Research into CVE-2017-2671
Ott Oopkaup

Supervisor: Kristjan Krips
Introduction
The Linux kernel is large and therefore has many exploits and bugs, some of which are tracked, some are not. This particular one (CVE-2017-2671) is not a major exploit, but has existed for quite some while – the current Linux kernel version 4.11, and this exploit has existed from 2.6 until 4.10. The exploit relies on improper socket locking in the Linux kernel. /net/ipv4/ping packet is used to manage ICMP ping messages, but due to an oversight, if the submitted ping request is invalid, it allows to leave the socket hanging.

The vulnerability
Presented as CVE-2017-2671 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-2671), the ping_unhash function in /net/ipv4/ping.c in the Linux kernel is too late in obtaining socket and ping lock, and therefore cannot ensure that disconnect calls are safe. This potentially allows users to overwhelm the system and cause a DOS attack through a kind-of slow loris attack, or a kernel panic through memory overflow.

Currently presented in the Linux kernel github, we can see the original code for ping_unhash (https://github.com/torvalds/linux/commit/43a6684519ab0a6c52024b5e25322476cabad893?diff=split).

void ping_unhash(struct sock *sk) {
    struct inet_sock *isk = inet_sk(sk);
    pr_debug("ping_unhash(isk=%p, isk->num=%u)\n", isk, isk->inet_num);
    if (sk_hashed(sk)) {
        - write_lock_bh(&ping_table.lock);
        hlist_nulls_del(&sk->sk_nulls_node);
        sk_nulls_node_init(&sk->sk_nulls_node);
        sock_put(sk);
        isk->inet_num = 0;
        isk->inet_sport = 0;
        sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
        - write_unlock_bh(&ping_table.lock);
    }
}
EXPORT_SYMBOL_GPL(ping_unhash);

We can quite clearly see the problem here, as the function is called, the lock for ping_table is obtained too late – right now the socket is locked after parsing has started.

The exploit
The exploit, although available only locally, can cause a Denial-of-Service attack. In essence, it allows the user to leave the socket occupied and the ping_table locked. Once the table is locked, further access to it is limited and the system will deny further connections. While denying ping seems minor, there are many protocols that rely on ping to determine the state, reachability and round trip time to
the server. A DOS attack to ping will mainly disrupt time-based applications, that rely on consistent and time-sensitive packets.

**The fix**

Surprisingly enough, the fix to this problem is quite simple – whilst disconnect is not allowed during the current operation, moving `write_lock_bh` outside the function is enough that any problem with unproper calls allows disconnect to be called again.

Corrected code:

```c
void ping_unhash (struct sock *sk) {
    struct inet_sock *isk = inet_sk(sk);
    pr_debug("ping_unhash(isk=%p,isk->num=%u)\n", isk, isk->inet_num);
    write_lock_bh(&ping_table.lock);
    if (sk_hashed(sk)) {
        hlist_nulls_del(&sk->sk_nulls_node);
        sk_nulls_node_init(&sk->sk_nulls_node);
        sock_put(sk);
        isk->inet_num = 0;
        isk->inet_sport = 0;
        sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
    }
    write_unlock_bh(&ping_table.lock);
}
EXPORT_SYMBOL_GPL(ping_unhash);
```
Sources:

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-2671

https://github.com/torvalds/linux/commit/43a6684519ab0a6c52024b5e25322476cabad893?diff=split

http://www.securityfocus.com/bid/97407