Pwn2Own’ing the TP-Link Archer A7

BARBHACK 2021

@0xMitsurugi - @swapgs
$ whoami

- @0xMitsurugi
  - Security Ninja at Synacktiv (Paris / Lyon / Rennes / Toulouse / Remote)
- @swapgs
  - Vulnerability Researcher at SonarSource (Geneva / Annecy / Bochum / Austin / Remote)
  - Work done during prior employment at Synacktiv, no affiliation between both companies
Summary

● We will guide you through our journey at Pwn2Own
  ○ Presentation of the competition and how it works
  ○ Initial setup
  ○ Discovery of CVE-2021-27246
  ○ Exploitation
  ○ Q&A

● Stay with us, it won’t be a crazy hardcore insane technical talk
Pwn2Own in 2 minutes

- Bi-annual competition organized by the Trend Micro Zero Day Initiative, taking place during CanSecWest
- A list of products is announced, along with rules
  - OS, browsers, consumer electronics (phones, watches, routers)
  - Products will be to be up-to-date (24 hours before) in default configuration
  - You have to prove remote code execution, without authentication
- Trend Micro isn’t a broker!
  - Acquisitions are disclosed to vendors with the goal of getting them fixed
- You get a cool challenge and maybe a few $$
Pwn2Own in 2 minutes

● We took part of Pwn2Own Tokyo 2020
  ○ Original announcement: July 28, 2020
  ○ Contest deadline: November 2, 2020

● Remote participation is now possible
  ○ ZDI will run it for you, everything is live streamed
  ○ Drawback: you are not allowed to fix the exploit(s) between attempts
  ○ You need to provide the exploit(s) and a full explanation of each bug beforehand

● Teams order is random, duplicates are not rewarded
  ○ “Partial win”

● Several bugs ready but only participated in the Routers category
TP-Link AC1750

- Mid-end Wi-Fi router
- Models A7 and C7 are very similar
  - The later has Alexa support (??), mostly sold on Amazon (15k+ evaluations)
  - 720 Mhz MIPS CPU, 8MB of flash, 128MB of RAM
  - 802.11ac, 4 LAN slots + 1 WAN
- < 100€, quite popular in the custom firmware scene
  - Some documentation related to the OpenWrt support is public
- Second year in a row at Pwn2Own
  - Bugs are found and disclosed every year
  - No major change between versions
Initial access - UART

- “Free” shell access on consumer electronics is rare
- First step of any research on embedded systems
  - UART / JTAG are often easy to locate
    - Physical presence, datasheets
    - Not always restricted
  - Debugging capabilities are incredibly useful
- We won’t cover the UART discovery
  - Check out Team Flashback’s great video https://www.youtube.com/watch?v=01mw0oTHwxg
- No downgrade protection, you can also use exploits from previous years
  - Requires persistence (not investigated)
Initial access - UART
Initial access - UART

- A good logic analyzer will help finding the right parameters to decode the serial communication
  - e.g. Saleae + Logic 2 + 5 minutes
Initial access - UART

- Plug everything, reboot the device
- minicom -8 -b 115200 -D /dev/tty.usbmodem*
- Access to the bootloader prompt
  - U-Boot 1.1.4
  - Useful if we need to reflash the device
- Shell access as root
- Limited OpenWrt environment
  - MIPS OpenWrt Linux-3.3.8
Initial access - Environment

- Compilation of useful tools (gdbserver, strace, busybox with all applets)
  - Target is a MIPS32 big endian CPU, supported by Buildroot
    - BR2_MIPS_SOFT_FLOAT=y
    - BR2_TOOLCHAIN_BUILDROOT_LIBC="musl"
- Customized Dropbear is already running, but authentication is disabled
  - Kill it and remove a few options over UART: remove -C, add -L
  - Use it to copy additional binaries
- Don’t enjoy it too much though
- Time to hunt for vulnerabilities!
Attack surface

- Previous work by other contestants
- Recent firmwares are available on tp-link.com
- DHCP on the WAN
- Only a few services listen on the LAN
  - dropbear, udpXY, uhttpd, tdpServer
  - /usr/bin/tdpServer
    - UDP/20002, LAN-side
    - Simple protocol (binary header, JSON payload)
      - Already documented (and patented!)
    - Runs as root

```c
struct tdp_packet {
    uint8_t version;
    uint8_t type;
    uint16_t opcode;
    uint16_t len;
    uint8_t flags;
    uint8_t _padding;
    uint32_t device_serial;
    uint32_t checksum;
    uint8_t data[1024];
};
```
Attack surface

- Ghidra = <3
- `tdpServer` decrypts data with a fixed key and parses it as JSON (kind of)

```
key = b'TPONEMESH_Kf!xn?gj6pMAt-wBNV_TDP'[0:16]
```

- Most handlers are related to OneMesh
  - It seems related to proprietary configuration synchronization for roaming
  - Devices advertise themselves
  - Crafted a bunch of `scapy` scripts
  - After a first review, a few DoS but nothing exploitible
  - Plot twist: last year's vulnerability was not really fixed, but we missed it

- Each advertised device is added in a shared memory area
  - Stores pairs of MAC / IP of clients as strings
  - Who's reading from it?
  - New attack surface: `sync-server`
Research and discovery of CVE-2021-27246

```
"method": "slave_key_offer",
"data": {
  "group_id": "1",
  "ip": "1.3.3.7",
  "slave_mac": "00:11:22:33:44:55",
  "slave_private_account": "a",
  "slave_private_password": "a",
  "want_to_join": true,
  "model": "p2o",
  "product_type": "tplink",
  "operation_mode": "whatever",
  "signal_strength_24g": 2,
  "signal_strength_5g": 2,
  "link_speed_24g": 1,
  "link_speed_5g": 1,
  "level": 3,
  "connection_type": "whatever"
}
```
Research and discovery of CVE-2021-27246

- `sync-server`: a vulnerable function is found!

```c
undefined4 _handle_request_clients_async(void)
{
    //(...)
    char *array_ip_mac[64];
    //(...)
    onemesh_listDevices(&devlist);
    if (head != NULL) {
        while( true ) {
            json_field_ip_ = json_object_object_get(main_json_object,"ip");
            json_type_mac = json_object_object_get(main_json_object,"mac");
            ip_as_str = json_object_get_string(json_field_ip_);
            i = i + 1;
            arr_ip_mac[i * 2] = ip_as_str;
            mac_as_str = json_object_get_string(json_type_mac);
            arr_ip_mac[i * 2 + 1] = mac_as_str;
            if (head == NULL) goto LAB_00404b48;
            //(...)
        }
    }
}
```
Research and discovery of CVE-2021-27246

- **sync-server**: a vulnerable function is found!

```c
#define MAX_IPS 32

void handle_request(char *request, int request_len) {
    char *request_ip;
    char *request_mac;
    int request_len;
    int request_ip_len;
    int request_mac_len;

    // Process the request...

    // Example code snippet
    char *array_ip_mac[64];
    // (possibly vulnerable code)

    oneshell_listDevices (&devl)
    if (head != NULL) {
        while (true) {
            json_field_ip = json_object_object_get(main_json_object,"ip");
            json_type_mac = json_object_object_get(main_json_object,"mac");
            ip_as_str = json_object_get_string(json_field_ip);
            i = i + 1;
            arr_ip_mac[i * 2] = ip_as_str;
            mac_as_str = json_object_get_string(json_type_mac);
            arr_ip_mac[i * 2 + 1] = mac_as_str;
            if (head == NULL) goto LAB_00404b48;
        }
    }
}
```

- **Fixed size array on stack**
- **As long as there is data to write...**
- **Overflow the stack if more than 32 records...**
Research and discovery of CVE-2021-27246

- Test scenario
  - Send more than 32 messages to tdpServer containing different IP / MAC
  - Wait for sync-server to read them
  - sync-server crash

- A PoC is written and confirms the bug
  - “Illegal instruction” and not “Segmentation Fault”?

- Time to exploit!

```
sync-server: handle_request_clients_async:2494: [DBG] count is 49
sync-server: handle_request_clients_async:2503: [DBG] Infile: /tmp-sync-server/
  -> request-input-2046063169-25104
  -> request-output-1502619911-25104
Illegal instruction
root@ArcherC7v5:~#
```
Exploitation

- This bug seems OK
  - Not in a network daemon, less likely to be found by another team

- Some good points
  - No stack canary
  - Non-PIE binary
  - IP and MAC formats are not validated, only limited in size

- And bad points
  - Full ASLR
  - Integrity checks on JSON data
  - No direct interaction with `sync-server`
  - Everything is sensitive: we must avoid crashing `tdpServer`
  - MAC addresses can’t be longer than 17 bytes = 4 MIPS32 instructions
Exploitation - ASLR

- ASLR is trivially bypassed!
- The stack overflow writes a pointer to data we control in the heap
  - $pc$ is restored and points to a MAC address we control
- Heap is RWX!
- Code execution? But devil lies in the details...
Exploitation - ASLR

array_ip_mac[64]  

... array_ip_mac[0]  array_ip_mac[1] ... array_ip_mac[63] ... $ra ...

heap

... "0.0.0.0\0" ... "00:00:00:00:00:00:00:00\0" ... "0.0.0.1\0" ... "00:00:00:00:00:31\0" ...

saved registers
Exploitation - ASLR

array_ip_mac[64]

... array_ip_mac[0] array_ip_mac[1] ... array_ip_mac[63] ... $ra ...

saved registers

heap

... "0.0.0.0\0" ... "00:00:00:00:00:00\0" ... "0.0.0.1\0" ... "00:00:00:00:00:31\0" ... "FE:FE:FF:FF:FF:FF\0" ...

Pwn2Own'ing the TP-Link Archer A7 - BARBHACK 2021
Exploitation - JSON encoding

● JSON checks
  ○ Format of MAC address is not validated, only its size (17 bytes mac)
  ○ But it must passes a “string” check, only \[\x20-\x7f\] is allowed
  ○ No more “chosen” code execution?

● ASCII Shellcoding is hard (impossible?) in this context

● Reversing a JSON parser is tedious but
  ○ It handles Unicode escape sequences
  ○ It accepts \u00xx for encoding any byte (except NULL bytes)
  ○ Shellcode without with NULLs is an acceptable constraint
Exploitation - shellcoding with 4 instructions

- Idea: why not `system(cmd)`?
- `sync-server` is not compiled as PIE
  - `0c 10 07 14 jal system`
  - No NULL byte
- `$s0, $s2, $s4` and `$s6` contains pointers to IPs we advertised
  - `02 40 20 25 move $a0, $s2`
  - No NULL byte
- Only two instructions needed
- We have to decide which command to execute
  - No `telnetd`, no `netcat`, a stripped down `busybox` with few applets…
Exploitation - Final Step

- TP-Link ships a debug daemon called `tddp` riddled with trivial vulnerabilities
  - Not started by default
- `system("tddp")`
- Inject a second stage through `tddp`
  - Start a reverse shell
  - Blink all the LEDs (`/sys/devices/platform/leds-gpio/leds/*/brightness`)
  - Profit \o/
- Exploit is reliable
  - Exploit takes time because `sync-server` is asynchronous and terribly slow
  - We can wait up to 80 seconds per attempt
Final Steps

- Whitepaper and exploit sent to ZDI the week before the event
- … but a new update is released a few days before the event
  - Most contestants cancel their participation
  - Our bug is still working (??)
  - *Plot twist of the plot twist: last year’s bug has been patched*
- Organisers schedule a Zoom call before the attempt
  - Explain the setup, show the hardware and the version
  - Different firmware but `sync-server` is the same binary
- Exploit is launched on live stream, without showing the script output
- 3 attempts, individual limit of 5 minutes
  - 2 x 80 seconds feels like an eternity
Win!
Aftermath

- To publish details, either
  - Wait for 3 months
  - OR
  - Vulnerability is patched by editor

- Patch is published
  - Analysis has been done
  - A simple counter is added
  - No more than 32 pair IP/MAC allowed, this bug is dead!

- But
  - No special hardening has been added
  - tddp still here…
Conclusion

● The 90’s are calling
  ○ Most ~ modern exploit mitigations are missing
  ○ Patches are both rushed... and delayed to the last minute

● Pwn2Own is fun
  ○ New categories are more accessible than ever (printers, routers)
  ○ Organizers will do everything to help you before / during the event
  ○ The TP-Link AC1750 is still here ;-)  

● We put everything on GitHub
  ○ https://github.com/synacktiv/CVE-2021-27246_Pwn2Own2020

● Many thanks to the Barbhack organizers!
Q&A

Thank you for your attention!

We’ll be happy to take questions :-)