TACFAM DB-120WL

PWN

HADESS

WWW.HADESS.IO

Executive Summary

This executive summary provides an overview of a critical code execution vulnerability discovered in the TACFAM DB-120WL networking device. The vulnerability allows remote attackers to execute arbitrary code on the device, potentially compromising the entire network. The analysis covers various aspects of the vulnerability, including technical details, exploitation process, and potential impact. The following key points are highlighted:

Vulnerability Overview: The TACFAM DB-120WL device, a popular networking solution, is affected by a severe code execution vulnerability in its firmware. This vulnerability enables remote attackers to inject arbitrary code into the device's memory, leading to unauthorized access and compromising the entire network infrastructure.

Technical Analysis: The analysis provides a detailed examination of UART communication, including its overview, parameters, and pin configuration. It also explains the process of connecting UART to USB using TTL converters. Additionally, the analysis explores the vulnerable areas within the device's firmware, such as XSS vulnerabilities, misconfigurations, authentication bypass, clear text password storage, CSRF to RCE, and buffer overflow.

Exploitation Process: The exploitation process involves crafting specially crafted requests to the TACFAM DB-120WL device to exploit its insecure input processing. By injecting malicious code into the device's memory, attackers gain control over its functionalities, leading to unauthorized access, data breaches, and complete device compromise.

Impact: The code execution vulnerability in the TACFAM DB-120WL device poses significant risks, including unauthorized access, data breaches, and compromise of sensitive information. Attackers can manipulate network configurations, access restricted functionalities, and retrieve sensitive data stored on the device.

Cover by **Michael Gaslenko**

hadess_security

01

Advisory

The TACFAM DB-120WL device is a popular networking solution widely used for wireless connectivity in homes and businesses. However, a severe code execution vulnerability has been identified in its firmware. This analysis aims to provide a detailed understanding of the vulnerability, its implications, and recommend effective countermeasures.

Abstract

This comprehensive technical analysis delves into a critical code execution vulnerability discovered in the TACFAM DB-120WL networking device. The vulnerability allows remote attackers to execute arbitrary code on the device, potentially compromising the entire network. This article provides an in-depth examination of the vulnerability, its impact, and suggests potential mitigations to protect users and organizations.

Introduction

The TACFAM DB-120WL device is a popular networking solution widely used for wireless connectivity in homes and businesses. However, a severe code execution vulnerability has been identified in its firmware. This analysis aims to provide a detailed understanding of the vulnerability, its implications, and recommend effective countermeasures.

02

Technical Analysis

The technical analysis provided covers the following topics:

- UART Communication: This section provides an overview of UART communication, including its components, parameters, and pin configuration. It also explains how to connect UART to a USB port using TTL converters.
- TACFAM DB-120WL Code Execution Vulnerability: This section discusses a critical code execution vulnerability in the TACFAM DB-120WL networking device. It explains how remote attackers can exploit this vulnerability to execute arbitrary code on the device, potentially compromising the entire network.
- Exploitation Process: This section describes the process through which an attacker can exploit the code execution vulnerability in the TACFAM DB-120WL device by sending specially crafted requests to manipulate the device's memory and inject arbitrary code.
- XSS Vulnerability: The analysis explores Cross-Site Scripting (XSS) vulnerabilities that may exist in the TACFAM DB-120WL device's firmware. It explains how improper handling of user input can lead to the injection of malicious JavaScript code into web pages.
- Cleartext Password Storage Vulnerability: This section discusses the security risk associated with storing passwords in clear text in the TACFAM DB-120WL device. It explains how passwords stored in clear text can be easily read by attackers if they gain access to the device.
- CSRF to RCE Vulnerability: The analysis explores a scenario in the TACFAM DB-120WL device where a Cross-Site Request Forgery (CSRF) vulnerability leads to Remote Code Execution (RCE). It explains how an attacker can inject malicious commands and achieve remote code execution.
- Buffer Overflow Vulnerability: The analysis discusses an OS Command Injection vulnerability in the TACFAM DB-120WL device, which occurs when untrusted user input is directly included in system commands without proper validation. It explains how the vulnerability can be exploited by leveraging the ';' character.

Technical Analysis

UART (Universal Asynchronous Receiver/Transmitter) is a widely used communication protocol that facilitates serial communication between electronic devices. It serves as a crucial interface for connecting devices such as microcontrollers, sensors, and other peripherals. This article provides an in-depth understanding of UART communication and guides you through the process of connecting UART to USB using TTL (Transistor-Transistor Logic) converters.

1. UART Communication Overview: UART communication is based on asynchronous serial communication, where data is transmitted in a sequential bit-by-bit manner. It involves two main components: a transmitter and a receiver. The transmitter converts data from parallel to serial format, while the receiver performs the reverse operation.

2. UART Communication Parameters: UART communication requires a set of parameters to establish a successful connection between devices. These parameters include:

- Baud Rate: Baud rate determines the speed at which data is transmitted. It represents the number of bits per second (bps) and must be set identically on both the transmitter and receiver to ensure data synchronization.

- Data Bits: Data bits define the number of bits used to represent each character. Common configurations include 7 bits, 8 bits, or even 9 bits for special cases.

- Parity: Parity allows for error detection during data transmission. It can be set to none, even, or odd parity, providing basic error checking capabilities.

- Stop Bits: Stop bits indicate the end of a data transmission. Common configurations include one or two stop bits.

3. UART Pin Configuration: UART communication requires specific pin connections between devices. The common UART pin configuration consists of:

- TX (Transmit): The TX pin is responsible for transmitting data from the UART transmitter to the receiver.

- RX (Receive): The RX pin receives data from the UART receiver.

c. Ground (GND): The GND pin establishes a common reference point for both devices, ensuring a stable electrical connection.

4. Connecting UART to USB using TTL: To connect UART to a USB port on a computer, a TTL converter is required. The TTL converter serves as an intermediary, facilitating the translation between the UART voltage levels and the USB interface. The following steps outline the process:

a. Identify the UART Pins: Determine the TX, RX, and GND pins on the UART device.

b. Choose a TTL Converter: Select a TTL converter module that matches the voltage levels of the UART device. Common options include 3.3V and 5V TTL converters.

c. Connect the TTL Converter: Connect the TTL converter to the UART device using jumper wires or appropriate connectors. Ensure the TX pin from the UART device is connected to the RX pin of the TTL converter, and the RX pin from the UART device is connected to the TX pin of the TTL converter. Also, connect the GND pins of both devices together.

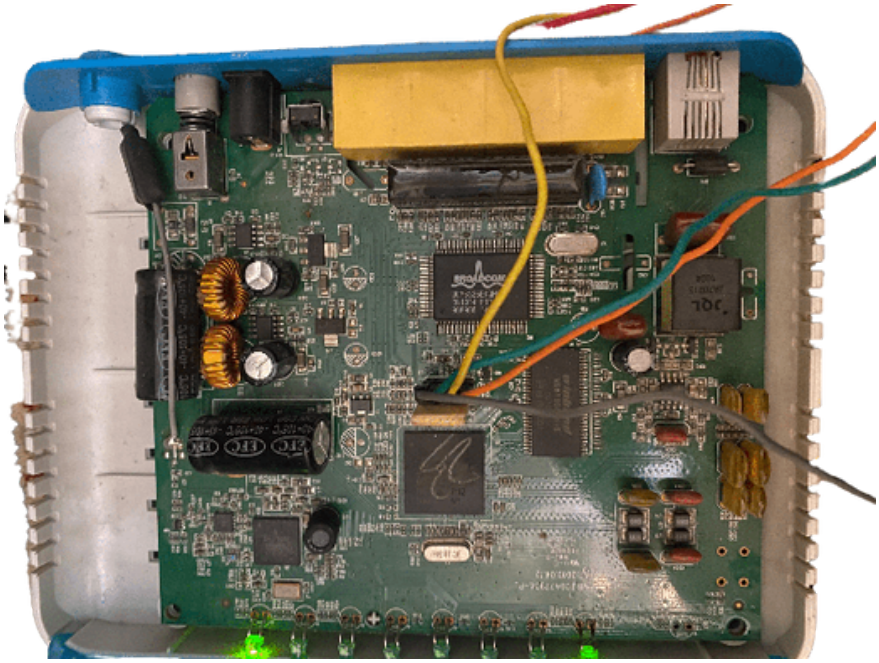
d. Connect the TTL Converter to USB: Connect the USB end of the TTL converter to an available USB port on the computer.

e. Install USB-to-UART Driver: If required, install the appropriate USB-to-UART driver for the TTL converter on your computer. The driver ensures the operating system recognizes the UART device connected via USB.

f. Configure Terminal Software: Use terminal software such as PuTTY, TeraTerm, or minicom to configure the serial port settings, including the baud rate, data bits, parity, and stop bits.

g. Test the Connection: Open the terminal software and establish a connection to the UART device using the configured settings. You should now be able to send and receive data between the UART device and the computer via the USB port.

The code execution vulnerability in the TACFAM DB-120WL device is rooted in the processing of user-supplied input within the firmware. By skillfully crafting malicious requests, an attacker can exploit this flaw to inject arbitrary code into the device's memory, thereby achieving remote code execution. This critical vulnerability grants unauthorized access to the device, potentially compromising the entire network infrastructure.



Exploitation Process

To exploit the vulnerability, an attacker initiates a specially crafted request to the TACFAM DB-120WL device, capitalizing on the device's insecure input processing. Exploiting this weakness allows the attacker to manipulate the device's memory and inject arbitrary code, thereby gaining control over its functionalities. This code execution capability enables unauthorized access, data breaches, and complete control of the device.

Introduction: Connecting a TTL (Transistor-Transistor Logic) device to a USB port on a computer allows for convenient serial communication and data exchange. By utilizing a terminal screen, such as "screen" command-line tool, you can establish a connection with the TTL device and interact with it via a terminal interface. This article provides a step-by-step guide on connecting TTL to USB and using the screen in the terminal for effective communication.

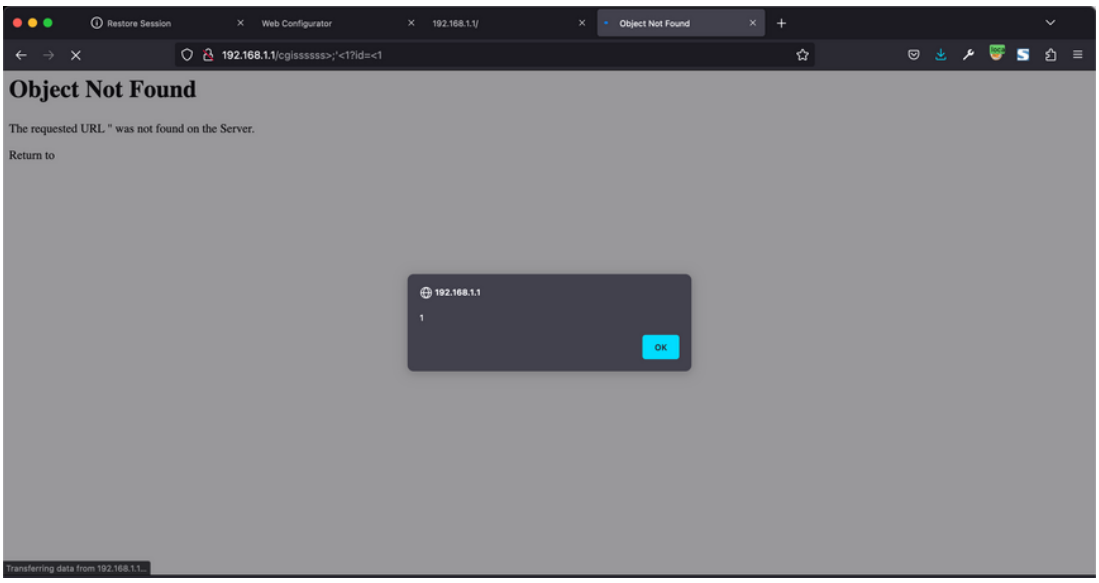
Run "screen" Command: In the terminal, type the following command to initiate a screen session with the TTL device:

screen /dev/ttyUSBX <baud_rate>

For access to os shell via admin

admin:admin

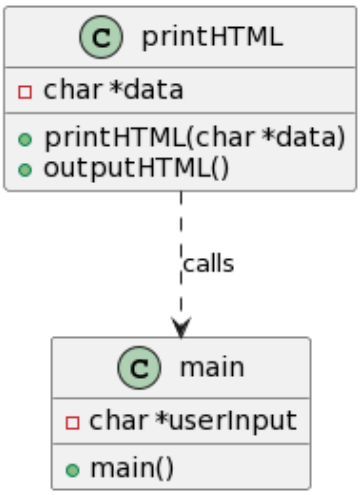
Through careful examination of the CGI scripts in the TACFAM DB-120WL device firmware, we have identified potential areas where XSS vulnerabilities may exist. These vulnerabilities typically stem from improper handling of user input, inadequate input validation, or lack of output encoding. The vulnerable C code snippet below demonstrates a common scenario:



```
#include <stdio.h>
#include <stdlib.h>

void printHTML(char *data) {
   printf("Content-Type: text/html\n\n");
   printf("<html>\n");
   printf("<body>\n");
   printf("%s\n", data); // Vulnerable line: Unsanitized user input directly embedded into HTML
response
   printf("</body>\n");
   printf("</html>\n");
}

int main() {
   char *userInput = getenv("QUERY_STRING");
   printHTML(userInput);
   return 0;
}
```

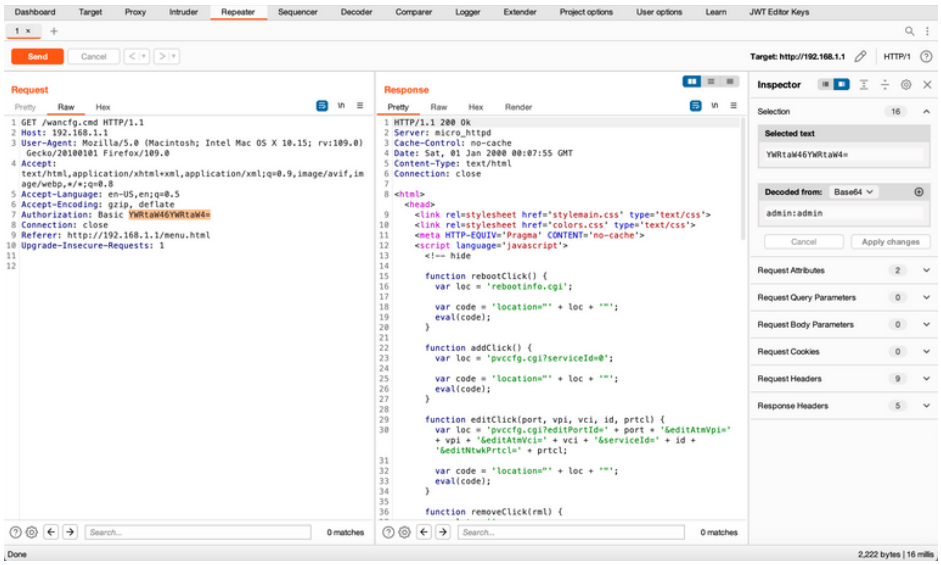
In this code snippet, the `printHTML()` function takes user input (`data`) and directly embeds it into the HTML response without proper sanitization or encoding. This allows an attacker to inject malicious JavaScript code into the web page.

Class Diagram: Code Vulnerable to XSS



Misconfigurations occur when system administrators or users inadvertently leave their systems in an insecure state by incorrectly configuring permissions, access controls, or file permissions. Such misconfigurations can provide attackers with unauthorized access to critical files, including password files, compromising the security of the device.

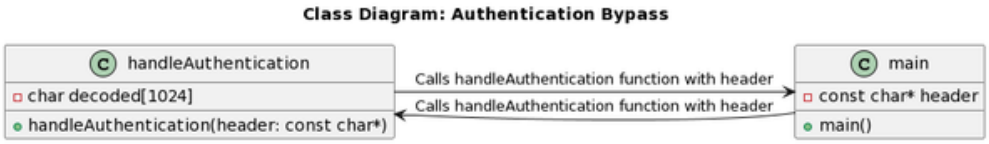
Authentication plays a crucial role in securing devices and systems. However, vulnerabilities in the authentication mechanisms can lead to unauthorized access and compromise the security of the device. This article explores a specific authentication bypass vulnerability in the TACFAM DB-120WL device, which involves the base64 decoding of an HTTP header. Through an in-depth analysis of vulnerable C code snippets, we will examine the root causes of this vulnerability and discuss effective mitigation strategies to enhance the device's security.



By examining relevant C code snippets in the TACFAM DB-120WL device's authentication mechanism, we can uncover the root causes of the vulnerability. The following code snippet illustrates a potential scenario where the vulnerability exists:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/bio.h>
#include <openssl/evp.h>

void handleAuthentication(const char *header) {
   char decoded[1024];

   // Base64 decode the header
   BIO *bio = BIO_new(BIO_f_base64());
   BIO *bioMem = BIO_new_mem_buf(header, strlen(header));
   bioMem = BIO_push(bio, bioMem);
   BIO_read(bioMem, decoded, strlen(header));
   BIO_free_all(bioMem);

   // Perform authentication
   if (strcmp(decoded, "admin:password") == 0) {
      printf("Authentication successful!\n");
      // Grant access to restricted functionalities
   } else {
      printf("Authentication failed!\n");
      // Deny access
   }
}

int main() {
         const char *header = "Basic YWRtaW46cGFzc3dvcmQ="; // Base64-encoded
"admin:password"
   handleAuthentication(header);
   return 0;
}
```

In this code snippet, the `handleAuthentication()` function performs the authentication process by base64 decoding the provided header and comparing it to a hardcoded username and password combination. However, the vulnerability lies in the improper handling of the decoded data, allowing for authentication bypass.



In token-based authentication, when a user successfully logs in to an application, they are issued a unique token, which is typically a long string of characters. This token serves as proof of the user's identity and is sent along with each subsequent request to the server. The server then verifies the authenticity and validity of the token to grant or deny access to the requested resources.
Token-based authentication offers several advantages over traditional username/password authentication:

1. Stateless: Tokens are self-contained, meaning the server does not need to maintain a session state for each user. This allows for easier scalability and reduces server-side storage requirements.
2. Cross-domain and cross-platform support: Tokens can be used across multiple domains or platforms since they are typically sent via HTTP headers or as part of the API request payload.
3. Enhanced security: Tokens can be designed to expire after a certain period, forcing users to reauthenticate regularly. They can also be revoked if compromised, reducing the potential impact of a security breach.

While token-based authentication provides enhanced security, there are still some risks and vulnerabilities to consider:

1. Token theft: If an attacker manages to obtain a user's token, they can impersonate that user and gain unauthorized access. This can occur through various means, such as cross-site scripting (XSS) attacks, man-in-the-middle (MITM) attacks, or server-side vulnerabilities.
2. Token leakage: Tokens should be carefully handled to prevent leakage. Storing tokens in client-side storage mechanisms like local storage or cookies can make them vulnerable to cross-site scripting attacks or cross-site request forgery (CSRF) attacks.

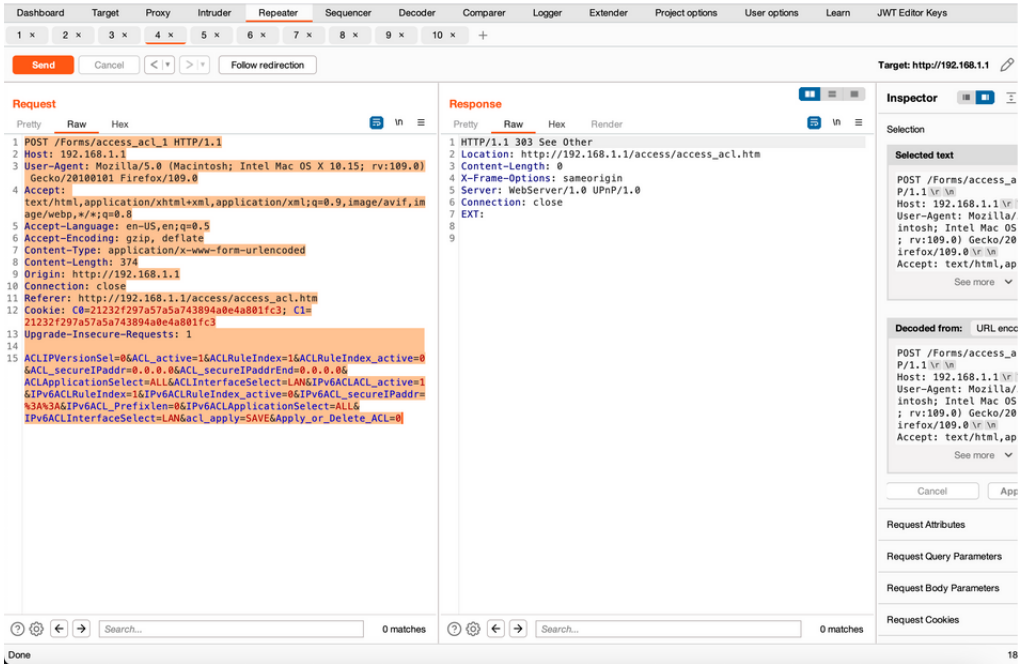
3. Insecure token storage: If tokens are not securely stored on the server side, they may be susceptible to data breaches. It is essential to properly encrypt and protect tokens to prevent unauthorized access.

4. Token replay attacks: If tokens are not properly protected against replay attacks, an attacker may intercept a token and use it multiple times to gain unauthorized access to resources.

5. nadequate token expiration and revocation: Tokens should have an appropriate expiration time and mechanism for revocation. If tokens have a lengthy expiration period or if there is no proper revocation mechanism in place, the risk of unauthorized access increases.

To mitigate these risks, it is crucial to implement best practices, such as securely transmitting tokens over HTTPS, using secure token storage mechanisms, employing measures like token encryption and signing, implementing strong input validation to prevent injection attacks, and regularly monitoring and auditing token usage to detect suspicious activities.

Cross-Site Request Forgery (CSRF) vulnerabilities can expose devices and systems to significant security risks. When combined with Remote Code Execution (RCE) capabilities, these vulnerabilities can result in severe consequences. This article explores a specific scenario in the TACFAM DB-120WL device where a CSRF vulnerability leads to RCE. Through an in-depth analysis of vulnerable C code snippets, we will examine the root causes of this vulnerability and discuss effective mitigation strategies to enhance the device's security.

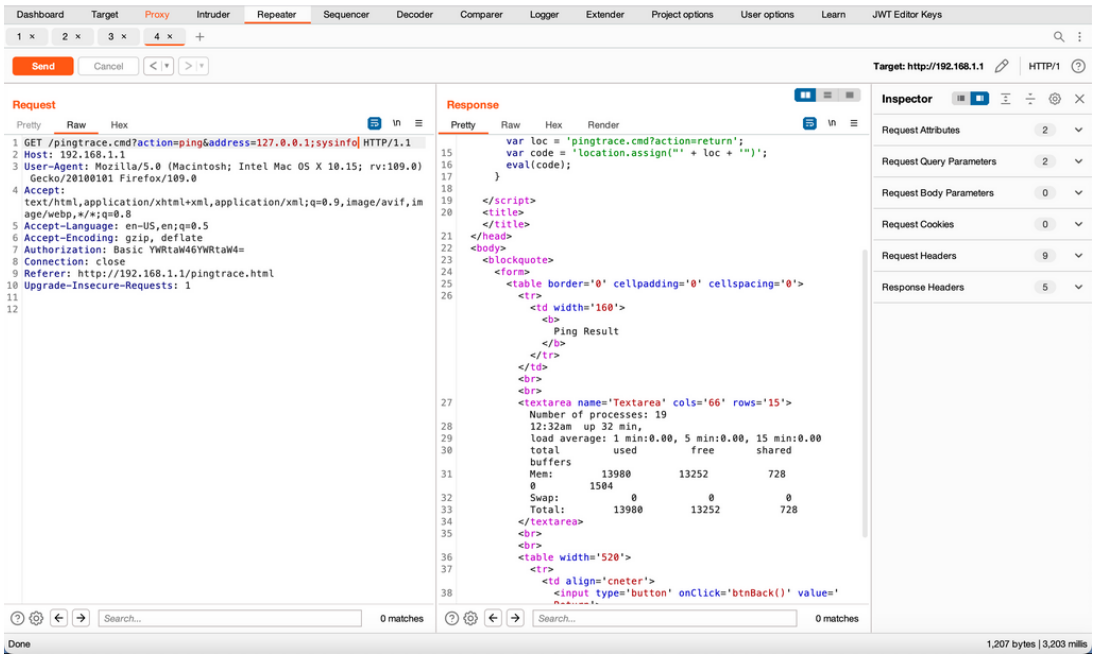


By examining relevant C code snippets in the TACFAM DB-120WL device, we can uncover the root causes of the vulnerability. The following code snippet illustrates a potential scenario where the vulnerability exists:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

void processRequest(const char *request) {
   char command[1024];
      snprintf(command, sizeof(command), "echo \"%s\" | sh", request); // Vulnerable line: Executing user-supplied command

   if (system(command) == -1) {
      printf("Failed to execute command.\n");
   }
}

int main() {
   const char *request = "rm -rf /"; // Malicious command
   processRequest(request);
   return 0;
}
```

In this code snippet, the `processRequest()` function takes user-supplied data (request) and executes it as a command using the `system()` function. The lack of proper input validation and access controls allows an attacker to inject malicious commands and potentially achieve remote code execution.

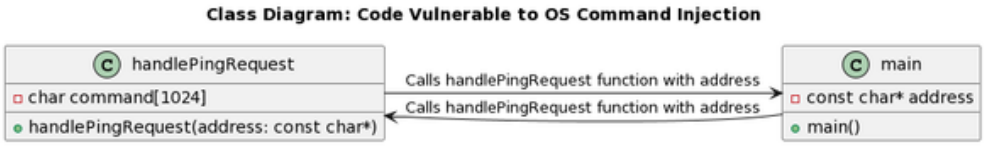
The OS Command Injection vulnerability in the TACFAM DB-120WL Device occurs when untrusted user input is directly included in system commands without proper validation or sanitization. In this case, the "action=ping" functionality allows attackers to inject arbitrary commands by leveraging the ';' character.



By examining relevant C code snippets in the TACFAM DB-120WL Device, we can uncover the root causes of the vulnerability. The following code snippet exemplifies a potential scenario where the vulnerability exists:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

void handlePingRequest(const char *address) {
   char command[1024];
    snprintf(command, sizeof(command), "ping -c 1 %s", address); // Vulnerable line: Command injection via user input

   if (system(command) == -1) {
      printf("Failed to execute command.\n");
   }
}

int main() {
   const char *address = "127.0.0.1; ls"; // Malicious input with injected command
   handlePingRequest(address);
   return 0;
}
```

In this code snippet, the `handlePingRequest()` function constructs a system command to execute a ping request using the provided address. However, the vulnerability lies in the lack of proper input validation or sanitization, allowing an attacker to inject arbitrary commands using the ';' character.



03

Impact

The code execution vulnerability in the TACFAM DB-120WL device exposes users and organizations to significant risks. Attackers can gain unauthorized access to the device, compromising its settings and potentially infiltrating the entire network. This allows them control over network configurations, device functionalities, and sensitive information. The vulnerability can lead to data breaches, with attackers retrieving login credentials, network configurations, and personally identifiable information. This compromised data can be exploited or sold on the black market, causing severe consequences. Attackers can also perform network reconnaissance and exploit additional vulnerabilities within the network infrastructure, compromising the security and integrity of the entire network.

The code execution vulnerability in the TACFAM DB-120WL device poses significant risks to users and organizations, including:

a. Unauthorized Access: Attackers can gain unauthorized access to the device, compromising its settings, and potentially infiltrating the entire network. This unauthorized access grants attackers control over network configurations, device functionalities, and sensitive information.

b. Data Breaches: Once in control, attackers may retrieve sensitive information stored on the device, such as login credentials, network configurations, or personally identifiable information. This compromised data can be exploited for malicious purposes or sold on the black market, leading to severe consequences for individuals and organizations.

c. Network Compromise: With control over the TACFAM DB-120WL device, attackers can perform network reconnaissance, mapping the network topology, identifying connected devices, and exploiting additional vulnerabilities within the network infrastructure. This unrestricted access jeopardizes the security and integrity of the entire network.

Mitigation Strategies: To mitigate the code execution vulnerability in the TACFAM DB-120WL device and safeguard against potential attacks, the following measures are recommended:

a. Firmware Updates: Users should ensure that their devices are running the latest firmware version provided by the manufacturer. Regularly checking for firmware updates and promptly applying them helps patch known vulnerabilities and bolster device security.

b. Network Segmentation: Implementing network segmentation limits the potential impact of a compromised device. By dividing the network into separate segments, an attacker's lateral movement and access to critical resources can be restricted, minimizing the overall network risk.

c. Network Monitoring: Deploy robust network monitoring solutions to detect and identify suspicious activities, such as unusual traffic patterns or unauthorized access attempts. Intrusion detection systems and anomaly detection mechanisms provide early warning signs of compromise, allowing for timely response and mitigation.

d. Vendor Security Best Practices: Engage with the device manufacturer and adhere to their recommended security best practices. This includes using strong, unique passwords, disabling unnecessary services, and implementing encryption protocols to strengthen the overall security posture of the TACFAM DB-120WL device.

04

Conclusion

The code execution vulnerability in the TACFAM DB-120WL device poses significant security risks to users and organizations relying on this networking solution. This analysis has provided an in-depth examination of the vulnerability, its potential impact, and suggested strategies to mitigate the risk. By proactively applying firmware updates, implementing network segmentation, monitoring network activities, and following vendor security best

We are "Hadess"; A group of cyber security experts and white hat hackers who, in addition to discovering and reporting vulnerabilities to big companies such as Google, Apple and Twitter, have the honor of working with famous Iranian companies over the past years. Ayman Burhan Rehiaft Azarakhsh Cyber Security Company provides its customers with integrated solutions in the field of cyber security, with a deep insight and understanding of the software development process as well as the development infrastructure.

WWW.HADESS.IO