Contents

1. Introduction .......................................................................................................................... 3
2. Description ............................................................................................................................ 3
3. Exploiting the vulnerability .................................................................................................... 4
4. Solution .................................................................................................................................. 4
5. References ............................................................................................................................. 6
1. Introduction

PuTTY is a free and open-source client program for the SSH, Telnet and Rlogin network protocols. These protocols are all used to run remote session on a computer, over a network.

2. Description

CVE-2017-6542 was a heap-corrupting integer overflow bug in PuTTY, which affected versions prior to 0.68. The vulnerability was reported by Tim Kosse.

The bug is in the `ssh_agent_channel_data` function which processes messages sent by remote SSH clients to a forwarded agent connection.

```c
static int ssh_agent_channel_data(struct ssh_channel *c, char *data, int length)
{
    while (length > 0) {
        if (c->u.a.lensofar < 4) {
            unsigned int l = min(4 - c->u.a.lensofar, (unsigned)length);
            memcpy(c->u.a.msglen + c->u.a.lensofar, data, l);
            data += l;
            length -= l;
            c->u.a.lensofar += l;
        }
        if (c->u.a.lensofar == 4) {
            c->u.a.totalen = 4 + GET_32BIT(c->u.a.msglen);
            c->u.a.message = snwcn(c->u.a.totalen, unsigned char);
            memcpy(c->u.a.message, c->u.a.msglen, 4);
        }
        if (c->u.a.lensofar >= 4 && length > 0) {
            unsigned int l = min(c->u.a.totalen - c->u.a.lensofar,
                                 (unsigned)length);
            memcpy(c->u.a.message + c->u.a.lensofar, data, l);
            data += l;
            length -= l;
            c->u.a.lensofar += l;
        }
        if (c->u.a.lensofar == c->u.a.totalen) {
            void *reply;
            int replylen;
            c->u.a.outstanding_requests++;
            c->u.a.pending = agent_query(
                c->u.a.message, c->u.a.totalen, &reply, &replylen,
                ssh_agentf_callback, c);
            if (!c->u.a.pending)
                ssh_agentf_callback(c, reply, replylen);
            sfrc(c->u.a.message);
            c->u.a.message = NULL;
            c->u.a.lensofar = 0;
        }
    }
    return 0; /* agent channels never back up */
}
```
Every message in agent protocol starts with a 32-bit length field, which only includes the message, but not the length field itself. In order to get the length field PuTTY adds 4 to the received length value, which will only return the message length inclusive of everything that message contained. This addition of 4 would be fine if the addition of 4 to the message length size is smaller than the maximum unsigned long size allowed in C, which is 4294967295, but if the addition exceeds the maximum size of unsigned long then the value wraps around back to 0 and will add remainder to it.

If the incorrect value of the length is received, it will cause PuTTY to record a value of the total message size a lot smaller than it actually is. PuTTY then would assume that writing to heap is safe, but in fact it will overwrite the allocated heap block. This will cause PuTTY to crash.

3. Exploiting the vulnerability

Heap overflow can be a security threat, which may lead to execution of attacker’s code.

On a Unix machine that has socat network utility installed, can be easily determined if PuTTY is affected by this vulnerability by running a following shell command:

```
(\ec\o\h\-\ne '\xFF\xFF\xFF\xFD\x0B'; cat /dev/zero) | socat stdio unix-connect:SSH_AUTH_SOCK
```

If PuTTY crashes to this shell command then it means that PuTTY is vulnerable to heap overflow bug.

4. Solution

The bug was fixed in 0.68 release in January 2017.

Rather than expecting entire message to be loaded at once, there was added a buffer chain, where each request the message parts are loaded. Buffer chain allows multiple buffers to be stored, this allows more data to be stored. Following code checks if the entire message is read, if it’s not then the size of the message is returned 0, so it can determine that client is still sending data.

```c
static int ssh_agent_channel_data(struct ssh_channel *c, char *data,
                                int length)
{
    buchain_add(&c->u.a.inbuffer, data, length);
    ssh_agentf_try_forward(c);

    return (c->u.a.pending ? buchain_size(&c->u.a.inbuffer) : 0);
}
```
Also there was added a check if the length field in message or entire message is not loaded then the returned size would be set to 0 until the buffer chain consists entire message including the message length as shown below. If the datalen is less than 4 the code breaks out of loop and the partial message is not read. After it is clear that entire message is read then it gets added to the buffer chain.

```c
while (!) {
    // Try to extract a complete message from the input buffer.
    data(len = bufchain_size(&c->u.a.inbuffer);
    if (data(len < 4) /* not even a length field available yet */
        break;

    bufchain_fetch(&c->u.a.inbuffer, msglen, 4);
    lengthfield = GET_32BIT(msglen);
    if (lengthfield > data(len - 4)) /* a whole message is not yet available */
        break;

    messagelen = lengthfield + 4;
    message = snew(messagelen, unsigned char);
    bufchain_fetch(&c->u.a.inbuffer, message, messagelen);
    bufchain_consume(&c->u.a.inbuffer, messagelen);
    c->u.a.pending = agent_query(
        message, messagelen, &reply, &replylen, ssh_agentf_callback, c);
    sfree(message);

    if (!c->u.a.pending)
        return; /* agent_query promised to reply in due course */

    ssh_agentf_got_response(c, reply, replylen);
}
```

Since PuTTY supports other protocols like Telnet and Rlogin, then another solution would be to turn off the SSH agent forwarding, which should be turned off by default, and use the Telnet for remote access, because this bug only affects the SSH protocol.
5. References


https://git.tartarus.org/?p=simon/putty.git;a=commitdiff;h=4ff22863d895cb7ebfced4cf923a012a614adaa8 (30.04.2018)
