Beginner's Guide on How to Start Exploring IoT Security

SECURITY STUDY GROUP

#! Print(“print aboutme”)

- Veerababu Penugonda
- Working @Aujas , IoT/OT security
- Working and R&D on IoT Security for past 2 years
- Not Expert just Learning everyday
- Published articles , writing blogs & GitHub pages
- Giving the talks for open communities

- Key skills – CTF player, CVE , Scripting and reverse engineering

IoT(Internet of things)

- A Device which connected to Internet and sharing the data directly or indirectly is called Internet of things
- IoT is having the lot of future scope to develop and speeding the world next level
- Smart things everywhere – smart bands , health industry , smart gadgets like amazon echo , etc
- Smart things all are user defined and vendor development
  - Which means according to our purpose only we are interest use the devices and vendor is creating a needed gadget for all

What is OT

- OT – Operational Technology
  - Which is hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices, processes and events in the enterprise.

Scenario	IoT	OT
security	Challenging	Challenging
Pentesting	Difficult	Difficult
malware	Critical	High

IoT/OT blooming day by day

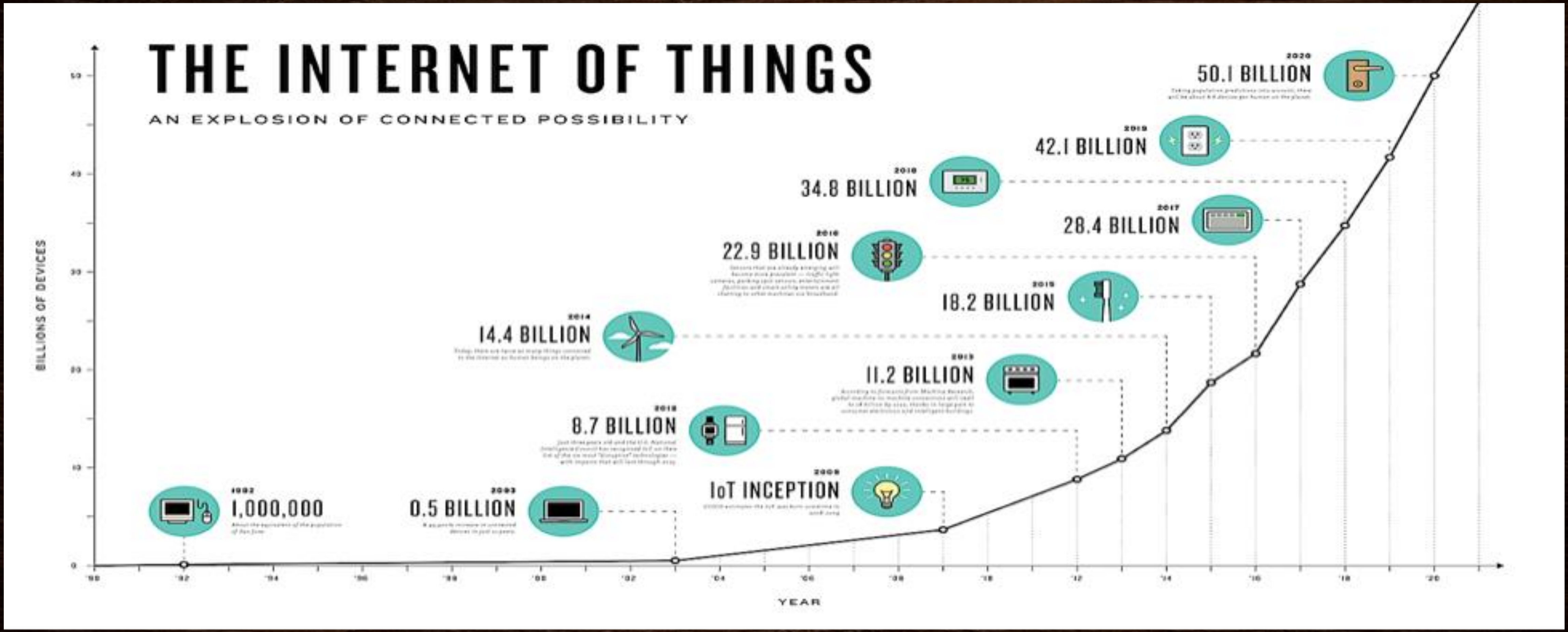


Image Source: http://www.nsr.com/upload/images/M2M5_BL3._graph_1.png

IoT/OT

Smart IoT

- Smart bands, BLE Devices,
- Connected clocks

OT

- ICS
- SCADA , PLC

Hardware

- PCB'S, CHIPS

- Key Points
- IoT/OT everywhere
- When its connected world anyway it will be vulnerable to hack
- Security always is challenging task compare to pentesting or hacking
- So will discuss about the security practices also

IoT attack vector

- Networks
- Radio & Wireless communications
- Embedded application and web services
- Mobile (android and iOS)
- Cloud , API
- Firmware (UEFI , filesystem, Bootloaders)
- Hardware

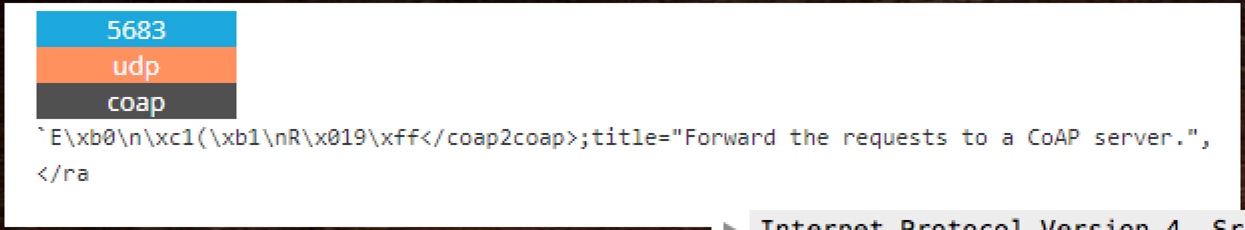
1. Network pentesting in IoT

- Finding open ports and running services with version
- Attacking with Metasploit with known vulnerabilities
- Writing fuzzing scripts to grab the information from the device
- Writing exploit code to trying to get reverse shell with different way

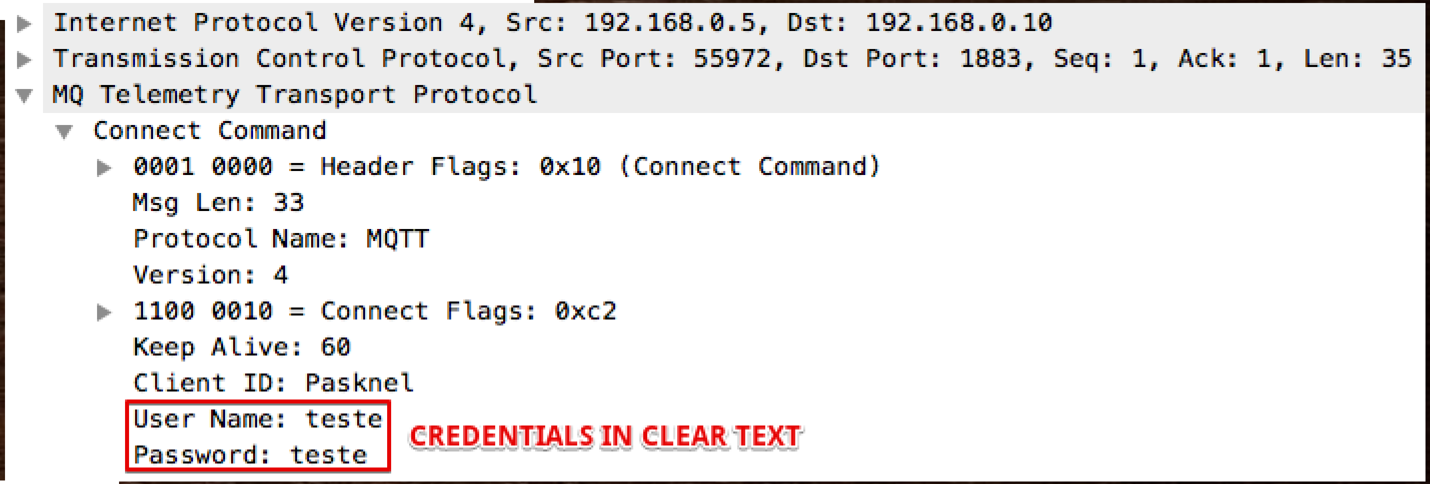
Tools to be used : Nmap , curl , NetCat , hydra, Metasploit , SEH etc

Running services in IoT - network level

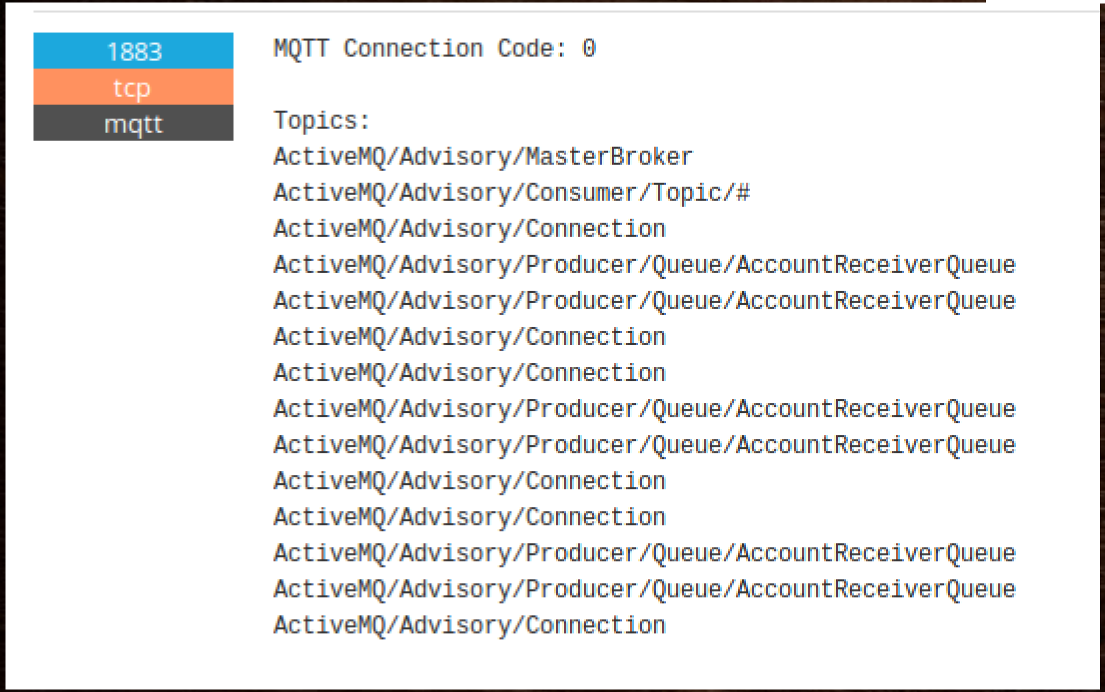
- FTP (21)
- telnet (23)
- SSH (22)
- RPC bind (111)
- XMPP (5222, 80 ,443)
- MQTT (1883 , 8883)
- CoAP (5683)



```
`E\xb0\n\xc1(\xb1\nR\x019\xff</coap2coap>;title="Forward the requests to a CoAP server.",
</ra
```

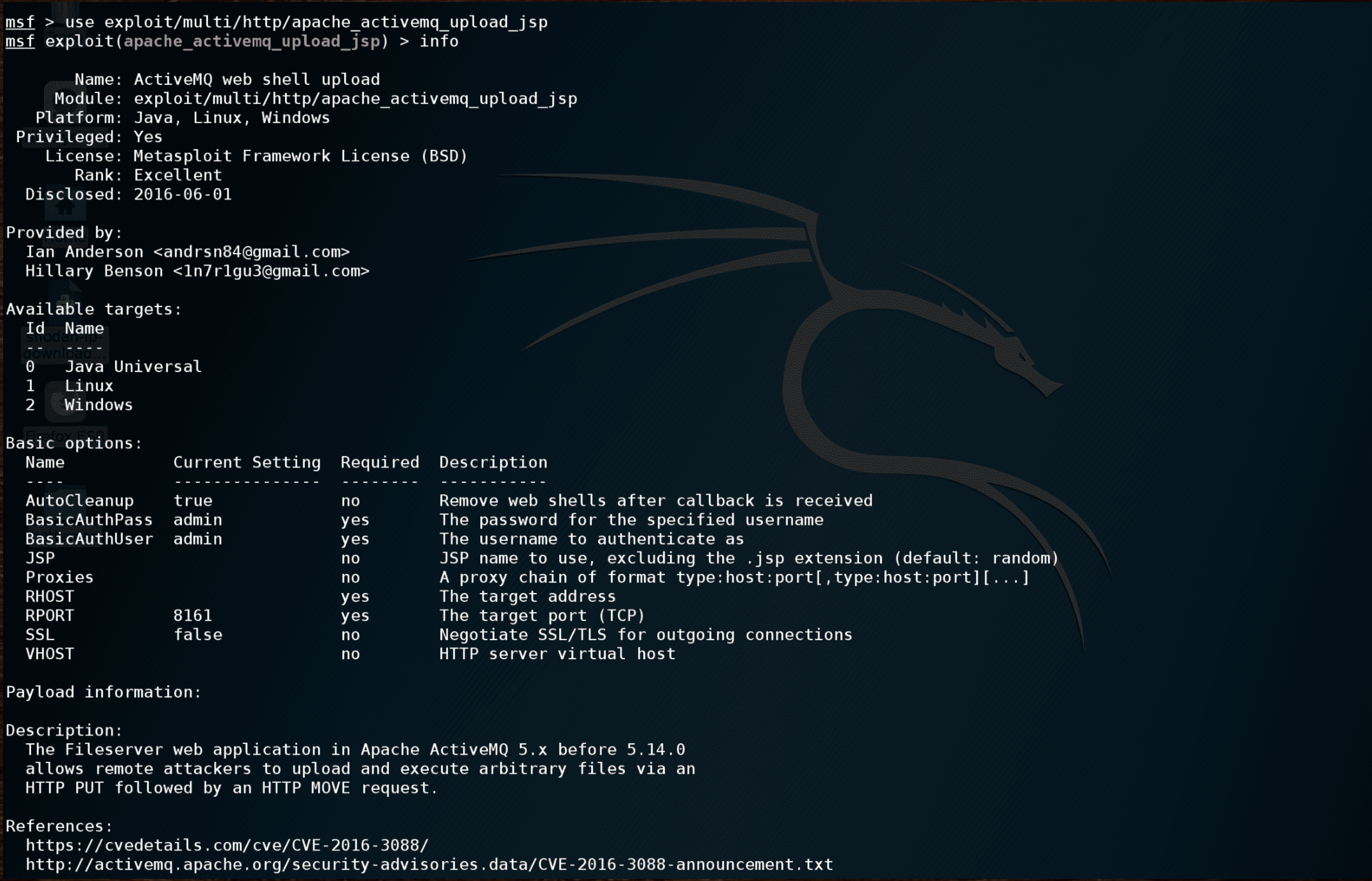


```
▶ Internet Protocol Version 4, Src: 192.168.0.5, Dst: 192.168.0.10
▶ Transmission Control Protocol, Src Port: 55972, Dst Port: 1883, Seq: 1, Ack: 1, Len: 35
▼ MQ Telemetry Transport Protocol
  ▼ Connect Command
    ▶ 0001 0000 = Header Flags: 0x10 (Connect Command)
      Msg Len: 33
      Protocol Name: MQTT
      Version: 4
    ▶ 1100 0010 = Connect Flags: 0xc2
      Keep Alive: 60
      Client ID: Pasknel
      User Name: teste
      Password: teste
```



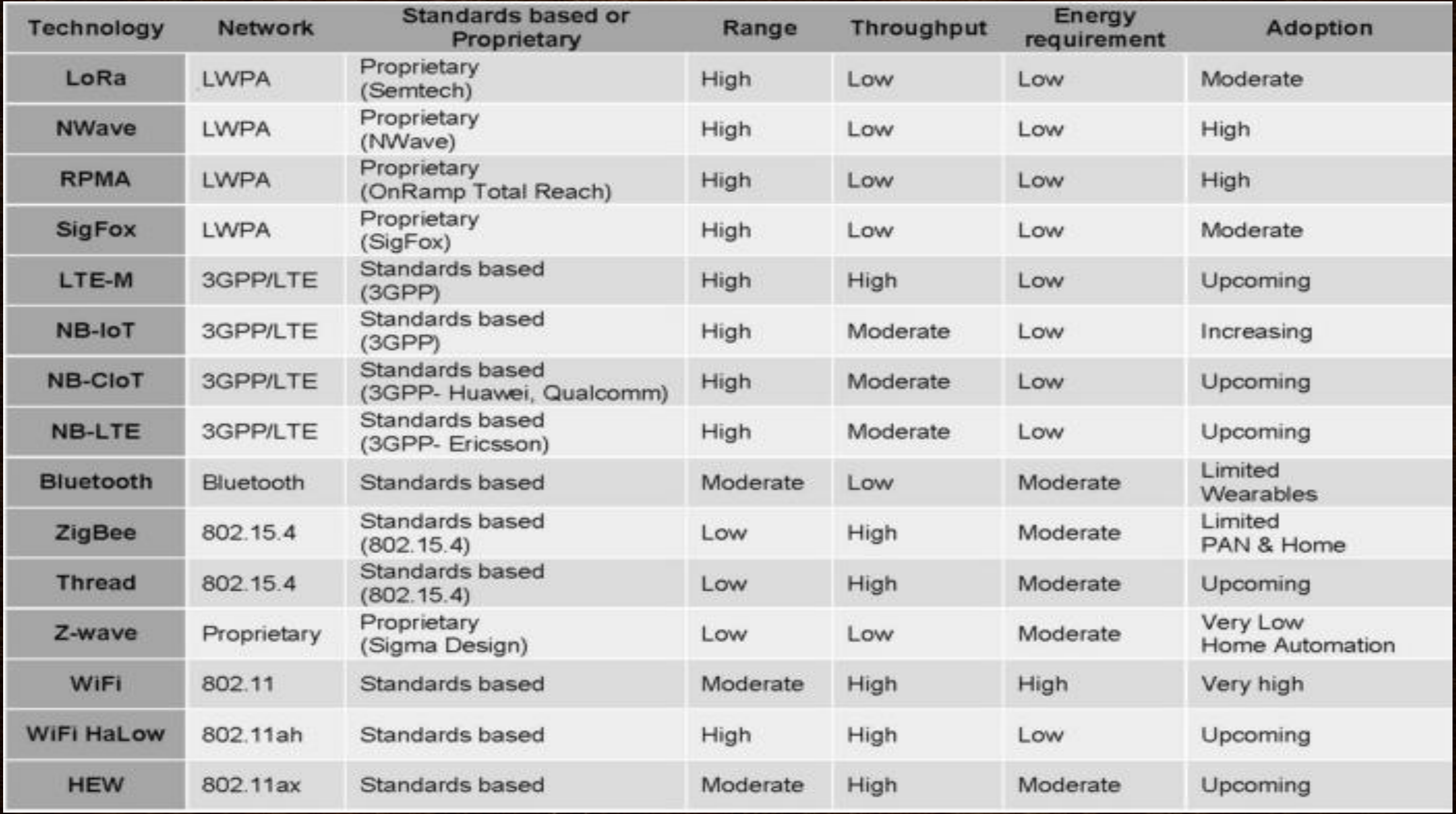
```
MQTT Connection Code: 0

Topics:
ActiveMQ/Advisory/MasterBroker
ActiveMQ/Advisory/Consumer/Topic/#
ActiveMQ/Advisory/Connection
ActiveMQ/Advisory/Producer/Queue/AccountReceiverQueue
ActiveMQ/Advisory/Producer/Queue/AccountReceiverQueue
ActiveMQ/Advisory/Connection
ActiveMQ/Advisory/Connection
ActiveMQ/Advisory/Producer/Queue/AccountReceiverQueue
ActiveMQ/Advisory/Producer/Queue/AccountReceiverQueue
ActiveMQ/Advisory/Connection
ActiveMQ/Advisory/Connection
ActiveMQ/Advisory/Producer/Queue/AccountReceiverQueue
ActiveMQ/Advisory/Producer/Queue/AccountReceiverQueue
ActiveMQ/Advisory/Connection
```

Maybe works



```
msf > use exploit/multi/http/apache_activemq_upload_jsp
msf exploit(apache_activemq_upload_jsp) > info

       Name: ActiveMQ web shell upload
     Module: exploit/multi/http/apache_activemq_upload_jsp
   Platform: Java, Linux, Windows
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2016-06-01

Provided by:
  Ian Anderson <andrsn84@gmail.com>
  Hillary Benson <1n7r1gu3@gmail.com>

Available targets:
  Id  Name
  --  ----
  0   Java Universal
  1   Linux
  2   Windows

Basic options:
  Name           Current Setting  Required  Description
  ----           ---------------  --------  -----------
  AutoCleanup    true             no        Remove web shells after callback is received
  BasicAuthPass  admin            yes       The password for the specified username
  BasicAuthUser  admin            yes       The username to authenticate as
  JSP                             no        JSP name to use, excluding the .jsp extension (default: random)
  Proxies                         no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOST                           yes       The target address
  RPORT          8161             yes       The target port (TCP)
  SSL            false            no        Negotiate SSL/TLS for outgoing connections
  VHOST                           no        HTTP server virtual host

Payload information:

Description:
  The Fileserver web application in Apache ActiveMQ 5.x before 5.14.0
  allows remote attackers to upload and execute arbitrary files via an
  HTTP PUT followed by an HTTP MOVE request.

References:
  https://cvedetails.com/cve/CVE-2016-3088/
  http://activemq.apache.org/security-advisories.data/CVE-2016-3088-announcement.txt
```

2. Radio & Wireless communication Pentesting in IoT



Technology	Network	Standards based or Proprietary	Range	Throughput	Energy requirement	Adoption
LoRa	LWPA	Proprietary (Semtech)	High	Low	Low	Moderate
NWave	LWPA	Proprietary (NWave)	High	Low	Low	High
RPMA	LWPA	Proprietary (OnRamp Total Reach)	High	Low	Low	High
SigFox	LWPA	Proprietary (SigFox)	High	Low	Low	Moderate
LTE-M	3GPP/LTE	Standards based (3GPP)	High	High	Low	Upcoming
NB-IoT	3GPP/LTE	Standards based (3GPP)	High	Moderate	Low	Increasing
NB-CIoT	3GPP/LTE	Standards based (3GPP- Huawei, Qualcomm)	High	Moderate	Low	Upcoming
NB-LTE	3GPP/LTE	Standards based (3GPP- Ericsson)	High	Moderate	Low	Upcoming
Bluetooth	Bluetooth	Standards based	Moderate	Low	Moderate	Limited Wearables
ZigBee	802.15.4	Standards based (802.15.4)	Low	High	Moderate	Limited PAN & Home
Thread	802.15.4	Standards based (802.15.4)	Low	High	Moderate	Upcoming
Z-wave	Proprietary	Proprietary (Sigma Design)	Low	Low	Moderate	Very Low Home Automation
WiFi	802.11	Standards based	Moderate	High	High	Very high
WiFi HaLow	802.11ah	Standards based	High	High	Low	Upcoming
HEW	802.11ax	Standards based	Moderate	High	Moderate	Upcoming

Wi-Fi

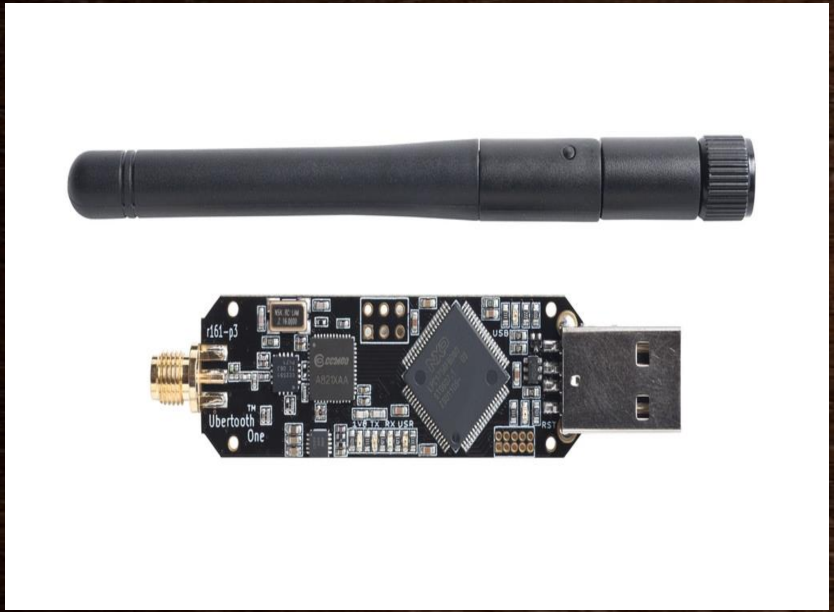
- KRACK vulnerability in WPA2
- MiTM attacks to get the confidential information such as login and keys
- Replay attacks
- DoS attacks to damage the device

BLE

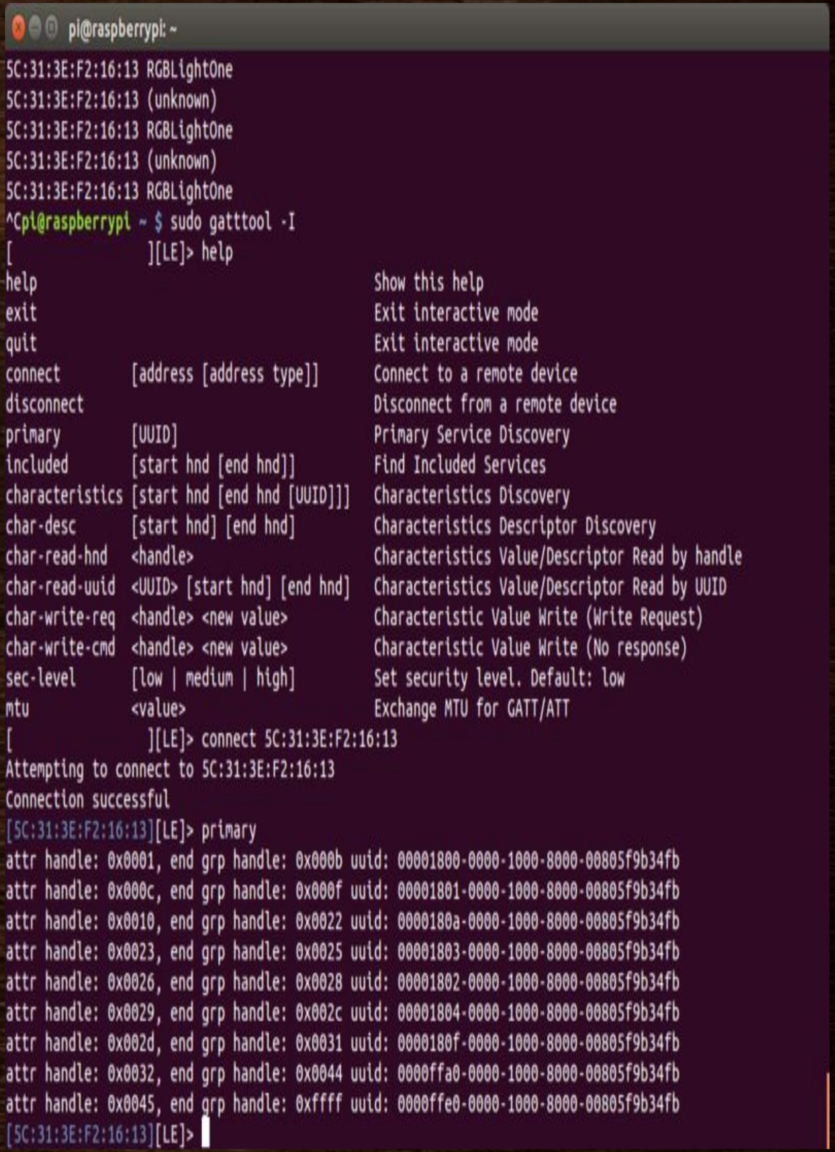
- Bluborne attack which is key pairing attack in BLE devices
- MiTM for reading the information about device and confidential info
- Finding the rx and tx characteristics to communicate or to gain the

BLE Testing:

Ubertooth



Gattool



```
pi@raspberrypi: ~
5C:31:3E:F2:16:13 RGBLightOne
5C:31:3E:F2:16:13 (unknown)
5C:31:3E:F2:16:13 RGBLightOne
5C:31:3E:F2:16:13 (unknown)
5C:31:3E:F2:16:13 RGBLightOne
^Cpi@raspberrypi ~ $ sudo gatttool -I
[                 ][LE]> help
help                                           Show this help
exit                                           Exit interactive mode
quit                                           Exit interactive mode
connect         [address [address type]]       Connect to a remote device
disconnect                                     Disconnect from a remote device
primary         [UUID]                         Primary Service Discovery
included        [start hnd [end hnd]]          Find Included Services
characteristics [start hnd [end hnd [UUID]]]   Characteristics Discovery
char-desc       [start hnd] [end hnd]          Characteristics Descriptor Discovery
char-read-hnd   <handle>                       Characteristics Value/Descriptor Read by handle
char-read-uuid  <UUID> [start hnd] [end hnd]   Characteristics Value/Descriptor Read by UUID
char-write-req  <handle> <new value>           Characteristic Value Write (Write Request)
char-write-cmd  <handle> <new value>           Characteristic Value Write (No response)
sec-level       [low | medium | high]          Set security level. Default: low
mtu             <value>                        Exchange MTU for GATT/ATT
[                 ][LE]> connect 5C:31:3E:F2:16:13
Attempting to connect to 5C:31:3E:F2:16:13
Connection successful
[5C:31:3E:F2:16:13][LE]> primary
attr handle: 0x0001, end grp handle: 0x000b uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x000c, end grp handle: 0x000f uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x0010, end grp handle: 0x0022 uuid: 0000180a-0000-1000-8000-00805f9b34fb
attr handle: 0x0023, end grp handle: 0x0025 uuid: 00001803-0000-1000-8000-00805f9b34fb
attr handle: 0x0026, end grp handle: 0x0028 uuid: 00001802-0000-1000-8000-00805f9b34fb
attr handle: 0x0029, end grp handle: 0x002c uuid: 00001804-0000-1000-8000-00805f9b34fb
attr handle: 0x002d, end grp handle: 0x0031 uuid: 0000180f-0000-1000-8000-00805f9b34fb
attr handle: 0x0032, end grp handle: 0x0044 uuid: 0000ffa0-0000-1000-8000-00805f9b34fb
attr handle: 0x0045, end grp handle: 0xffff uuid: 0000ffe0-0000-1000-8000-00805f9b34fb
[5C:31:3E:F2:16:13][LE]>
```

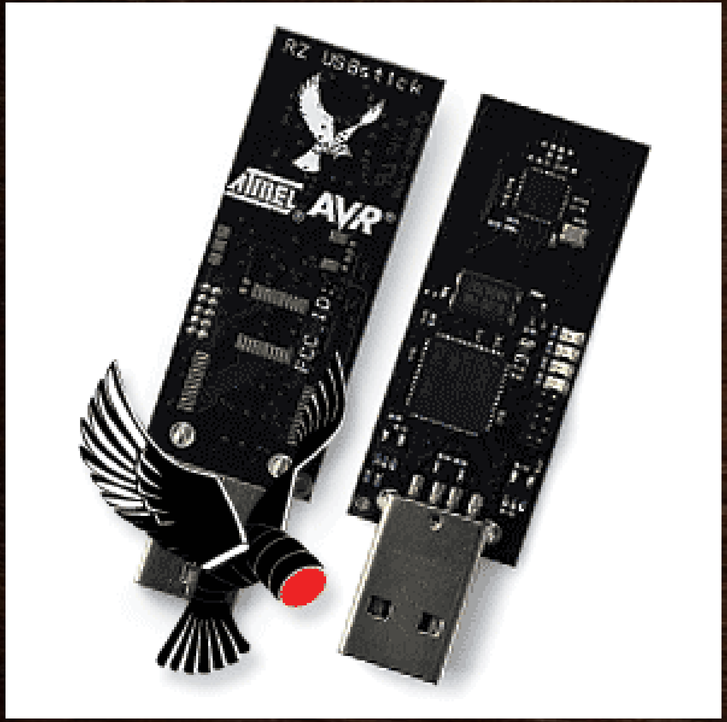
ZigBee

- Network layer security (AES Encryption – AES CCM Mode)
- Application Support Sublayer Security
- Unauthorized access

Z-Wave

- ZShave attack which is recently happened – key pairing value 00000000
- UZB (Zwave USB Disk) attacks

Pentesting Zigbee



+ KillerBee =

Rz Raven USB Stick

Philips Hue

Radio Pentesting..

- Radio waves
- GSM signals
- ADS-B (automatic dependence surveillance – broadcasting)
- Commonly
  - Capturing
  - Extract the text data from the wave file
  - Replay attacks
  - Fake GSM (BTS)

Tools for Radio Pentesting

- Skywave Linux
- Gnuradio companion
- GQRX
- etc

- www.rtl-sdr.com
- https://www.owasp.org/images/2/29/AppSecIL2016_HackingTheIoT-PenTestingRFDevices_ErezMetula.pdf

Devices which we have to use for Radio Pentesting

3. Embedded application and application Pentesting in IoT….

- Embedded application means software or hardware web interface
- Firmware known as application with UI

- Key findings in IoT Embedded application
- Command Injection (Most)
- CSRF(Tentative)
- XSS (firm)
- Etc

Emulating Firmware

- Emulating firmware for pentesting the application
- QEMU , Firmadyne , Firmware analysis toolkit(FAT) etc
- Demo with AttifyOS (https://www.youtube.com/watch?v=mxe7nErtXmw)
- Pentesting demo with Burpsuite

4. Mobile IoT (android , iOS and windows hardware, bootloader)

- Android static and dynamic application pentesting
- Static and dynamic analysis Android
  - Andorid SDK , Android Emulator, MobSF , enjarify , burpsuite. Owasp ZAP
- Static and dynamic analysis iOS
  - Idb, Mob-SF, Burpsuite, ZAP , Xcode tools

Identifying threats

- Eavesdrop on API calls
- Expose sensitive user details
- Delete camera playback feeds
- Change user information's
- Gain access to other user accounts
- Track users in the vendor’s cloud environment

MOBSF

A Heartful Thanks to - ajin Abraham

Demo on fitness app

5. Cloud & API

- Infrastructure as a Service (IaaS): Infrastructure APIs provision raw computing and storage.
- Software as a Service (SaaS): Software or application APIs provision connectivity and interaction with a software suite.
- Platform as a Service (PaaS): Platform APIs provide back-end architecture for building intensive and feature rich applications

Service	IaaS	SaaS	PaaS
Pentesting	Yes	No	Yes

Important tools to pentest cloud

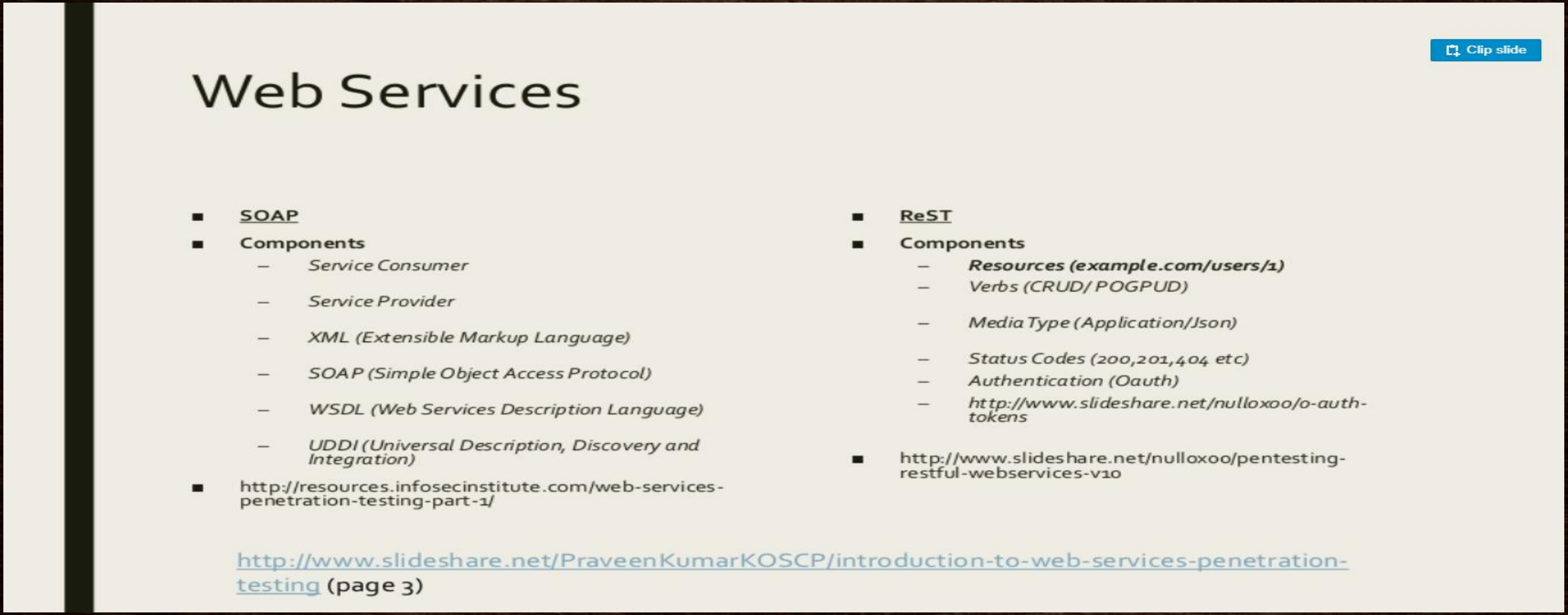
- SOASTA CloudTest:
- LoadStorm:
- BlazeMeter:
- Nexpose:
- AppThwack:

Check List

https://intrinium.com/pen-testing-checklist-for-the-cloud/

API (Application Programmable Interface)

is a set of subroutine definitions, protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication between various components.



Web Services

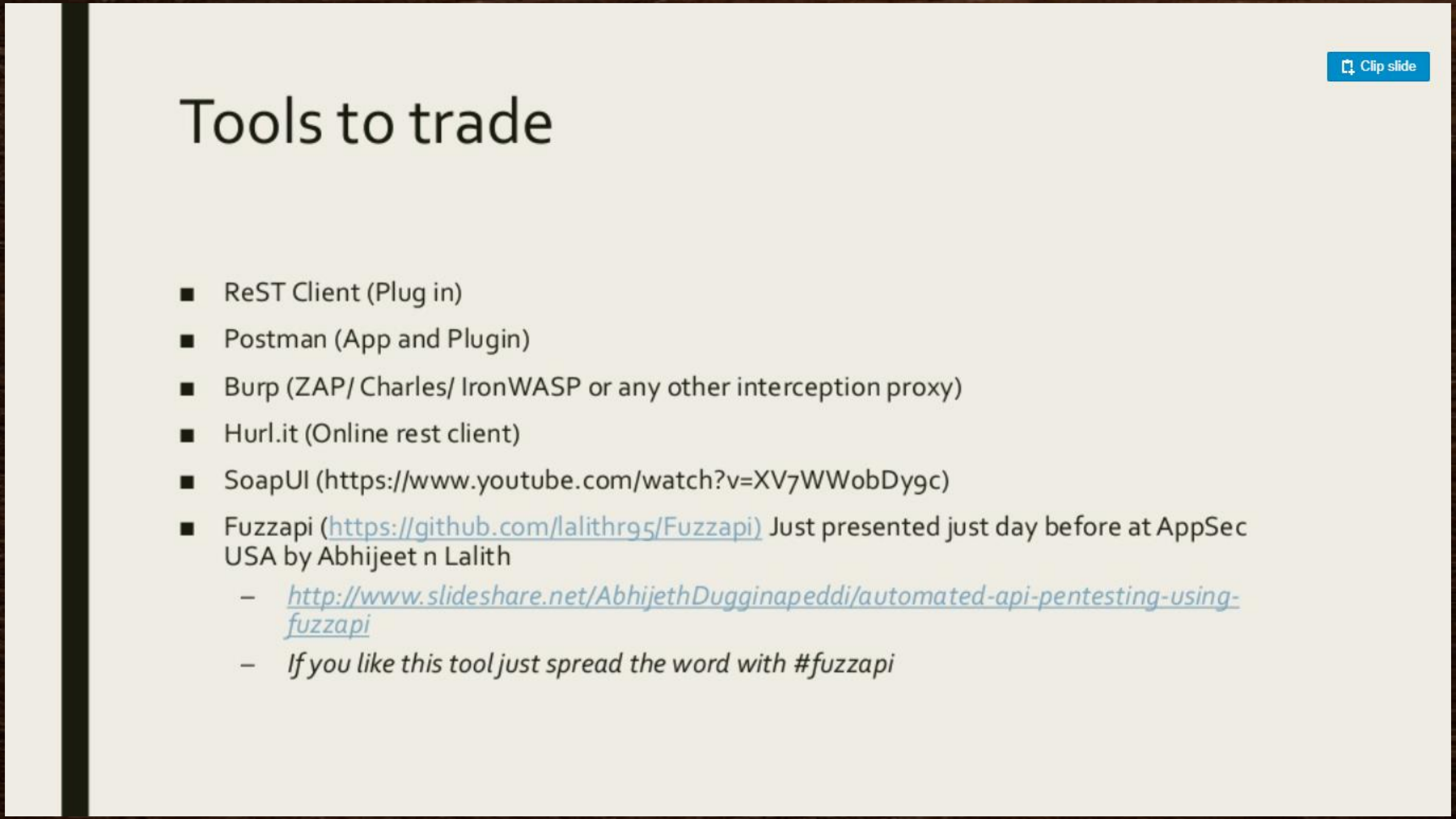
- SOAP
- Components
  - Service Consumer
  - Service Provider
  - XML (Extensible Markup Language)
  - SOAP (Simple Object Access Protocol)
  - WSDL (Web Services Description Language)
  - UDDI (Universal Description, Discovery and Integration)
- http://resources.infosecinstitute.com/web-services-penetration-testing-part-1/

- ReST
- Components
  - Resources (example.com/users/1)
  - Verbs (CRUD/ POGPUD)
  - Media Type (Application/Json)
  - Status Codes (200,201,404 etc)
  - Authentication (Oauth)
  - http://www.slideshare.net/nulloxoo/o-auth-tokens
- http://www.slideshare.net/nulloxoo/pentesting-restful-webservices-v10

http://www.slideshare.net/PraveenKumarKOSCP/introduction-to-web-services-penetration-testing (page 3)

https://www.slideshare.net/NutanKumarPanda/pentesting-rest-api

Tools to Use API Pentesting



Clip slide

Tools to trade

- ReST Client (Plug in)
- Postman (App and Plugin)
- Burp (ZAP/ Charles/ IronWASP or any other interception proxy)
- Hurl.it (Online rest client)
- SoapUI (https://www.youtube.com/watch?v=XV7WWobDy9c)
- Fuzzapi (https://github.com/lalithr95/Fuzzapi) Just presented just day before at AppSec USA by Abhijeet n Lalith
  - *http://www.slideshare.net/AbhijethDugginapeddi/automated-api-pentesting-using-fuzzapi*
  - *If you like this tool just spread the word with #fuzzapi*

https://www.slideshare.net/NutanKumarPanda/pentesting-rest-api

6. Firmware analysis

- Firmware is software of hardware
- Dump from vendor website , sniff the while updating , capture by OTA, pull from the hardware
- Firmware filesystems are consisting the data of hardcoded and sensitive
- Commonly we check for
  - Architecture
  - Filesystem
  - Hardcoded information like passwords or token info or certificate info or remote connect ip address or database addresses
  - Reversing and buffer over flow

Firmware Analysis with tools

- Binwalk – extracting and check the information
- Readelf – reading the elf(executable and likable format) file
- Strings – to print readable characters
- Hexdump – hex analysis on firmware
- dd – copy or separating required data from the firmware
- Radare2 – reverse engineering (required ROP knowledge)
- IDA Pro – reverse engineering and fuzzing (required assembly and em c and c++)
- etc

Content of Firmware security 101

1. what is firmware
2. dig deep into firmware
3. firmware importance
4. how many ways we can obtain the firmware
5. firmware emulation
6. finding the bugs in embedded application
7. firmware reversing
    - i. extraction
    - ii. identifying the architecture
    - iii. finding the key info
    - iv. looking into hardcoded data
    - v. backdooring the file
    - vi. reverse engineering

What is a firmware..?

Firmware is a software of hardware

(Or)

permanent software programmed into a read-only memory.

- Mainly firmware consists
  - Low level languages programmed
  - File systems
  - Root Directory
  - Compression
  - Application data files
  - Architecture information
  - Busybox (important)
  - Encrypted data

Filesystems Type..?

- SquashFS
- JFFS
- JFFS2
- CPIO
- YAFFS
- UBIFS
- XFS

- These are commonly used in Firmware

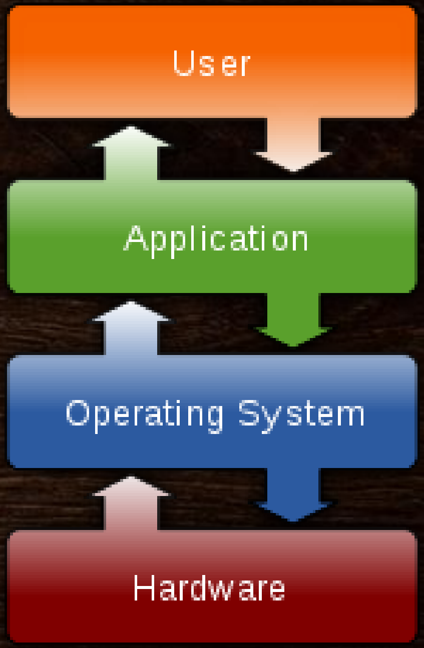


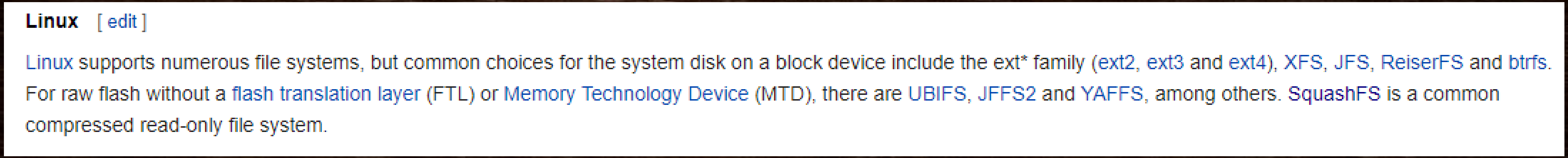
Image Source : https://upload.wikimedia.org/wikipedia/commons/thumb/e/e1/Operating_system_placement.svg/165px-Operating_system_placement.svg.png

Detailed in Filesystem..

SquashFS:

Squashfs is a compressed read-only file system for Linux. Squashfs compresses files, inodes and directories, and supports block sizes up to 1 MB for greater compression. Several compression algorithms are supported. Squashfs is also the name of free software, licensed under the GPL, for accessing Squashfs filesystems.

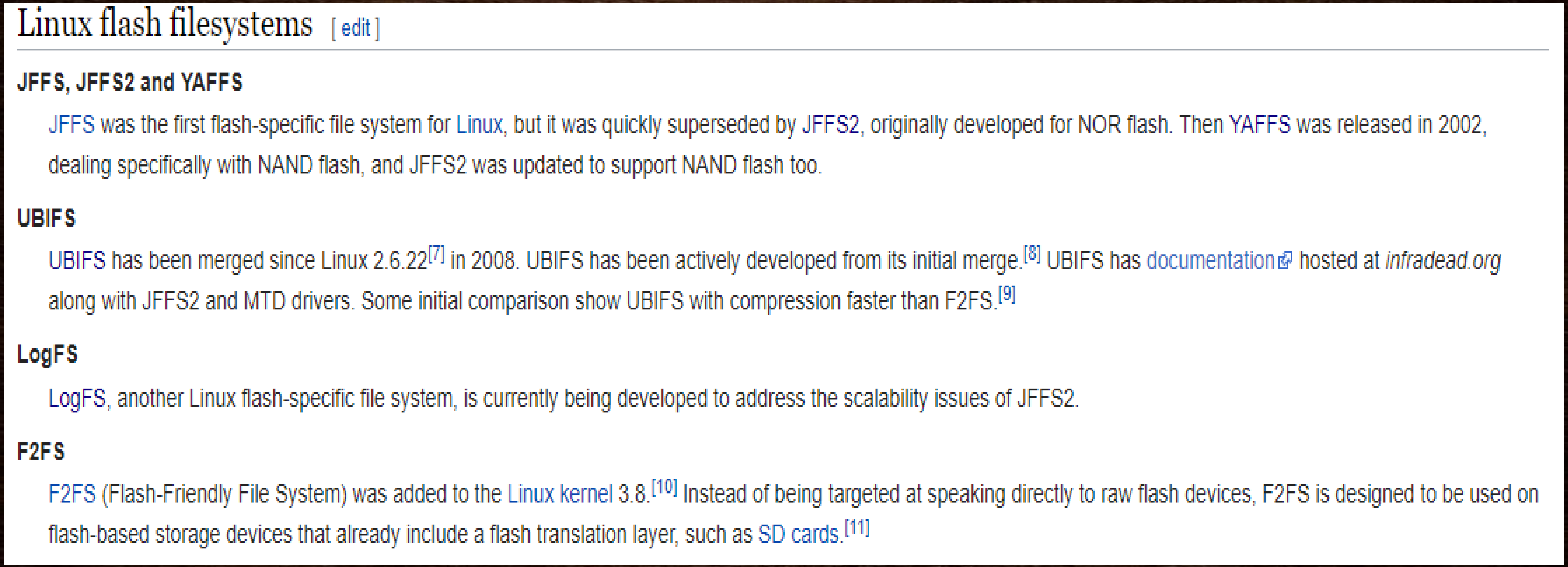
Squashfs is intended for general read-only file-system use and in constrained block-device memory systems (e.g. embedded systems) where low overhead is needed.



Linux [edit]

Linux supports numerous file systems, but common choices for the system disk on a block device include the ext* family (ext2, ext3 and ext4), XFS, JFS, ReiserFS and btrfs. For raw flash without a flash translation layer (FTL) or Memory Technology Device (MTD), there are UBIFS, JFFS2 and YAFFS, among others. SquashFS is a common compressed read-only file system.

Detailed with flashsystem ..



Linux flash filesystems [edit]

JFFS, JFFS2 and YAFFS

JFFS was the first flash-specific file system for Linux, but it was quickly superseded by JFFS2, originally developed for NOR flash. Then YAFFS was released in 2002, dealing specifically with NAND flash, and JFFS2 was updated to support NAND flash too.

UBIFS

UBIFS has been merged since Linux 2.6.22[7] in 2008. UBIFS has been actively developed from its initial merge.[8] UBIFS has documentation hosted at *infradead.org* along with JFFS2 and MTD drivers. Some initial comparison show UBIFS with compression faster than F2FS.[9]

LogFS

LogFS, another Linux flash-specific file system, is currently being developed to address the scalability issues of JFFS2.

F2FS

F2FS (Flash-Friendly File System) was added to the Linux kernel 3.8.[10] Instead of being targeted at speaking directly to raw flash devices, F2FS is designed to be used on flash-based storage devices that already include a flash translation layer, such as SD cards.[11]

Root Directory

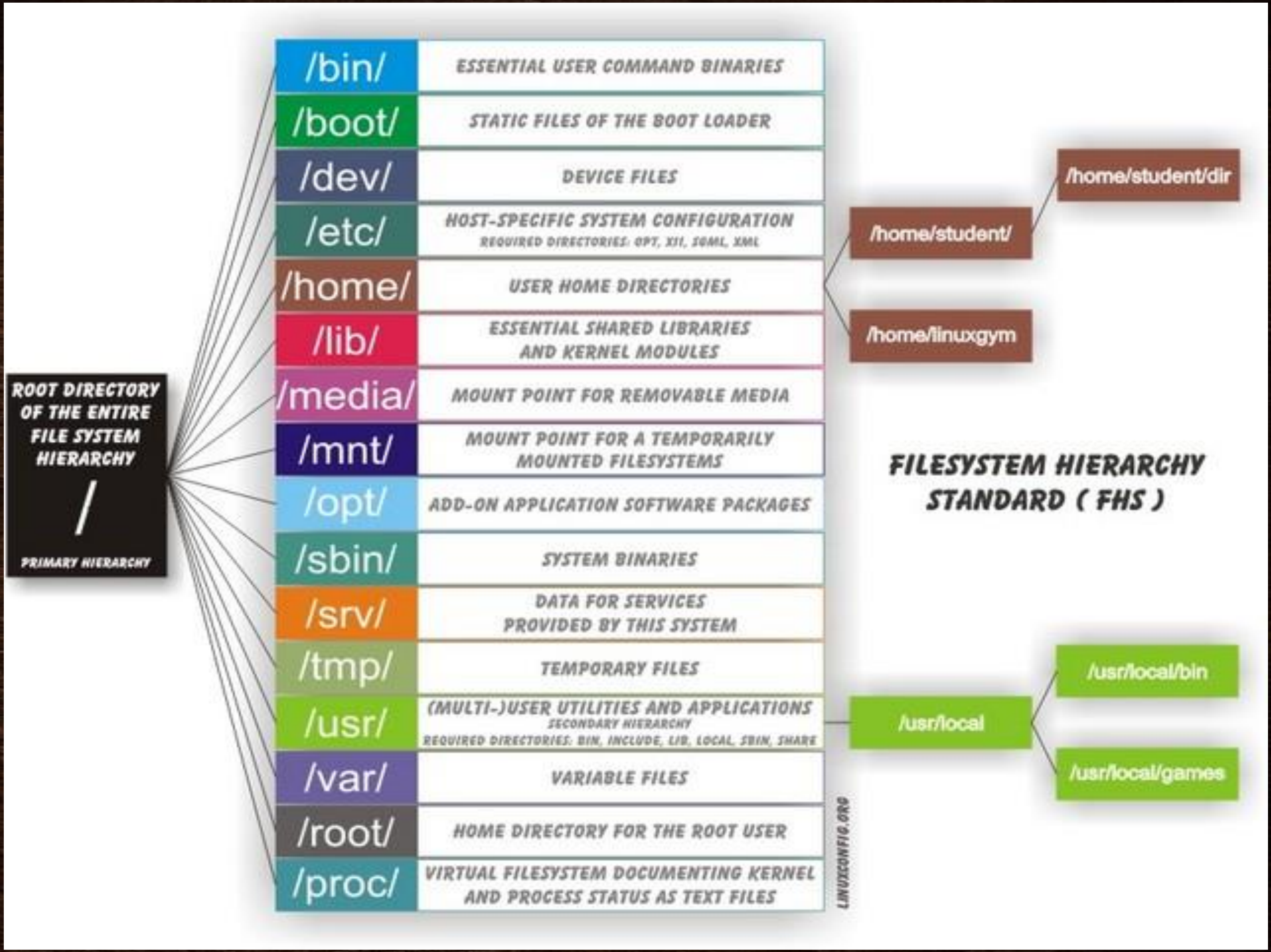
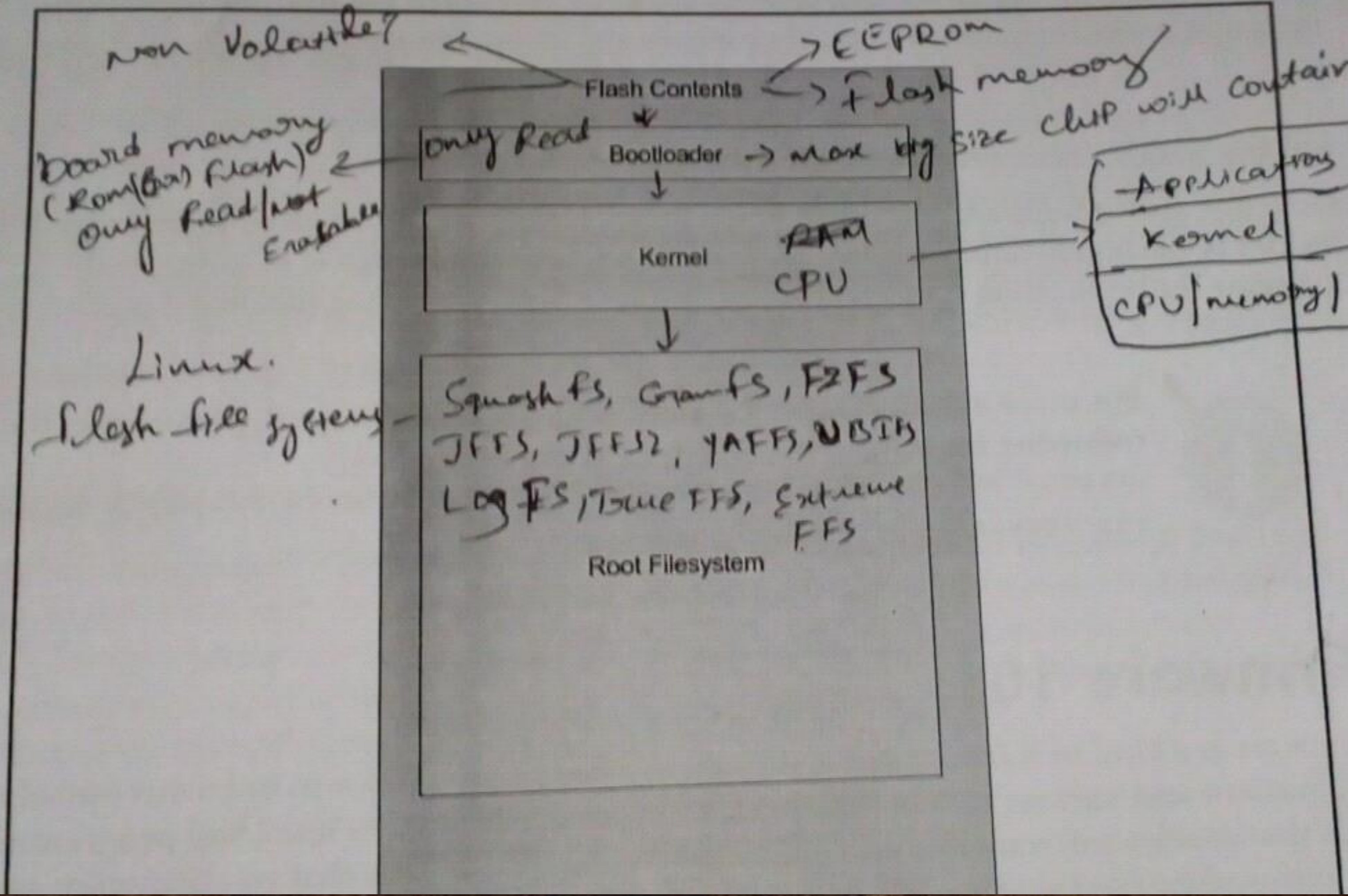

ROOT DIRECTORY OF THE ENTIRE FILE SYSTEM HIERARCHY
/
PRIMARY HIERARCHY
/bin/
ESSENTIAL USER COMMAND BINARIES
/boot/
STATIC FILES OF THE BOOT LOADER
/dev/
DEVICE FILES
/etc/
HOST-SPECIFIC SYSTEM CONFIGURATION
REQUIRED DIRECTORIES: OPT, X11, SGML, XML
/home/
USER HOME DIRECTORIES
/lib/
ESSENTIAL SHARED LIBRARIES AND KERNEL MODULES
/media/
MOUNT POINT FOR REMOVABLE MEDIA
/mnt/
MOUNT POINT FOR A TEMPORARILY MOUNTED FILESYSTEMS
/opt/
ADD-ON APPLICATION SOFTWARE PACKAGES
/sbin/
SYSTEM BINARIES
/srv/
DATA FOR SERVICES PROVIDED BY THIS SYSTEM
/tmp/
TEMPORARY FILES
/usr/
(MULTI-)USER UTILITIES AND APPLICATIONS
SECONDARY HIERARCHY
REQUIRED DIRECTORIES: BIN, INCLUDE, LIB, LOCAL, SBIN, SHARE
/var/
VARIABLE FILES
/root/
HOME DIRECTORY FOR THE ROOT USER
/proc/
VIRTUAL FILESYSTEM DOCUMENTING KERNEL AND PROCESS STATUS AS TEXT FILES
/home/student/
/home/student/dir
/home/linuxgym
FILESYSTEM HIERARCHY STANDARD (FHS)
/usr/local
/usr/local/bin
/usr/local/games
LINUXCONFIG.ORG

Image Source: https://www.gocit.vn/wp-content/uploads/2015/09/linux-file-system.jpg

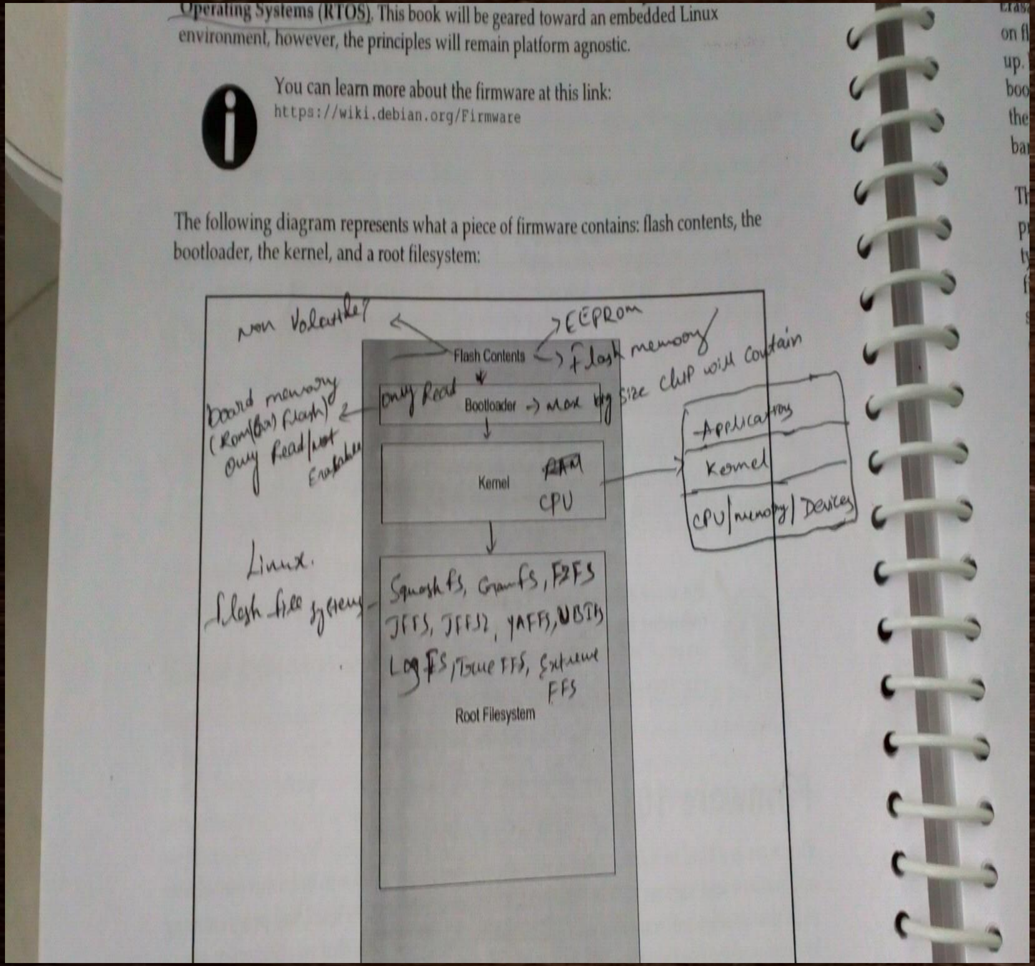
Operating Systems (RTOS). This book will be geared toward an embedded Linux environment, however, the principles will remain platform agnostic.

You can learn more about the firmware at this link:
https://wiki.debian.org/Firmware

The following diagram represents what a piece of firmware contains: flash contents, the bootloader, the kernel, and a root filesystem:



Firmware Importance ..



Operating Systems (RTOS). This book will be geared toward an embedded Linux environment, however, the principles will remain platform agnostic.

You can learn more about the firmware at this link: https://wiki.debian.org/Firmware

The following diagram represents what a piece of firmware contains: flash contents, the bootloader, the kernel, and a root filesystem:

non Volatile?
EEPROM
Flash Contents
flash memory
Board memory (Rom(bios) flash)
Only Read/not Erasable
Only Read
Bootloader
max big size chip will contain
Application
Kernel
CPU|memory|Devices
Kernel
RAM
CPU
Linux.
flash file system
Squash fs, Cram fs, F2FS
JFFS, JFFS2, YAFFS, UBIFS
Log FS, True FFS, Extreme FFS
Root Filesystem

- Firmware working for running the hardware device to bootup
- Firmware where we can store the most important data like credentials and certificates
- When back door is injected for firmware attacker will take always reverse connection

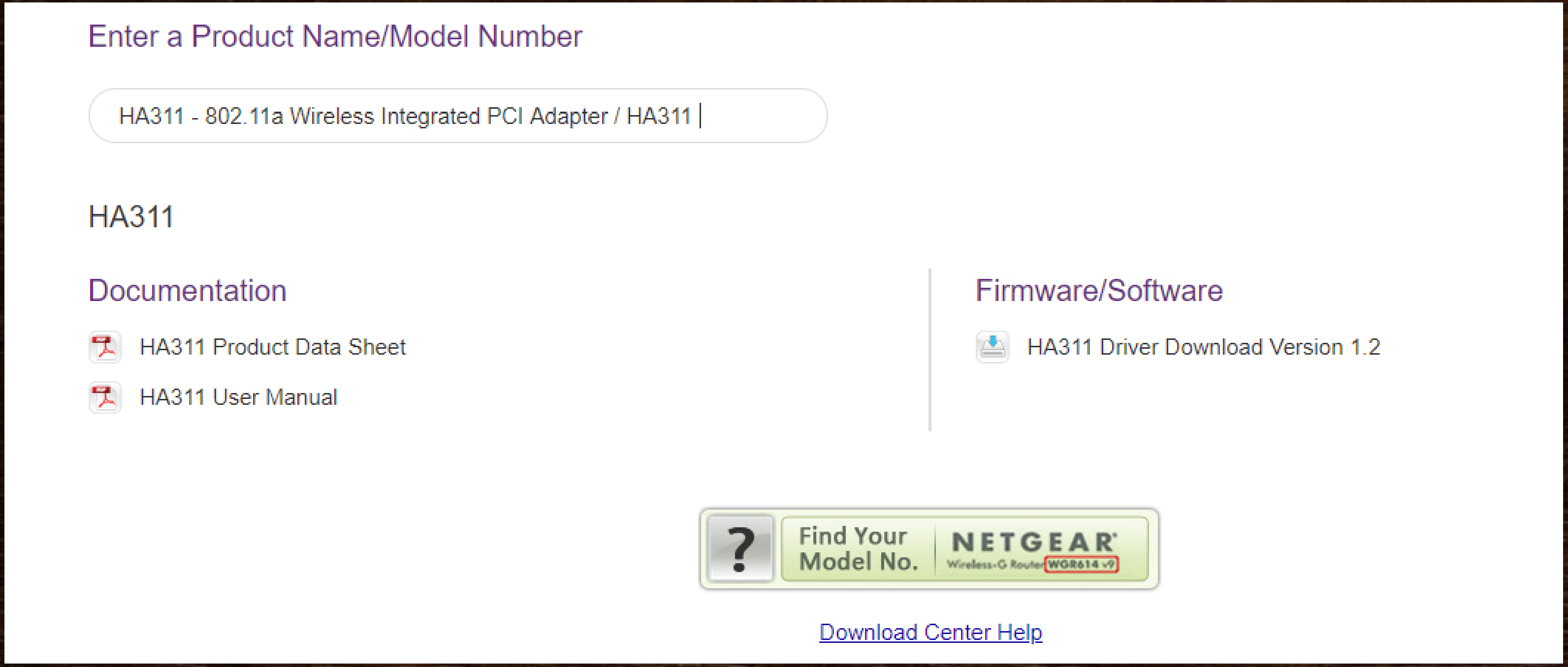
Setting UP Lab

- Use Attify OS
  - https://github.com/adi0x90/attifyos
- Kali Linux
  - https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-hyperv-image-download/
- Ubuntu is best for IoT(LTS)
  - https://www.ubuntu.com/download/desktop/thank-you?version=18.04&architecture=amd64

How many ways we can obtain firmware..

- Downloading from vendor websites
- Capturing the firmware data while updating
- Extracting form the hardware
- Social Engineering

Downloading from the vendor site..


Enter a Product Name/Model Number

HA311 - 802.11a Wireless Integrated PCI Adapter / HA311

HA311

Documentation

HA311 Product Data Sheet

HA311 User Manual

Firmware/Software

HA311 Driver Download Version 1.2

Download Center Help

Demo

- Capturing the firmware data while updating

Tools to used

1. Wireshark
2. Ettercap
3. Device
4. Internet
5. Host as a Linux OS
6. IP tables

Explaining Topic

Explaining the Topic

- Extracting from the hardware
  - Debuggers – Buspirate, Shikra, Jtag,
  - Connectors -- UART, Spi, I2C connectors
  - EEPROM Chip Reader - CH341A
  - http://iotpentest.com/category/firmware/page/2/
- Social Engineering
  - Need a telephone
  - Company email id
  - Creating a valid reason

Firmware Emulation…

One of the challenging task now a days , emulating the firmware

1. Download Attify OS
2. Use FAT (Firmware analysis Toolkit)
3. Qemu also one of the best Emulation tools for all
4. After Getting Web Interface start pentesting it

Firmware Reverse Engineering

i. extraction and analyzing
ii. identifying the architecture
iii. finding the key info
iv. looking into hardcoded data
v. backdooring the file
vi. reverse engineering

Requirements

Tools

1. Binwalk
2. Attify OS
3. Kali Linux
4. Qemu
5. dd
6. Angr
7. Hexedit
8. Hexdump
9. IDA pro
10. Radare2
11. Firmwalker
12. etc

Languages learn to pentest

1. ARM
2. MIPS
3. Assembly
4. C, C ++
5. Python
6. ROP

What need to looking for in the firmware

- Looking for file return data
- Looking for Signatures
- Checking for printable data
- Identify firmware build
- Filesystem
- Hardcoded info
- Authorized key info
- "etc/passwd" and "etc/shadow"
- "etc/ssl"
- grep -rnw '/path/to/somewhere/' -e "pattern" like password, admin, root, etc.
- find . -name '*.conf' and other file types like *.pem, *.crt, *.cfg, .sh, .bin, etc.

Extracting && analyzing the firmware..

- If file downloaded as Zip Unzip for the binary
- Use binwalk to extract the firmware
- Analyze the binary with the binwalk

Useful commands

-B, --signature

-A, --opcodes

-Y, --disasm

-E, --entropy

-Mre ,

https://github.com/ReFirmLabs/binwalk/wiki/Usage

identifying the architecture

Firmware architecture mainly

1. MIPS
2. ARM

Demo

finding the key info

Certification information
Hardcoded url
Api information
IP information
Telnet and SNMP info

Demo

looking into hardcoded data

Passwords and Api information mainly

/etc/passwd
/etc/shadow
/etc/ssl
/ proc/
/sbin/

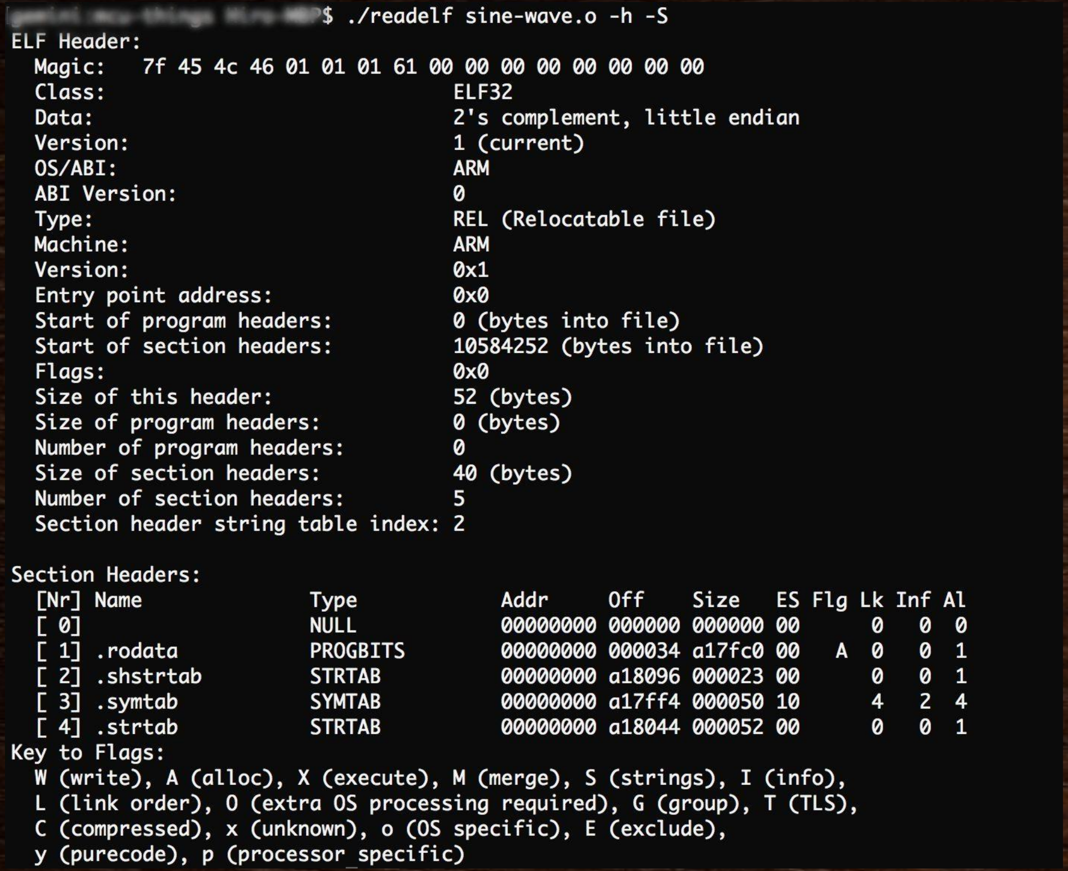
Demo

Reverse engineering firmware

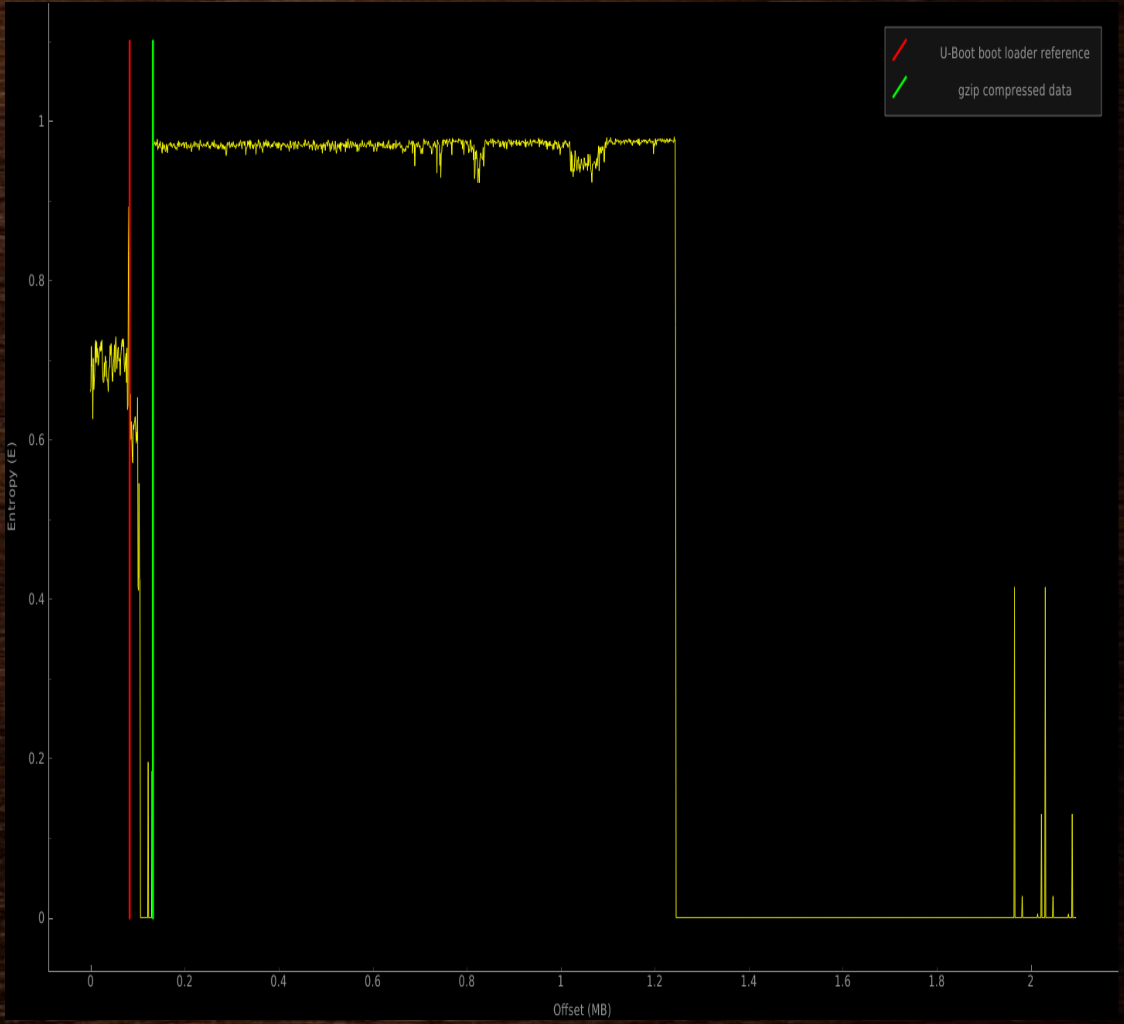
Objdump
(http://www.tutorialspoint.com/unix_commands/objdump.htm)

Radare2 basics
(https://radare.gitbooks.io/radare2book/content/introduction/basic_usage.html)

ODA
(Online Disassembler(https://onlinedisassembler.com/static/home/index.html))



```
[redacted]$ ./readelf sine-wave.o -h -S
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 61 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            ARM
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          10584252 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           40 (bytes)
  Number of section headers:         5
  Section header string table index: 2

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .rodata           PROGBITS        00000000 000034 a17fc0 00   A  0   0  1
  [ 2] .shstrtab         STRTAB          00000000 a18096 000023 00      0   0  1
  [ 3] .symtab           SYMTAB          00000000 a17ff4 000050 10      4   2  4
  [ 4] .strtab           STRTAB          00000000 a18044 000052 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  y (purecode), p (processor specific)
```

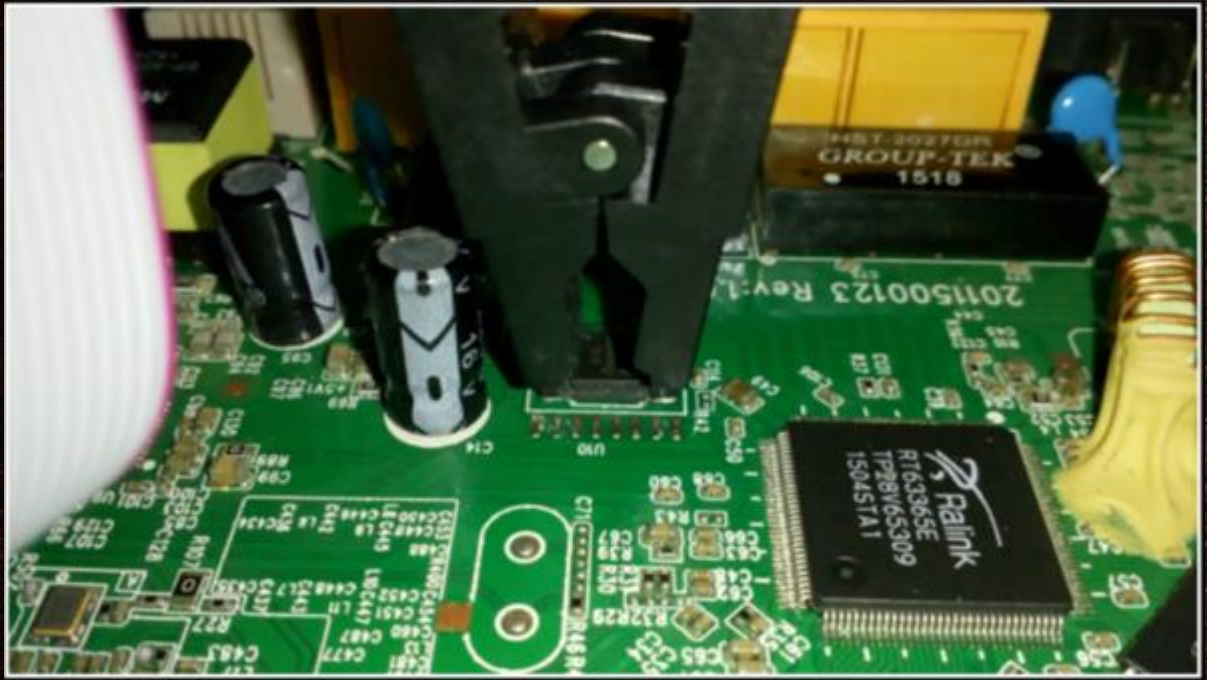


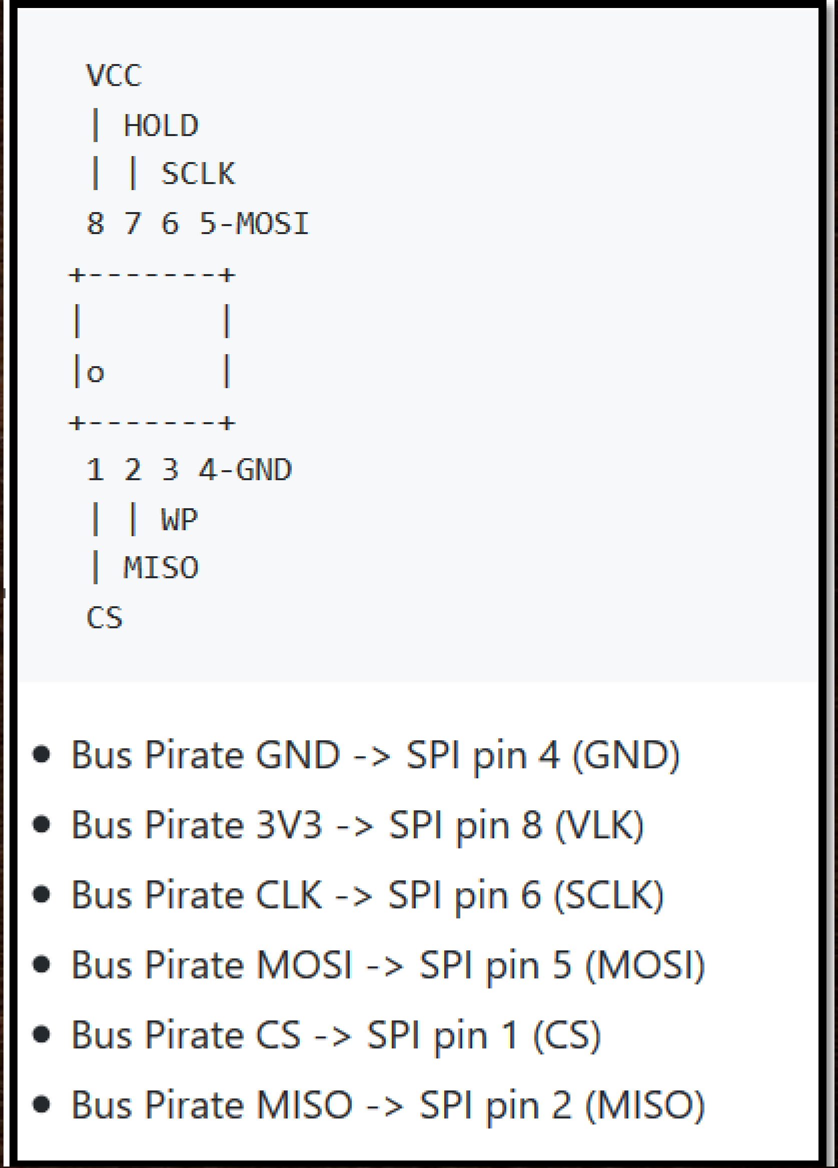
7. Hardware pentesting 101

- One of my favorite part
- Need to know about basic of electronics like resistor , diode and chips
- And screw types and PCB design understanding

- Commonly
  - Spi , i2c and Uart , JTAG will required communicating
    - Dumping and reading the data
    - Getting the shell and glitching attacks
    - Analyzing the binaries after we got shell or dump the data
    - Serial port and USB port attacks

SPI and I2C connection

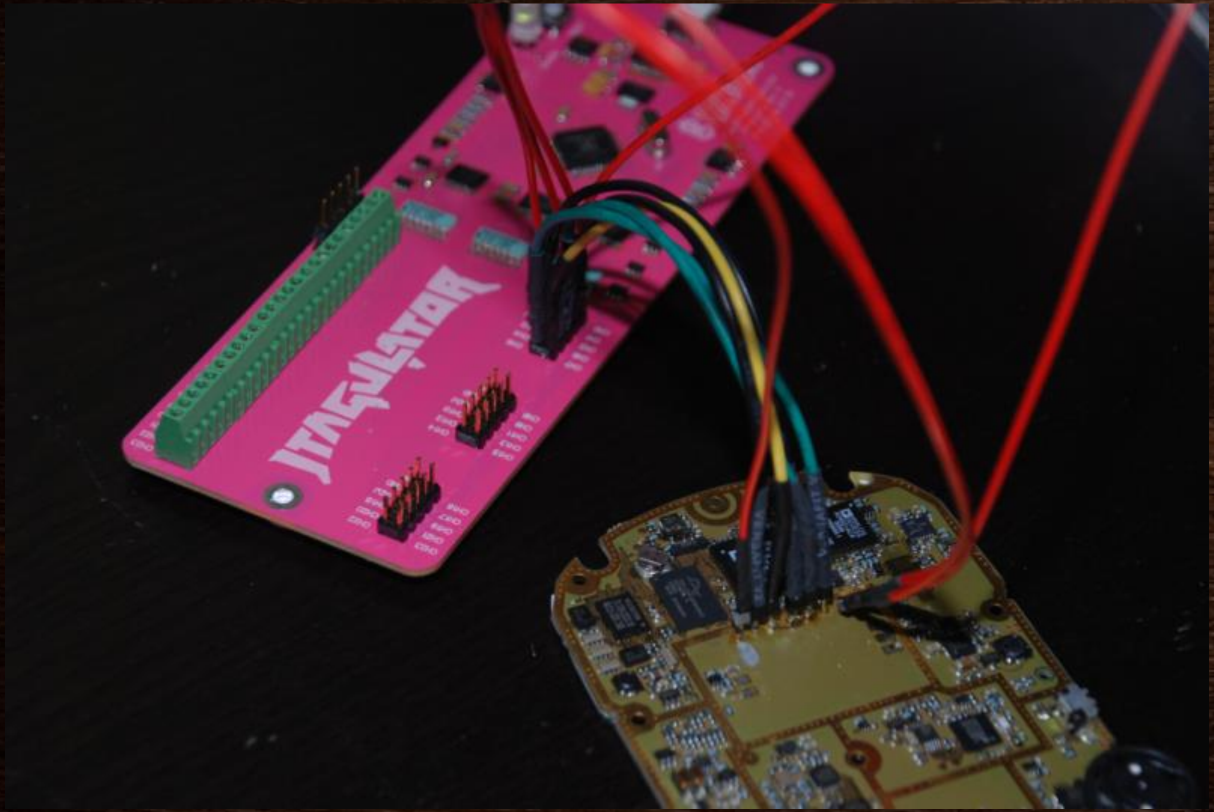


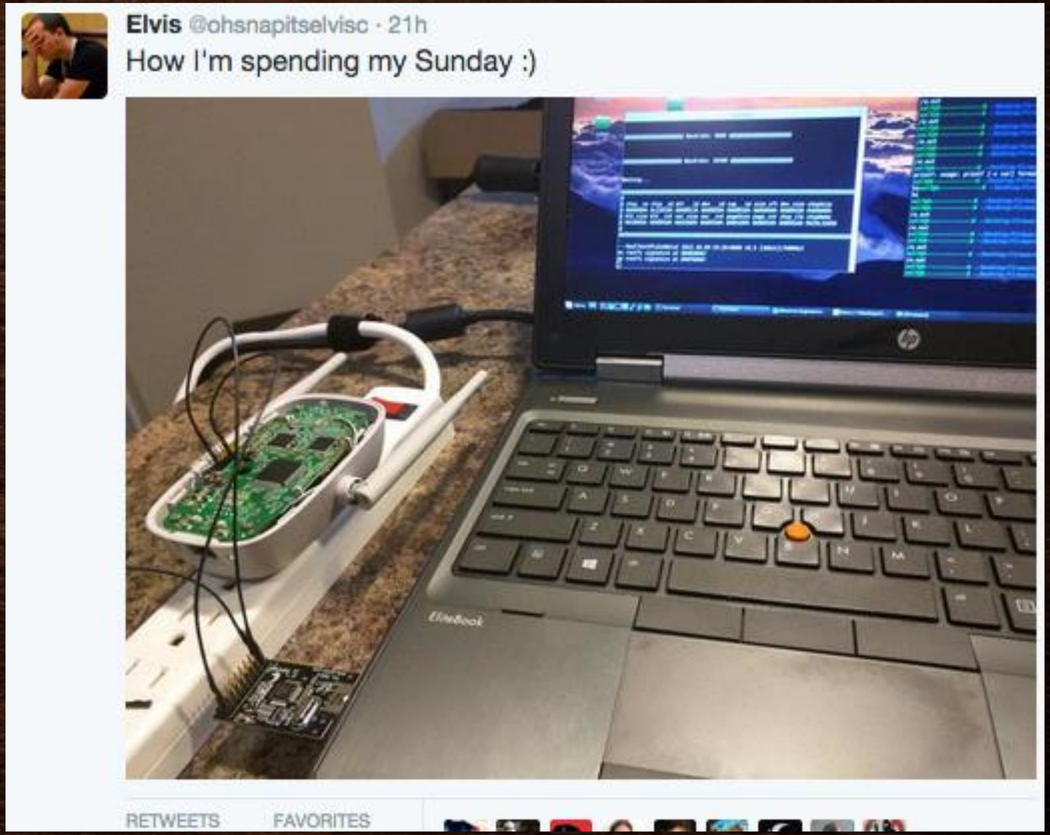


```
 VCC
 | HOLD
 | | SCLK
 8 7 6 5-MOSI
+-------+
|       |
|o      |
+-------+
 1 2 3 4-GND
 | | WP
 | MISO
 CS
```

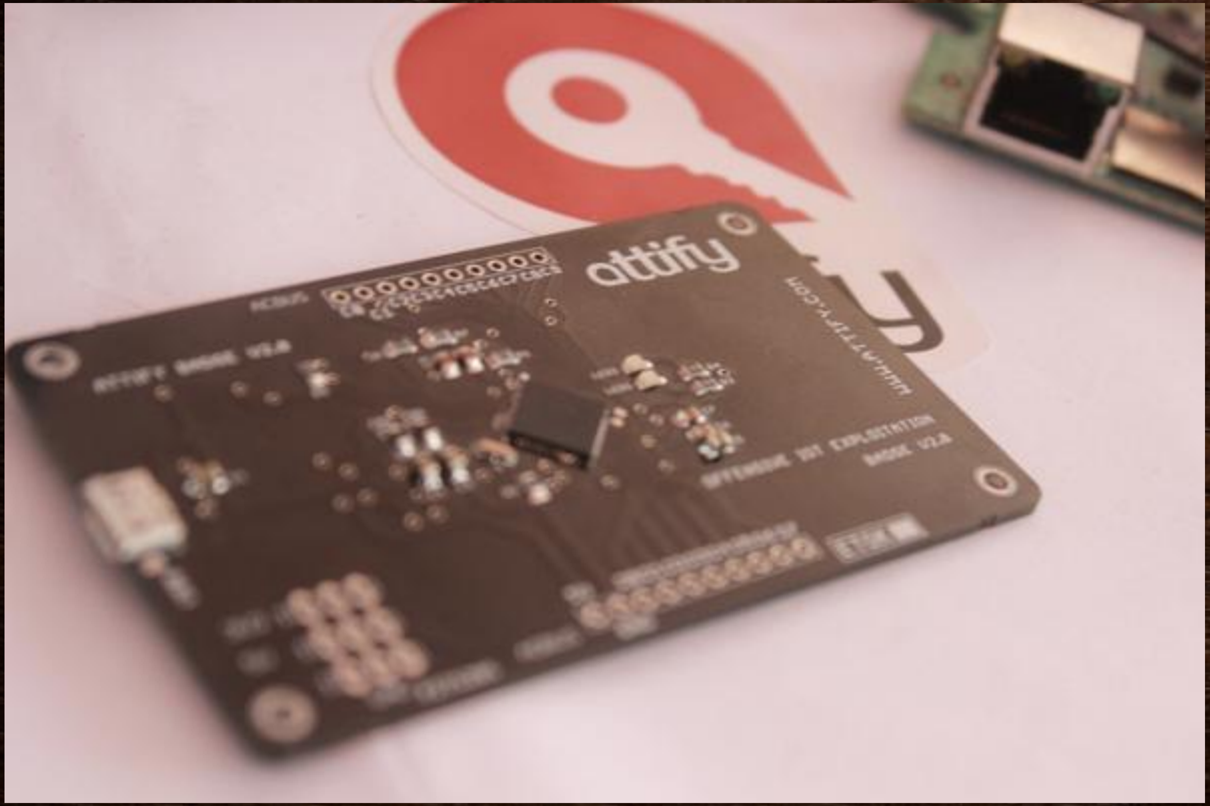
- Bus Pirate GND -> SPI pin 4 (GND)
- Bus Pirate 3V3 -> SPI pin 8 (VLK)
- Bus Pirate CLK -> SPI pin 6 (SCLK)
- Bus Pirate MOSI -> SPI pin 5 (MOSI)
- Bus Pirate CS -> SPI pin 1 (CS)
- Bus Pirate MISO -> SPI pin 2 (MISO)

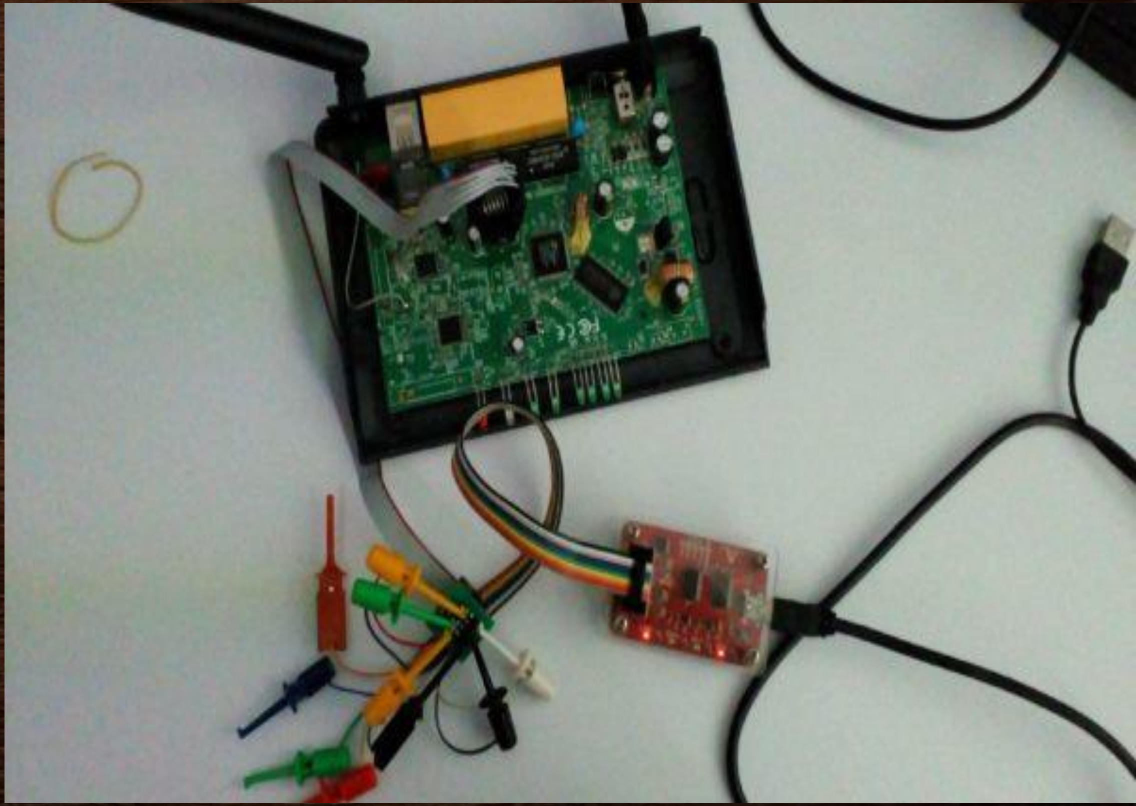
Jtagulator connection and shikra




Elvis @ohsnapitselvisc · 21h
How I'm spending my Sunday :)

Attify badge and buspirate





Security Practices to remediate the attacks of IoT

Network Level

- Close the unnecessary ports which is not required like telnet and ftp , ssh
- Maintain complex password with authentication Key certificate
- Remove un necessary services like UpNP

IoT Hardware security practices

- Check The Uncommon Screws types availability
- Anti Tampering
- Side Channel Attacks
- Encrypting Communication data and TPM

Thank You

contact info : veeru.rockstar249@gmail.com