WiFi Exploitation: How passive interception leads to active exploitation
SecTor Canada
Solomon Sonya @Carpenter1010

The problem always seems to become more tractable when presented with the solution...
What to Expect

- Hand-Waving!
- Intro / Background
- Building Knowledge Requirement
- Deep Dive into 802.11 Protocol
- Developing the Sensor
- Live Demos!
- Tagging and Geotracking people
- 802.11 Vulnerability Exposure
- Security Protocol Enhancement
- Future Work
- Questions
Whoami...
Security at Present...

- How far have we come?

- Is it good enough?
  - Discovered vulnerabilities are fixed with patches
  - Known malware removed with AV (signature based)
  - Emerging malware “detected” via baselining (anomaly)
  - Digitally signed software

- We still believe “Detection is the key”
  - Avg malware lifespan (depending on source) ~294-300+ days still!

- Fallacy with Security:
  - Current [incorrect] view: start-state is secure, bolt on security from here
  - We’ll remain ahead of the adversary ;-)
Anatomy of a Cyber Attack

Reconnaissance

Scanning

Penetration

Stealth & Persistence

Privileges+

Pillage

Pivot

Paralyze

Cover Tracks

Source: Solomon Sonya @Carpenter1010
Updated Anatomy of a Cyber Attack

Reconnaissance/Research

Scan Targets

Penetration

Privileges++
Pillage
Pivot
Paralyze

Stage Exploits

Evade Detection

Maintain Access

Social Engineering
Water-Hole
Drive-By
Phishing
XSS
Protocols
Trojan
Insider
Embedded Devices

Active & Passive
Ping Sweep
ARP Scan
Port Knock
DNS Lookups
IP Reservations
Management Protocols

Source: Solomon Sonya @Carpenter1010
Security of the Future

- Root of the problem lies with how security is considered during creation and deployment:
  - Bolted-on vrs. Built-in approaches

- Integration of Smart Devices
  - A country’s greatest spy

- IoT… Are we ready?
Cell Phones: A country’s greatest spy

- When was the last time you audited the permissions granted to your apps?
- Is all of this necessary to show a light? (I would avoid apps like these!)

Really?!!!

WiFi Leakage: How passive interception leads to active exploitation

Solomon Sonya * @Carpenter1010
Thought Question...

Even if we secured it all…
What are we doing still
to secure the protocols
these devices are using to communicate?
How did we get here...

- Researching threat intelligence last year
- Sitting at an airport enroute to a conference, watching people pass by, I wondered if it is possible if I could determine where each person is coming from *a priori*...
- Knew people usually carried cell-phone, smart device, and/or laptop on travels and these devices are constantly probing to connect to a known network
- Hacker’s Mantra: “I wonder what happens if…”
Developed the following research questions to direct new research project:

- Is it possible to intercept PNL (preferred network list) probes to fingerprint certain people?

- If so, can a profile be created to reveal the area-of-habitation (likely places of work, live, play)

- Can we determine a likely device and alert on likely “places of interest” such that we can identify a person that works/lives at specific places? (Think Intel, Google, military, etc)

- Can we expand profile on a person to determine their previous geolocations, SSIDS, and activity times within an area such that we can know when to expect a person within a particular area? (think home and work, etc…)

- **Determining each devices’ PNL, can we establish a rogue AP and MiTM a user’s device to route all traffic through our machine without the victim’s knowledge?!!!!!**

**Spoiler Alert:** YES YOU CAN!!!! Let’s see how!
Initial Knowledge Required…

- **802.11 Frames:**
  - **Management Frames:** Setup and maintain communications
    - Authentication, Deauthentication
    - Association, Disassociation
    - Synchronization Messages
    - Probe
    - Beacon
  
  - **Control Frames:** Assist in frame delivery and reduces collisions
    - Acknowledgements, Request/Clear to Send, Block, Poll, End
  
  - **Data Frames:** Transport data from higher layers (HTTP, etc)

- **802.11 Client Authentication Process**
Initial Knowledge Required...

- Distributed Computing (Efficiency, Optimization, Updates)
- Socket Programming (Connections, Tokenization)
- Threads
- Wrappers (Worker Process, Conversion, Parsing, Encryption, etc)
- Coding not your thing? No problem, just use Theia! Demo coming in a few slides from now!
Deep Dive: Management Frames - 1

- Request / Response Frames
- Authentication Frame
  - Network member (wireless device NIC) signifies intention to join membership with access point (AP)
- Deauthentication Frame
  - Access point sends frame to member to terminate <secure> connection
  - This packet must be accepted and immediately terminates communications
- Association Frame
  - Synchronize resources between AP and NIC
  - NIC exchanges supported data rates, SSID, Encryption Protocol
  - If accepted, response from AP allows NIC to communicate with AP
  - Reassociation similar – used when NIC roams to AP with stronger signal
- Disassociation Frame
  - NIC wishes to gracefully terminate the association to allow the AP to reallocate memory
Deep Dive: Management Frames - 2

- **Probe Request Frame**
  - NIC queries for available AP’s or specific AP containing SSID within range
  - Transmitted on every channel the NIC supports to discover every compatible AP and AP with requested SSID
  - Supports roaming (with reassociation) to maintain established connection

- **Probe Response**
  - APs respond to requesting clients and provides synchronization information (data rates, SSID*, Encryption Protocol, etc)
  - Cloaked: AP will respond if probe includes correct SSID
  - Discover cloaked AP when associated member joins and probes for “hidden” SSID

- **Beacon Frame**
  - AP periodically broadcasts its presence and connection information (BSSID, supported data rates, Encryption Protocol, SSID (if not hidden))
  - Cloaked: AP sends beacons, but omits SSID
Deep Dive: Control / Data Frames

- **Control Frames**
  - Optional Frames: Request to Send (RTS) and Clear to Send (CTS)
  - Reduces frame collision
  - Not too common, but seen when AP has **hidden** SSID

- **Data Frames**
  - Transport data frames after NIC has associated with AP
802.11 Pertinent Frame Subtype Identifiers

- Authentication 0x0b
- Deauthentication 0x0c
- Association Request 0x00
- Association Response 0x01
- Reassociation Request 0x02
- Reassociation Response 0x03
- Probe Request 0x04 *** (NIC is in the area)
- Probe Response 0x05 (now know AP is in the area)
- Beacon 0x08 (now know AP is in the area)
- Request to Send (RTS) 0xb0 (usually present with hidden Aps)
- Clear to Send (CTS) 0xc0

- Control and Data frames handled in future research

So... What do we do with this information?

Let’s bring it together by understanding the 802.11 Authentication Process, PNL, and then let’s demo!
Device routinely probes to discover available access points in the area and rejoin previously associated networks.

**Client Probe Activity**

- **BROADCAST**
  - Probe requests to different access points (NETGEAR, linksys, Free_Airport_WiFi).

**Client Authentication Process**

1. **Probe** from device to access point.
2. **Probe Response** from access point to device.
3. **Authentication Request** from device to access point.
4. **Authentication Response** from access point to device.
5. **Association Request** from device to access point.
6. **Association Response** from access point to device.
7. **Data** transmission between device and access point.

**Timeline:**  
- **Time ↓**
Let’s Build the Sensor!
Hardware Setup on the cheap…

- Minimum
  - Wireless Card (Alfa Card)
  - Wireshark (Tshark)
  - *NIX OS (Kali 2.x.x)

- Extra Credit
  - Long Range, High Gain WiFi Antennae “BAA” (12dbi Antennae)
  - GPS Receiver (GlobalSat BU-353-S4 USB GPS Receiver)
Intercepting Frames to Begin Analysis

- Collection process: acquire/query, parse, analyze/interpret, present, visualize

- What data should we collect?

- How do we collect it?

Let’s begin with the protocol analyzer Tshark (shortlist)

- `-Y wlan.fc.type –T fields` Focus and filter on 802.11 protocol
- `-e wlan.fc.subtype` Display different frame types (mgt, ctrl)
- `-e wlan.sa` Display transmitter MAC address
- `-e wlan.da` Display receiver MAC address
- `-e wlan` Display WiFi protocol information
- `-e wlan_mgt.ssid` Display the SSID (if present)
What can we do at this point?

- Active devices within range
- Preferred Network List broadcasted by active devices
- MAC addresses for devices and access points
- SSIDs broadcasted by non-hidden Access Points
- Vendor identifying information (when linked to OUI database)
Initial Sensor Construction

- What did we first capture?

```bash
root@kali:~# ifconfig wlan0 down
root@kali:~# iwconfig wlan0 mode monitor
root@kali:~# ifconfig wlan0 up
root@kali:~# tshark -i wlan0 -Y wlan.fc.type -T fields -e wlan.fc.subtype -e wlan.ta -e wlan.da -e wlan -e wlan_mgt.ssid
```

Running as user "root" and group "root". This could be dangerous.

```
tshark: Lua: Error during loading: [string "/usr/share/wireshark/Init.lua"]:44: dofile has been disabled due to running Wireshark as superuser. See https://wiki.wireshark.org/CaptureSetup/CapturePrivileges for help in running Wireshark as an unprivileged user.
Capturing on wlan0
```

<table>
<thead>
<tr>
<th>Subtype</th>
<th>Transmitter</th>
<th>Destination</th>
<th>Details (snip)</th>
<th>SSID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How do we process this information to derive meaning from the data?

“Everything should be made as simple as possible, but not simpler.” – Albert Einstein
Frame Acquisition Process

- **Process:**
  - Acquire/Query, Parse, Analyze/Interpret, Present, Visualize

- **Recall:**
  - If we build our own sensor, what components are required?
  - Wouldn’t it be nice if we had a tool to process all of this information for us?
  - Options you could use: Kismet, NetStumbler, inSSIDer, Airocrack-ng suite... OR, you can build your own and add custom features specific to your environment... I chose the latter!
Components to Build the Sensor Suite

- 802.11 Receiver/Antennae (Sensor)
- GPS Receiver (GPS Collector)
- IEEE OUI Specification
- Geo-Location Retrieval Agent
- Collector: Acquire/Query, Parse, Analyze/Interpret, Present
- User Interface: Visualize

**Diagram:**

- GPS Collector
- Sensor
- OUI Agent
- Geo-Retrieval
- Collector
- Log
- Theia Interface
Demo

Enough Talk... DEMO!!!!!

Theia
Theia Sensor Suite

- Distributed Wireless Sensor Suite to acquire, analyze, process, and visualize data extracted from 802.11 frames
- Extensible: Allows multiple sensor integration back into the central collector(s)
- Displays linkage between PNL from each device and APs within the area
- Provides geolocation ability to display and map locations where a device (and user) have been
- Provides tagging, alert, and notification capabilities for entry and departure of devices and base stations
Theia Introduction and Live Demo
Theia Introduction and Live Demo

WiFi Leakage: How passive interception leads to active exploitation * Solomon Sonya * @Carpenter1010
WiFi Leakage: How passive interception leads to active exploitation

* Solomon Sonya * @Carpenter1010
Theia Introduction and Live Demo

WiFi Leakage: How passive interception leads to active exploitation

* Solomon Sonya * @Carpenter1010
So What Have We Discovered so far...

Theia
So What Have We Discovered so far...

- Device Profile (Vendor, MAC, etc)
- Device PNL (full probing list)
- Relative Proximity of devices within range of sensor (RSSI)
- Arrival and Departure Activity Window
- Encryption Specifications for each requested SSID
- Frequency and communication channel for each device

If we can determine this amount of information, can we tag a device to a person?

**Spoiler Alert: YES YOU CAN!**
Profiling People (e.g., social engineer connection to planted AP, you can now specifically tag an individual person to a device)

Intercepting frames from a device can reveal much more than details about the device, but locations of the user!

By analyzing PNL, can we create a geo-location profile on people?!??! If so, how?

- Process PNL for each device, submit query to Theia_GEO to acquire geolocation of where SSID (AP) has been detected before
- Link Geolocation data to SSID and device... PWN!
- Special thanks to Wigle.net and @bobzilla!
Hello there, simply because you left your WiFi enabled…
I now know where you’ve been…
Enriching the Database

- War-Driver
- Wigle.net
Populated SSIDs

Total Number SSIDs loaded: 57

Total Number SSIDs requested by device: 14

Access Point SSID: Pullman-Cairns-Guest
Latitude, Longitude: 146.92464286, 14.8760903
Number of Probing Devices: 1
STATUS: DEPARTED
Encryption Type (if known): null
Date Last Seen: 9/15/2014 10:07
Probing Device: 54:26:96:CE:52:3F - Apple Inc
Probing Device Channel List: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Probing Device Frequency List: 2412, 2417, 2422, 2427, 2437, 2442, 2447, 2452, 2457, 2462
Theia Live Geo-Tracking Demo

Populated SSIDs

Total Number SSIDs loaded: 819

Access Point SSID: Ryk Hgt
Date Last Seen: 7/29/2015 017
Latitude, Longitude: 64.1444738, -21.9219559
Theia Live Geo-Tracking Demo

SSID: 5g Mountain Dew

Total Number MACs requesting for this SSID: 7

Access Point SSID: 5gMountainDew
Latitude, Longitude: 38.056316, -104.6955613
Number of Probing Device(s): 7
STATUS: ACTIVE
Date Last Seen: 5/14/2015 2:47
Probing Device(s):
- Last Contact: May 11, 2016 22:39:09 | 5D:1A:5D:60:5A:E3 - Microsoft
- Last Contact: May 12, 2016 01:13:07 | C0:1A:DA:00:84:07 - Apple, Inc.
**Theia Live Geo-Tracking Demo**


Total Number SSIDs requested by device: 5

*Access Point SSID: Hotel Alex Johnson*
*Latitude, Longitude: 44.06028793, -103.2287761*
*Number of Probing Devices: 1*
*STATUS: DEPARTED*
*Date Last Seen: 10/20/2015 14:09*
*Probing Device Channel List: 1, 4, 8, 11*
*Probing Device Frequency List: 2422, 2447, 2462, 2484, 2500*
Vulnerability Exposure
Vulnerability Exposure [2]

- 802.11 protocol fails to tuple SSID to geo-location, BSSID
- Your PNL is blindly built on a system of trust... surely the AP that responds to your probe is legit right?... → This leaves room for exploitation since any device will suffice!
- Thus, if another device “suddenly” comes into the area that matches your probe, your device will attempt to authenticate with it
- This is especially dangerous if authenticating to OPEN WiFi networks!
- Don’t believe me? Welp, let’s start further exploitation with the WiFi Pineapple and then build our own attacks to go after victim devices

This is the access point you are looking for...

http://www.reidlitchfield.com/these-aren’t-the-droids-you’re-looking-for/

https://www.wifipineapple.com/
Demo

MiTM attack via WiFi Pineapple

Theia
802.11 Protocol Security Recommendations
Security Recommendation

- **802.11 Protocol Enhancement**
  - Devices should restrict full broadcast of entire PNL unless within range of known BSSID
  - Update required to tuple SSID, BSSID(MAC), and GPS Coordinates to known SSID and base station
  - Notify when anomaly detected

- **802.11 MiTM Mitigation**
  - Never store OPEN WiFi SSIDs on your devices
  - If so, device should notify when reassociated to “previous” base station

- **Geo-Location tracking and Personnel Profiling Mitigation**
  - Remove all inactive SSIDs from devices
  - Disable WiFi when not in active use

- Audit active WiFi connections on devices

- Rename AP SSID broadcasts (it turns out less unique is actually safer for you in this case…!)

- A new capability is required and we must audit 802.11 spectrum… (stay tuned, Connectionless (C)overt Channel: data-exfil and C2 module release in the works coming next summer) simply from 802.11 frames… whose auditing this right now???

“We can't solve problems by using the same kind of thinking we used when we created them.” – Albert Einstein
Configuring and Running Theia...
Establishing Theia Sensor Suite

- **theia_oui.exe** [*NIX*] or [windows]
  - Will autostart and establish serversocket for new connections

- **theia_geo.exe** [*NIX*] or [windows]
  - Will autostart, establish serversocket for new connections, and allow you to import geo-location database table

- **theia_sensor.exe** [*NIX only] – because of the wifi drivers
  - Will autostart, search for wireless interfaces, bind to interface, establish serversocket for new connections, display 802.11 collected frames ready for transmission to collector

- **GPS_collector.exe** [*NIX only]
  - GPS agent, autostarts, binds to GPS receiver, displays telemetry data from receiver, establishes serversocket to relay data through sockets
  - First time: will download and configure dependencies for GPS reception
  - Connect theia_sensor.exe to GPS_collector.exe

- **theia_collector.exe** [*NIX*] or [windows]
  - Will autostart and establish serversocket for new connections
  - Connects to theia_oui.exe, theia_geo.exe, theia_sensor.exe, theia_interface.exe
  - Collected frames will now be processed by the system ready for visualization

- **theia_interface.exe** [windows only until Java 1.8 FX release on *NIX]
  - User Interface providing visualization from collector
Upcoming Features / Releases

- Light-weight deployment module to Raspberry Pi
- Sensor distribution package for Windows OS
- Simultaneous multi-antennae integration
- Crowd-source SSID database (distributed sharing)
- Increased accuracy: GEO lat/lon to address
- Additional wardriving compatibility with Kismet and Netstumbler
Questions?

Acknowledgements:
- SecTor, Toronto, Ontario, Canada
- Wigle.net, @bobzilla
- Google
- Vivek Ramachandaran, @securitytube
- Sushail M. “The Boss” @hackcon
WiFi Leakage: How passive interception leads to active exploitation

Solomon Sonya
@Carpenter1010
Theia_OUI

- Provides Vendor lookup information for detected MAC’s
- `java -jar theia_oui.jar`
- (I have already prepackaged the OUI table within the program.)

```
root@ali:/Desktop# java -jar theia_oui.jar

Welcome to Theia_OUI vrs 0.110 created by Solomon Sonya @Carpenter1010

Listing Network Interfaces now...
- > Network Interface Map Complete.
OUI resource was successfully located within archive at /Resources/OUI/2016-02-08/oui.txt
Commencing Parser functions on file: /Resources/OUI/2016-02-08/oui.txt

Parser complete on file: /Resources/OUI/2016-02-08/oui.txt. Num lines read: 129755. Unique OUI MAC addresses: 21580

Ready to establish server socket. Run command "establish_server_socket" or "help" if necessary...

I will attempt to establish server socket for you. Remember to use the help if you need to configure the server socket differently. Standby...

Listing Network Interfaces now...
- > Network Interface Map Complete.
Listing Network Interfaces now...

Listing Available IP addresses...
IP Address: /fe80::0:0:0:29c:29ff:fe29:6eb9%eth0
IP Address: /192.168.0.130
IP Address: /192.168.0.9%lo
IP Address: /127.0.0.1

ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /fe80::0:0:0:29c:29ff:fe29:6eb9%eth0 on port <8081>...
ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /192.168.0.130 on port <8081>...
ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /0:0:0:0:0:0:0:1%lo on port <8081>...
SUCCESS! ThdServerSocket is bound to 192.168.0.130:8081. Ready for new connections across port 8881.
ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /127.0.0.1 on port <8081>...
SUCCESS! ThdServerSocket is bound to 127.0.0.1:8081. Ready for new connections across port 8881.
```
Theia_Geo

- Provides Geo coordinates lookup for detected SSIDs
- `javac -jar theia_geo.jar`
- Import geo-database table configuration file(s)

```
root@kali:/Desktop# java -jar theia_geo.jar

Listing Network Interfaces now...
   --> Network Interface Map Complete.

Ready to establish server socket. Run command "establish_server_socket" or "help" if necessary.
Commencing Parser functions on file: /root/Desktop/colorado.txt

I will attempt to establish server socket for you. Remember to use the help if you need to configure the server socket differently. Standby...

Listing Network Interfaces now...
   --> Network Interface Map Complete.
Listing Network Interfaces now...

Listing Available IP addresses...
IP Address: /fe80:0:0:0:0:0:fe29:6eb9%eth0
IP Address: /192.168.0.130
IP Address: /0:0:0:0:0:0:0:1%eth0
IP Address: /127.0.0.1
ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /fe80:0:0:0:0:0:fe29:6eb9%eth0 on port 8282...
ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /192.168.0.130 on port 8282...
ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /0:0:0:0:0:0:0:1%eth0 on port 8282...
ThdServerSocket thread started. Configuring Environment to listen for new sockets across interface /127.0.0.1 on port 8282...
SUCCESS! ThdServerSocket is bound to 0:0:0:0:0:0:1%eth0. Ready for new connections across port 8282
SUCCESS! ThdServerSocket is bound to 192.168.0.130:8282. Ready for new connections across port 8282
SUCCESS! ThdServerSocket is bound to 127.0.0.1:8282. Ready for new connections across port 8282
```
Theia_Collector

- Heart and Engine of the entire system - Processes raw sensor data from theia_sensor to establish appropriate linkages ready to visualize in the interface
- Self configures environment ready for you to connect to theia_geo, theia_oui, theia_sensor, theia_interface
- java –jar theia_collector.jar
GPS_Collector (OPTIONAL)

- Interfaces with GPS receiver to supply GPS data to all connected sockets
- java –jar gps_collector.jar
- NOTE: If this is the first time running, I programmed it to configure your system for you, and then interface with GPS receiver
Actual sensor capturing 802.11 frames and transmitting to connected Theia_Collectors

`java -jar theia_sensor.jar`

After started, connect to GPS_Collector.jar (if applicable) using command:

`gps_connect <IP> <PORT>`
Theia_Interface

- User Interface to visualize data intercepted by Theia Sensor Suite
- Connects to Theia_Collector
- `java -jar theia_interface.jar`
Building the Sensor - 1

- Determine wireless interfaces and place into monitor mode
  - `ifconfig`
  - `ifconfig wlan<#> down`
  - `iwconfig wlan<#> mode monitor`
  - `ifconfig wlan<#> up`

- Frequency Hopping!
  - Very important otherwise, you’ll capture only on 1 channel
  - There are scripts to do this, we’ll implement programmatically based on our interrupt timer
  - `iwconfig wlan<#> channel <#>`
  - (we’ll code this later in the presentation)

- GPS Collector

- ServerSocket

- Transmission of frames across connected sockets
Building Sensor

- Remaining slides will cover development of the sensor programatically
  - **GPS_Collector**
    - Interfacing with GPS daemon, process and present telemetry
  - **Sensor**
    - Reading 802.11 frames directly from wireless interface card
  - **Geo**
    - Wigle.net API
    - Database construction
    - Google Map API construction
  - **OUI**
    - IEEE specification, HashTable construction for efficient retrieval
  - **Collector**
    - Most critical and robust component - HashMap, TreeMap, LinkedList required to efficiently store and link devices
  - **Interface**
    - Processing all data from Collector into User Interface
  - **WarDriving**
Upcoming Features

Theia