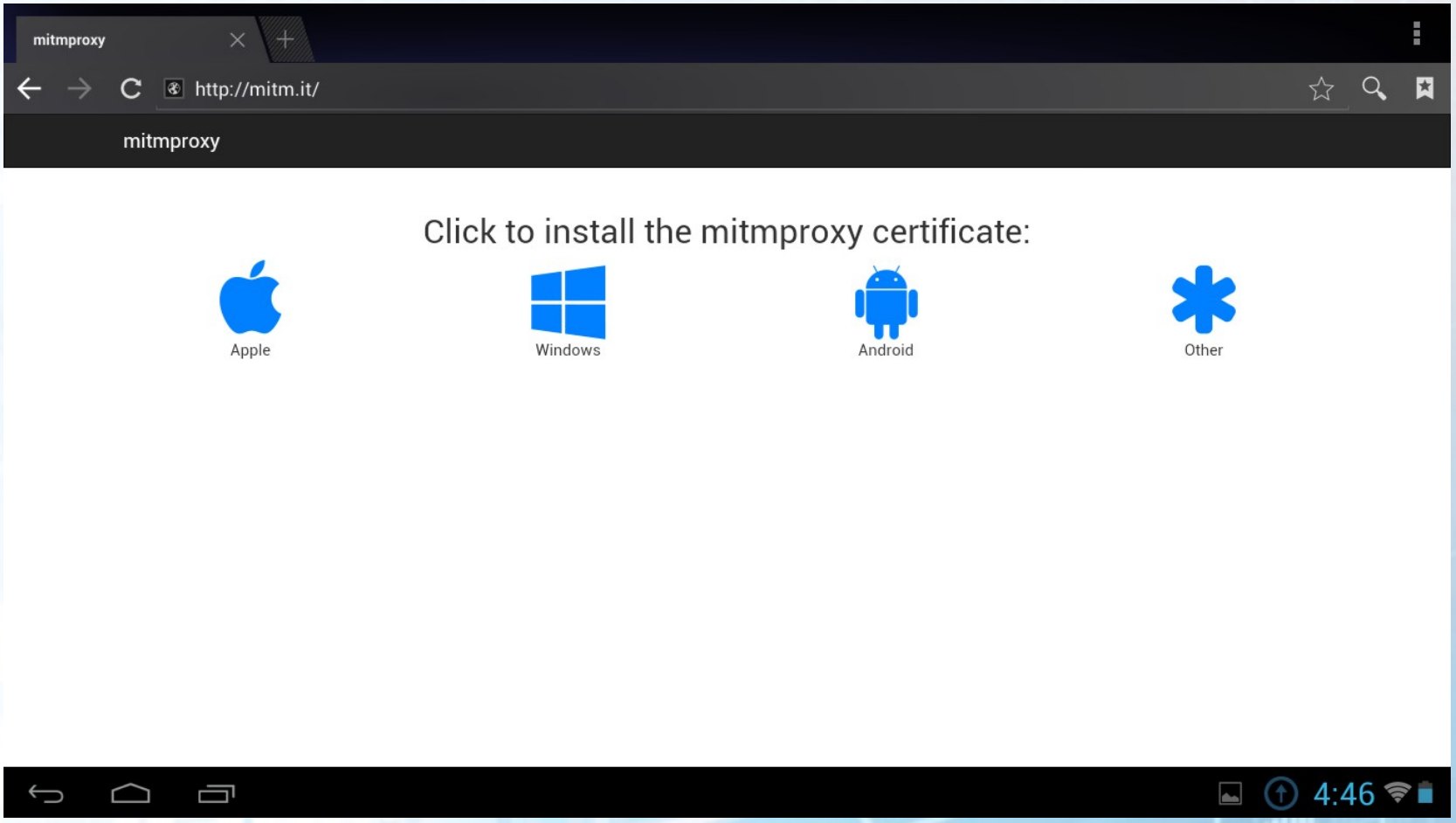


Ensure Mitmproxy is in transparent mode!

Android Browser Certificate Warning

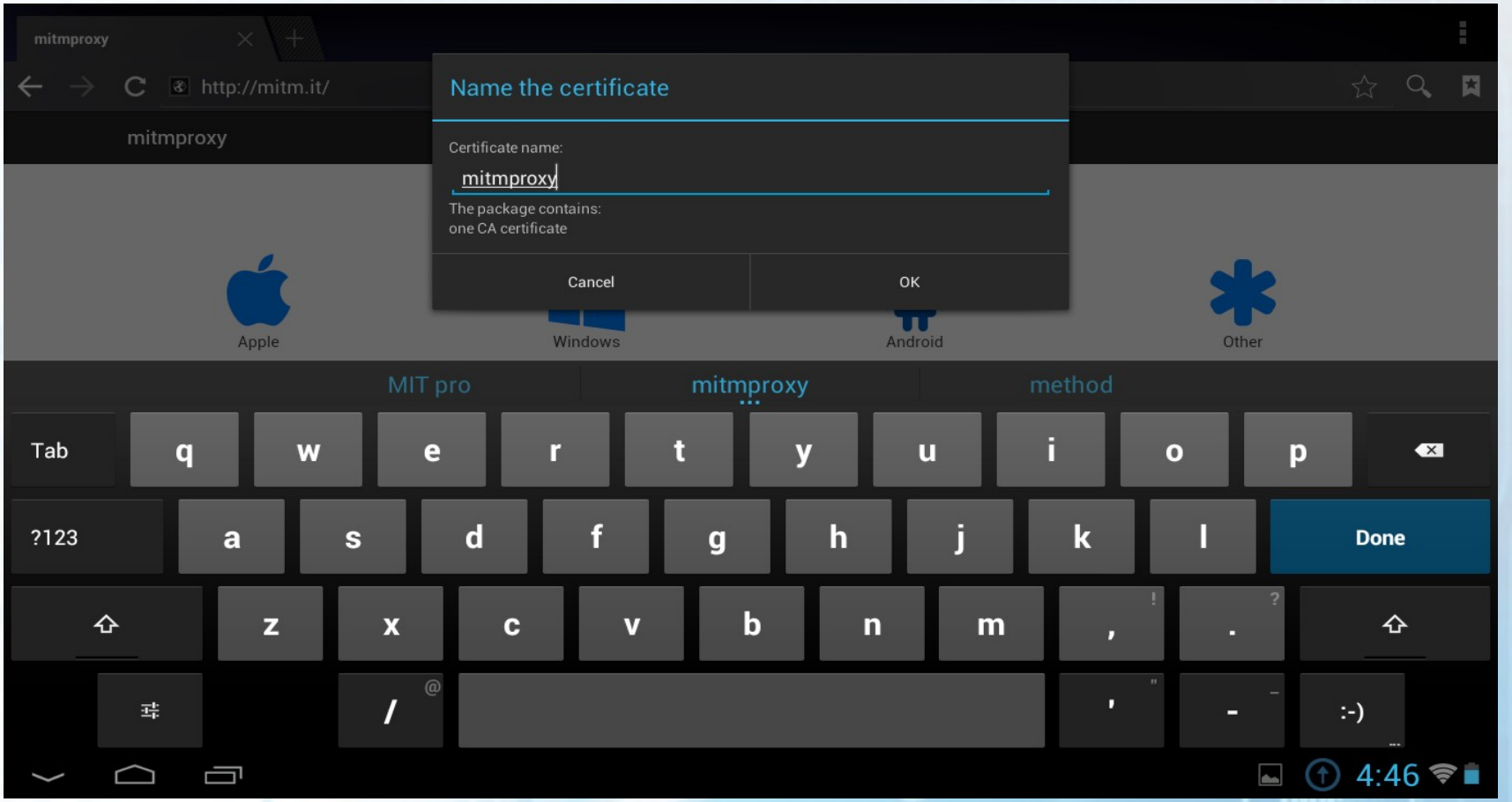


Mitmproxy CA Download Page

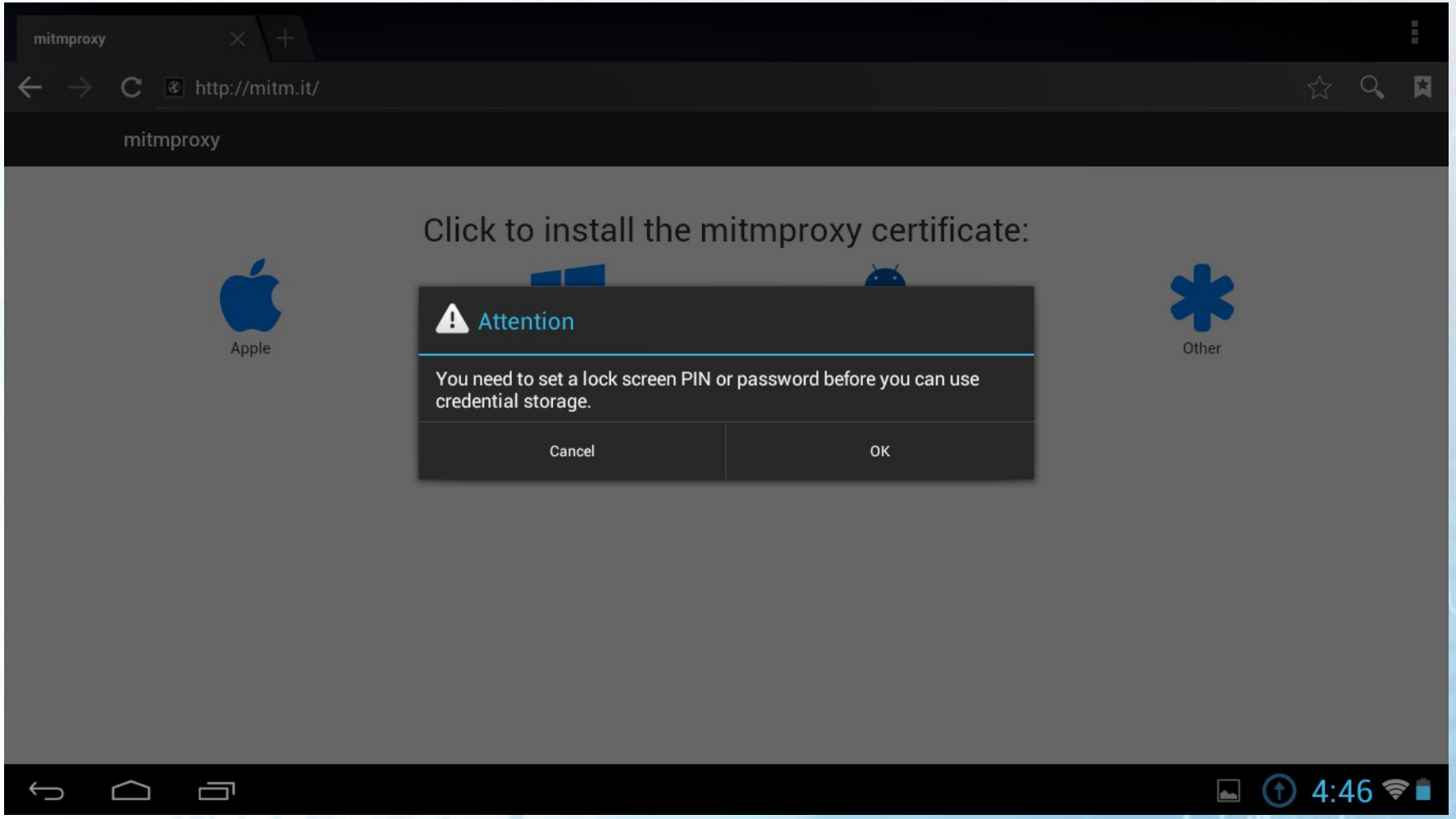


http://mitm.it/

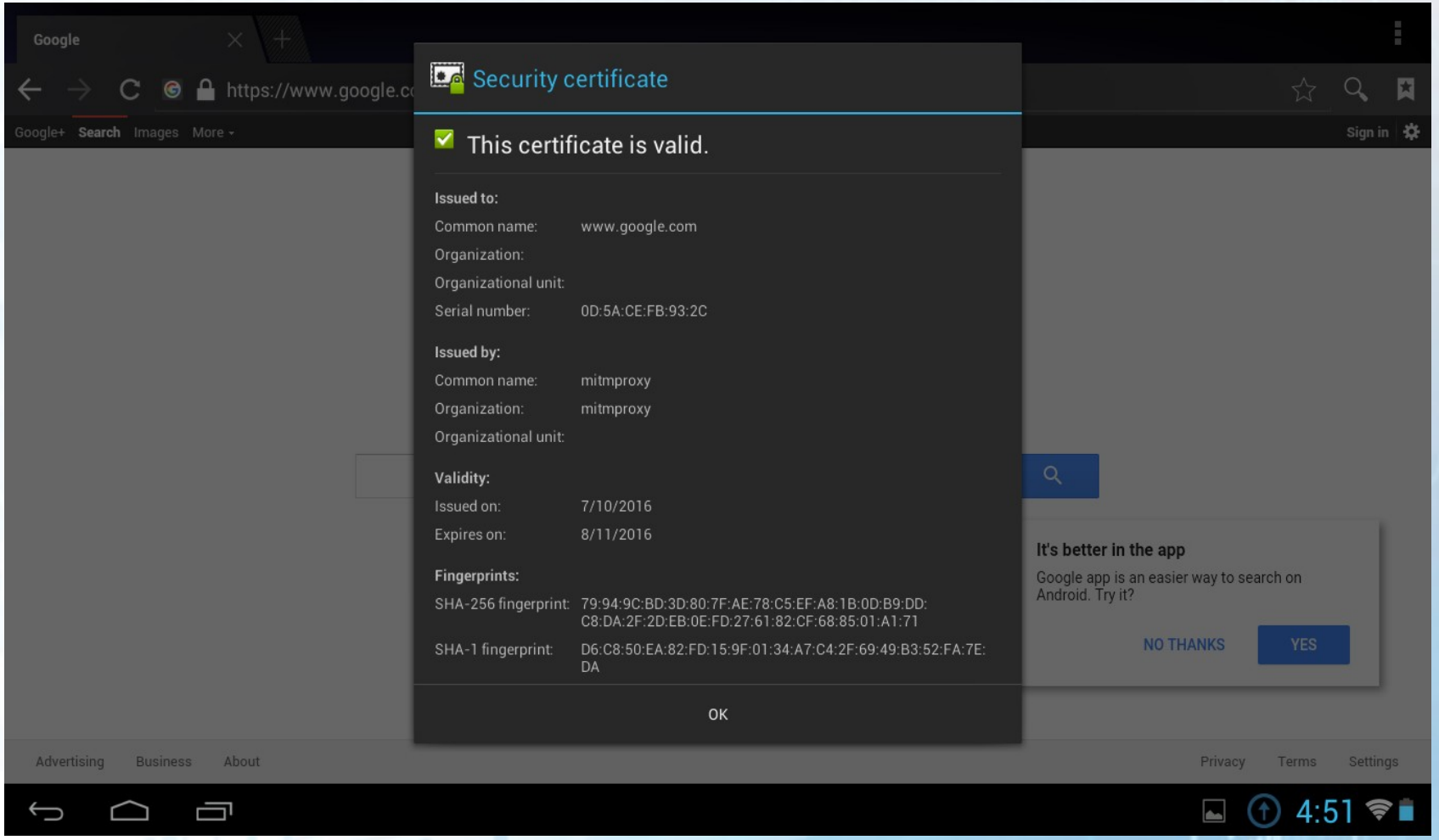
Mitmproxy CA Installation



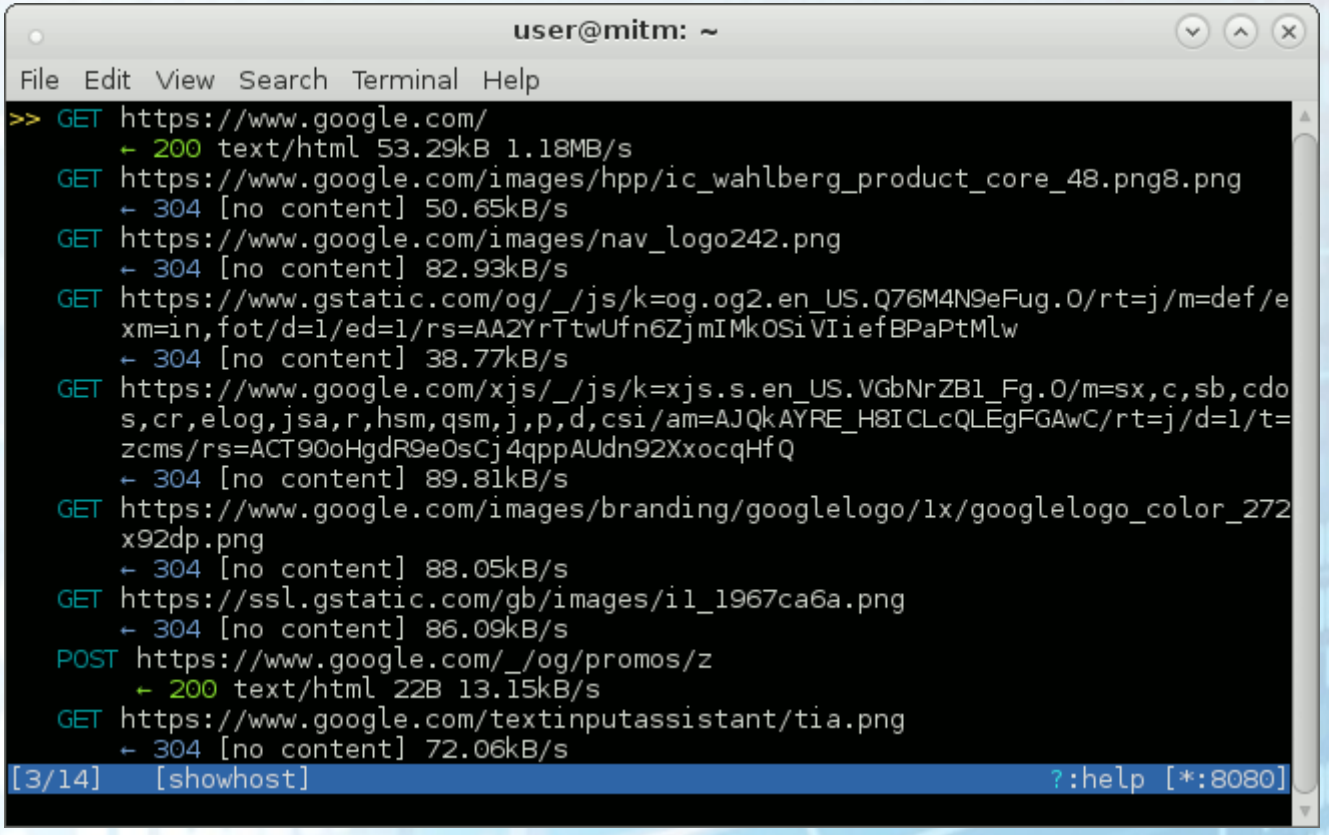
Android Credential Storage PIN



Android Browser



Visiting an HTTPS Site

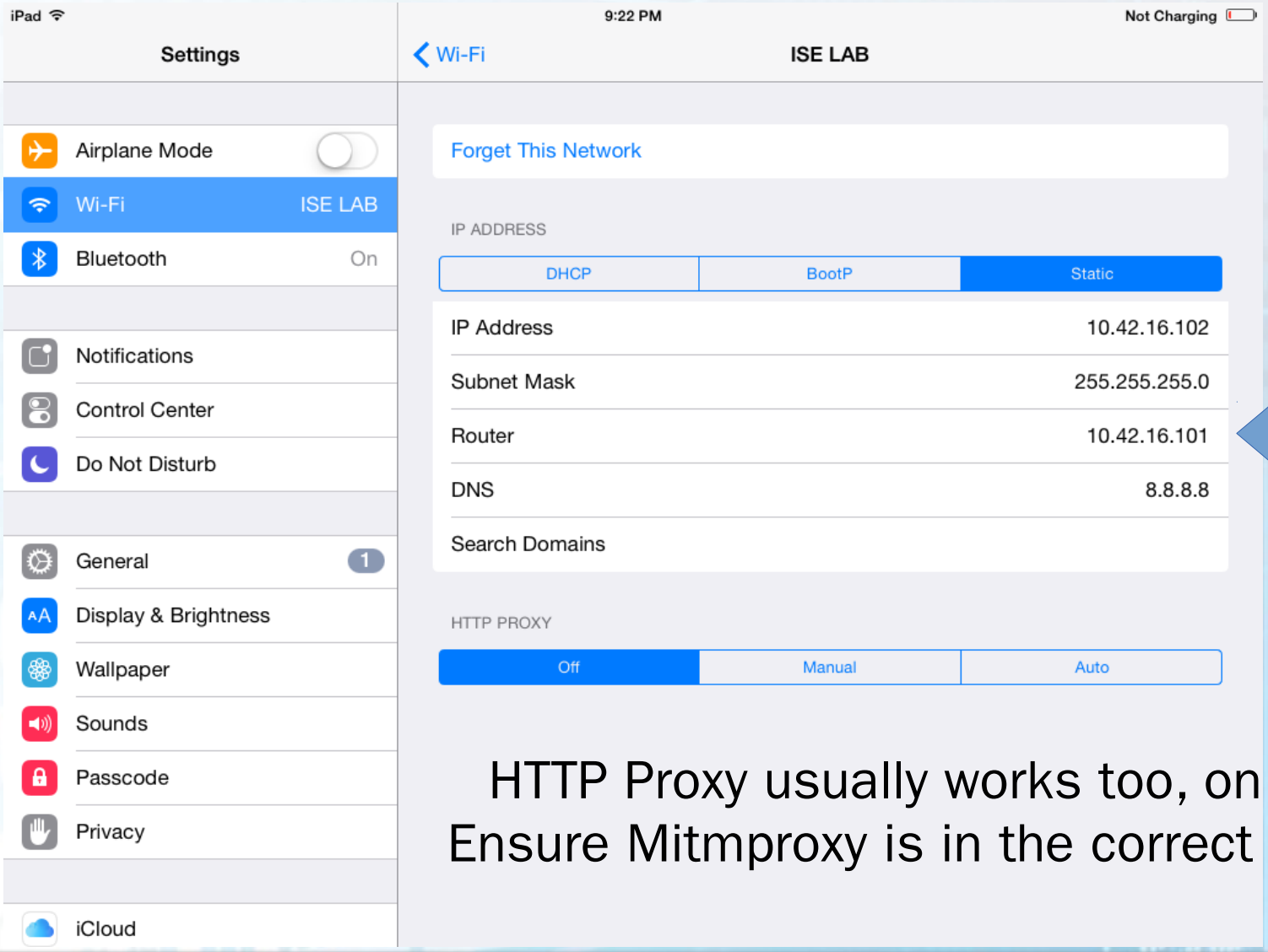


Troubleshooting

- Proxy setting on Android is unreliable
  - Notice we used default gateway
- Android Nougat (7) trusted CA changes
- Certificate pinning
- JNI

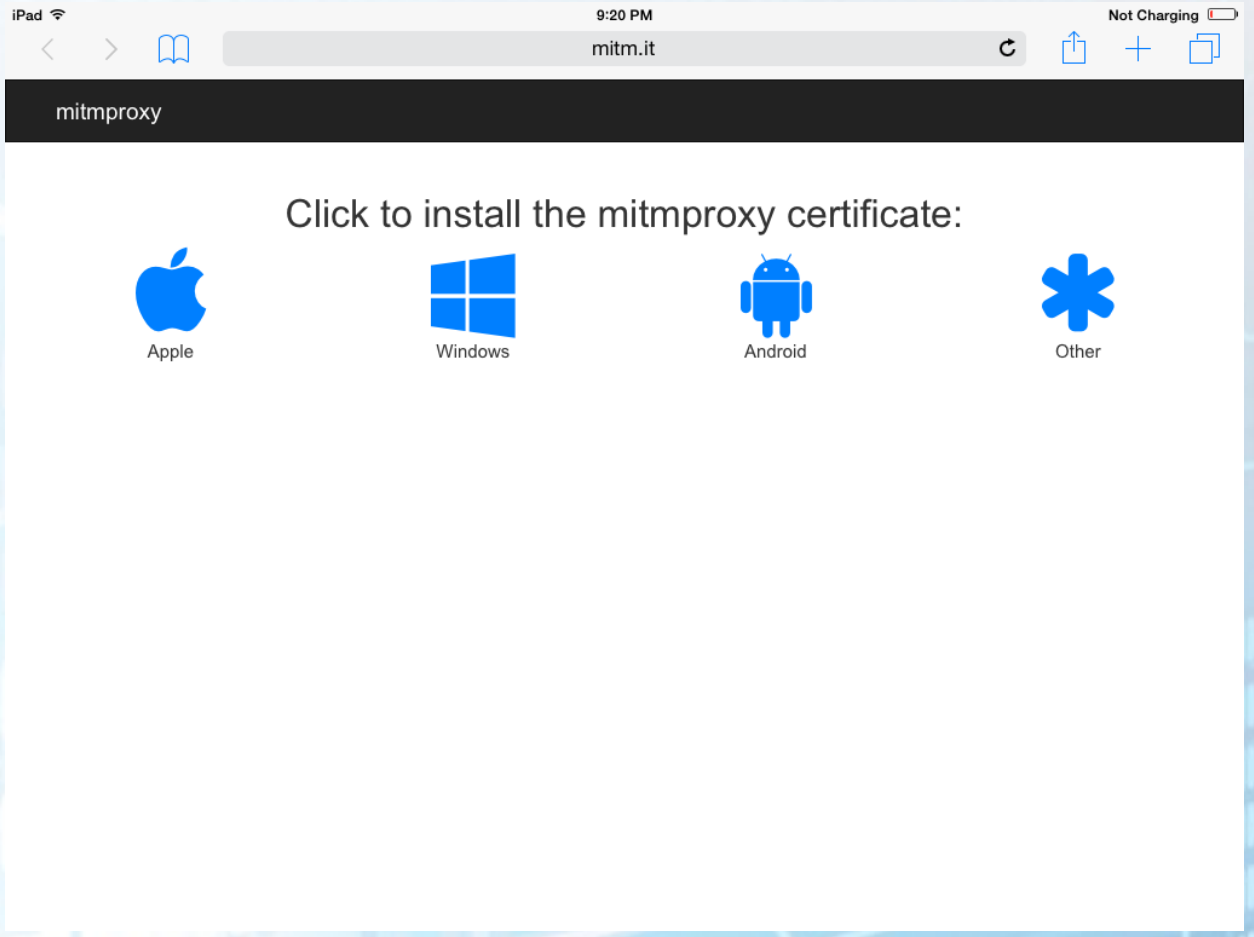
iOS Applications

iOS Network Settings



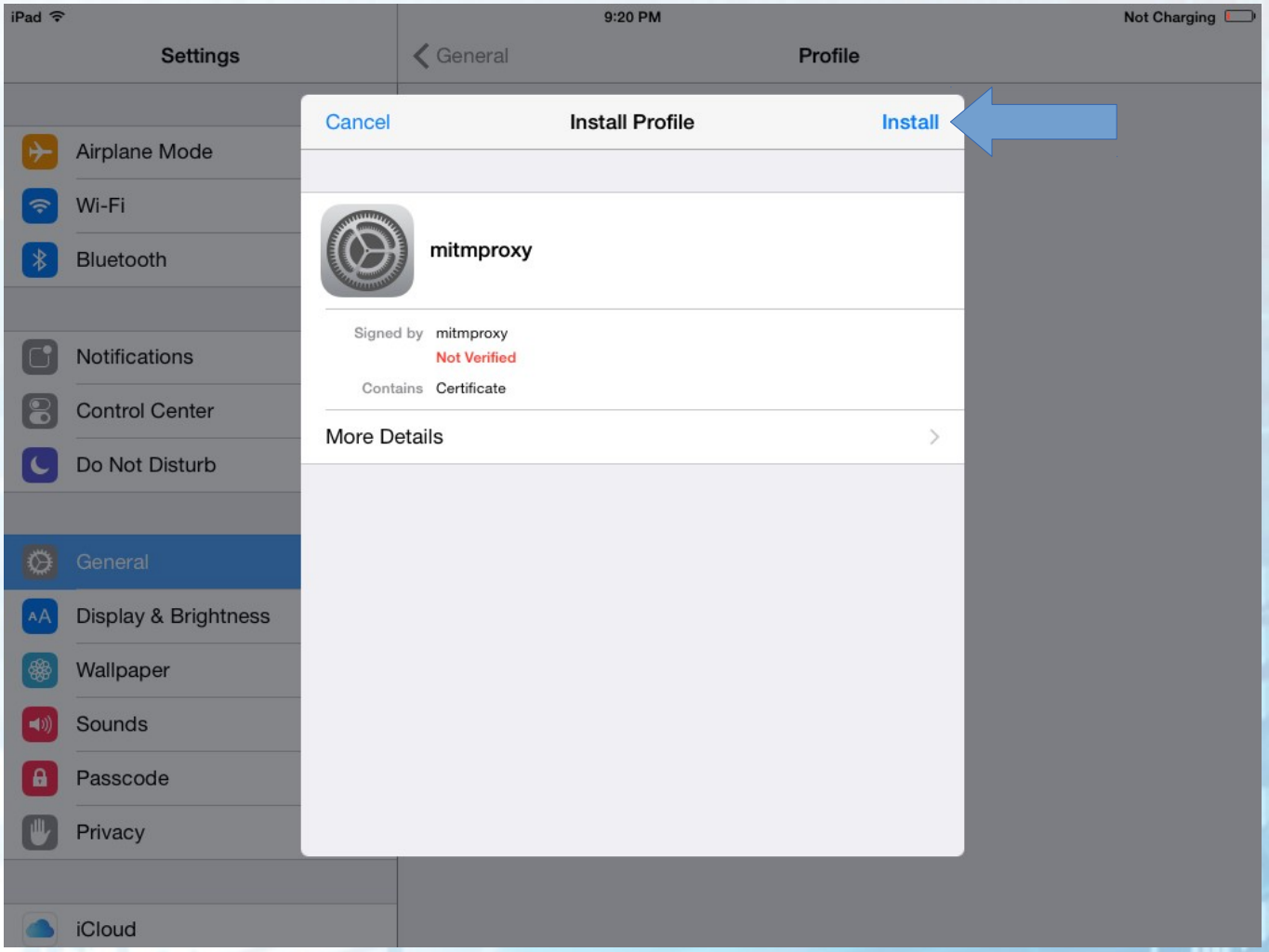
HTTP Proxy usually works too, on iOS.
Ensure Mitmproxy is in the correct mode!

Mitmproxy CA Download Page

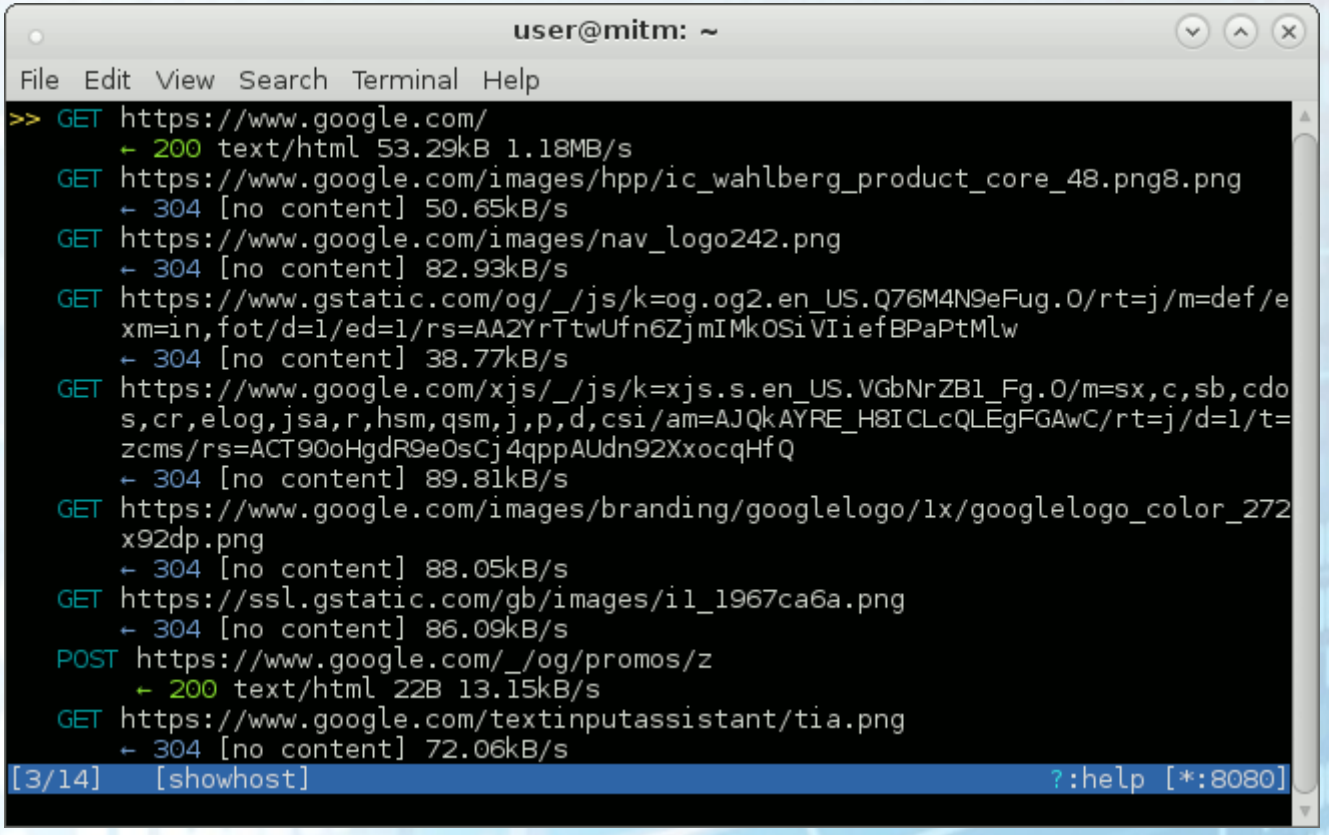


http://mitm.it/

Mitmproxy CA Installation



Visiting an HTTPS Site



Troubleshooting

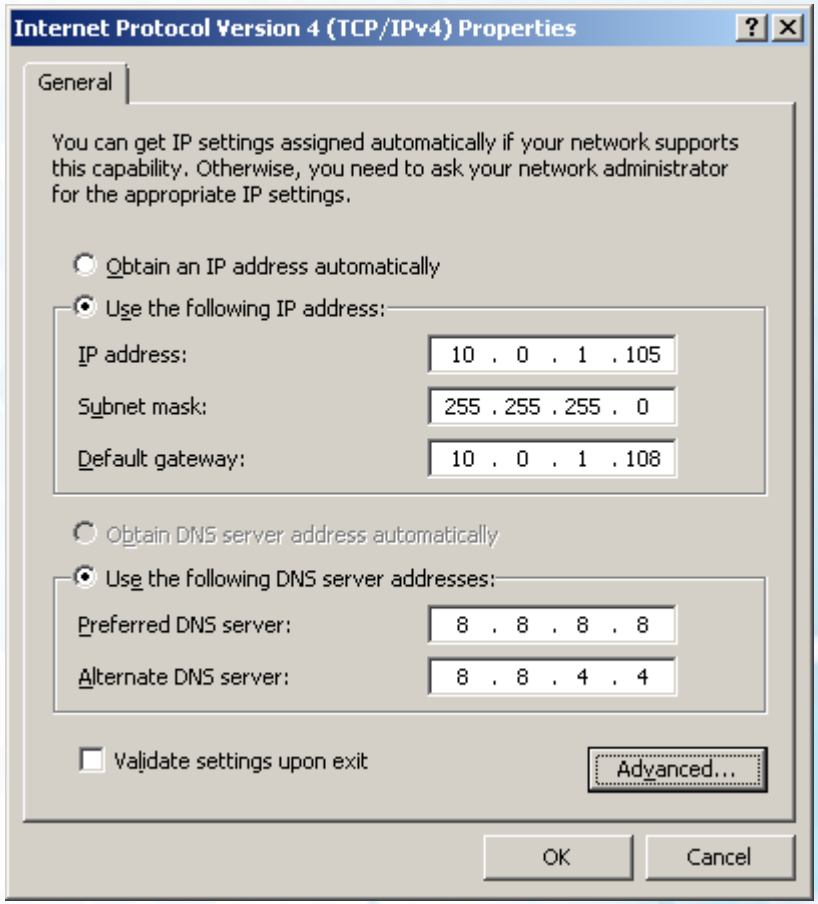
- Certificate pinning
  - See SSL Kill Switch and similar tools

Java Applications

Java Applications

- Java does *not* always use Internet Explorer or other system-wide proxy and certificate settings

Configure Default Gateway



Could also configure Java proxy settings.

Ensure Mitmproxy is in transparent mode!

Sample Java Application (jcurl)

```
C:\>java -jar jcurl-all.jar -e url https://www.google.com/
```

Sample Java Application (jcurl)

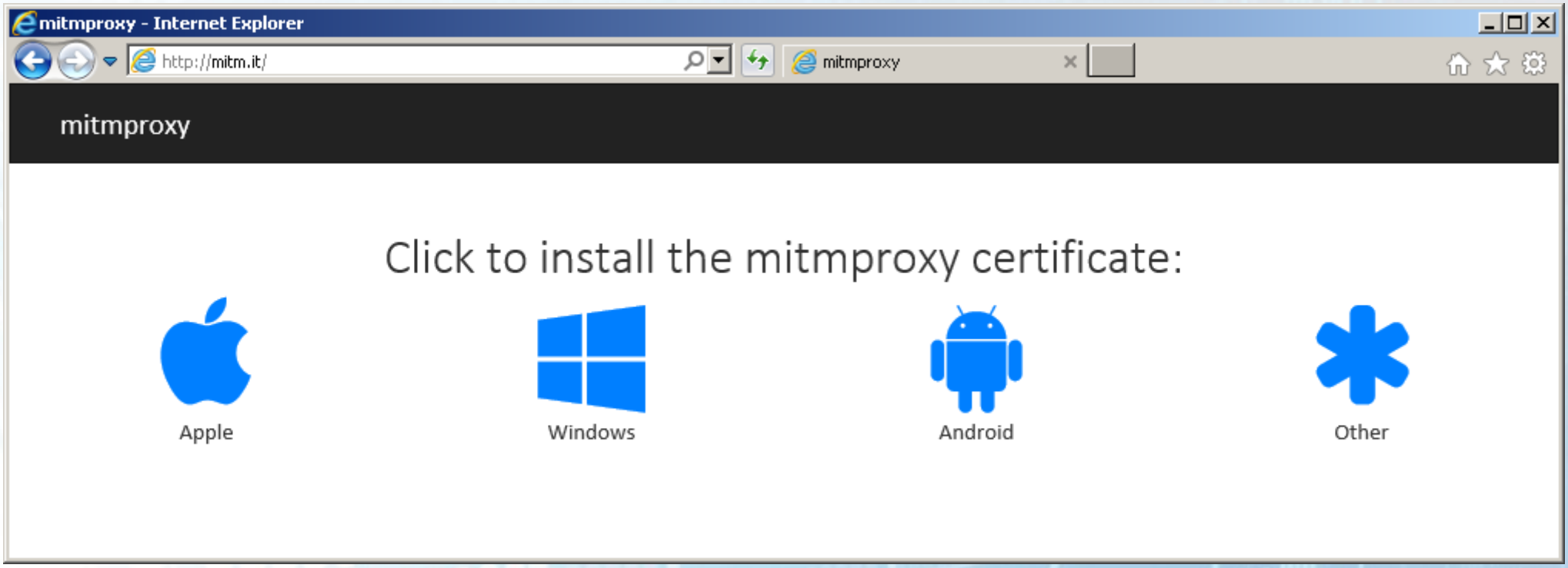
```
C:\>java -jar jcurl-all.jar -e url https://www.google.com/
Starting jCurl in C:\

Sending 'GET' request to URL : https://www.google.com/
Exception in thread "main" javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
    at sun.security.ssl.Alerts.getSSLException(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.fatal(Unknown Source)
    at sun.security.ssl.Handshaker.fatalSE(Unknown Source)
    at sun.security.ssl.Handshaker.fatalSE(Unknown Source)
    at sun.security.ssl.ClientHandshaker.serverCertificate(Unknown
Source)
    at sun.security.ssl.ClientHandshaker.processMessage(Unknown
Source)
    at sun.security.ssl.Handshaker.processLoop(Unknown Source)
    at sun.security.ssl.Handshaker.process_record(Unknown Source)
...
```

Java Certificate Authority Store

`$JAVA_HOME/lib/security/cacerts`

Different paths for 32-bit and 64-bit JREs
JDK vs. JRE location varies

Downloading Mitmproxy CA

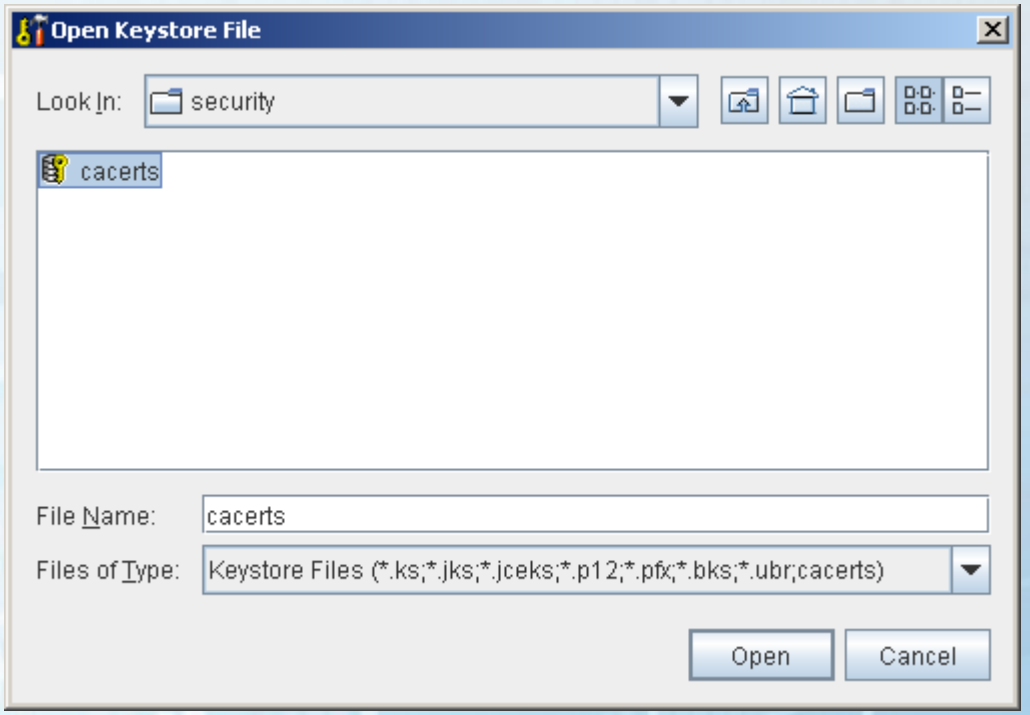


http://mitm.it/

Portecle

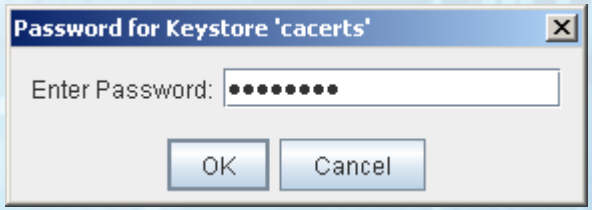
```
java -jar portecle.jar
```

Open cacerts File



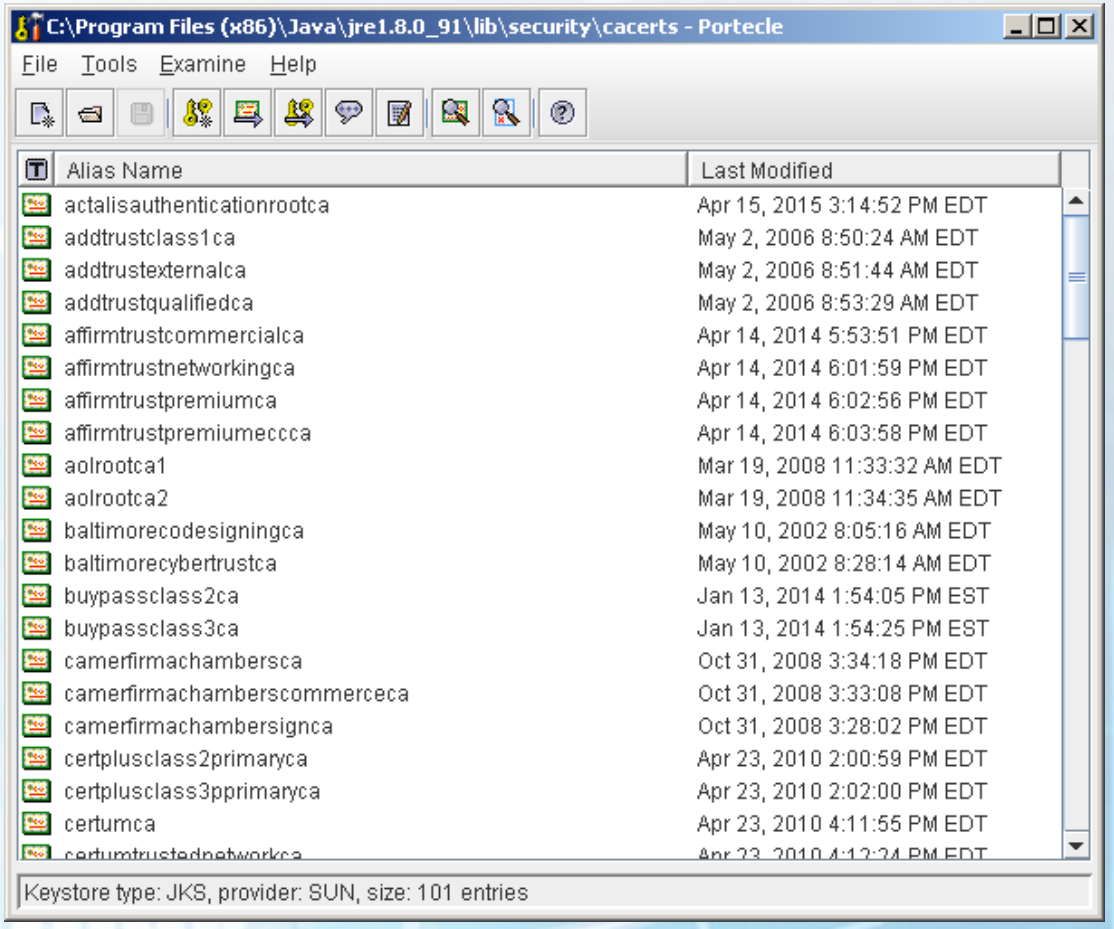
Must have sufficient privileges to edit the file!

CA Store Password



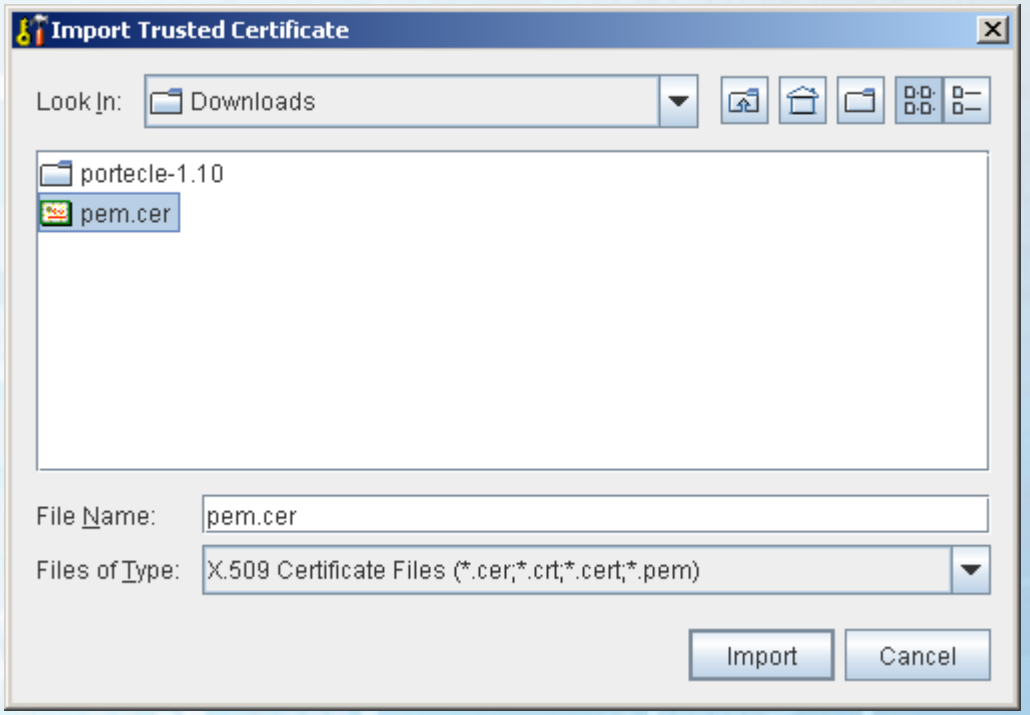
“changeit”

Viewing Trusted CAs



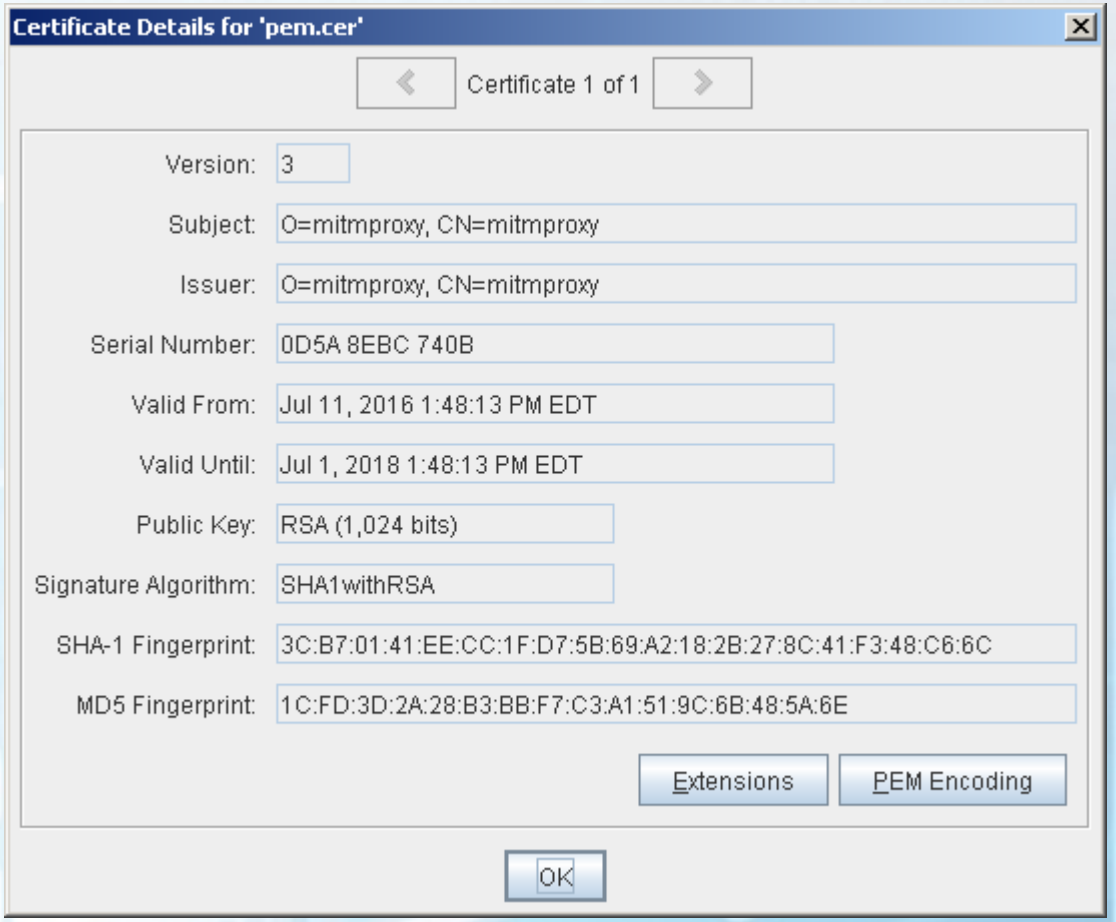
Tools → Import Trusted Certificate

Select Mitmproxy CA PEM File



Accept Trust Warning

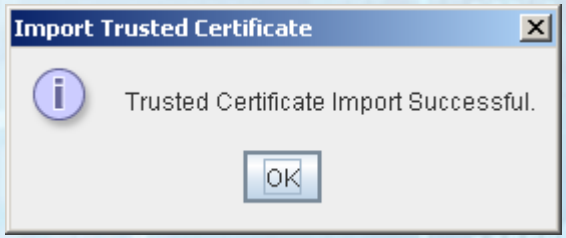
Accept Trust Warning



Accept Trust Warning

Name CA Certificate

Import Certificate



Retry jCurl HTTPS Download

```
C:\>java -jar jcurl-all.jar -e url https://www.google.com/
Starting jCurl in C:\

Sending 'GET' request to URL : https://www.google.com/
Response Code : 200
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"
lang="en">
…
```

Advanced Techniques

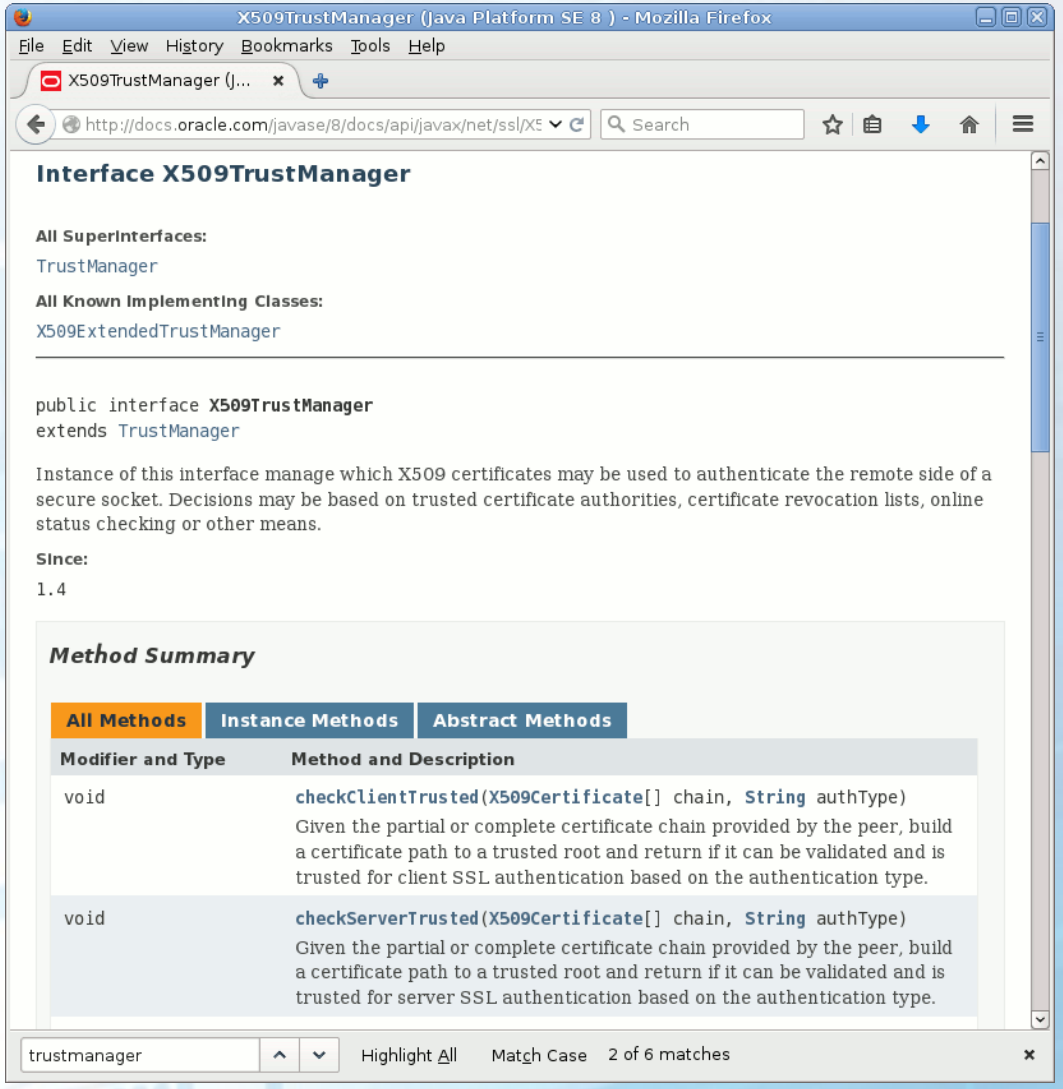
Certificate Pinning

Certificate Pinning

- Or “certificate authority pinning”
- Overcoming certificate pinning
  - Debugging
  - Patching

Java: Custom TrustManager

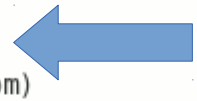
Java: Custom TrustManager



SSLContext.init

init

```
public final void init(KeyManager[] km,
                       TrustManager[] tm,
                       SecureRandom random)
                throws KeyManagementException
```



Initializes this context. Either of the first two parameters may be null in which case the installed security providers will be searched for the highest priority implementation of the appropriate factory. Likewise, the secure random parameter may be null in which case the default implementation will be used.

Only the first instance of a particular key and/or trust manager implementation type in the array is used. (For example, only the first javax.net.ssl.X509KeyManager in the array will be used.)

Parameters:

`km` - the sources of authentication keys or null

`tm` - the sources of peer authentication trust decisions or null

`random` - the source of randomness for this generator or null

Throws:

`KeyManagementException` - if this operation fails

null TrustManager = default

Suppressing Custom TrustManager

```
$ jdb CustomCA
Initializing jdb ...
>
```

Suppressing Custom TrustManager

```
$ jdb CustomCA
Initializing jdb ...
> stop in javax.net.ssl.SSLContext.getInstance(java.lang.String)
Deferring breakpoint
javax.net.ssl.SSLContext.getInstance(java.lang.String).
It will be set after the class is loaded.
```

Suppressing Custom TrustManager

```
$ jdb CustomCA
Initializing jdb ...
> stop in javax.net.ssl.SSLContext.getInstance(java.lang.String)
Deferring breakpoint
javax.net.ssl.SSLContext.getInstance(java.lang.String).
It will be set after the class is loaded.
> run
run CustomCA
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
>
```

Suppressing Custom TrustManager

```
$ jdb CustomCA
Initializing jdb ...
> stop in javax.net.ssl.SSLContext.getInstance(java.lang.String)
Deferring breakpoint
javax.net.ssl.SSLContext.getInstance(java.lang.String).
It will be set after the class is loaded.
> run
run CustomCA
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
>
VM Started: Set deferred breakpoint
javax.net.ssl.SSLContext.getInstance(java.lang.String)

Breakpoint hit: "thread=main",
javax.net.ssl.SSLContext.getInstance(), line=155 bci=0

main[1]
```

Suppressing Custom TrustManager

```
main[1] step up
>
Step completed: "thread=main", CustomCA.main(), line=29 bci=19
29          SSLContext sctx = SSLContext.getInstance("TLS");

main[1] locals
Method arguments:
args = instance of java.lang.String[0] (id=940)
Local variables:
tmf = instance of javax.net.ssl.TrustManagerFactory(id=941)
trustManagers = instance of javax.net.ssl.TrustManager[1]
(id=942)
main[1]
```

Suppressing Custom TrustManager

```
main[1] step up
>
Step completed: "thread=main", CustomCA.main(), line=29 bci=19
29          SSLContext sctx = SSLContext.getInstance("TLS");

main[1] locals
Method arguments:
args = instance of java.lang.String[0] (id=940)
Local variables:
tmf = instance of javax.net.ssl.TrustManagerFactory(id=941)
trustManagers = instance of javax.net.ssl.TrustManager[1]
(id=942)
main[1] set trustManagers=null
 trustManagers=null = null
main[1] cont
> <!doctype html><html itemscope=""
itemtype="http://schema.org/WebPage" lang="en">
```

C: OpenSSL

C: OpenSSL


```
long SSL_get_verify_result(const SSL *ssl);
```

Returns X509_V_OK (0) on successful validation
Non-zero otherwise

wget Call to OpenSSL

```
  430e84:       e8 a7 31 fd ff          callq  404030
<SSL_get_verify_result@plt>
  430e89:       48 85 c0                test   %rax,%rax
  430e8c:       48 89 c3                mov    %rax,%rbx
  430e8f:       c6 44 24 0f 01          movb   $0x1,0xf(%rsp)
  430e94:       0f 85 a6 02 00 00       jne    431140
<SSLv3_client_method@plt+0x2c270>
```

wget Call to OpenSSL

```
  430e84:       90                      nop
  430e85:       90                      nop
  430e86:       90                      nop
  430e87:       31 c0                   xor    %eax,%eax
  430e89:       48 85 c0                test   %rax,%rax
  430e8c:       48 89 c3                mov    %rax,%rbx
  430e8f:       c6 44 24 0f 01          movb   $0x1,0xf(%rsp)
  430e94:       0f 85 a6 02 00 00       jne    431140
<SSLv3_client_method@plt+0x2c270>
```

Certificate Magic Numbers

Certificate Magic Numbers

- PEM format
  - -—BEGIN CERTIFICATE—-
  - -—END CERTIFICATE—-
- DER format
  - 0x30 0x82 0x01 (1024-bit RSA certificate)
  - 0x30 0x82 0x02 (2048-bit RSA certificate)

Mistakes in Implementing HTTPS

Mistakes in Implementing HTTPS

- Failing to Verify Certificates
  - No verification at all
  - CA, but not hostname
    - OpenSSL limitations
  - Hostname, but not CA

Mistakes in Implementing HTTPS

- Allowing anonymous cipher suites
  - Confusing OpenSSL APIs

Mistakes in Implementing HTTPS

- Allowing NULL encryption cipher suites

Mistakes in Implementing HTTPS

- Outdated Libraries
  - Apple “goto fail”
  - ChangeCipherSpec Injection
  - DROWN
  - Logjam

Conclusion

Bypassing TLS

- Modify trusted CA list
  - Windows
  - Android
  - iOS
  - Java
- Overcome certificate pinning
  - Debugging (e.g., Java)
  - Patching (e.g., C + OpenSSL, certificate magic numbers)
- Leverage implementation mistakes

Bypassing TLS

- Questions?