DEVCORE
SECURITY
CONSULTING

Your NAS is not your NAS !

Angelboy

angelboy@chroot.org

@scwuaptx

Whoami

- Angelboy
  - Researcher at DEVCORE
  - CTF Player
    - HITCON / 217
  - Chroot
  - Pwn2Own 2020 Tokyo
  - Co-founder of pwnable.tw
  - Speaker
    - HITB GSEC 2018/AVTokyo 2018/VXCON/HITCON

PWNABLE.TW

Outline

- Introduction
- Recon
- Netatalk
  - Vulnerability
  - Exploitation
- Mitigation
- Conclusion


STATUS
DISK 1
DISK 2
DISK 3
DISK 4
DS418play

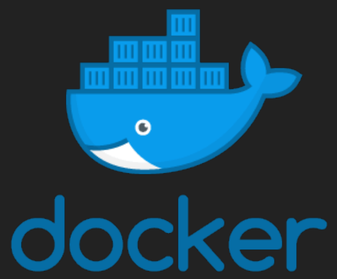
Introduction

NAS

- Network Attached Storage
  - In the early days, users could simply access data and share files directly on the Internet
  - NAS also provides a variety of services, which not only makes file sharing more convenient, but also closer to IoT

Introduction

NAS



Introduction

Motivation

- Intranet
  - For the red team, NAS is one of the most common devices in the intranet
  - Confidential corporate information is often stored in the NAS

Introduction

Motivation

- Ransomware
  - NAS become a ransomware target in recent years
    - Synolocker
    - Qlocker

再爆發 AgeLocker、eCh0raix 加密勒索 QNAP NAS 一週內連發 5 項安全漏洞更新

文: Matthew Chan / 新聞中心

文章索引： IT要聞 IT港聞 網路產品 QNAP

IT 安全新聞網站 Bleeping Computer 報導，繼 Qlocker 之後近日再出現 QNAP NAS 苦主被勒索軟體加密檔案的報告，QNAP 官方在短短1 星期內共發佈了 5 項安全性警告更新，其中 3 項仍在調查中、 2 項已被解決，調查中的包括 AgeLocker 勒索軟體、eCh0raix 勒索軟體，Roon Server Zero-Day 漏洞，已解決的有 QNAP Malware Remover 4.x 存在命令注入漏洞及 Music Station 訪問控制漏洞，各位 QNAP NAS 用家需要加強注意防範。

據 Bleeping Computer 指出 QNAP NAS 近年飽受勒索軟體的攻擊威脅，2019 年 7 月與 2020 年 6 月被 eCh0raix 勒索攻擊，2021 年 4 月初被 Qlocker 勒索攻擊，黑客 5 日內捲款 26 萬美元，2021 年 4 月下旬緊接出現 AgeLocker 勒索攻擊，5月初再出現 eCh0raix 勒索攻擊，令不少 QNAP NAS 用家提心吊膽。

Introduction

Motivation

- Pwn2Own Mobile
  - Home Automation
  - Televisions
  - Routers
  - NAS Server
    - W D
    - Synology

Recon

Environment

- Environment
  - DS918+
  - DSM 6.2.3-25426
  - Default setting

Recon

Attack surface

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      9611/smbd
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      9640/nginx: master
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      8471/sshd
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      9506/postgres
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      9640/nginx: master
tcp        0      0 127.0.0.1:4700          0.0.0.0:*               LISTEN      9275/cnid_metad
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN      9611/smbd
tcp        0      0 0.0.0.0:3262            0.0.0.0:*               LISTEN      11607/iscsi_snapsho
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN      9640/nginx: master
tcp        0      0 0.0.0.0:5001            0.0.0.0:*               LISTEN      9640/nginx: master
tcp        0    196 192.168.86.42:22        192.168.86.55:38656     ESTABLISHED 8622/sshd: angelboy
tcp6       0      0 :::139                  :::*                    LISTEN      9611/smbd
tcp6       0      0 :::80                   :::*                    LISTEN      9640/nginx: master
tcp6       0      0 :::22                   :::*                    LISTEN      8471/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      9640/nginx: master
tcp6       0      0 :::445                  :::*                    LISTEN      9611/smbd
tcp6       0      0 :::3261                 :::*                    LISTEN      -
tcp6       0      0 :::3263                 :::*                    LISTEN      -
tcp6       0      0 :::3264                 :::*                    LISTEN      -
tcp6       0      0 :::3265                 :::*                    LISTEN      11627/scsi_plugin_s
tcp6       0      0 :::548                  :::*                    LISTEN      9274/afpd
tcp6       0      0 :::5000                 :::*                    LISTEN      9640/nginx: master
tcp6       0      0 :::5001                 :::*                    LISTEN      9640/nginx: master
```

Recon

Attack surface

```
udp        0      0 0.0.0.0:1900          0.0.0.0:*                   15584/minissdpd
udp        0      0 0.0.0.0:49171         0.0.0.0:*                   9393/synosnmpcd
udp     2944      0 0.0.0.0:68            0.0.0.0:*                   11493/dhclient
udp        0      0 0.0.0.0:68            0.0.0.0:*                   7990/dhclient
udp        0      0 192.168.86.42:123     0.0.0.0:*                   9087/ntpd
udp        0      0 127.0.0.1:123         0.0.0.0:*                   9087/ntpd
udp        0      0 0.0.0.0:123           0.0.0.0:*                   9087/ntpd
udp        0      0 192.168.86.255:137    0.0.0.0:*                   10793/nmbd
udp        0      0 192.168.86.42:137     0.0.0.0:*                   10793/nmbd
udp        0      0 0.0.0.0:137           0.0.0.0:*                   10793/nmbd
udp        0      0 192.168.86.255:138    0.0.0.0:*                   10793/nmbd
udp        0      0 192.168.86.42:138     0.0.0.0:*                   10793/nmbd
udp        0      0 0.0.0.0:138           0.0.0.0:*                   10793/nmbd
udp        0      0 0.0.0.0:43169         0.0.0.0:*                   10870/avahi-daemon:
udp        0      0 127.0.0.1:161         0.0.0.0:*                   9341/snmpd
udp        0      0 0.0.0.0:5353          0.0.0.0:*                   10870/avahi-daemon:
udp        0      0 0.0.0.0:9997          0.0.0.0:*                   9068/findhostd
udp        0      0 0.0.0.0:9998          0.0.0.0:*                   9068/findhostd
udp        0      0 0.0.0.0:9999          0.0.0.0:*                   9068/findhostd
```

Recon

Attack surface

- DSM Web
  - Developed by Synology
  - Huge function, but relatively safe
- SMB
  - Open source project
  - There have been vulnerabilities in SambaCry that are more harmful, and there are many CVEs every year, but they are not very harmful recently.

Recon

Attack surface

- iSCSI Manager
  - Developed by Synology
  - There are many vulnerabilities recently
- Netatalk
  - Open source project
  - There was only one RCE vulnerability in 2018
    - CVE-2018-1160

Netatalk

Introduction

- Apple Filing Protocol is a proprietary network protocol, and part of the Apple File Service (AFS), that offers file services for macOS and the classic Mac OS.
- Netatalk is a freely-available Open Source AFP fileserver.
- Capable of serving many Macintosh clients simultaneously as an AppleShare file server
- We can see it on most of NAS.

Netatalk

Synology

- Default enable
- Version
  - Modified from 3.1.8
  - Full Security Patch
- Protection
  - ASLR
  - NX
  - Stack Guard

Netatalk

Data Structure

- DSI
  - The Data Stream Interface (DSI) is a session layer used to carry AFP traffic over TCP.
  - DSI is spoken between a client and an AFP server. All DSI communication contains the DSI header.

Netatalk

Data Structure

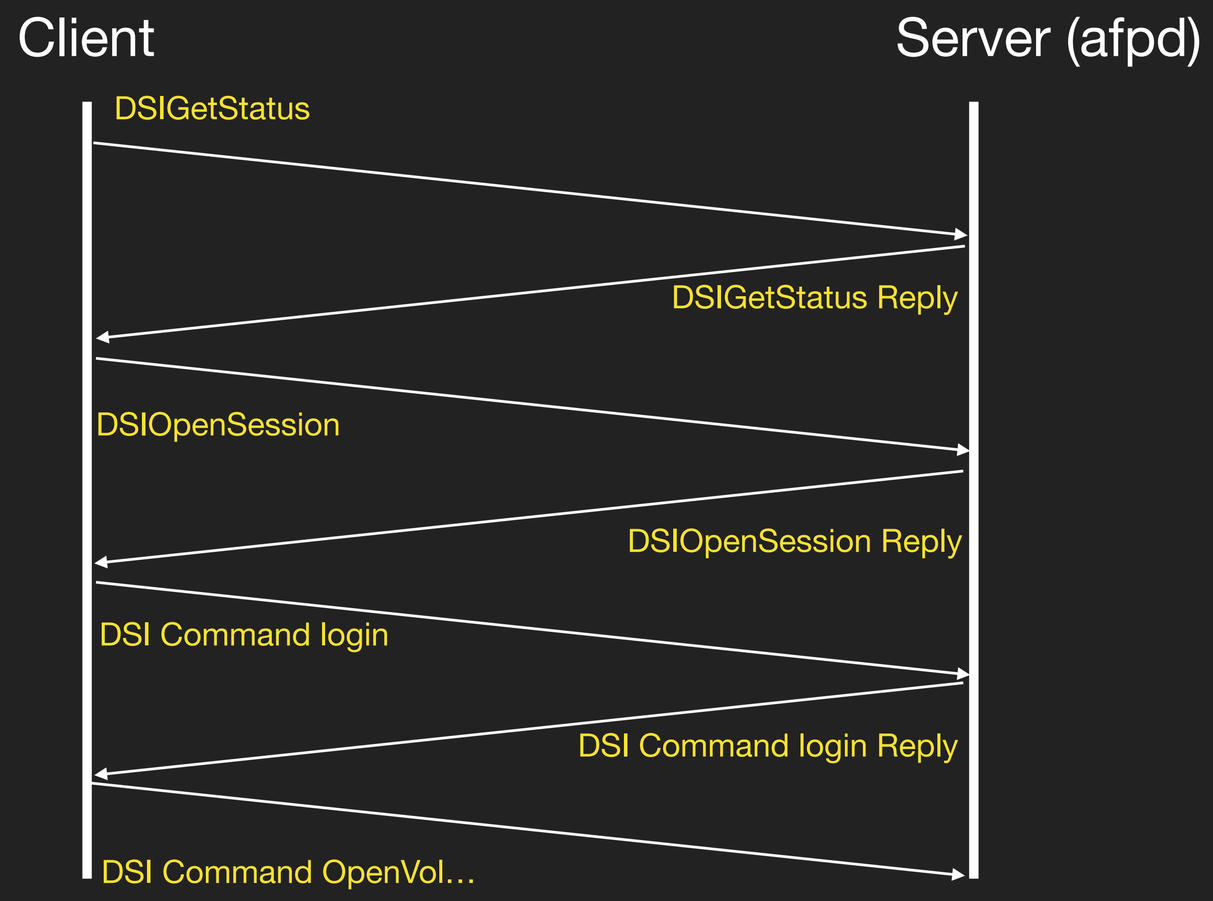
- DSI Packet Header

Bit 0 - 7	Bit 8 - 15	Bit 16 - 23	Bit 24 - 31
Flags	Command	Request ID	
Error code/data offset			
Total data length			
Reserved			
Payload (Variable)			

Netatalk

Architecture

- AFP over DSI



DEVCORE SECURITY CONSULTING

Netatalk

Data Structure

- dsi_flag
  - Whether the packet is a request (0x00) or a reply (0x01)

```
struct dsi_block {
    uint8_t dsi_flags;      /* packet type: request or reply */
    uint8_t dsi_command;    /* command */
    uint16_t dsi_requestID; /* request ID */
    union {
        uint32_t dsi_code;  /* error code */
        uint32_t dsi_doff;  /* data offset */
    } dsi_data;
    uint32_t dsi_len;       /* total data length */
    uint32_t dsi_reserved;  /* reserved field */
};
```

Netatalk

Data Structure

- dsi_command
  - DSICloseSession
  - DSICommand
  - DSIGetStatus
  - DSIOpenSession
  - …

```
struct dsi_block {
    uint8_t dsi_flags;        /* packet type: request or reply */
    uint8_t dsi_command;      /* command */
    uint16_t dsi_requestID;   /* request ID */
    union {
        uint32_t dsi_code;    /* error code */
        uint32_t dsi_doff;    /* data offset */
    } dsi_data;
    uint32_t dsi_len;         /* total data length */
    uint32_t dsi_reserved;    /* reserved field */
};
```

Netatalk

Data Structure

- dsi_code
  - Error code
  - For reply
- dsi_doff
  - DSI data offset
  - Using in DSIWrite
- dsi_len
  - The Length of Payload

```
struct dsi_block {
    uint8_t dsi_flags;       /* packet type: request or reply */
    uint8_t dsi_command;     /* command */
    uint16_t dsi_requestID;  /* request ID */
    union {
        uint32_t dsi_code;   /* error code */
        uint32_t dsi_doff;   /* data offset */
    } dsi_data;
    uint32_t dsi_len;        /* total data length */
    uint32_t dsi_reserved;   /* reserved field */
};
```

Netatalk

Data Structure

- DSI
  - A descriptor of dsi stream.

```
typedef struct DSI {
    struct DSI *next;            /* multiple listening addresses */
    AFPObj   *AFPobj;
    ...

    uint32_t attn_quantum, datasize, server_quantum;
    uint16_t serverID, clientID;
    uint8_t  *commands; /* DSI recieve buffer */
    uint8_t  data[DSI_DATASIZ];    /* DSI reply buffer */
    ...
} DSI;
```

Netatalk

Data Structure

- DSI
  - The size of command buffer is taken from the afp configuration file (afp.conf)

```c
typedef struct DSI {
    struct DSI *next;              /* multiple listenin
    AFPObj   *AFPobj;
    ...

    uint32_t attn_quantum, datasize, server_quantum;
    uint16_t serverID, clientID;
    uint8_t  *commands; /* DSI recieve buffer */
    uint8_t  data[DSI_DATASIZ];    /* DSI reply buffer
    ...
} DSI;
```

```c
/*!
 * Allocate DSI read buffer and read-ahead buffer
 */
static void dsi_init_buffer(DSI *dsi)
{
    if ((dsi->commands = malloc(dsi->server_quantum)) == NULL) {
        LOG(log_error, logtype_dsi, "dsi_init_buffer: OOM");
        AFP_PANIC("OOM in dsi_init_buffer");
    }
```

Fixed size depend on conf

Netatalk

Data Structure

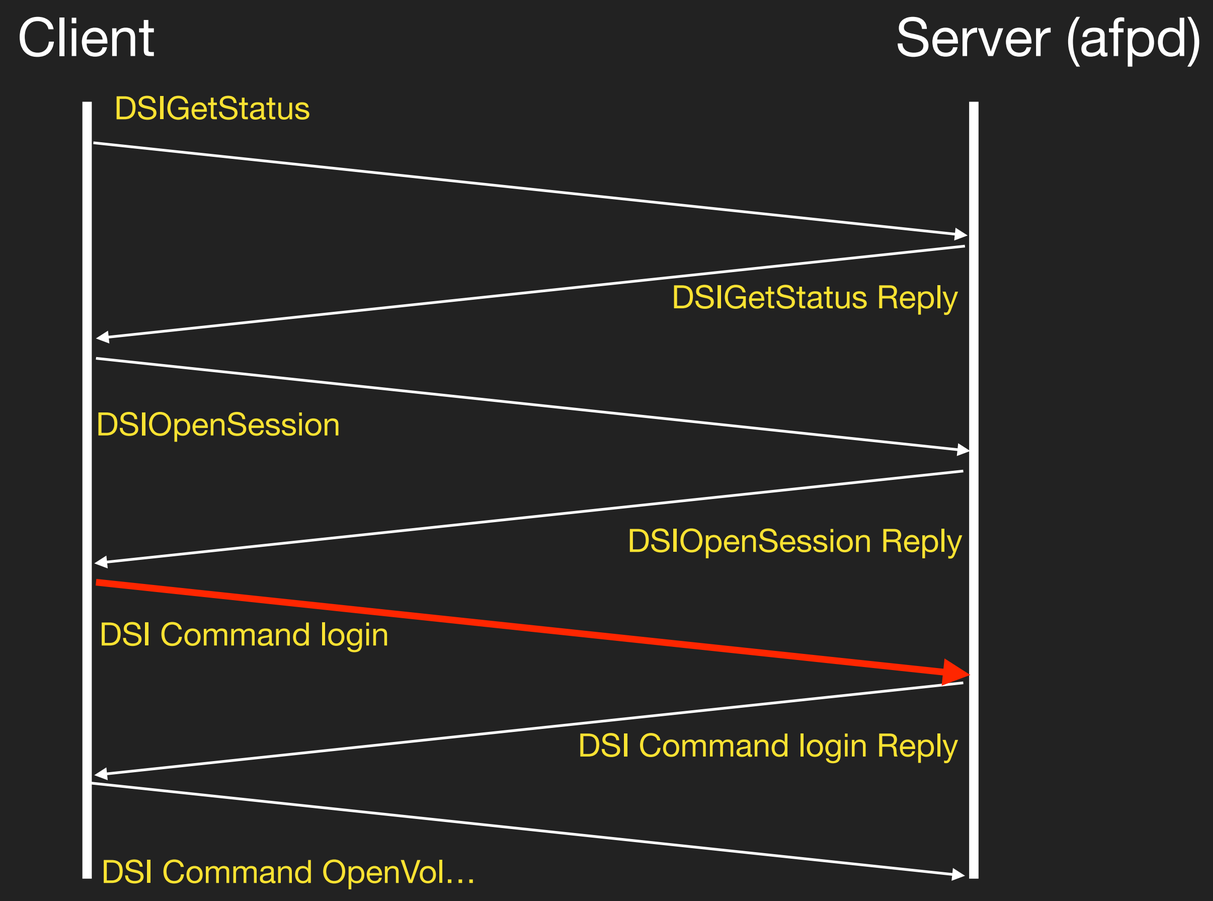
- DSI

```
/* server and client quanta */
#define DSI_DEFQUANT        2           /* default attention quantum size */
#define DSI_SERVQUANT_MAX   0xffffffff  /* server quantum */
#define DSI_SERVQUANT_MIN   32000       /* minimum server quantum */
#define DSI_SERVQUANT_DEF   0x100000L   /* default server quantum (1 MB) */
```

Netatalk

Vulnerability

- dsi_stream_receive



DEVCORE SECURITY CONSULTING

Netatalk

Vulnerability

- dsi_stream_receive - heap overflow

```
int dsi_stream_receive(DSI *dsi)
{
  ...
  /* make sure we don't over-write our buffers. */
  dsi->cmdlen = MIN(ntohl(dsi->header.dsi_len), dsi->server_quantum);

  /* Receiving DSIWrite data is done in AFP function, not here */
  if (dsi->header.dsi_data.dsi_doff) {
      LOG(log_maxdebug, logtype_dsi, "dsi_stream_receive: write request");
      dsi->cmdlen = dsi->header.dsi_data.dsi_doff;
  }

  if (dsi_stream_read(dsi, dsi->commands, dsi->cmdlen) != dsi->cmdlen)
    return 0;
```

Netatalk

Vulnerability

- dsi_stream_receive - heap overflow

```
int dsi_stream_receive(DSI *dsi)
{
  ...
  /* make sure we don't over-write our buffers. */
  d

  /
  i
                                                    ;
    dsi->cmdlen = dsi->header.dsi_data.dsi_doff;
  }

  if (dsi_stream_read(dsi, dsi->commands, dsi->cmdlen) != dsi->cmdlen)
    return 0;
```

Heap overflow

Size = 0x 100000

Can be controlled

Netatalk

Exploitation

- Memory Allocator in DSM
  - The memory allocator used by Netatalk in DSM is glibc 2.20
  - When the malloc size exceeds 0x20000, mmap will be used to allocate memory space
    - It will use mmap to allocate dsi->command
      - Default 0x100000

Netatalk

Exploitation

```
...
0x00007fcadb716000 0x00007fcadb719000 r-xp  /usr/lib/libdl-2.20-2014.11.so
0x00007fcadb719000 0x00007fcadb918000 ---p  /usr/lib/libdl-2.20-2014.11.so
0x00007fcadb918000 0x00007fcadb919000 r--p  /usr/lib/libdl-2.20-2014.11.so
0x00007fcadb919000 0x00007fcadb91a000 rw-p  /usr/lib/libdl-2.20-2014.11.so
0x00007fcadb91a000 0x00007fcadb999000 r-xp  /usr/lib/libatalk.so.17.0.0
0x00007fcadb999000 0x00007fcadbb99000 ---p  /usr/lib/libatalk.so.17.0.0
0x00007fcadbb99000 0x00007fcadbb9a000 r--p  /usr/lib/libatalk.so.17.0.0
0x00007fcadbb9a000 0x00007fcadbb9d000 rw-p  /usr/lib/libatalk.so.17.0.0
0x00007fcadbb9d000 0x00007fcadbbae000 rw-p  mapped
0x00007fcadbbae000 0x00007fcadbbcf000 r-xp  /usr/lib/ld-2.20-2014.11.so
0x00007fcadbda9000 0x00007fcadbdce000 rw-p  mapped
0x00007fcadbdce000 0x00007fcadbdcf000 r--p  /usr/lib/ld-2.20-2014.11.so
0x00007fcadbdcf000 0x00007fcadbdd1000 rw-p  /usr/lib/ld-2.20-2014.11.so
0x00007ffdc2eed000 0x00007ffdc2f0e000 rw-p  [stack]
0x00007ffdc2f65000 0x00007ffdc2f68000 r--p  [vvar]
0x00007ffdc2f68000 0x00007ffdc2f6a000 r-xp  [vdso]
0xffffffffff600000 0xffffffffff601000 r-xp  [vsyscall]
```

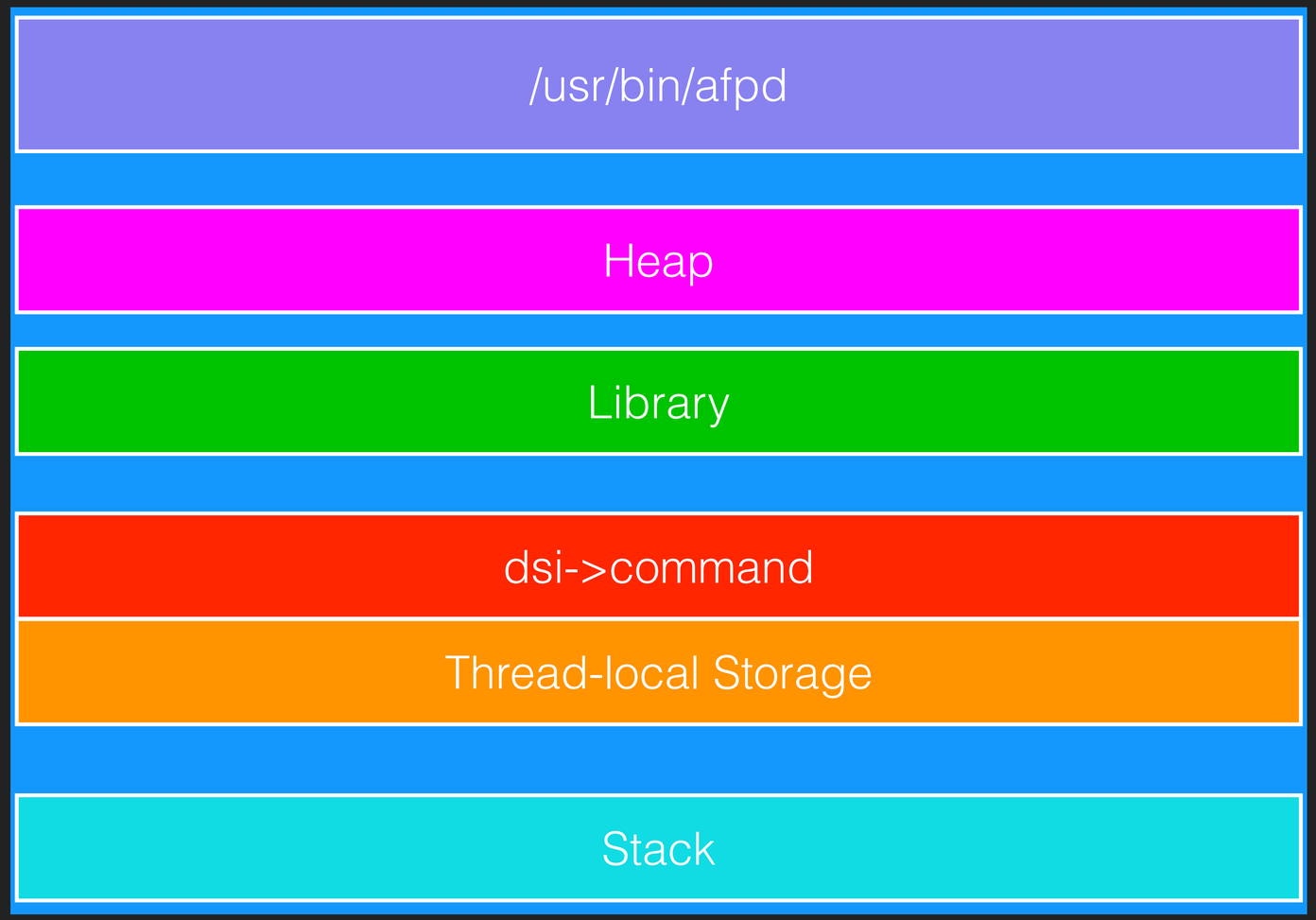
dsi->commands and TLS

DEVCORE SECURITY CONSULTING

Netatalk

Exploitation

Memory Layout of afpd



DEVCORE SECURITY CONSULTING

Netatalk

Exploitation

- Thread-local Storage (TLS)
  - Thread local variable、destructor 、 main arena pointer
  - Each thread will have one
  - It will be released when Thread is destroyed

Netatalk

Exploitation

- Target in TLS
  - main_arena
    - forge main_arena to achieve arbitrary writing
  - pointer guard
  - tls_dtor_list
  - ...

Netatalk

Exploitation

- Overwrite tls_dtor_list
  - This technique was proposed by project zero in 2014
  - Overwrite tls_dtor_list in TLS to control RIP
    - It will be triggered after the process exit.

Netatalk

Exploitation

- Overwrite tls_dtor_list
  - dtor_list

```
struct dtor_list
{
    dtor_func func;
    void *obj;
    struct link_map *map;
    struct dtor_list *next;
};
```

Netatalk

Exploitation

```
__call_tls_dtors (void)
{
  while (tls_dtor_list)
    {
      struct dtor_list *cur = tls_dtor_list;
      tls_dtor_list = tls_dtor_list->next;

      cur->func (cur->obj);

      __rtld_lock_lock_recursive (GL(dl_load_lock));

      /* Allow DSO unload if count drops to zero.  */
      cur->map->l_tls_dtor_count--;
      if (cur->map->l_tls_dtor_count == 0 && cur->map->l_type == lt_loaded)
        cur->map->l_flags_1 &= ~DF_1_NODELETE;

      __rtld_lock_unlock_recursive (GL(dl_load_lock));

      free (cur);
    }
}
```

DEVCORE SECURITY CONSULTING

Netatalk

Exploitation

- Overwrite tls_dtor_list
  - But in the new version of glibc, the function pointer in this structure is protected by pointer_guard
    - We need to leak pointer_guard to control the RIP

Netatalk

Exploitation

```
__call_tls_dtors (void)
{
  while (tls_dtor_list)
    {
      struct dtor_list *cur = tls_dtor_list;
      dtor_func func = cur->func;
#ifdef PTR_DEMANGLE
      PTR_DEMANGLE (func);
#endif

      tls_dtor_list = tls_dtor_list->next;
      func (cur->obj);

      /* Ensure that the MAP dereference happens before
    l_tls_dtor_count decrement.  That way, we protect this access from a
    potential DSO unload in _dl_close_worker, which happens when
    l_tls_dtor_count is 0.  See CONCURRENCY NOTES for more detail.  */
      atomic_fetch_add_release (&cur->map->l_tls_dtor_count, -1);
      free (cur);
    }
}
```

DEVCORE SECURITY CONSULTING

Netatalk

Exploitation

- Overwrite tls_dtor_list
  - In fact, the pointer_guard used in decoding is also on
    - pointer_guard is in the tcbhead_t structure on TLS
  - Therefore, we can overwrite tls_dtor_list and clear the pointer_guard at the same time.
    - We don’t have to deal with the pointer_guard problem.

Netatalk

Exploitation

- tcbhead_t
  - Thread Control Block (TCB)
  - A thread descriptor
  - Used to store various information of thread
  - On x86_64 linux, fs register will point to this structure
    - When we access thread local variable, it accesses through fs register

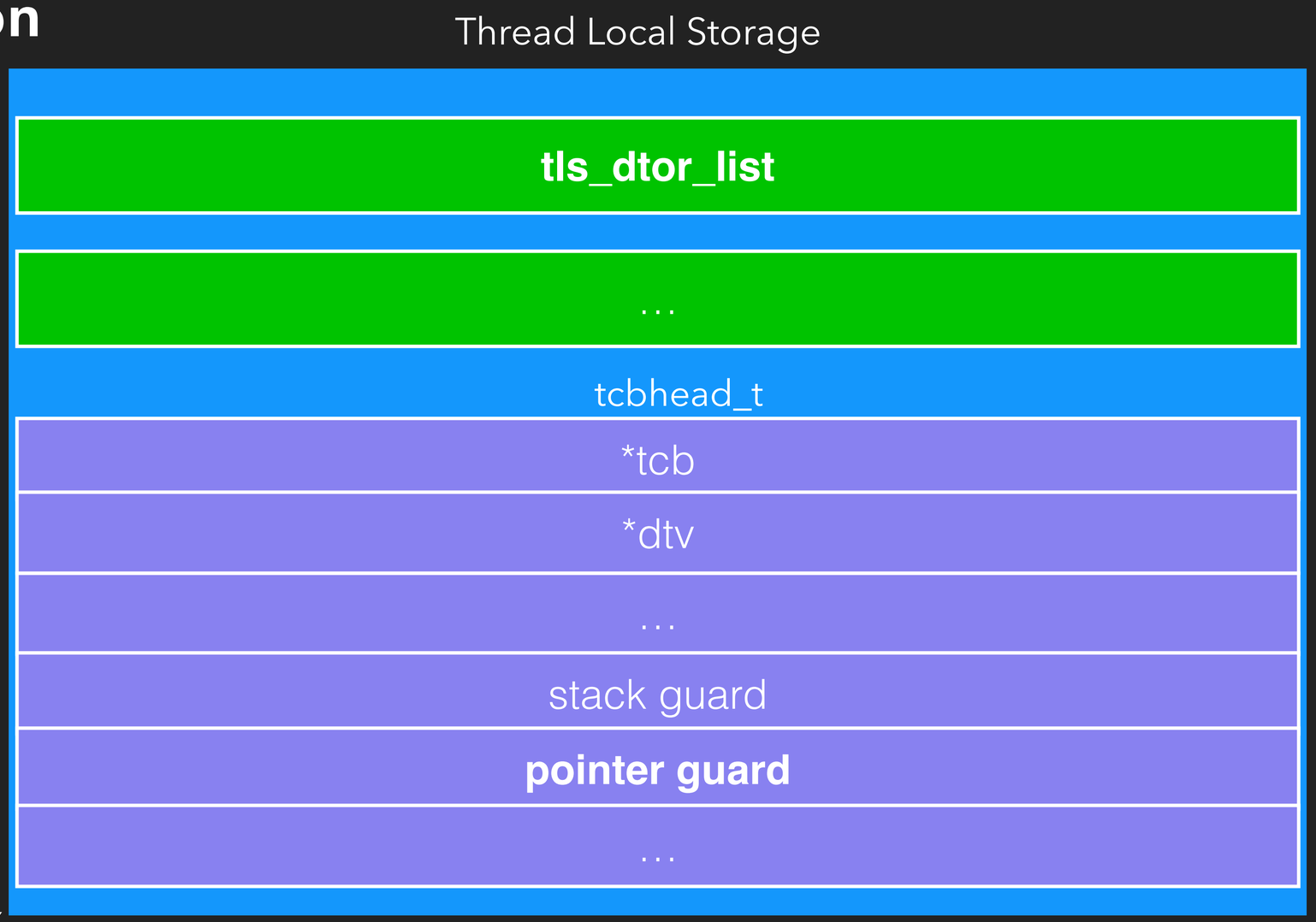
Netatalk

Exploitation

```
typedef struct
{
  void *tcb;    /* Pointer to the TCB.  Not necessarily the
                   thread descriptor used by libpthread.  */
  dtv_t *dtv;
  void *self;   /* Pointer to the thread descriptor.  */
  int multiple_threads;
  int gscope_flag;
  uintptr_t sysinfo;
  uintptr_t stack_guard;
  uintptr_t pointer_guard;
  ...
} tcbhead_t;
```

Netatalk

Exploitation



DEVCORE SECURITY CONSULTING

Netatalk

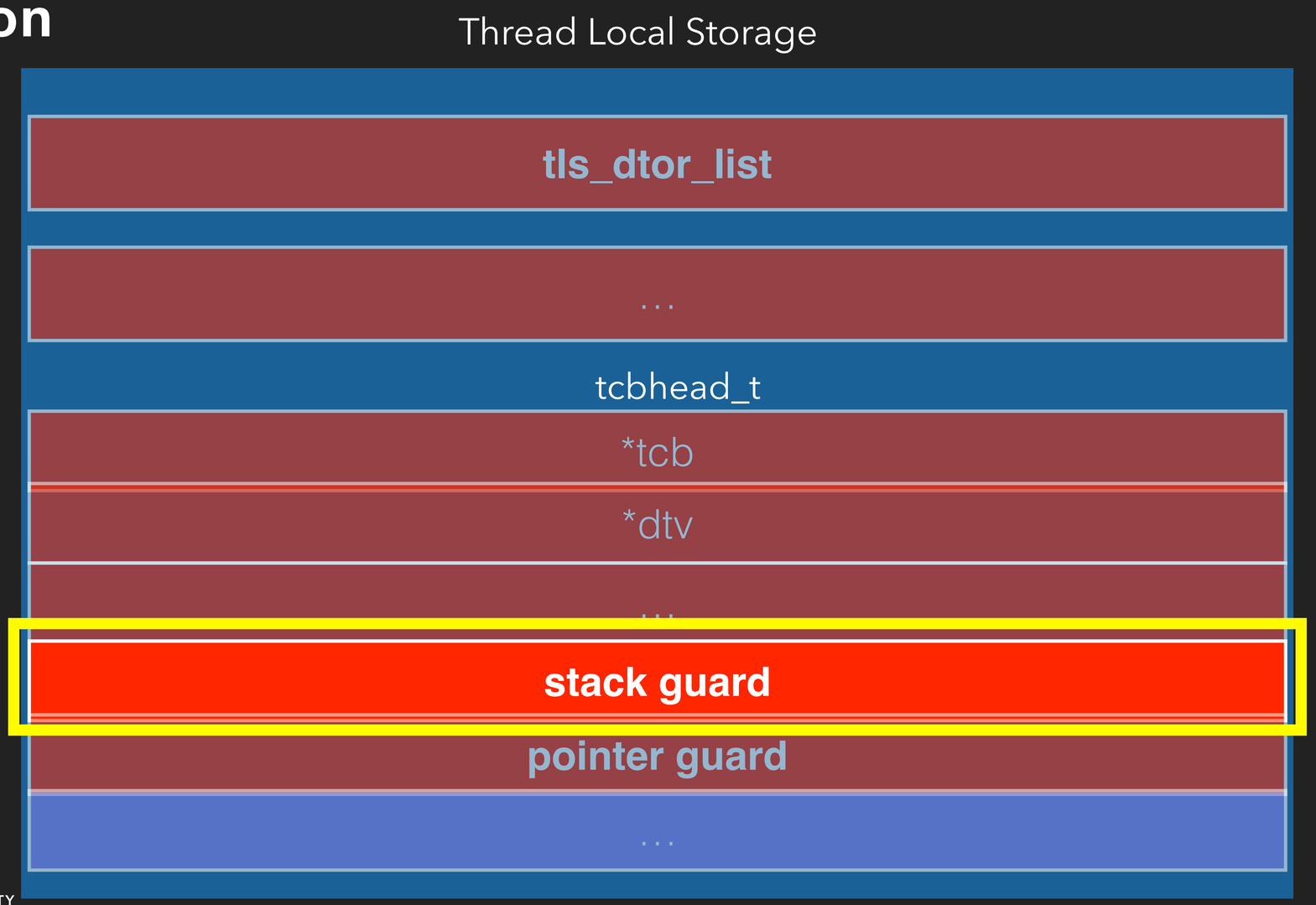
Exploitation

Thread Local Storage

DEVCORE SECURITY CONSULTING

Netatalk

Exploitation



DEVCORE SECURITY CONSULTING

Netatalk

Exploitation

```
int dsi_stream_receive(...){
  //mov rax, fs:28h
  //mov [rsp+18h],rax
  dsi_stream_read(...){
    readt(...){
      recv(fd,dsi->commands,xxx)
    }
  }
  //mov rcx, [rsp+18h]
  //xor rcx, fs:28h
  //jnz stack_chk_fail
}
```

Take stack_guard from TLS

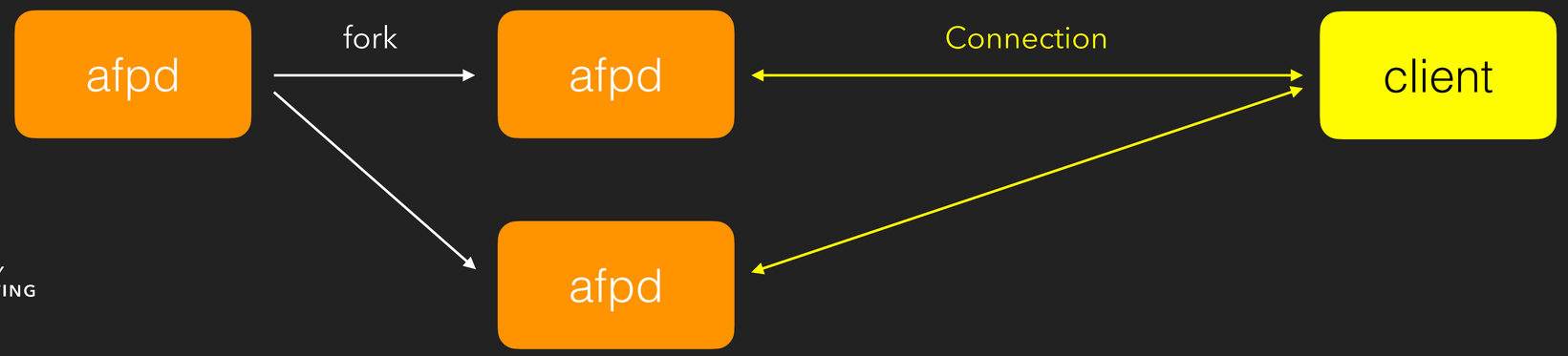
overflow

Check whether the stack_guard on stack is identical to the one in TLS

Netatalk

Exploitation

- Bypass stack guard
  - Netatalk forks a new process for each user's connection
  - The memory address and stack guard of each connection are the same as the parent process
  - We can leak the stack guard bytes one by one with brute-force



DEVCORE SECURITY CONSULTING

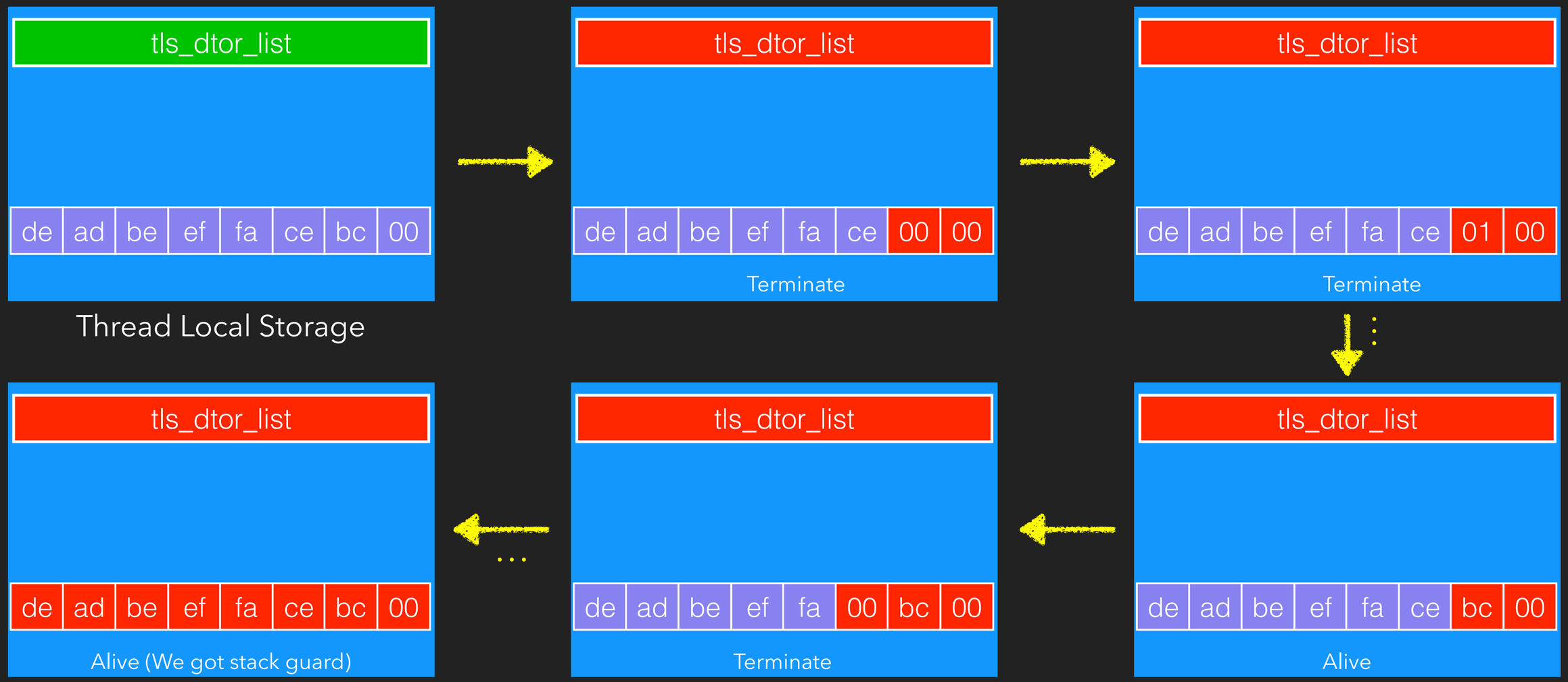
Netatalk

Exploitation

- Bypass stack guard - Brute-force stack guard
  - After overflow, we can overwrite only the last byte of the stack guard in TLS.
  - Each connection overwrites a different value. Once it is different from the stack guard, the connection will be disconnected.
    - We can determine whether the value we overwrite is correct based on its behavior.

Netatalk

Exploitation



Netatalk

Exploitation

- Construct the _dtor_list to control RIP
  - In our target, afpd does not enable PIE
    - The _dtor_list can be constructed in the data section of afpd
  - When using the login function of dhx2, the username will be copied to the global username buffer
    - We can forge this structure to a known fixed location along with login username

Netatalk

Exploitation

```c
static int passwd_login(void *obj, struct passwd **uam_pwd,
                        char *ibuf, size_t ibuflen,
                        char *rbuf, size_t *rbuflen)
{
    char *username;
    size_t len, ulen;

    *rbuflen = 0;

    /* grab some of the options */
    if (uam_afpserver_option(obj, UAM_OPTION_USERNAME, (void *) &username, &ulen) < 0) {
        LOG(log_info, logtype_uams, "DHX2: uam_afpserver_option didn't meet uam_option_use
            strerror(errno));
        return AFPERR_PARAM;
    }
    ...
    memcpy(username, ibuf, len );
    ibuf += len;
    username[ len ] = '\0';
    ...
    return (login(obj, username, ulen, uam_pwd, ibuf, ibuflen, rbuf, rbuflen));
}
```

Copy payload to username
which is pointed to a fixed address

uames_dhx2_passwd.c

Netatalk

Exploitation

- Trigger Exit
  - DSICloseSession

```
case DSIFUNC_CLOSE:
    LOG(log_debug, logtype_afpd, "DSI: close session request");
    afp_dsi_close(obj);
    LOG(log_note, logtype_afpd, "done");
    exit(0);
```

Netatalk

Exploitation

- Control RIP to get reverse shell
  - In the target glibc, it uses __tls_get_addr to get the tls_dtor_list and this function takes the value from the div field in the tcbhead_t
    - We need to construct it in a fixed address together
    - Although there is no system available in afpd, execl is available
      - The parameters are a bit more complicated

Netatalk

Exploitation

```
result = (struct_result *)__tls_get_addr((__int64)&stru_39ED90);
for ( dtor = (struct dtor_list *)result->tls_dtor_list; dtor; dtor = (struct dtor_list *)result
{
  v2 = (struct_result *)__tls_get_addr((__int64)&stru_39ED90);
  obj = dtor->obj;
  dtor_func = (void (__fastcall *)(void *))(__readfsqword(0x30u) ^ __ROR8__(dtor->func, 17));
  v2->tls_dtor_list = &dtor->next->func;
  dtor_func(obj);
```

Netatalk

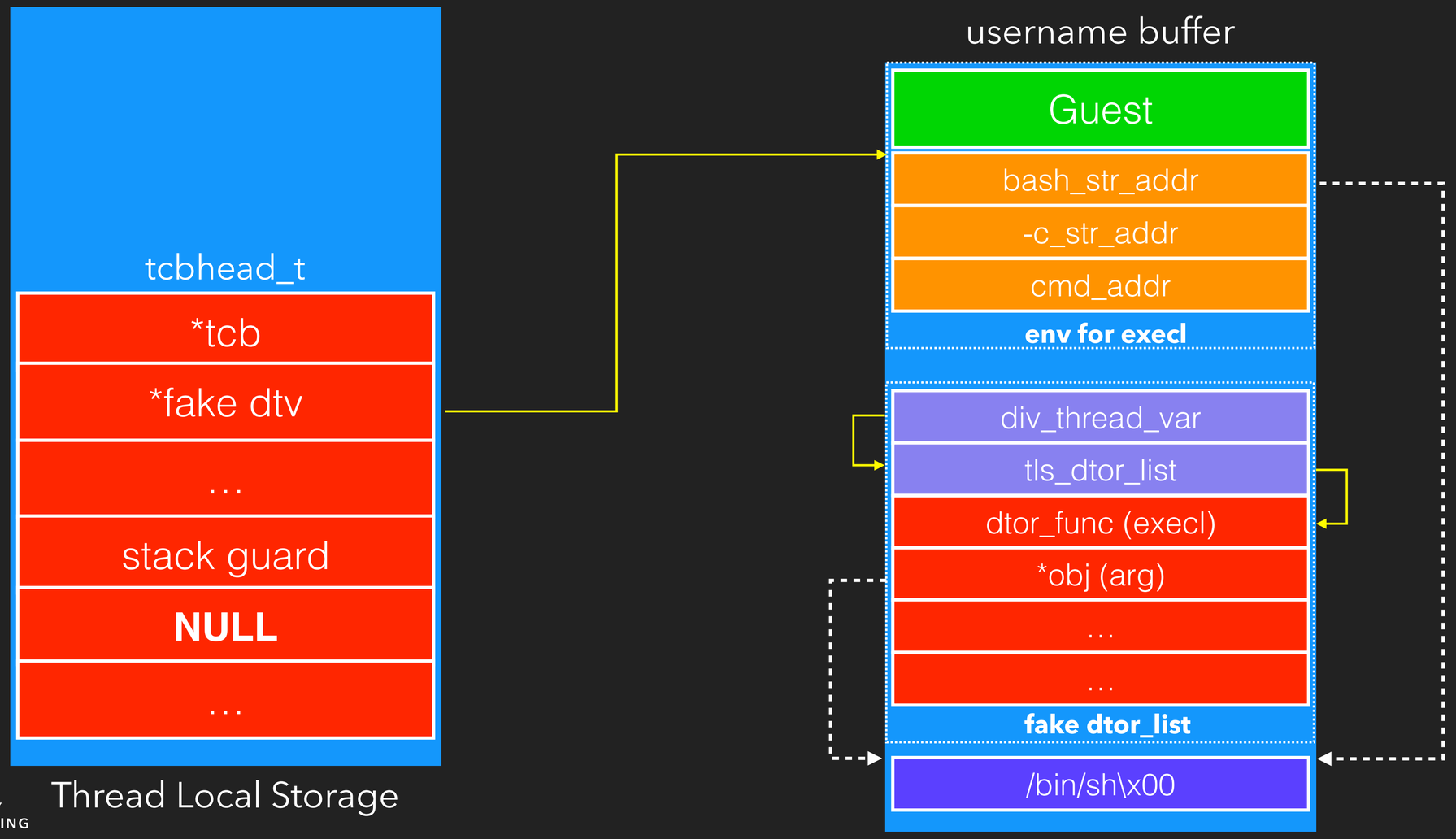
Exploitation

```
void *
__tls_get_addr (GET_ADDR_ARGS)
{
  dtv_t *dtv = THREAD_DTV ();

  if (__glibc_unlikely (dtv[0].counter != GL(dl_tls_generation)))
    return update_get_addr (GET_ADDR_PARAM);

  void *p = dtv[GET_ADDR_MODULE].pointer.val;

  if (__glibc_unlikely (p == TLS_DTV_UNALLOCATED))
    return tls_get_addr_tail (GET_ADDR_PARAM, dtv, NULL);

  return (char *) p + GET_ADDR_OFFSET;
}
```

tcb->div

Netatalk

Exploitation

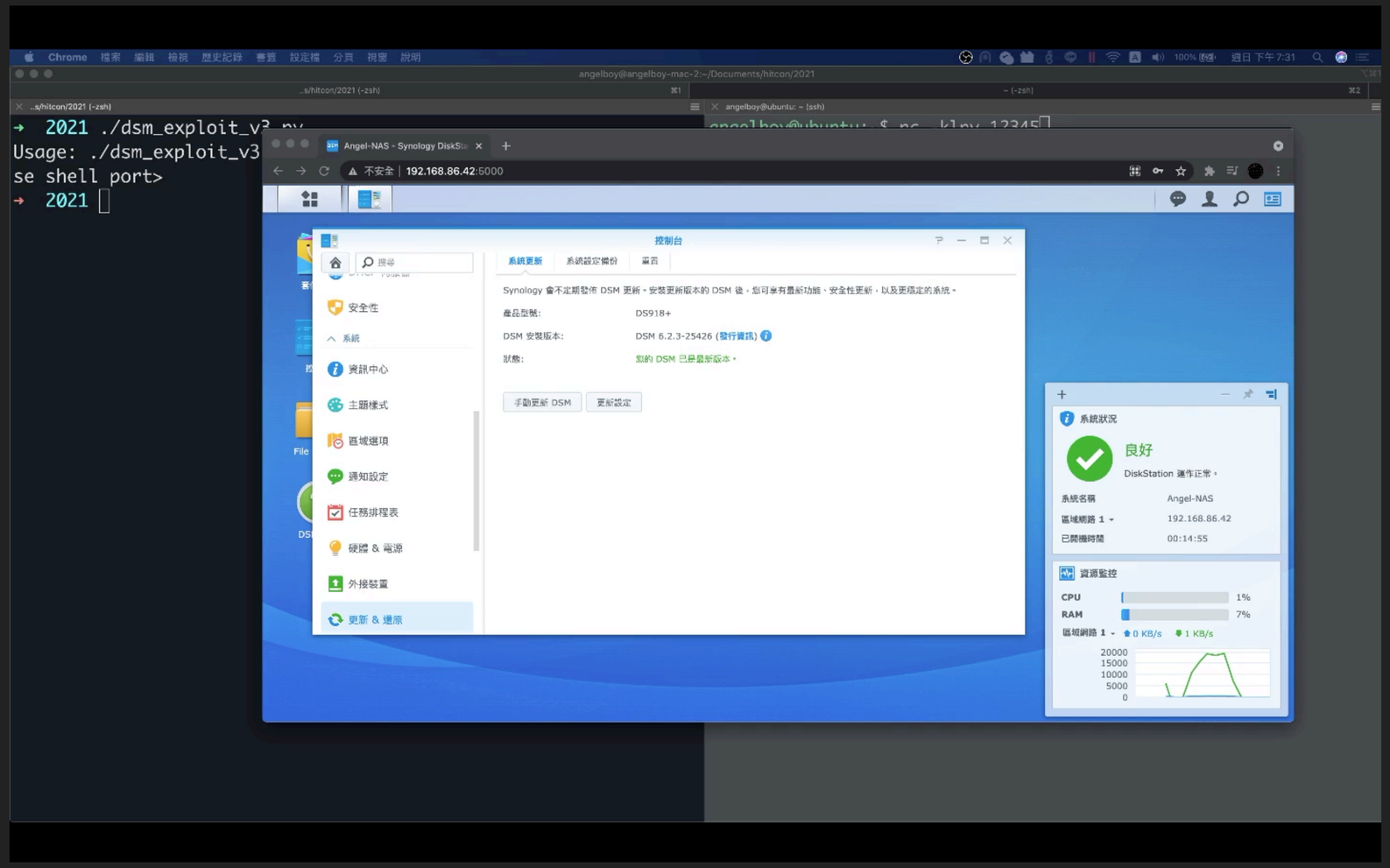


DEVCORE SECURITY CONSULTING

Netatalk

Exploitation

- Demo



DEVCORE SECURITY CONSULTING

Netatalk

Exploitation

- Remark
  - In general, PIE protection is enabled, and it is not easy to construct _dtor_list in a known address
  - In fact, you can also use a similar method to leak out the libc address
    - It is still exploitable
  - The vulnerability not only affects Synology, but also most devices that use Netatalk
    - QNAP
    - Asustor

Netatalk

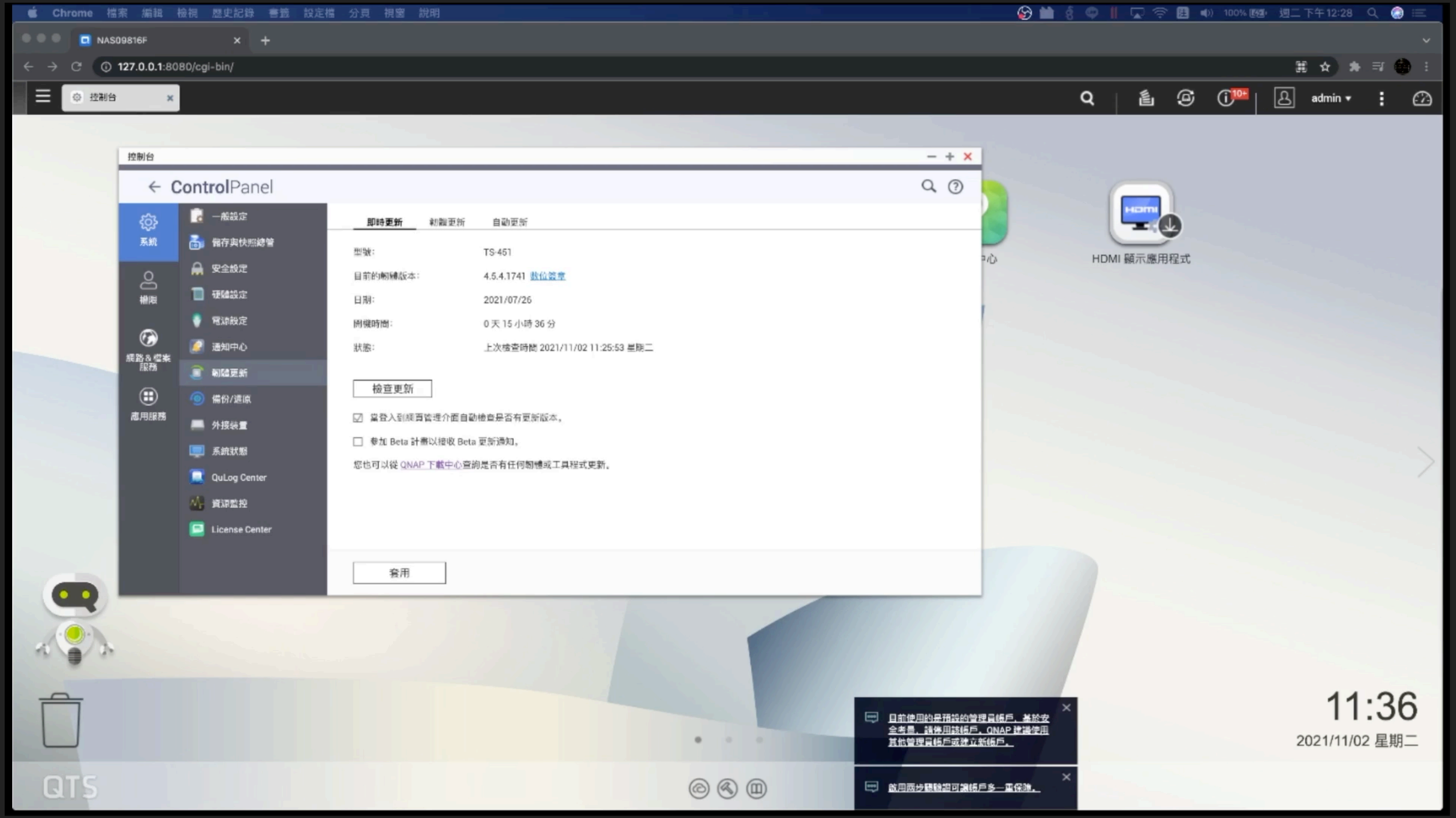
Exploitation

- Netatalk in QNAP
  - We tested on TS451
    - QTS 4.5.4.1741
    - Not enable by default
  - Protection
    - No Stack Guard
    - No PIE

Netatalk

Exploitation

- Demo



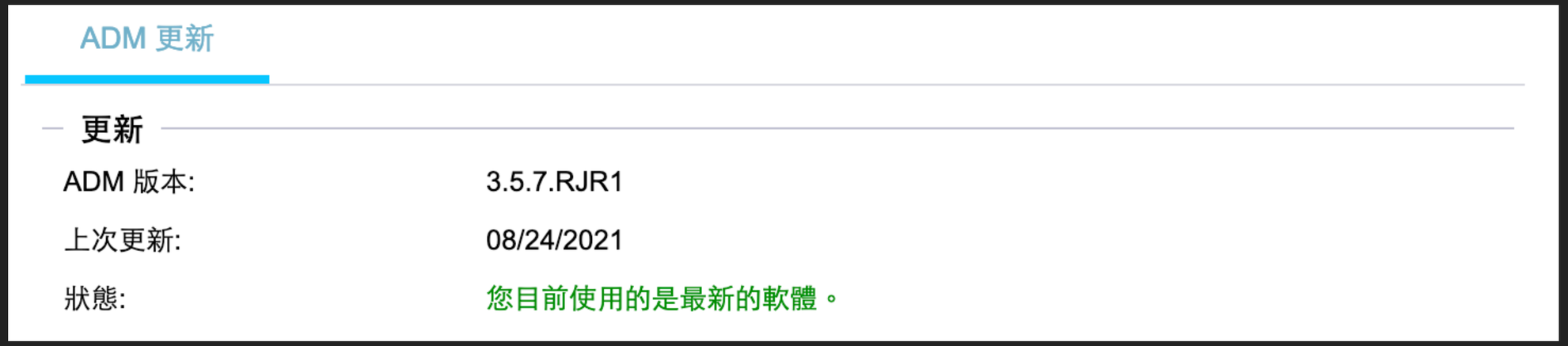
Netatalk

Exploitation

- Netatalk in Asustor
  - We tested on AS5202T
    - ADM 3.5.7.RJR1
    - Not enable by default
  - Protection
    - No Stack Guard
    - No PIE

Netatalk

Exploitation




```
angelboy@217-x:~$ nc -kln 55688 -v
Listening on 0.0.0.0 55688
Connection received on 192.168.86.49 38988
uname -a
Linux AS5202T-D305-Angel 4.14.x #1 SMP Tue Jul 27 00:31:17 CST 2021 x86_64 GNU/Linux
```

Netatalk

Exploitation

- Summary
  - Synology is exploitable by default
  - Although QNAP and Asustor are not turned on by default, many Mac users still turn it on for convenience
  - Your NAS is my NAS !

Netatalk

Exploitation

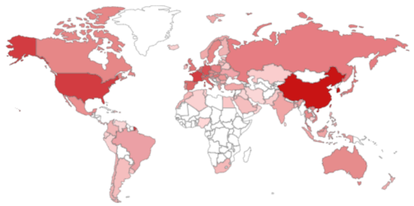
SHODAN | Explore | Downloads | Pricing

port:548 product:"Netatalk"

TOTAL RESULTS

131,672

TOP COUNTRIES



Korea, Republic of	26,800
China	24,062
Taiwan	13,004
United States	9,836
France	9,379

More...

View Report | View on Map

New Service: Keep track of what you have connected to the Internet. Check out **Shodan Monitor**

Korea Telecom

Korea, Republic of, Seoul

```
AFP:
  Server Name: biofarm_nas
  UTF-8 Server Name: biofarm_nas
  Machine Type: Netatalk3.1.8
  Network Addresses:
    192.168.0.4
  AFP Versions:
    AFP2.2
    AFP3.1
    AFP3.2
    AFP3.3
    AFP3.4
    AFPX03
  UAMs:
    DHCAST128
    DHX2
  Server Signature: da5f4f8ddf34c73067aae88e6689b0f3...
```

Mitigation

- Update
  - The above three have been patched, please update to the latest
  - Synology
    - https://www.synology.com/zh-hk/security/advisory/Synology_SA_20_26
  - QNAP
    - https://www.qnap.com/en/security-advisory/qsa-21-50
  - Asustor
    - https://www.asustor.com/service/release_notes#ADM%203.5.7.RKU2

Mitigation

- Disable AFP
  - It is best to disable it directly,
    - The project is almost not under maintenance
    - The risk of continuing to use it is extremely high
  - SMB is relatively safe
    - But it is recommended to only open the intranet

Mitigation

- For vendor
  - When using an Open Source project, it’s best to review the code by yourself
  - Turn on various protections as much as possible to increase the difficulty of exploitation

Conclusion

- We found a critical vulnerability in Netatalk
  - We can pwn many NAS in the world
- Netatalk is a new backdoor in NAS !

To be continue

- It not only one vulnerability !
- We will release more vulnerability in the future

Reference

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-31439
- https://googleprojectzero.blogspot.com/2014/08/the-poisoned-nul-byte-2014-edition.html
- https://www.zerodayinitiative.com/advisories/ZDI-21-492/
- https://www.zerodayinitiative.com/blog/2020/7/28/announcing-pwn2own-tokyo-2020-live-from-toronto
- https://www.youtube.com/watch?v=2inCVgsosyk

Q & A

Thank you for listening

@scwuaptx

DEVCORE SECURITY CONSULTING