I Know Where You Live

Sniffing & geolocating saved SSIDs

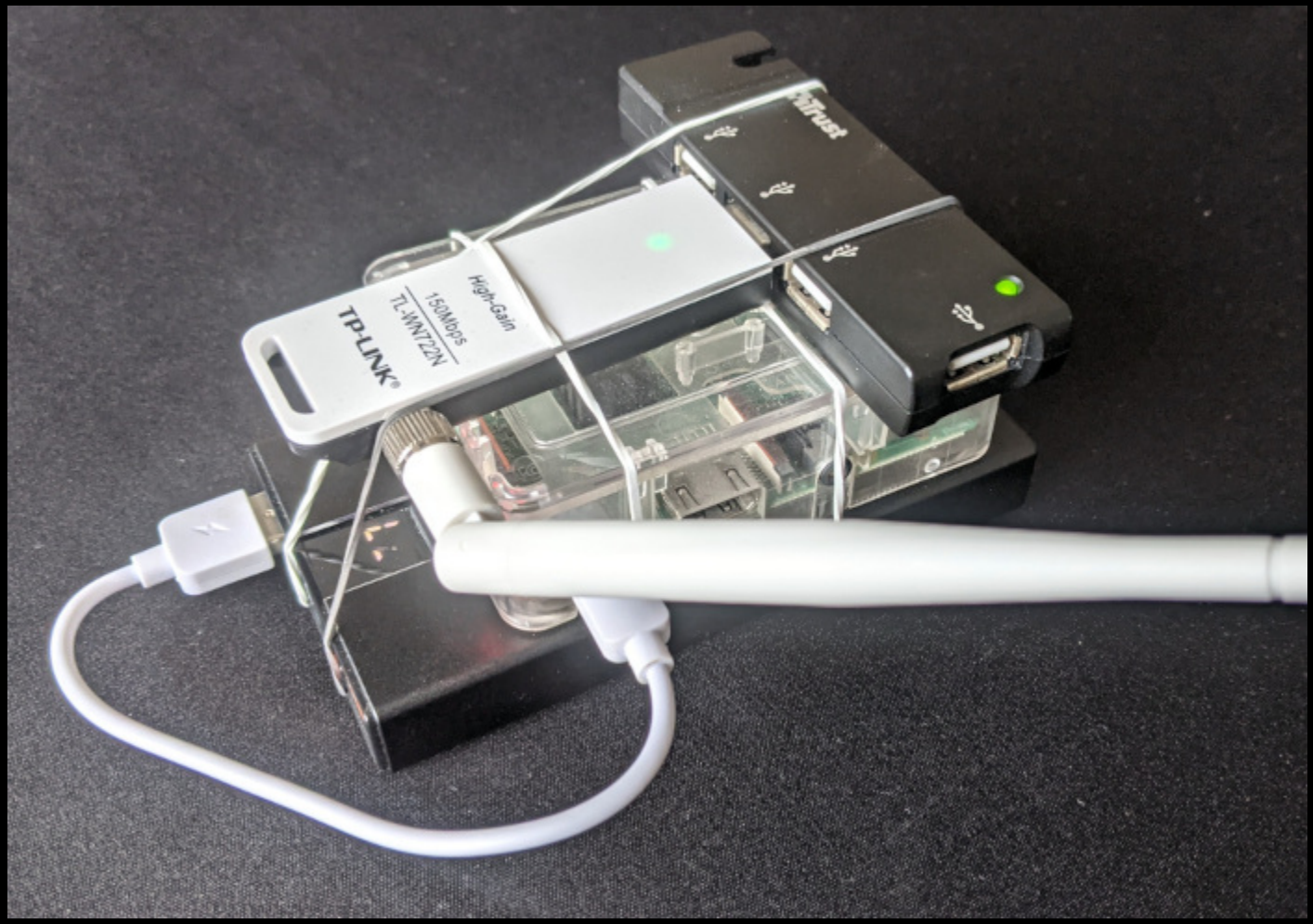
OWASP Czech Chapter Meeting

August 5, 2021 | By *@vavkamil*

Whoami?

- Kamil Vavra (@vavkamil)
  - Application Security Engineer @ Kiwi.com
  - Offensive Web Application Security
  - Burp Suite Certified Practitioner
  - Moderator of reddit.com/r/bugbounty

Proof of Concept



Agenda

- Revisiting KARMA attacks
  - History & Defense
- Building custom AP (Raspberry Pi & Python & PHP)
  - Sniffing 802.11 Probe Requests
  - SSID geolocation using OSINT
  - Rogue captive portal to fingerprint the victims

Wtf is the KARMA attack ?

- Details were first published in 2004
  - by *Dino Dai Zovi* & Shaun Macaulay
- Vulnerable client devices broadcast a "preferred network list" (PNL) containing previously connected SSIDs of access points to which they have previously connected and are willing to reconnect automatically
- These broadcasts are not encrypted and hence may be received by any Wi-Fi access point in the range

Wtf is "evil twin" attack ?

- The client receives the malicious access point's signal more strongly than that of the genuine access point
- Once the victim is connected, the attacker continues with MiTM or phishing via Captive Portal
- How to clone the SSID?
  - Plain Evil Twin targeting any network in range
  - KARMA: sniff PNL and respond with any SSID
  - Known Beacons: broadcast a list of most common SSIDs (CDWiFi, Regiojet - zluty, ...)
- You can check if you were victim of the attack by comparing history of BSSIDs

```
$ nmcli connection show CDWiFi | grep -i seen-bssids
```

Defenses applied by vendors

- Apple became the first major Vendor to deploy MAC address randomization (iOS 8, in 2014)
  - each probe request would appear to be unique
- Linux & Windows 10 support for MAC randomization arrived in 2016
- Google added an option for pre-association MAC address randomization (Android 8, in 2017)
  - made them the default for Android 10 in 2019

Defense applied by vendors

- Each Vendor does MAC address randomization differently
  - the lack of a standard approach has led to inconsistent implementations
- Some devices ...
  - randomize only the second part of the MAC
  - keep the prefix indicating the Vendor
  - keep the same MAC across re-connections
  - randomize MAC based on a predefined list
  - change randomized MAC only after an X period of time
  - randomize MAC only when switching channel

tldr; it might be still possible to track the victims

Defense applied by vendors

- Using a hidden SSID network may create a security risk
  - your phone will regularly broadcast its signal to find the network
- Using custom wpa_supplicant.conf - scan_ssid=0 (default) or 1
  - option 1 uses a directed Probe Request frame
  - many tutorials suggest using "insecure" option 1
- Older devices broadcast probe requests all the time
  - especially when walking
  - when opening network settings
  - or when the screen is off
- Interval of scanning is different based on many indicators

Building custom AP

- Main idea is to scan for 802.11 Probe Requests
  - to see if they are still being broadcasted in the wild
  - try to identify the vendors
  - try to de-anonymize clients based on poor MAC randomization
- If the SSID is being broadcasted, try to geolocate it
  - I don't know who you are, but I might know where you live :)
- Create WLAN forcing clients to visit Captive Portal when connected
  - Present the victim with a list of saved SSIDs based on MAC
  - just as a Proof of Concept (you might try with hidden SSID)

  It "might" be possible to de-anonymize the victim

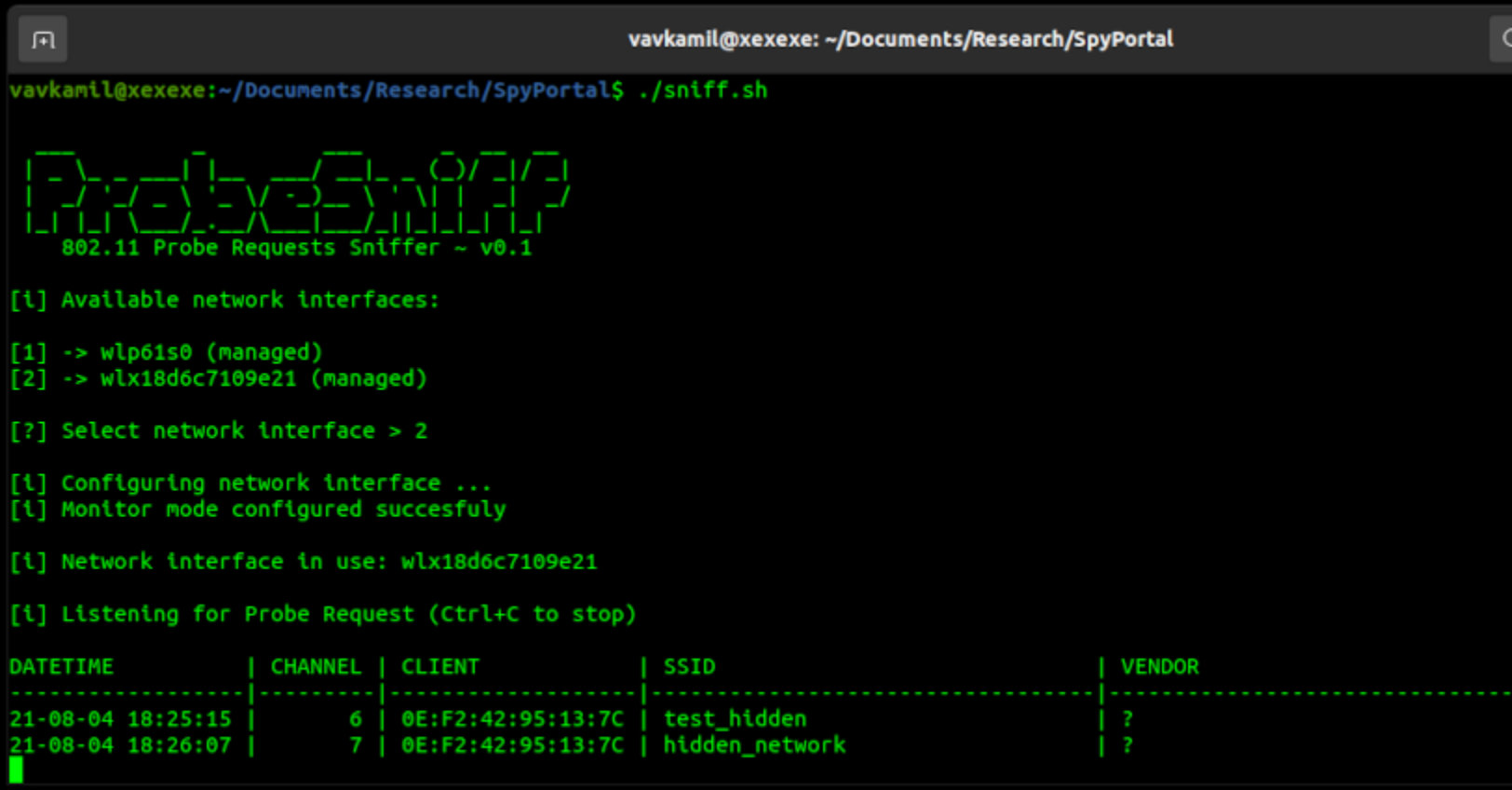
Sniffing 802.11 Probe Requests

- Using Python (Scapy)
  - Awesome library that enables the user to send, sniff and dissect and forge network packets
- Wi-Fi adapter supporting monitor mode

```
# Directory /sys/class/net/<interface>/type contains
  # the mode in which the interface is operating:
    #   1   -> managed
    #   803 -> monitor

os.system("ifconfig " + interface + " down")
os.system("iw " + interface + " set type monitor")
os.system("ifconfig " + interface + " up")

os.system("iw " + interface + " set channel " + str(channel))
```

- Change channel every 0.5 seconds
- Save Date & Time, Channel, SSID, MAC, Vendor

Sniffing 802.11 Probe Requests



Sniffing 802.11 Probe Requests

- It might be possible to identify
  - SSID of your home network
  - where you work
  - where and how often are you shopping (welcome@kaufland)
  - which public transport do you use
  - pubs that you usually visit
  - where you were on vacation (Resort Kranjska Gora, Hotel XYZ :)

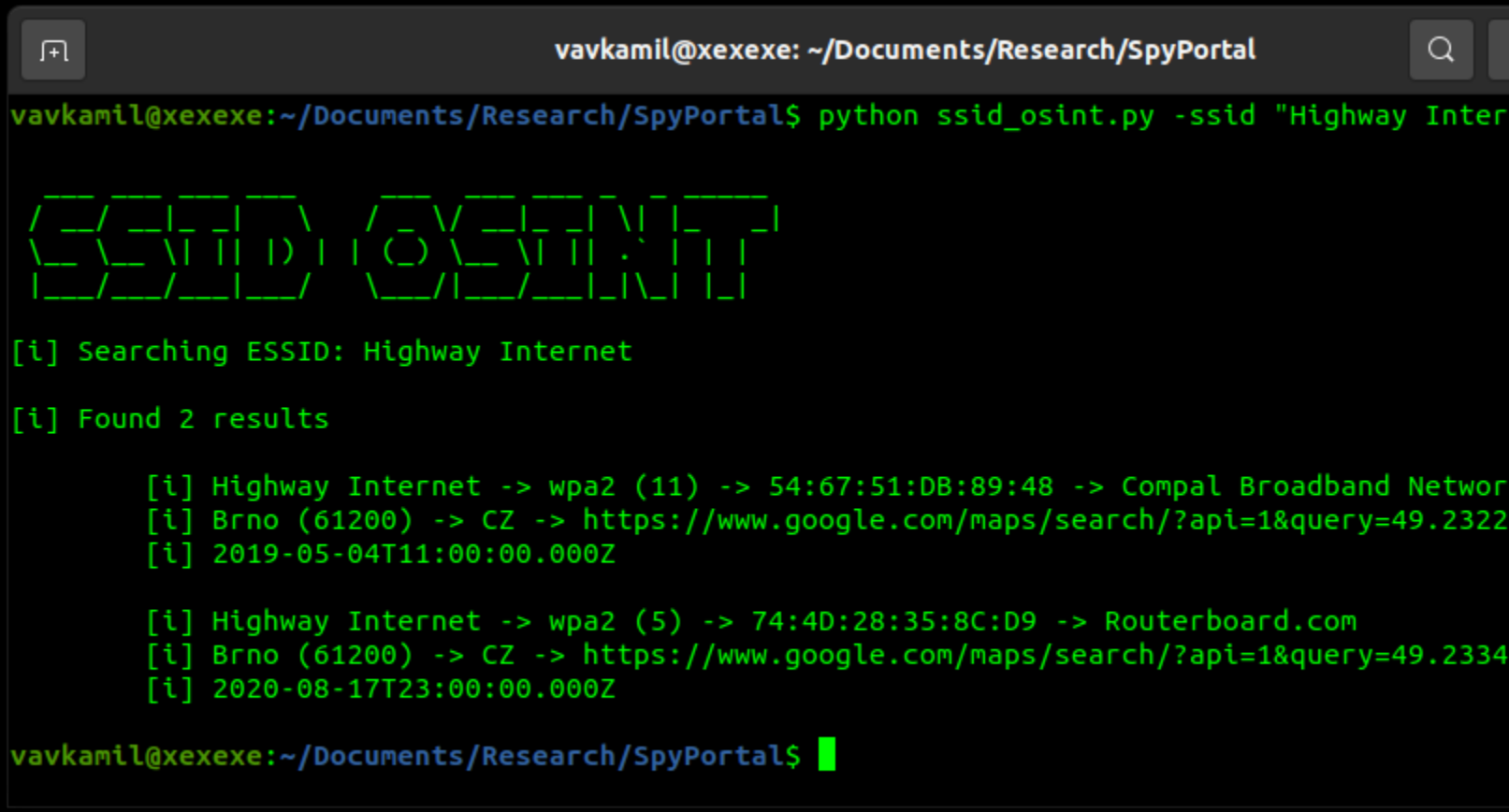
Make sure to use latest OS from reputable Vendor

SSID geolocation using OSINT

- WiGLE: Wireless Network Mapping (*wigle.net*)
  - *All the networks. Found by Everyone.*
- Android app: *WiGLE WiFi Wardriving*
  - Uses GPS to estimate locations of observed networks
  - Observations logged to the local database to track your networks
  - Upload and compete on the global WiGLE.net leaderboard
  - collecting and mapping network data since 2001
  - currently has over 350m networks
- WiGLE API (*api.wigle.net/swagger*)
  - Search, upload, and integrate statistics from WiGLE
  - You can search for any location, SSID, BSSID, ESSID, ...
  - Limit of 5 API requests for new users

SSID geolocation using OSINT

- How to find my old address



SSID geolocation using OSINT

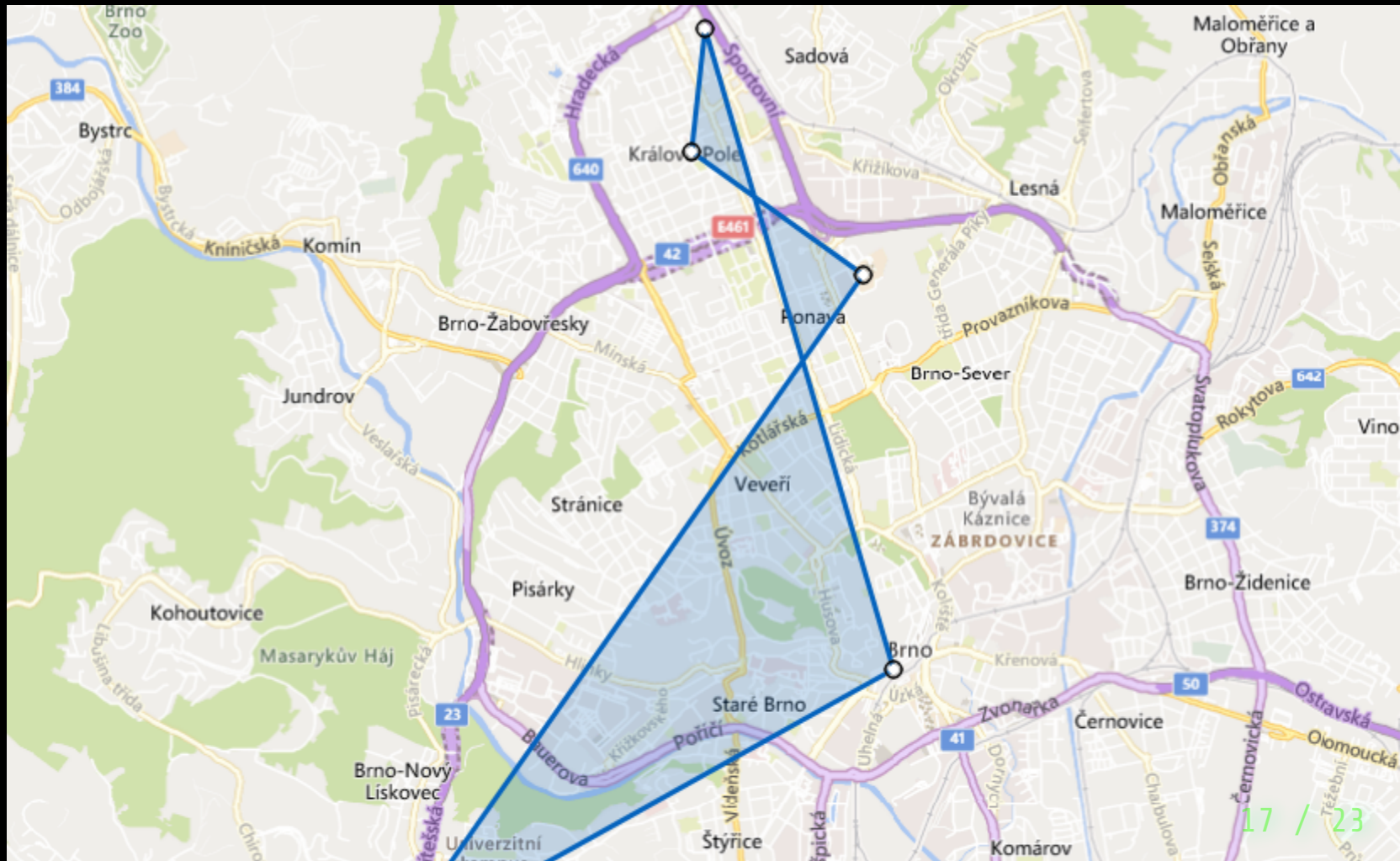
- It might be possible to
  - geolocate "any" SSID with a good enough proximity
  - track where you moved if you reuse router (BSSID)
  - check if you renamed your ESSID
  - check if you bought a new router
  - potentially find out where you created a mobile hotspot

Don't use identifiable SSID (e.g.: surname)

Randomize MAC of your router, if possible

Map area measurements

- Proof of Concept: tracking usual movements of the victim



Captive "Spy" Portal

- Work in progress (TODO: port to Python :)
  - Reused an awesome project by *@jerryryle* to save time
  - *Building a Rogue Captive Portal with the Raspberry Pi*

- Serving custom Captive Portal is an awesome idea
  - Browsers & devices will automatically request HTTP URLs such as

```
http://www.msftncsi.com/ncsi.txt
http://www.apple.com/library/test/success.html
http://detectportal.firefox.com/success.txt
http://google.com/generate_204
http://www.gstatic.com/generate_204
http://connectivitycheck.android.com/generate_204
http://connectivitycheck.gstatic.com/generate_204
```

- You can immediately present them with a phishing page
  - Especially useful to target work colleagues returning from HO

Captive "Spy" Portal

- Main idea is to present "captured" data to the victim
  - It's possible to get the MAC of the connected client from the ARP table
    - Look up the data from sniffing and show it to them
- Captive Portal might be used to further de-anonymize the victim
  - An attacker could
  - detect hostname / scan open ports
  - fingerprint installed browser extensions & installed apps
  - check if the victim is logged-in to various websites
  - steal passwords from bad passwords managers ;)

Connecting to an unknown SSID might compromise your privacy!

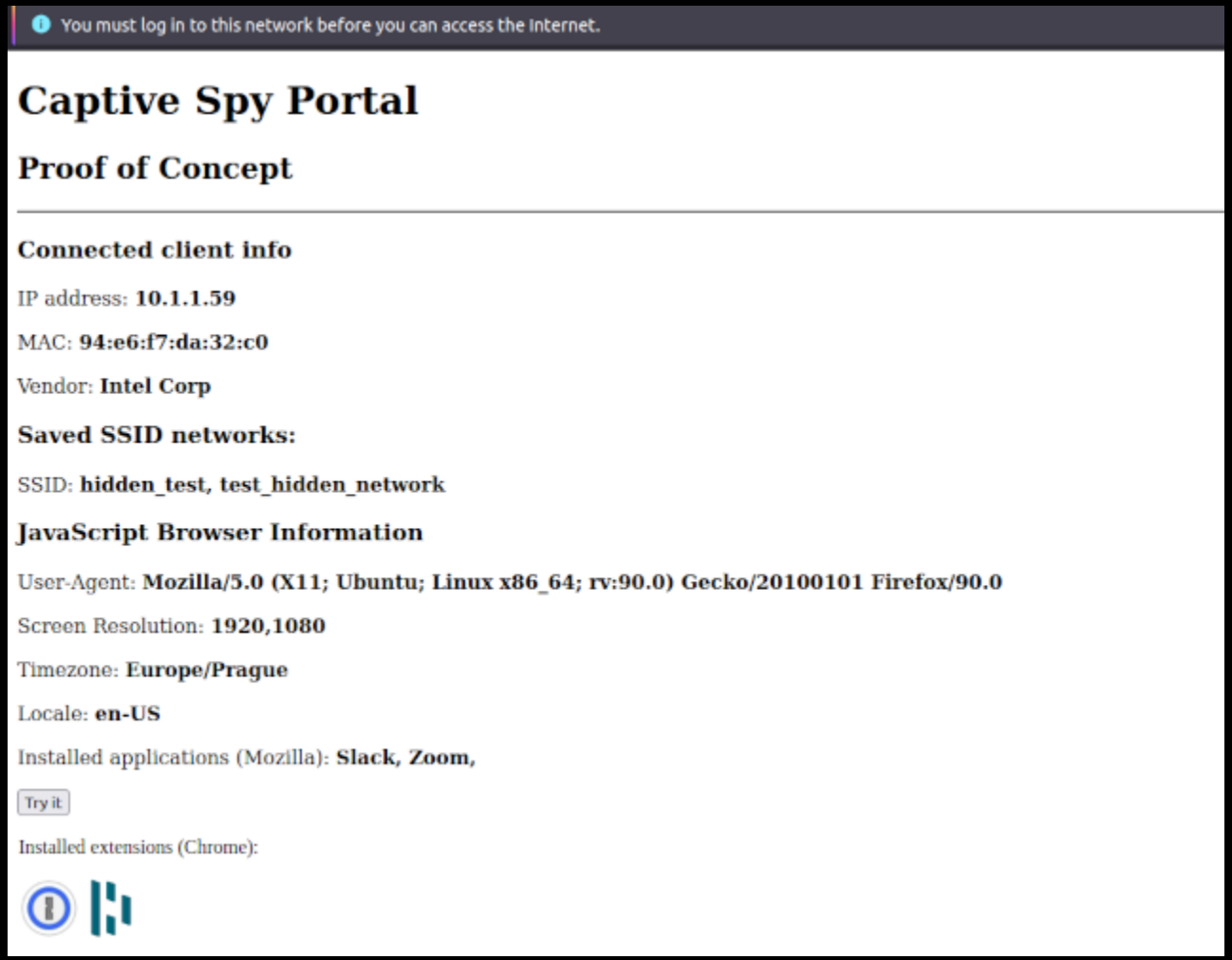
Captive "Spy" Portal

- Get MAC address of the connected client

```php
# https://stackoverflow.com/questions/1420381

$ipAddress = $_SERVER['REMOTE_ADDR'];
$macAddr = false;

#run the external command, break output into lines
$arp = `arp -a $ipAddress`;
$lines = explode("\n", $arp);

#look for the output line describing our IP address
foreach($lines as $line) {
  $cols = preg_split('/\s+/', trim($line));
  $cols_ip = str_replace( array("(", ")"), "", $cols[1]);

  if ($cols_ip == $ipAddress) {
    $macAddr = $cols[3];
  }
}
```

Captive "Spy" Portal

- Fingerprinting and possibly de-anonymizing the victim



Don't Be Evil

Proof of Concept:

Connect to the "OWASP" Wi-Fi network

- I will try my best to publish everything as soon as possible
  - Slides available at *xss.vavkamil.cz/owasp*
  - Source code: *github.com/vavkamil*

Thank you!

Any questions?