INTERCEPTING MOBILE APP NETWORK TRAFFIC

# (AKA "THE SQUIRREL IN THE MIDDLE")

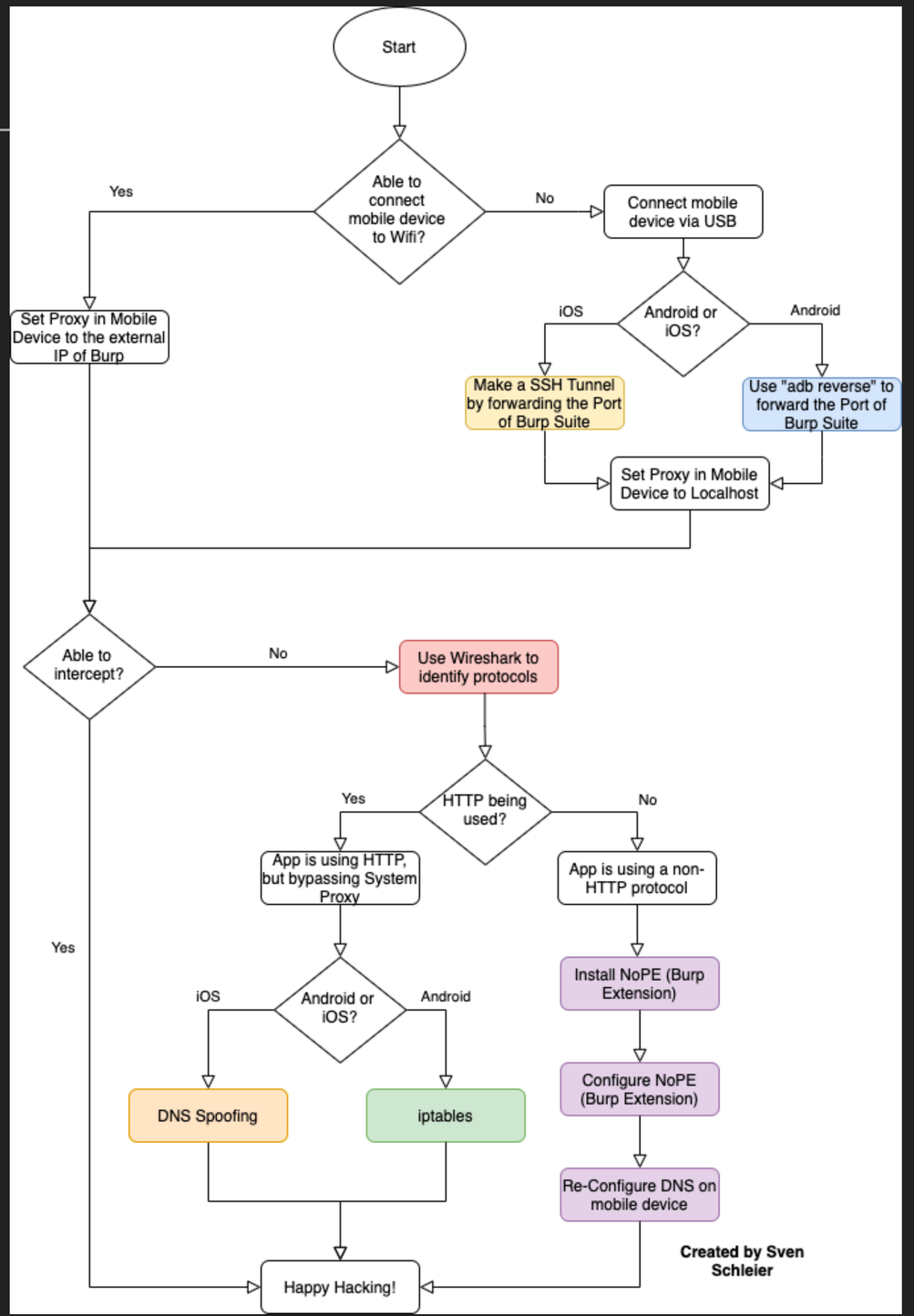
SVEN SCHLEIER – BSIDES SINGAPORE – SEPTEMBER 2021

# $ /USR/BIN/WHOAMI

Hi everyone, my name is Sven!

- Previous roles: Unix Admin, Penetration Tester, Security Architect for Web and Mobile Apps during SDLC
- Technical Director at F-Secure in ☀️ Singapore
- Project leader together with Carlos Holguera of:
  - OWASP Mobile Security Testing Guide (MSTG) and
  - OWASP Mobile AppSec Verification Standard (MASVS)
- Blogging on http://bsddaemonorg.wordpress.com/

# THE ULTIMATE DECISION TREE FOR MOBILE APP NETWORK TESTING




https://bsddaemonorg.wordpress.com/2021/02/11/the-ultimate-decision-tree-for-mobile-app-network-testing-aka-the-squirrel-in-the-middle/

# USUALLY YOU WILL FIND LOT OF INFORMATION AROUND ANDROID, SO I WILL BE FOCUSING MAINLY ON IOS TODAY!

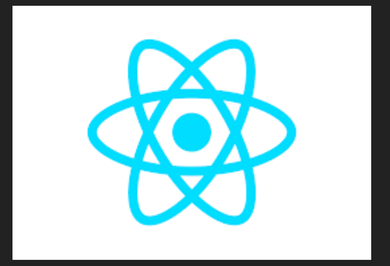
# INTERCEPTING NETWORK COMMUNICATION – USUAL TEST SETUP

The flow for testing an iOS app is similar to web apps:

1. Configuring an iOS Device to Work With Burp:
   - https://portswigger.net/support/configuring-an-ios-device-to-work-with-burp
2. Installing Burp's CA Certificate in an iOS Device
   - https://portswigger.net/support/installing-burp-suites-ca-certificate-in-an-ios-device
3. Happy Hacking!

# FRAMEWORKS THAT ALSO PRODUCE “NATIVE” BINARIES:

| Framework | Programming Language |
| --- | --- |
| Flutter (Google) | Dart |
| React Native (Facebook) | JavaScript |
| Xamarin (Microsoft) | C# |



# Which of the following mobile app frameworks are not using the system proxy of iOS?

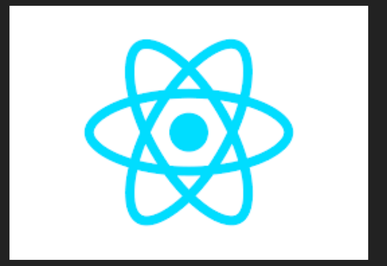
Flutter (Google)

Xamarin (Microsoft)

React Native (Facebook)

# FRAMEWORKS THAT ALSO PRODUCE “NATIVE” BINARIES:

| Framework | Programming Language | Bypasses System Proxy? |
|---|---|---|
| Flutter (Google) | Dart | Yes |
| React Native (Facebook) | Java Script | No |
| Xamarin (Microsoft) | C# | Yes |



# DIFFERENT SCENARIOS WHEN YOU ARE TESTING THE NETWORK COMMUNICATION

The default setup to intercept mobile apps might not work all the time and there are several scenarios you should be able to tackle when testing mobile apps:

You are not able / not allowed to connect your iOS device to the WiFi

System Proxy is ignored (Google Flutter or Xamarin)

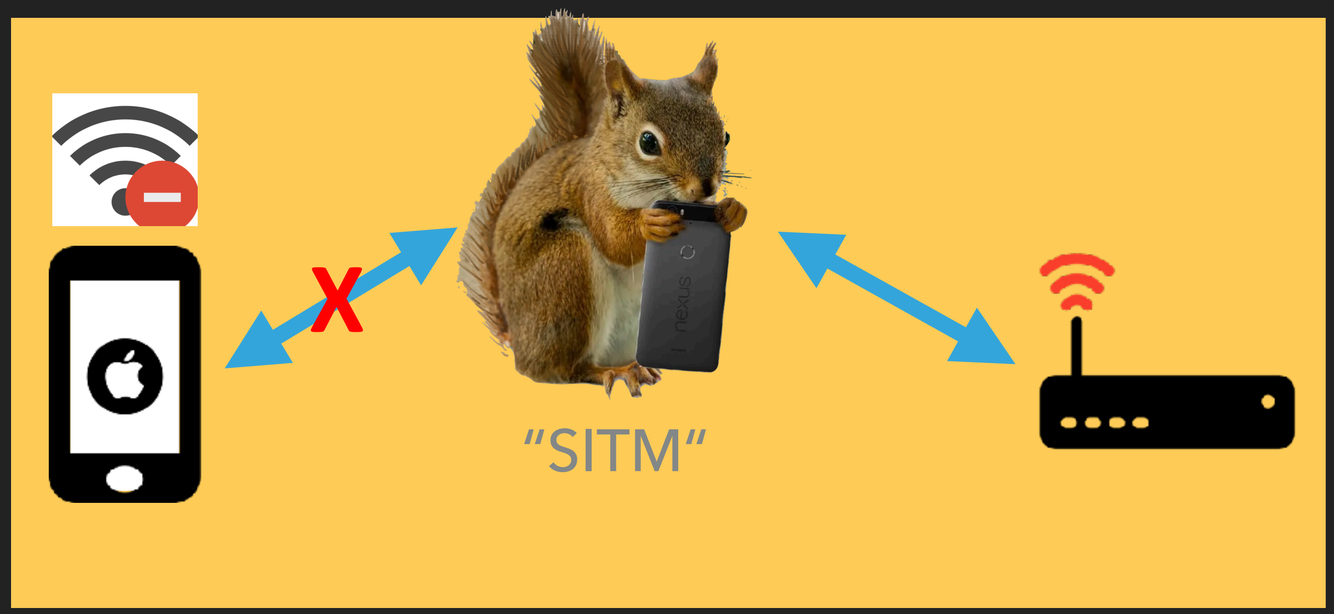

XMPP or other non-HTTP protocols are used

~~HTTP~~

Let’s go through it one by one!

# YOU ARE NOT ABLE / NOT ALLOWED TO CONNECT YOUR DEVICE TO THE WIFI NETWORK

# YOU ARE NOT ABLE / NOT ALLOWED TO CONNECT YOUR IOS DEVICE TO THE WIFI

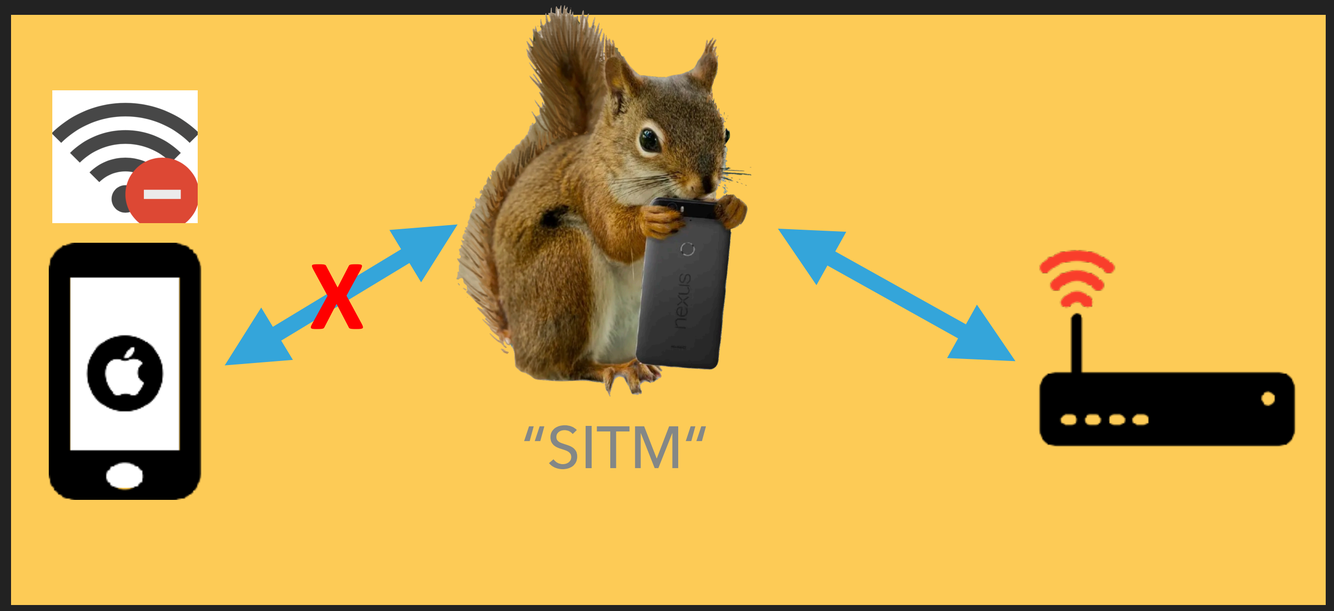



During a penetration test there might be limiting factors:

- You are not allowed to use your jailbroken device in the client's network
- There is "client isolation" activated in the Wireless network and the iOS device and your laptop are not able to communicate.
- The client Wifi is "full" and you cannot connect any more devices (yes, seriously!)

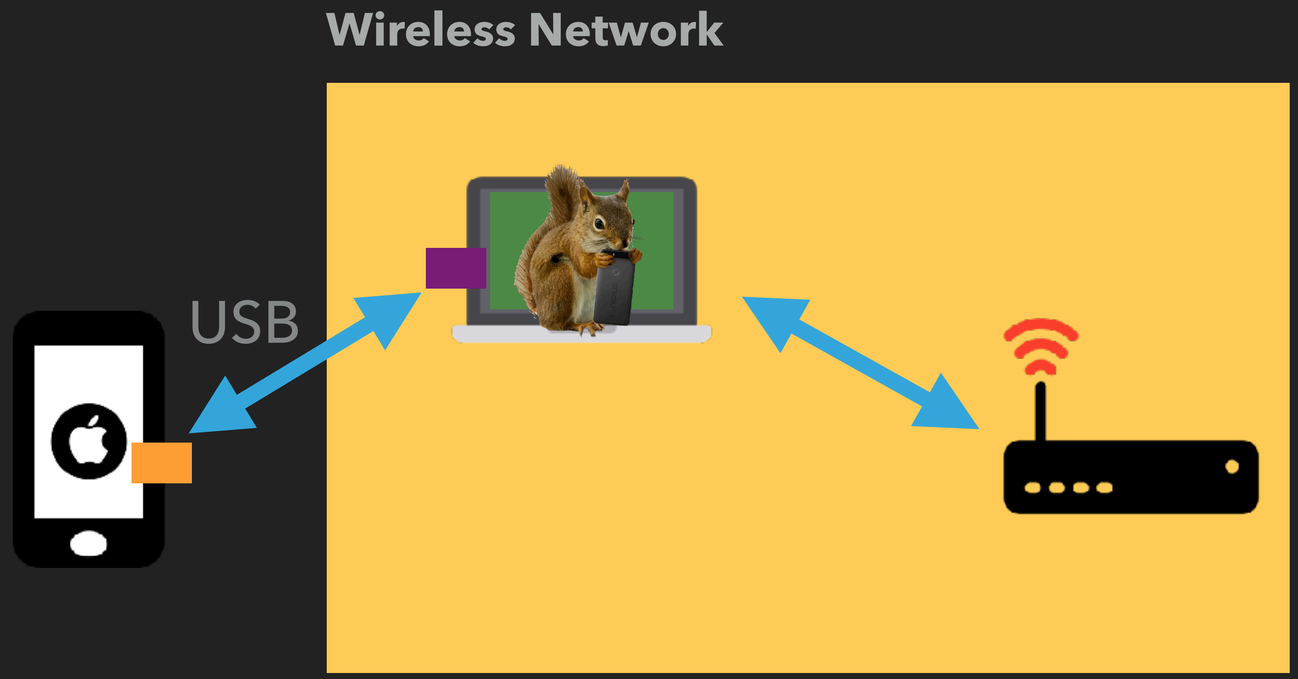
# WIFI WITH CLIENT ISOLATION

**Wireless Network**




- What to do if we cannot intercept the traffic of the iOS device in the WiFi network?

# TEST VIA SSH TUNNEL




- Connect your iOS device via USB to your laptop
- Execute iproxy and SSH:

```
iproxy (iproxy)
Last login: Sat Sep  5 13:28:10 on ttys007
~ ❯ iproxy 2222 22
waiting for connection
accepted connection, fd = 4
waiting for connection
Number of available devices == 1
Requesting connecion to device handle == 1671 (serial: c82
e058d175c5a86634b1f59c), port 22
run_ctos_loop: fd = 4
run_stoc_loop: fd = 4
```

```
8080:localhost:8080 (ssh)
Last login: Sat Sep  5 13:28:47 on ttys005
~ ❯ ssh -R 8080:localhost:8080 root@localhost -p 2222
root@localhost's password:
Svens-iPhone:~ root#
```

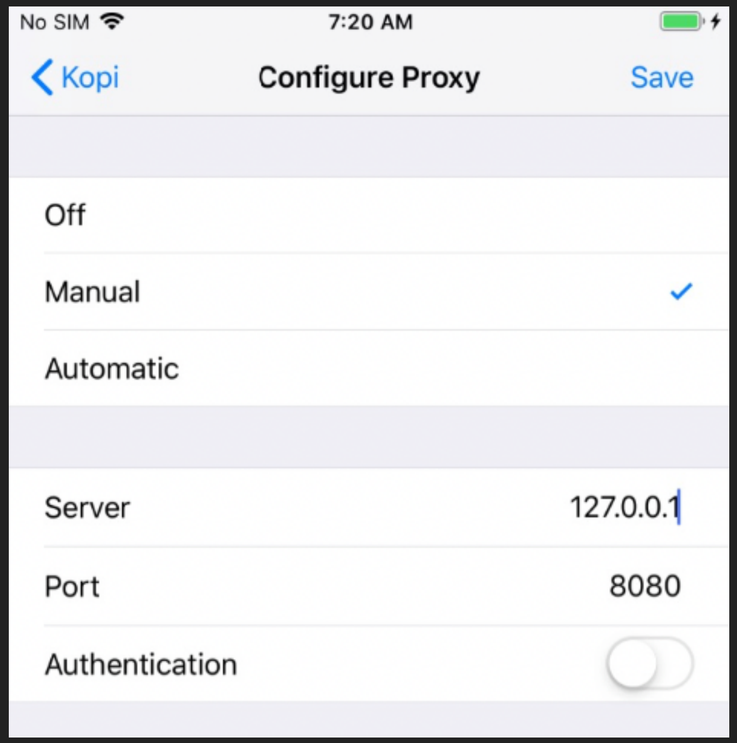
Remote port forwarding of port 8080 (Burp) to the iOS device

```
$ ssh -R 8080:localhost:8080 root@localhost -p 2222
```

# TEST VIA SSH TUNNEL

Re-configure the Proxy settings:

1. Open the “Settings” App
2. Click on Wi-Fi
3. Connect to any Wi-Fi Network
4. Click on ”Configure Proxy” and select “Manual”
5. Key in the IP address 127.0.0.1 and the port we just forwarded (port 8080)
6. Open Safari and browse to any page.

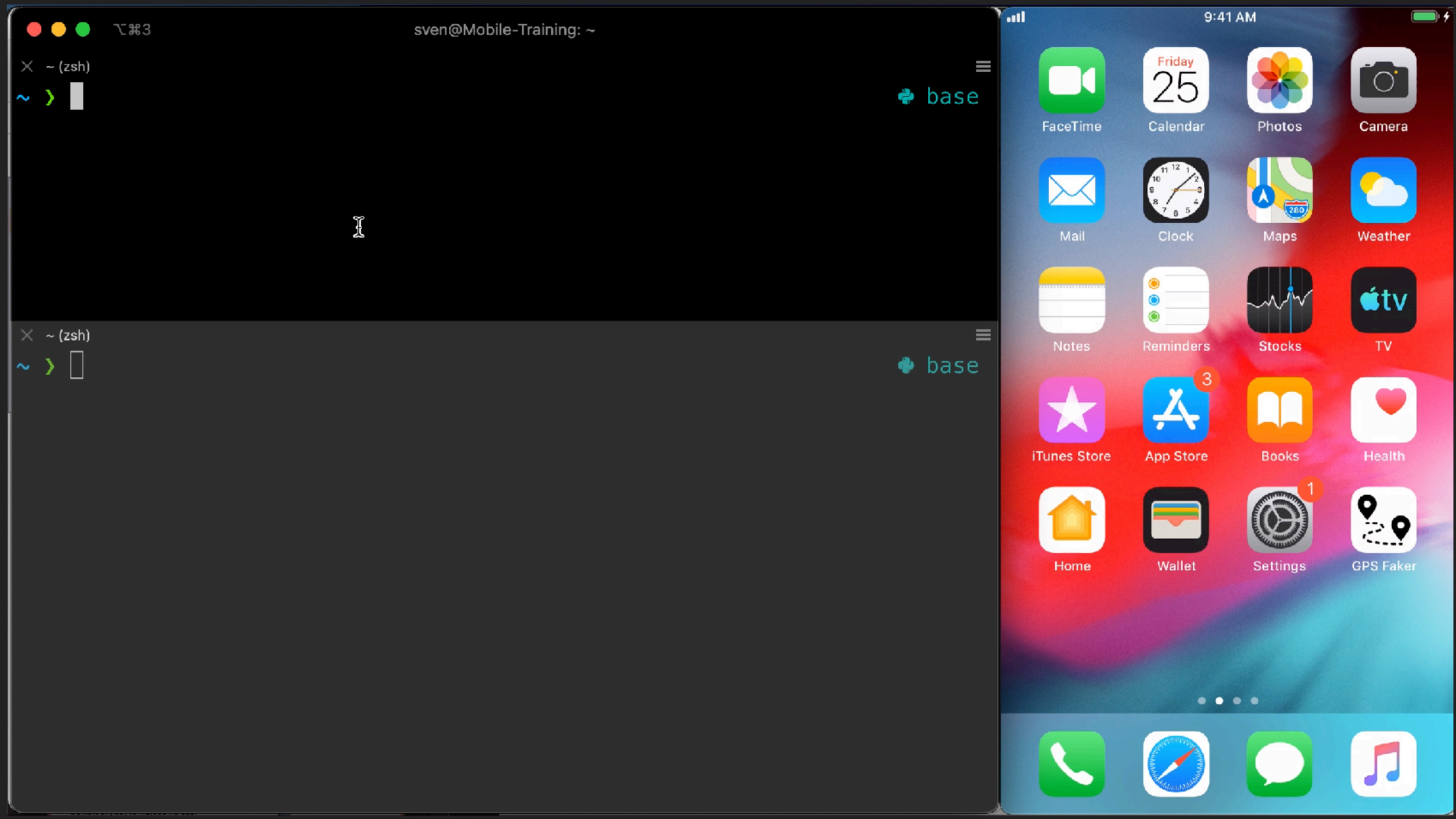



- **All your HTTP(S) traffic is now going via localhost through the SSH tunnel and the port forwarding to your Burp (and all via USB)!**
- **Your iOS device can also now literally be connected to any Wifi, as you just need to set the proxy and all traffic is anyway forwarded via USB.**

# TEST VIA SSH TUNNEL

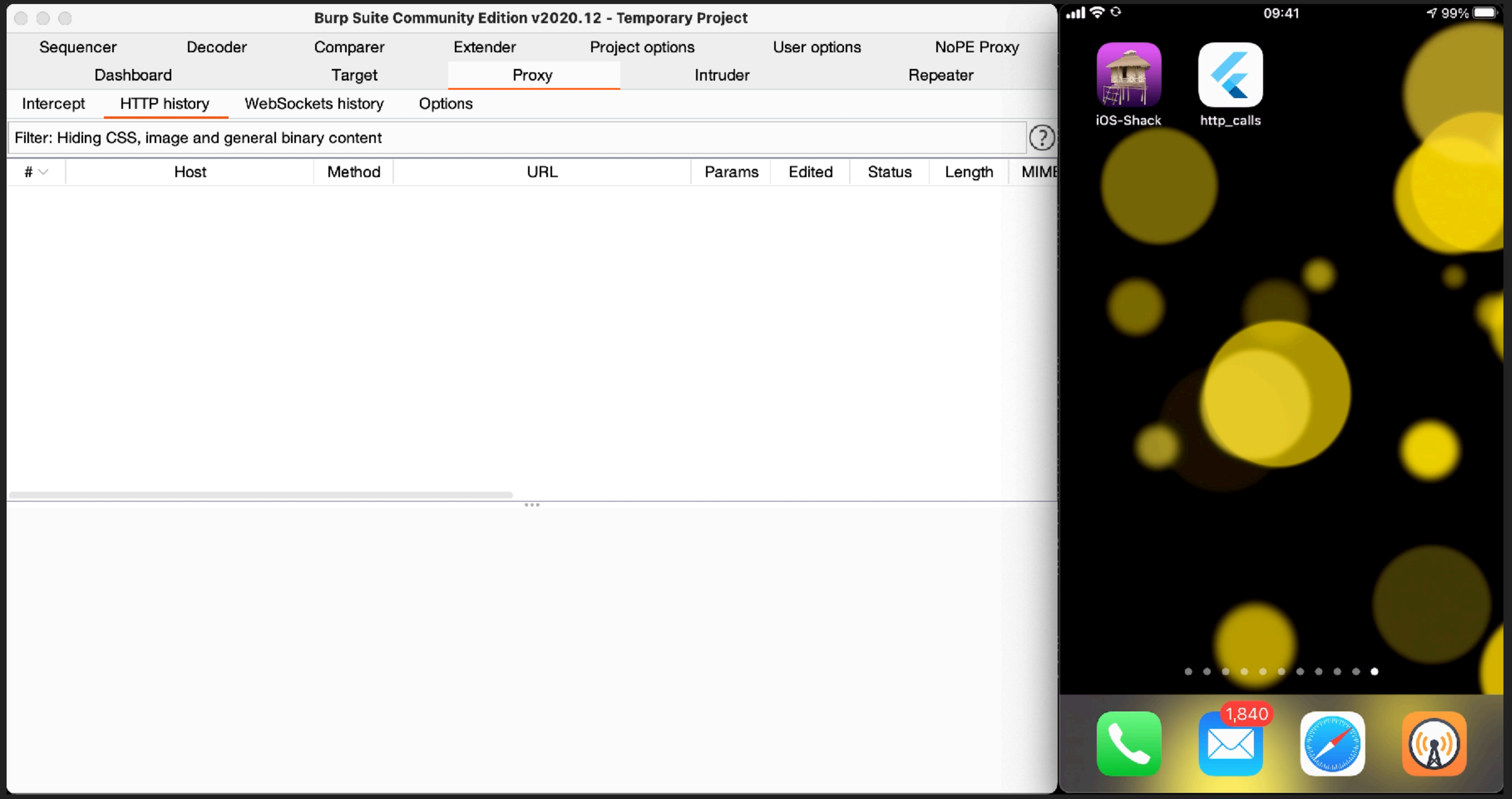
You will now be able to reach Burp on your iOS device. Open Safari on iOS and go to http://127.0.0.1:8080 and you will see the Burp Suite Landing Page.

# DEMO


⌥⌘3
sven@Mobile-Training: ~
~ (zsh)
base
~ (zsh)
base
9:41 AM
FaceTime
Calendar
Friday
25
Photos
Camera
Mail
Clock
Maps
Weather
Notes
Reminders
Stocks
TV
iTunes Store
App Store
3
Books
Health
Home
Wallet
Settings
1
GPS Faker


SYSTEM PROXY IS IGNORED (GOOGLE FLUTTER OR XAMARIN)
Flutter

# THE SQUIRREL IN THE MIDDLE



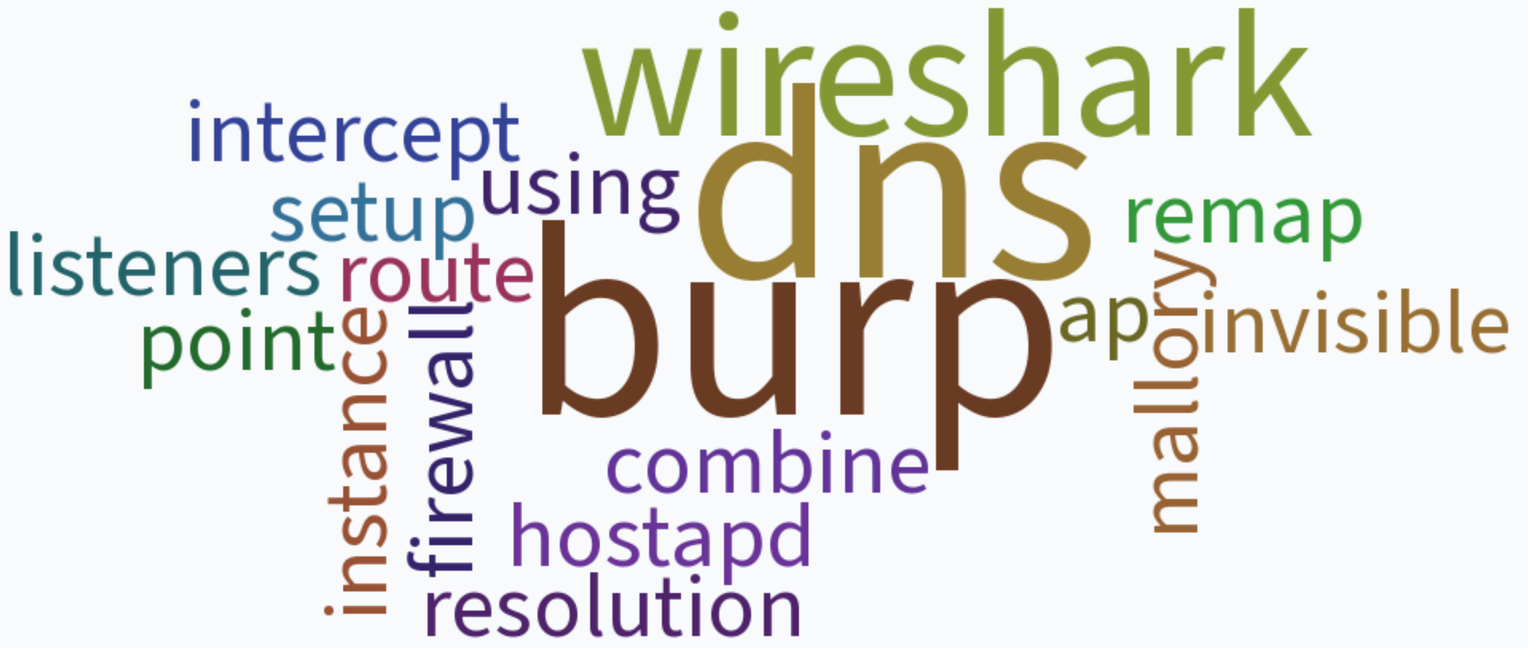
# SYSTEM PROXY IS IGNORED (GOOGLE FLUTTER OR XAMARIN)

- Mobile App Frameworks such as Google's Flutter or Microsoft's Xamarin are ignoring the System Proxy!
- If you set the proxy in the WiFi settings it will have no effect


What to do…?

See also: https://blog.nviso.eu/2019/08/13/intercepting-traffic-from-android-flutter-applications/

# How would you try to intercept the traffic in this scenario?



# SYSTEM PROXY IS IGNORED (GOOGLE FLUTTER OR XAMARIN)

Possible solutions:

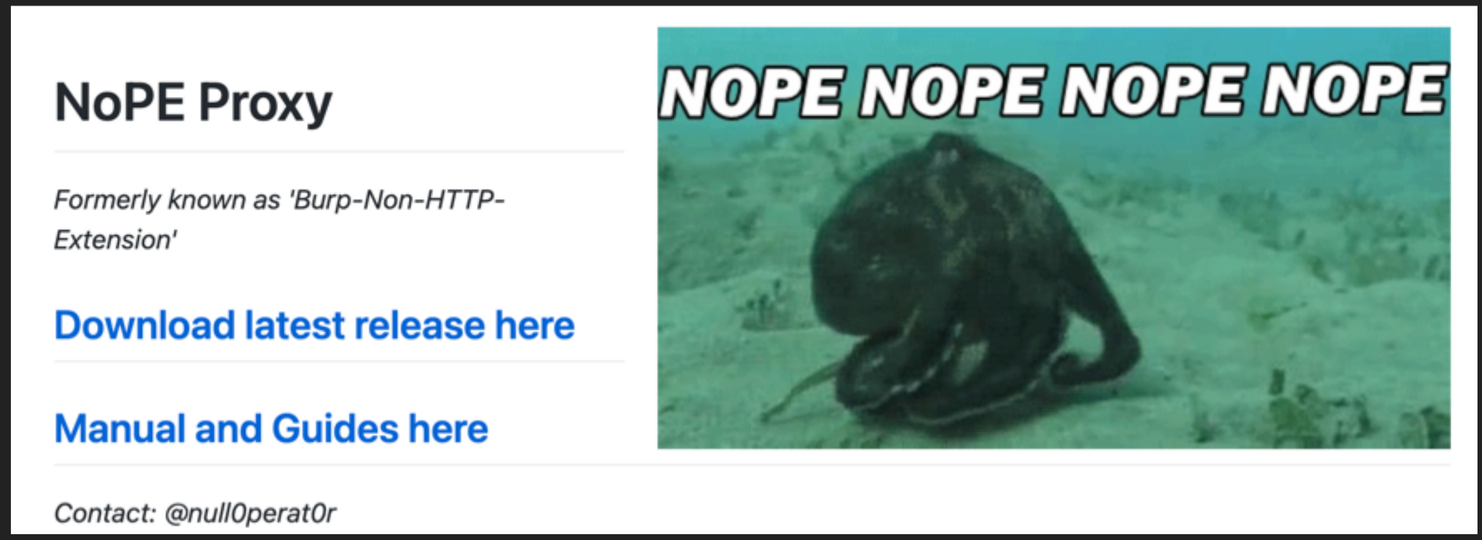
- ARP Poisoning (“The classic”) - We can attack on layer 2, by using (b)ettercap:
  - https://mobile-security.gitbook.io/mobile-security-testing-guide/general-mobile-app-testing-guide/0x04f-testing-network-communication#simulating-a-man-in-the-middle-attack
- Or DNS Spoofing by using NoPE Proxy
  - https://bsddaemonorg.wordpress.com/2021/02/03/intercepting-non-http-network-traffic-of-mobile-apps

**Both works on non-jailbroken and jailbroken devices!**

# NOPE - BURP PLUGIN

This burp extension adds two new features to BurpSuite.

- A configurable DNS server.
- A Non-HTTP MiTM Intercepting proxy.

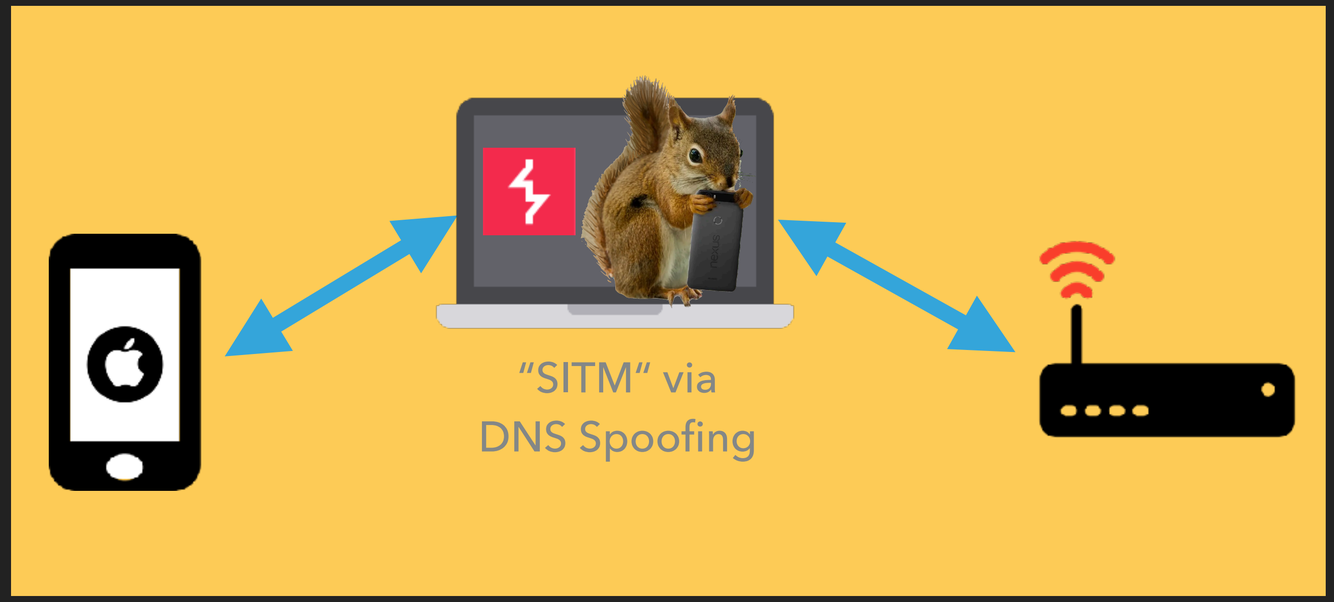



https://github.com/summitt/Burp-Non-HTTP-Extension

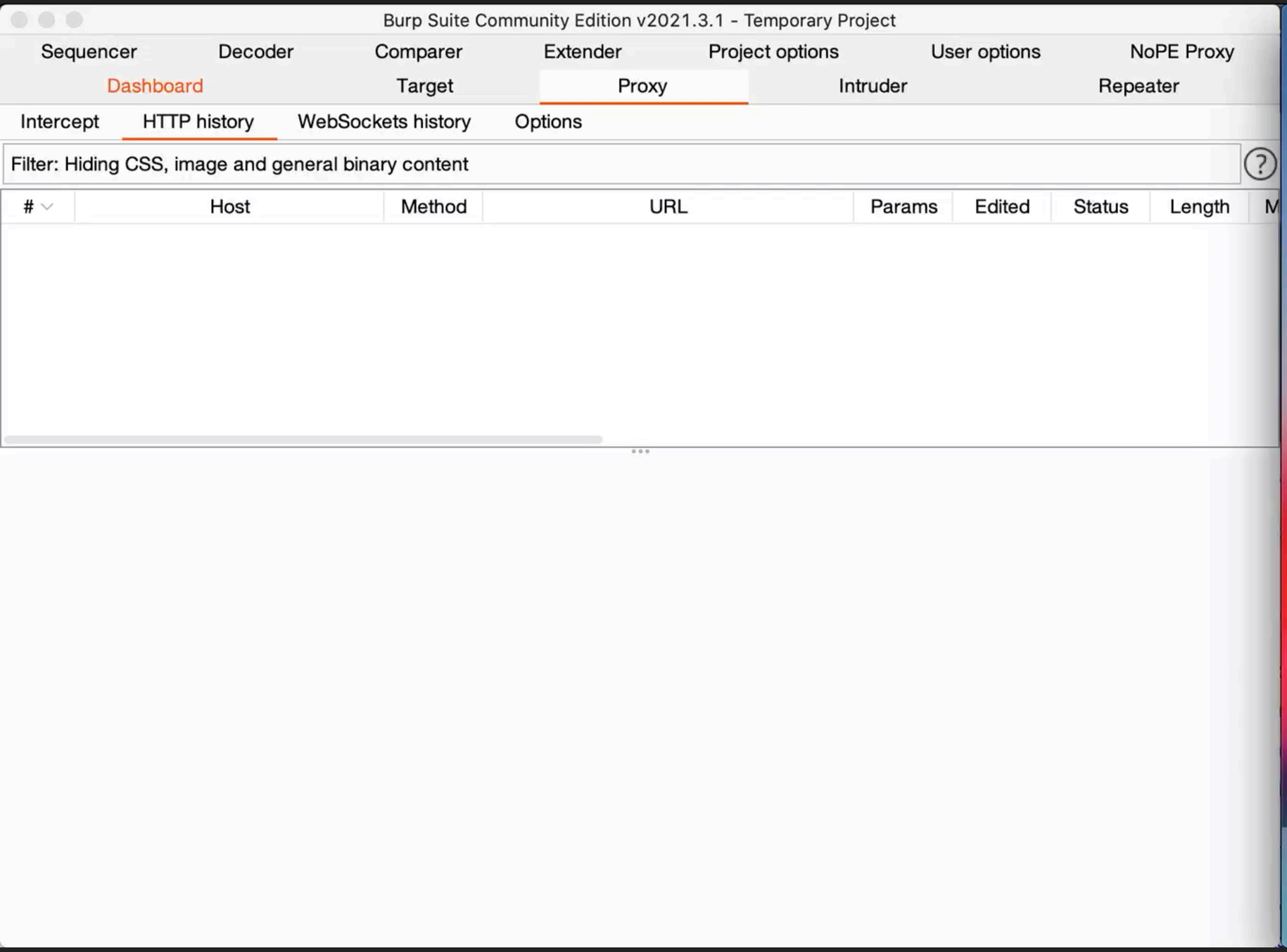
https://github.com/summitt/Burp-Non-HTTP-Extension/wiki#basic-set-up-for-mobile-testing-testing-on-two-machines

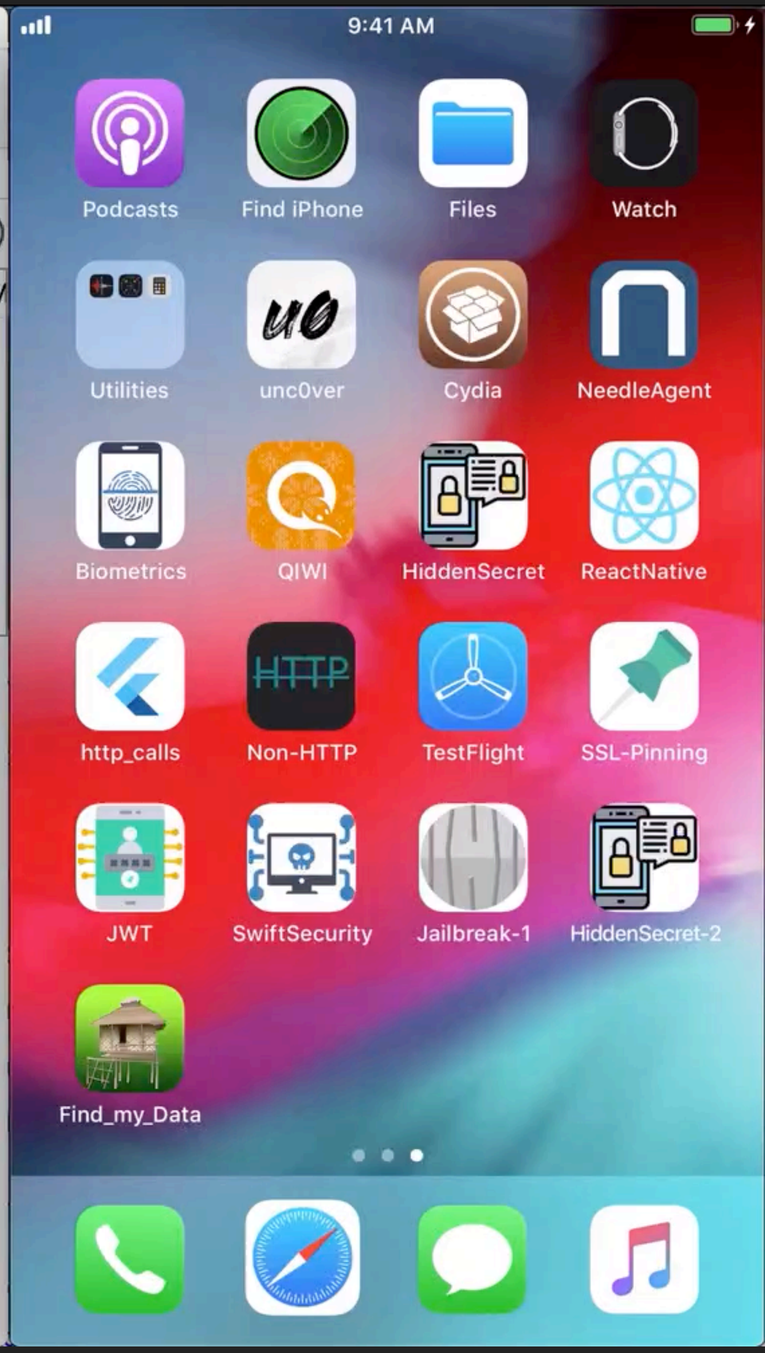
# SYSTEM PROXY IS IGNORED (GOOGLE FLUTTER OR XAMARIN)

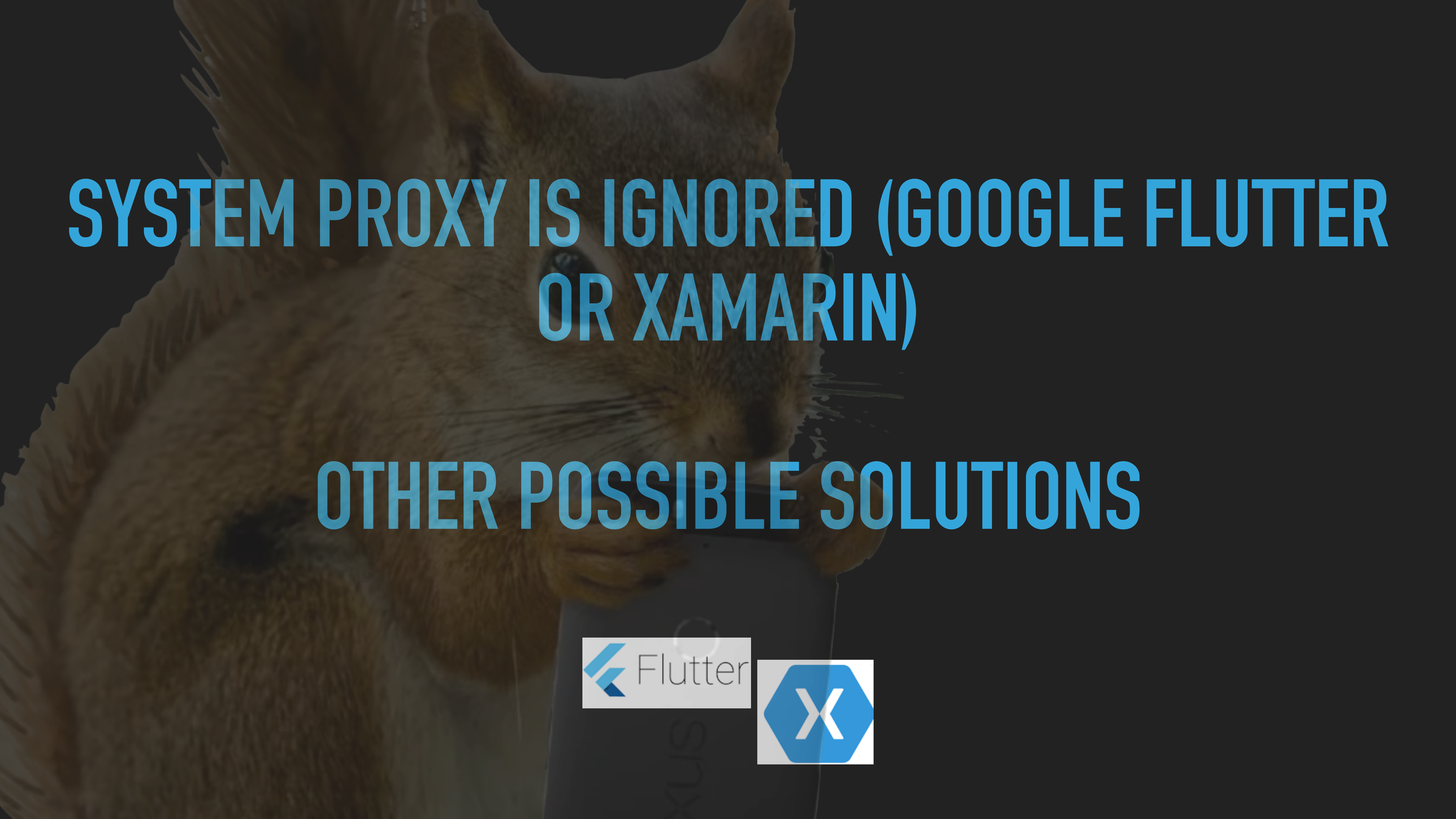
**Wireless Network**




- Start DNS Server with NoPE Extension in Burp Suite
- Configure DNS Server in mobile device
- Be the "Squirrel in the Middle" and intercept all DNS based traffic


Burp Suite Community Edition v2021.3.1 - Temporary Project
Sequencer
Decoder
Comparer
Extender
Project options
User options
NoPE Proxy
Dashboard
Target
Proxy
Intruder
Repeater
Intercept
HTTP history
WebSockets history
Options
Filter: Hiding CSS, image and general binary content
#
Host
Method
URL
Params
Edited
Status
Length


9:41 AM
Podcasts
Find iPhone
Files
Watch
Utilities
unc0ver
Cydia
NeedleAgent
Biometrics
QIWI
HiddenSecret
ReactNative
http_calls
Non-HTTP
TestFlight
SSL-Pinning
JWT
SwiftSecurity
Jailbreak-1
HiddenSecret-2
Find_my_Data


SYSTEM PROXY IS IGNORED (GOOGLE FLUTTER OR XAMARIN)
OTHER POSSIBLE SOLUTIONS
Flutter

# CONFIGURE /ETC/HOSTS

- You can configure the **/etc/hosts** on your jailbroken iOS device
- Add an entry into **/etc/hosts** for the target domain and point it to the IP address of your interception proxy.
- This creates a SITM attack on domain basis. In case you only need to attack a few domains, this is a simple way of redirecting traffic without configuring any daemons on your Mac.
- The Burp listener need to run on port 80 / 443 in invisible proxy mode (https://portswigger.net/burp/documentation/desktop/tools/proxy/options/invisible).

```
Svens-iPhone:~ root# cat /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
192.168.1.66    example.com
192.168.1.66    www.example.com
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
```

# SETUP AN ACCESS POINT

You can setup an Access Point the iOS device is connecting to that will redirect all traffic to your Burp:

https://mobile-security.gitbook.io/mobile-security-testing-guide/general-mobile-app-testing-guide/0x04f-testing-network-communication#simulating-a-man-in-the-middle-attack-with-an-access-point

# SUMMARY

| Technique | Jailbroken device needed? | What can be intercepted? |
|---|---|---|
| **ARP Poisoning** | No | All Traffic from iOS device |
| **DNS Spoofing** | No | All traffic based on DNS |
| **/etc/hosts** | Yes | Domains listed in hosts file |
| **Access Point** | No | All Traffic from iOS device |

And there are still more ways to intercept traffic, e.g. setting up a VPN server where the mobile device is connecting to etc…

# INTERCEPTING NON-HTTP TRAFFIC

~~HTTP~~

# NON-HTTP AND NON-STANDARD PORTS?

What if we face the following:

- XMPP or other non-HTTP protocols are being used
- Different port used than 80 (HTTP) or 443 (HTTPS)


Interception proxies such as Burp and OWASP ZAP won't show non-HTTP traffic, because they aren't capable of decoding it by default.

Examples:

- Build-in chat functions might rely on XMPP
- Some apps are using raw TCP to reduce overhead that HTTP Headers produce

In this case you need to monitor and analyze the network traffic first in order to decide what to do next.

# RVICTL + WIRESHARK (IOS)

iOS doesn't let you record a packet trace directly. However, you can use your Mac to record a packet trace on an attached iOS device using the Remote Virtual Interface (RVI) mechanism.

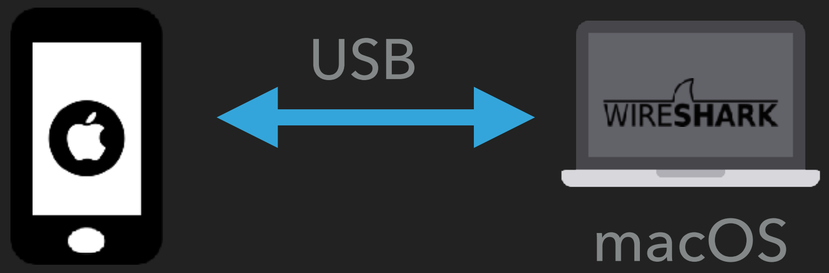
- Connect your iOS device via USB to your Mac
- Find the UDID of your iOS device

```
~ ❯ xcrun xctrace list devices                                    15s
== Devices ==
Mobile Training (C1F5F36[redacted]C8A)
Sven's iPhone (12.4) (c82b854417[redacted]f59c)
```

- Execute the command rvictl:

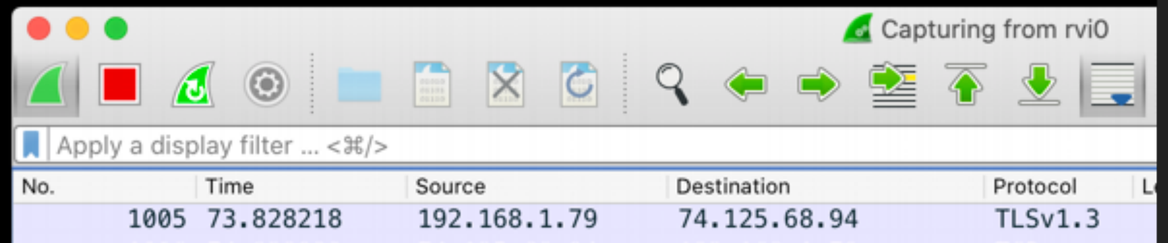
```
~ ❯ rvictl -s c82b8544171[redacted]

Starting device c82b8544171[redacted] [SUCCEEDED] with interface rvi0
```




- Start Wireshark and capture packets from rvi0

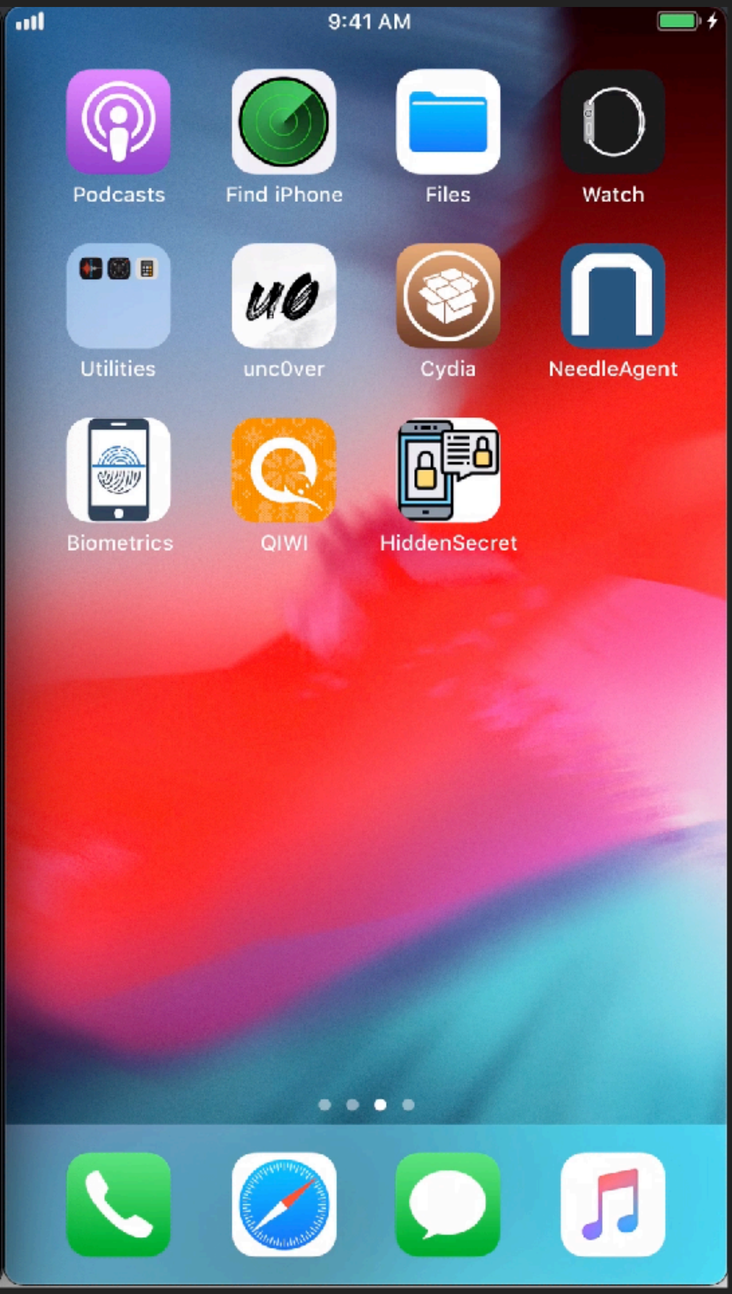
```
~ ❯ wireshark -i rvi0 -k
```



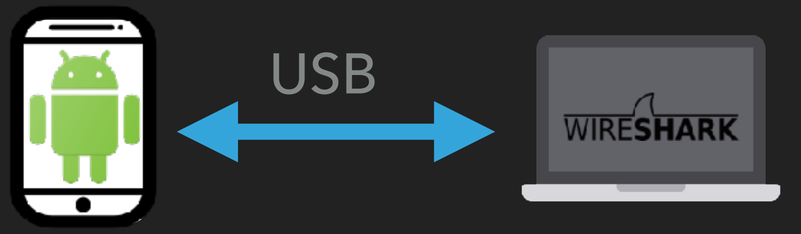

https://developer.apple.com/documentation/network/recording_a_packet_trace

# MONITOR TRAFFIC OF IOS

RVICTL + WIRESHARK

# TCPDUMP + WIRESHARK (ANDROID)



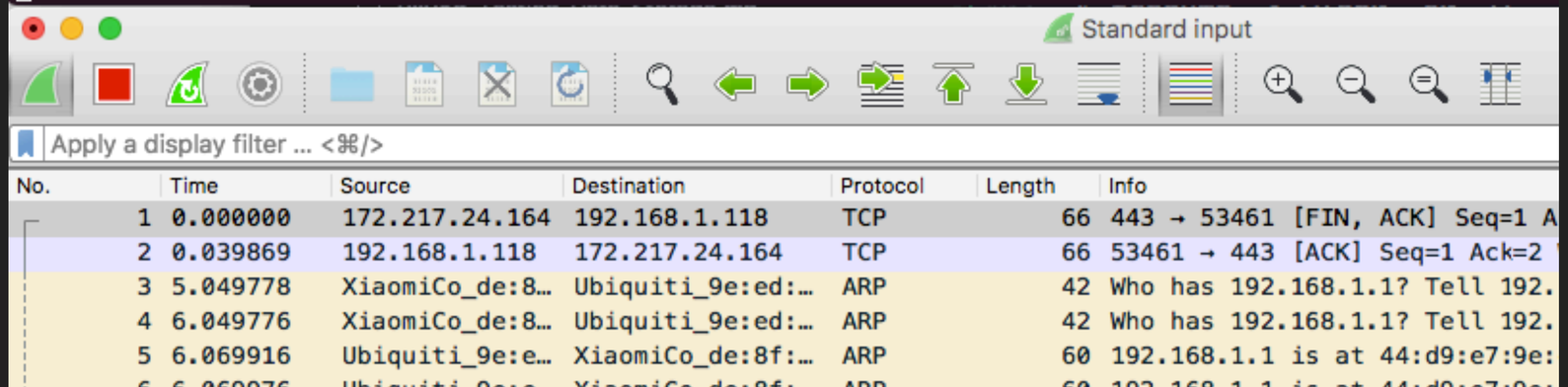

Remotely sniffing all Android traffic in real-time is possible with tcpdump, netcat (nc), and Wireshark.

To remotely sniff the Android phone's network traffic, first execute tcpdump and pipe its output to netcat (nc):

```
$ tcpdump -i wlan0 -s0 -w - | nc -l -p 11111
```

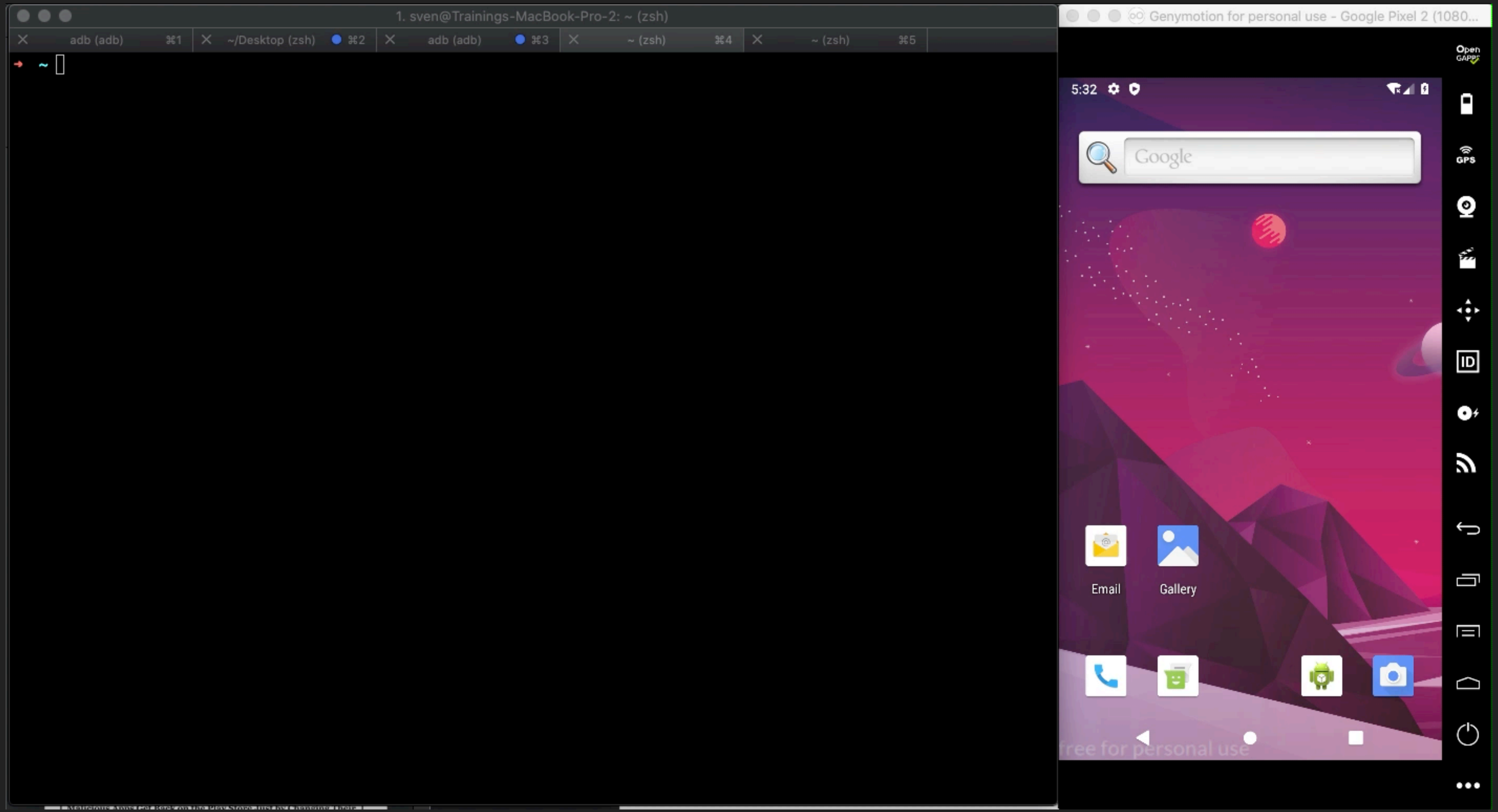
To access port 11111, you need to forward the port to your host computer via adb and the following command connects you to the forwarded port via netcat and piping to Wireshark.

```
➜ bin adb forward tcp:11111 tcp:11111
➜ bin nc localhost 11111 | wireshark -k -S -i -
13:02:21  Capture Warn sync_pipe_wait_for_child: waitpid returned EINTR. retrying.
```




Detailed explanation: https://bit.ly/3Am8APv

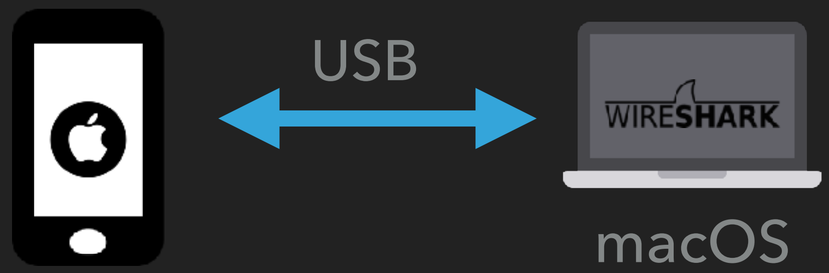
# MONITOR TRAFFIC OF ANDROID



# WIRESHARK – TIPS

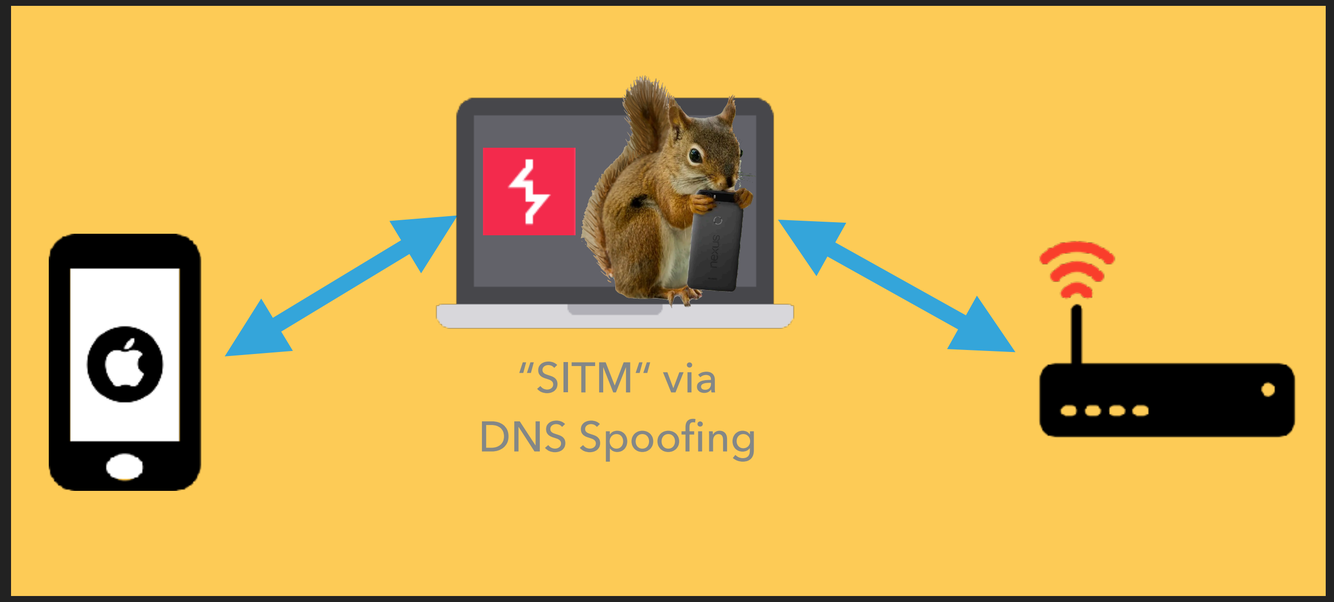
Using Wireshark to identify non-HTTP traffic:

- Close all apps in the background
- Identify IP address of the server the app is communicating with
- Filter traffic according to the IP (e.g. filter: ip.addr == 192.168.0.102)
- Follow TCP stream (http://bit.ly/3683BDj)
- Use the app and inspect the traffic in Wireshark:
  - Which protocols are used?
  - Is the traffic encrypted?



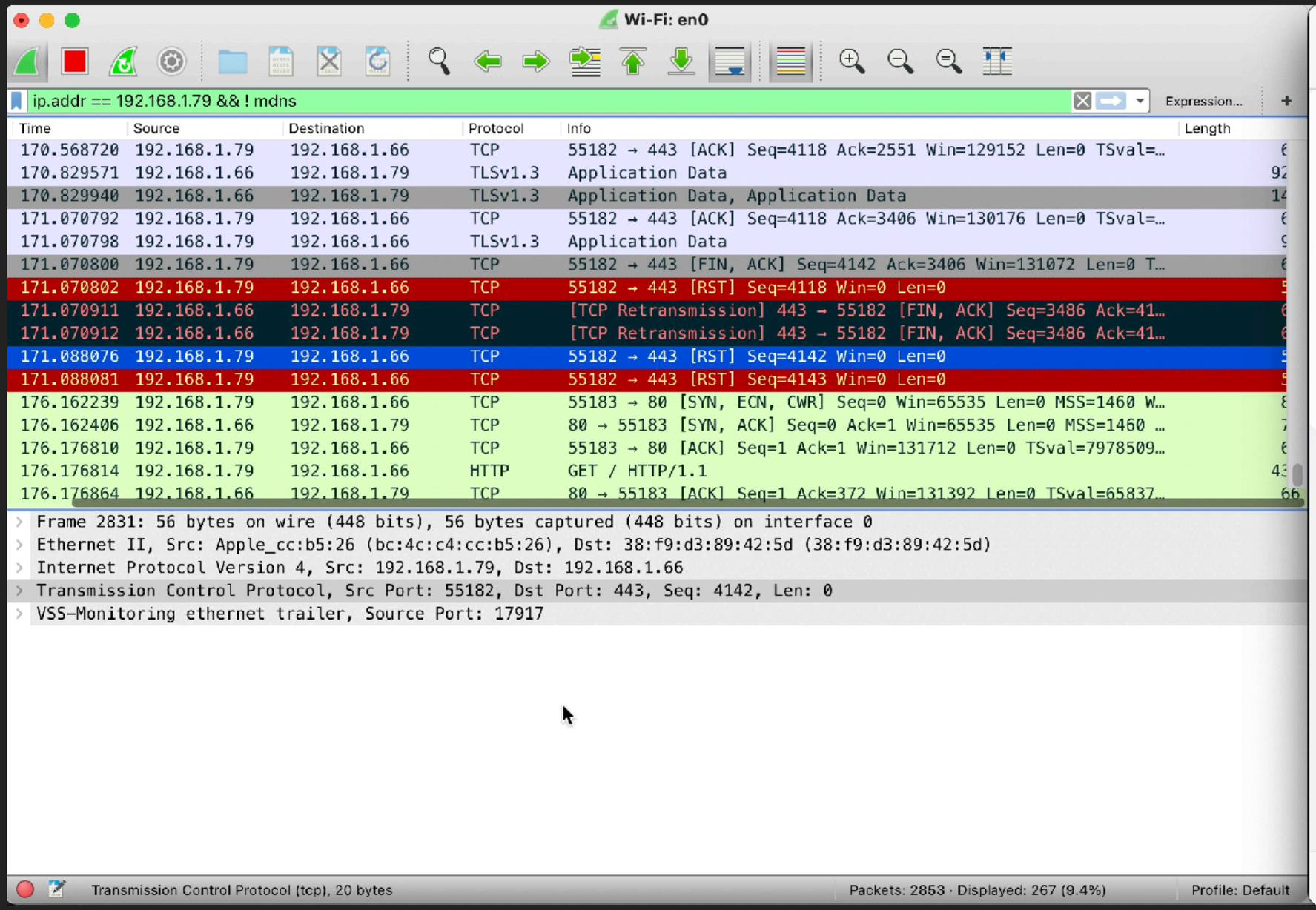
# INTERCEPTING NON-HTTP TRAFFIC

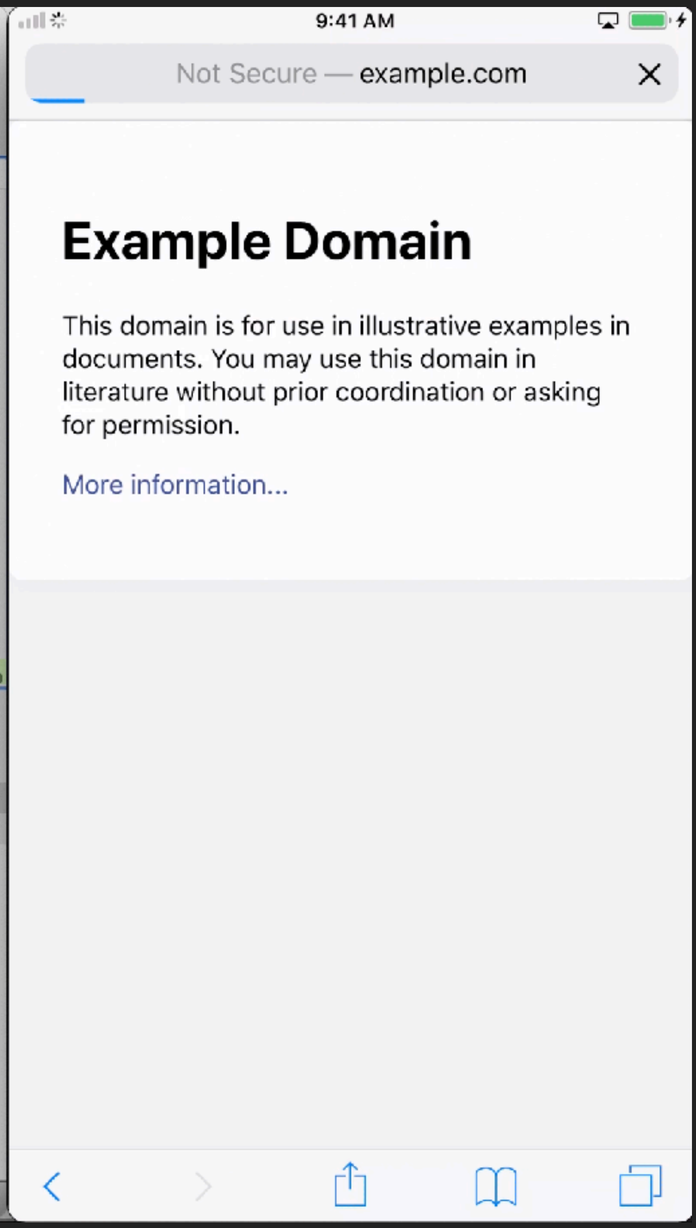
**Wireless Network**



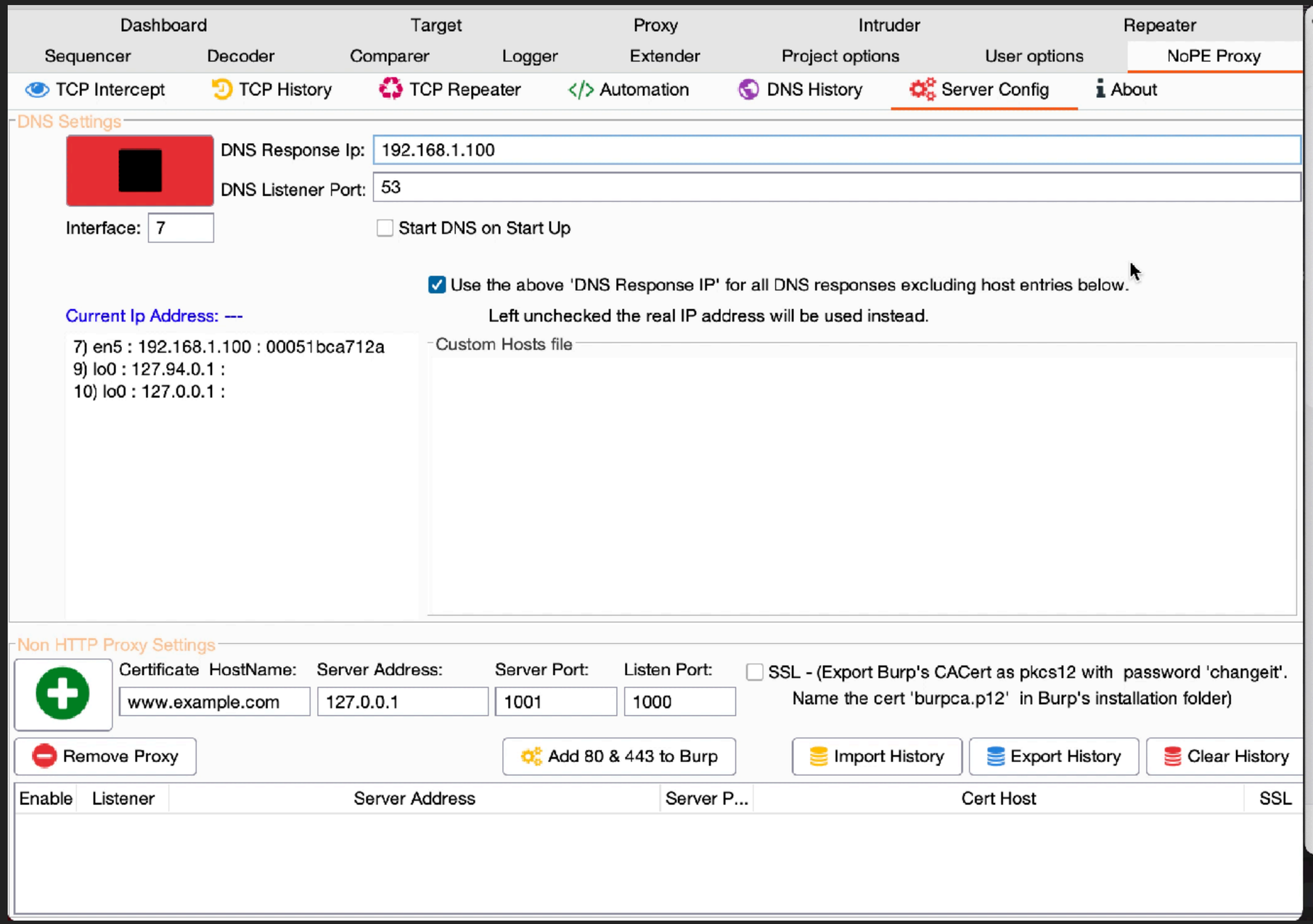

- Start DNS Server with NoPE Extension in Burp Suite
- Configure DNS Server in mobile device
- Add a listener for the port you want to monitor in NoPE Extension
- Be the "Squirrel in the Middle" and intercept all DNS based traffic

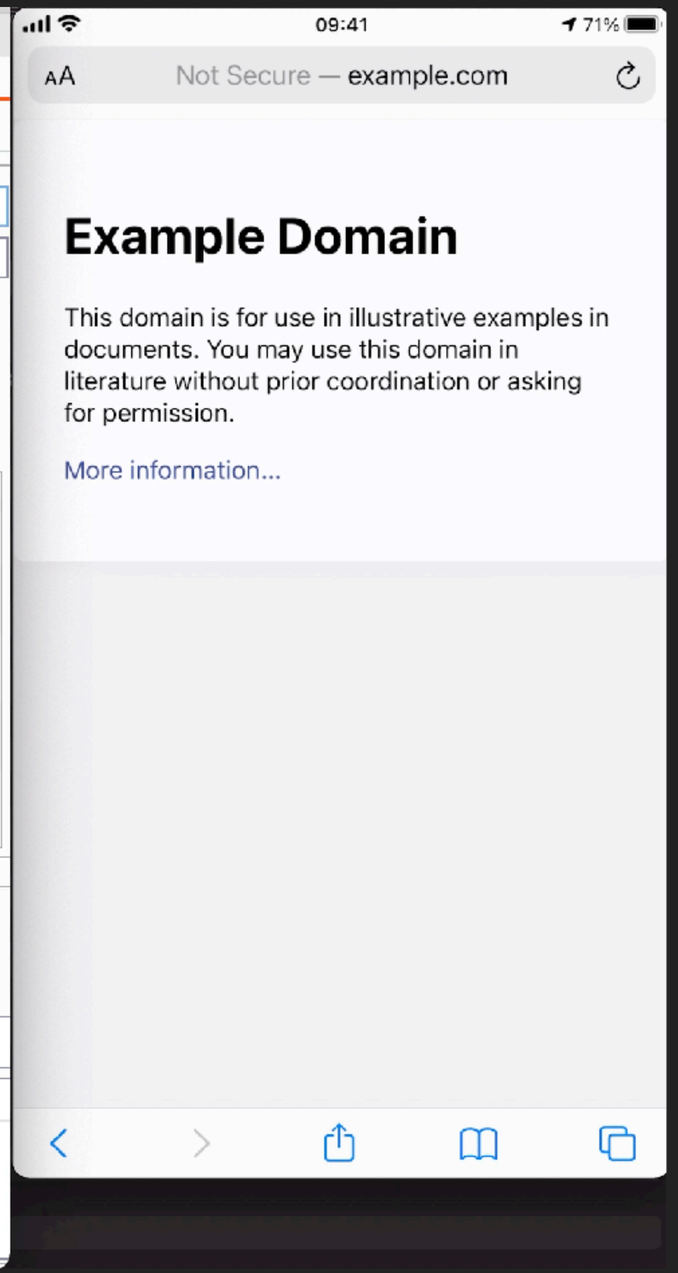
# GET THE IP AND PORT



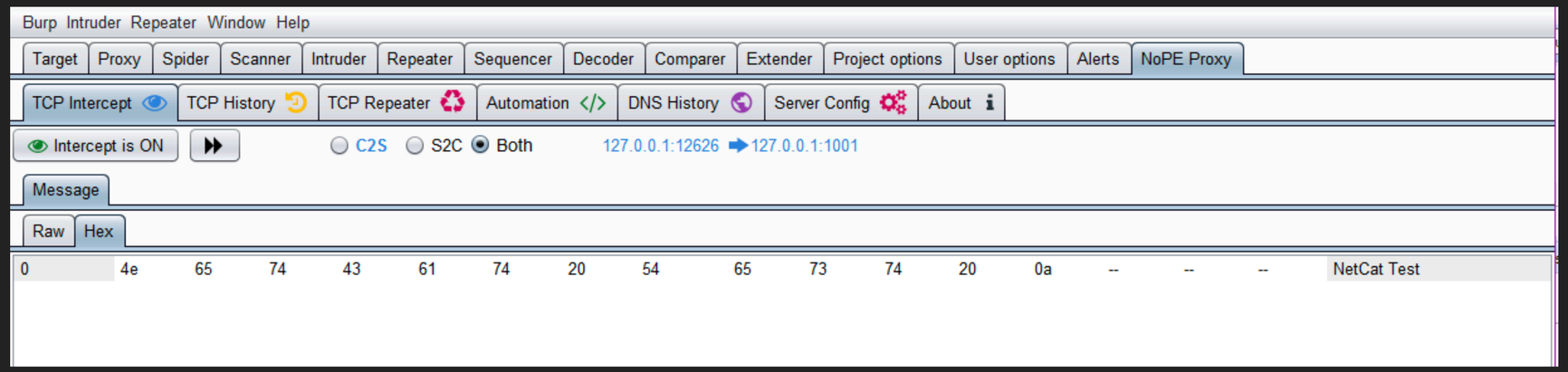
# INTERCEPT NON-HTTP TRAFFIC

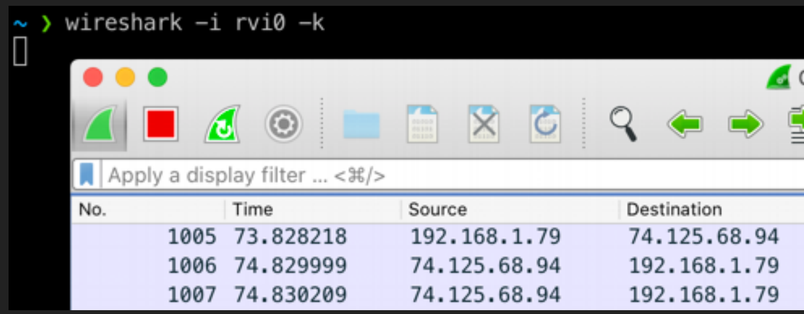


# SUMMARY FOR INTERCEPTION OF NON-HTTP TRAFFIC

- RVICTL can help us to trace and monitor all network requests coming from an iOS device
- Tcpdump can help us to trace traffic in real time on Android devices
- Tools like Wireshark are very powerful for network analysis and have a strong filter mechanism to identify the traffic we need.
- The Nope extension for Burp is one way to be able to intercept and analyse protocols that are not based on HTTP.

# KEY TAKEAWAYS



Download slide deck here:

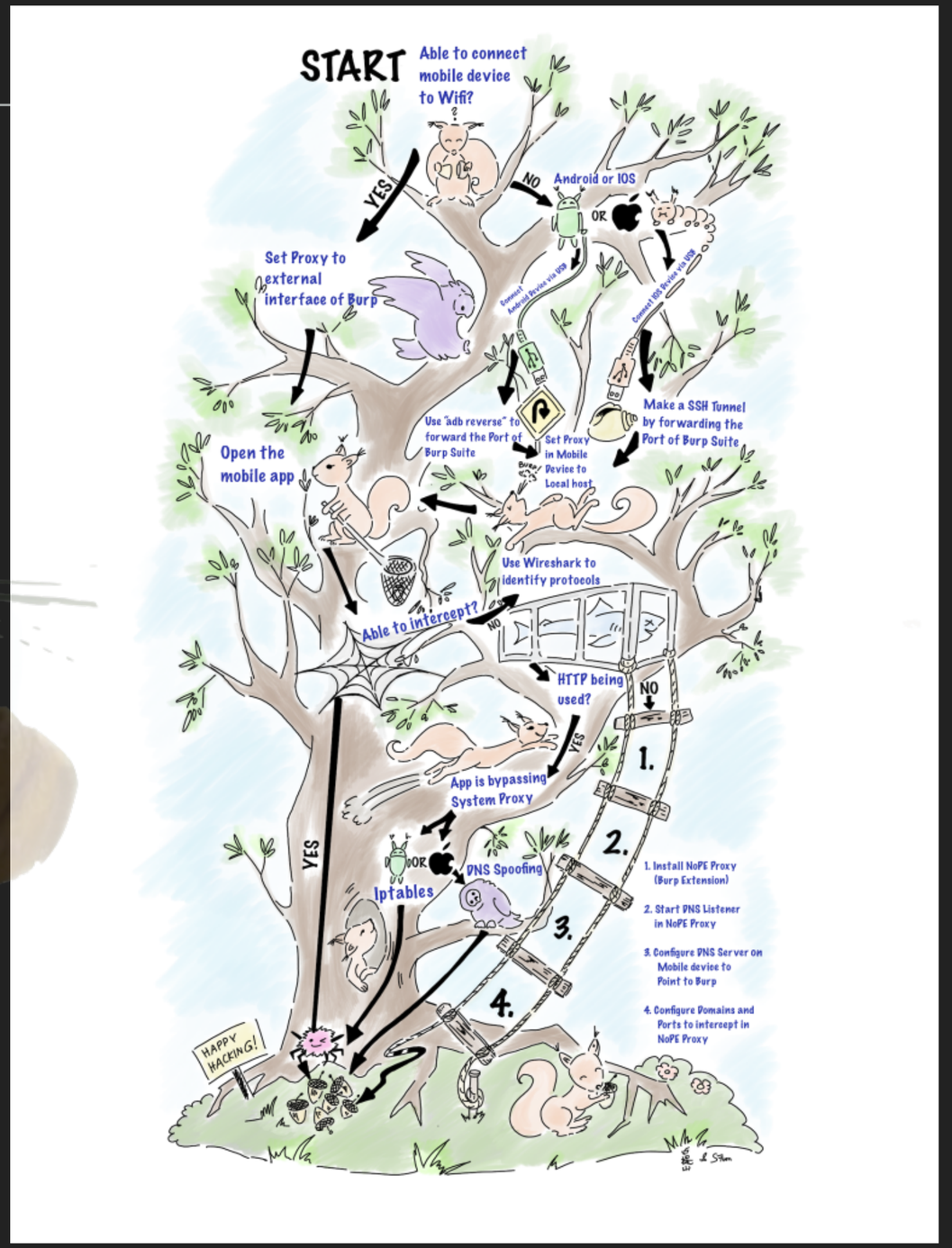
https://bit.ly/3CgoUSC

- Intercepting network communication has many edge cases!
- NoPE Proxy is an easy to use extension for Burp to intercept network communication (including non-HTTP)
- Monitor network traffic from the mobile device with Wireshark to identify all communication protocols from the app

- Additional Information:
  - Bypass SSL Pinning in Flutter Apps:
    - Android - https://blog.nviso.eu/2019/08/13/intercepting-traffic-from-android-flutter-applications/
    - iOS - https://blog.nviso.eu/2020/06/12/intercepting-flutter-traffic-on-ios/
  - Proxying Android app traffic – Common issues / checklist: https://blog.nviso.eu/2020/11/19/proxying-android-app-traffic-common-issues-checklist/

# SQUIRREL IN THE MIDDLE

- Play the game: https://bsddaemonorg.wordpress.com/2021/02/11/the-ultimate-decision-tree-for-mobile-app-network-testing-aka-the-squirrel-in-the-middle/



# Thank you!

sven.schleier@f-secure.com

 @bsd_daemon