strongSwan - Bug #301

No data traffic in tunnel

01.03.2013 10:53 - Stefan Tomas

Status:	Closed	**Start date:**	01.03.2013
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	android		
Target version:	5.0.3		
Affected version:	5.0.2	**Resolution:**	Fixed

Description

I have seen this issue happening quite rearely, and only after re-authentication. There is no traffic in or out of the tunnel after this. I've added some logs in the code to better understand the issue.
Here is the log showing this:

```
03-01 04:20:53.235: I/charon(9751): 12[IKE] CHILD_SA android{8} established with SPIs 7c1975db_i cd71364b_o and TS 172.1.1.11/32 === 0.0.0.0/0 
03-01 04:20:53.235: I/charon(9751): 12[DMN] setting up TUN device for CHILD_SA android{8}
03-01 04:20:53.325: I/charon(9751): 12[DMN] closing TUN device: 55
03-01 04:20:53.325: I/charon(9751): 12[DMN] successfully created TUN device: 53
03-01 04:20:53.325: I/charon(9751): 12[IKE] received AUTH_LIFETIME of 1232s, scheduling reauthentication in 632s
03-01 04:20:53.325: I/charon(9751): 12[IKE] peer supports MOBIKE
03-01 04:21:26.415: I/charon(9751): 06[DMN] select on TUN device failed: Bad file number, tunfd: 55, this->tunfd: 53
03-01 04:31:08.405: I/charon(9751): 05[NET] received packet: from 10.10.10.5[4500] to 10.10.10.11[40894] (348 bytes)
03-01 04:31:08.420: I/charon(9751): 05[ENC] parsed CREATE_CHILD_SA request 0 [ N(REKEY_SA) SA No TSi TSr ]
```

I've added logging of TUN device fd when it is created and closed and also tunfd mismatch on select. This happens in android_service.c handle_plain(). Here is code snippet, so you would see what logging I added.

```
    old = thread_cancelability(TRUE);
    len = select(tunfd + 1, &set, NULL, NULL, &tv);
    thread_cancelability(old);

    if (len < 0)
    {
        DBG1(DBG_DMN, "select on TUN device failed: %s, tunfd: %d, this->tunfd: %d", strerror(errno), tunfd, this->tunfd);
        return JOB_REQUEUE_NONE;
    }
```

Note that failure happens good 30 seconds after the re-authentication, which is somewhat puzzling.
charon is oblivious to this problem and continues to operate normally, but the tunnel never recovers, not even after repeated re-authentications (tunnel is still there but there is no traffic). Only way to restore connectivity on the device is to disconnect and optionally reconnect the tunnel.
Here is the ip route and ip addr output:

```
shell@android:/ $ ip route
ip route
0.0.0.0/1 dev tun0  scope link
default via 10.10.10.1 dev wlan0
default via 10.10.10.1 dev wlan0  metric 320
10.10.10.0/24 dev wlan0  proto kernel  scope link  src 10.10.10.11  metric 320
10.10.10.1 dev wlan0  scope link
128.0.0.0/1 dev tun0  scope link
shell@android:/ $ ip addr
```

```
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: sit0: <NOARP> mtu 1480 qdisc noop state DOWN
    link/sit 0.0.0.0 brd 0.0.0.0
3: ip6tnl0: <NOARP> mtu 1452 qdisc noop state DOWN
    link/tunnel6 :: brd ::
4: rmnet0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ppp
5: rmnet1: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ppp
6: rmnet2: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ppp
19: p2p0: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state DORMANT qle
n 1000
    link/ether 5e:0a:5b:94:1f:d1 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::5c0a:5bff:fe94:1fd1/64 scope link
       valid_lft forever preferred_lft forever
20: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 5c:0a:5b:94:1f:d1 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.11/24 brd 10.10.10.255 scope global wlan0
    inet6 fe80::5e0a:5bff:fe94:1fd1/64 scope link
       valid_lft forever preferred_lft forever
58: tun0: <POINTOPOINT,UP,LOWER_UP> mtu 1400 qdisc pfifo_fast state UNKNOWN qlen 500
    link/none
    inet 172.1.1.11/32 scope global tun0
```

And ipsec statusall on the other end:

```
Status of IKE charon daemon (strongSwan 5.0.2, Linux 3.2.0-37-generic, x86_64):
  uptime: 21 hours, since Feb 28 13:07:17 2013
  malloc: sbrk 385024, mmap 0, used 304400, free 80624
  worker threads: 6 of 16 idle, 9/1/0/0 working, job queue: 0/0/0/0, scheduled: 4
  loaded plugins: charon sqlite aes des blowfish sha1 sha2 md4 md5 random nonce x509 revocation co
nstraints pubkey pkcs1 pkcs8 pgp dnskey pem fips-prf
  gmp xcbc cmac hmac attr kernel-netlink resolve socket-default farp stroke updown eap-identity ea
p-sim eap-aka eap-aka-3gpp2 eap-simaka-sql eap-simaka-pseudonym
  eap-simaka-reauth eap-md5 eap-gtc eap-mschapv2 eap-dynamic eap-radius eap-tls eap-ttls eap-peap
eap-tnc xauth-generic tnc-tnccs dhcp
Virtual IP pools (size/online/offline):
  172.1.1.10/28: 14/1/0
Listening IP addresses:
  192.168.10.5
  10.10.10.5
Connections:
    cool-vpn:  %any...%any  IKEv2
    cool-vpn:   local:  [coolswan1.coolbits.com] uses public key authentication
    cool-vpn:    cert:  "C=SR, O=coolbits, CN=coolswan1"
    cool-vpn:   remote: uses EAP_DYNAMIC authentication
    cool-vpn:   child:  0.0.0.0/0 === dynamic TUNNEL
Security Associations (1 up, 0 connecting):
    cool-vpn[593]: ESTABLISHED 99 seconds ago, 10.10.10.5[coolswan1.coolbits.com]...10.10.10.11[te
stuser]
    cool-vpn[593]: IKEv2 SPIs: 9359bf743cee8c23_i d689e135ed18468c_r*, public key reauthentication
 in 20 minutes
    cool-vpn[593]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_2048
    cool-vpn{58}:  INSTALLED, TUNNEL, ESP in UDP SPIs: c518437c_i 2380ad3d_o
    cool-vpn{58}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 9 minutes
    cool-vpn{58}:   0.0.0.0/0 === 172.1.1.11/32
```

Associated revisions

Revision e88b529a - 01.03.2013 17:06 - Tobias Brunner

android: Mitigate race condition on reauthentication

If the TUN device gets recreated while another thread in handle_plain()
has not yet called select(2) but already stored the file descriptor of the
old TUN device in its FD set, select() will fail with EBADF.

Fixes #301.

History

#1 - 01.03.2013 12:48 - Tobias Brunner

- Status changed from New to Assigned

- Assignee set to Tobias Brunner

Thanks for the report. I suppose the return value there is a bit too hard.

This is a race condition that happens in a situation like this:

1. In handle_plain() the FD of the current TUN device (TUNFD0) is read and the RW-lock is released
2. **Context switch**
3. The thread waiting in setup_tun_device() acquires the lock, closes TUNFD0 and stores the new FD
4. **Context switch(es)**, perhaps not directly to the thread in handle_plain() which would explain the delay
5. Finally the initial thread gets its turn and select() is called, which fails with the error seen above as TUNFD0 is now closed
6. JOB_REQUEUE_NONE causes the callback job to be terminated, hence no packets will be read afterwards

The easiest fix is probably to just return JOB_REQUEUE_FAIR so the new FD is read on the next call to handle_plain(). If there really is some fatal problem this will result in tons of error messages, so another option would be to just retry a few times and then give up.

#2 - 01.03.2013 17:13 - Tobias Brunner

- Status changed from Assigned to Resolved

- Target version set to 5.0.3

- Resolution set to Fixed

I pushed a fix that retries reading packets if errno is EBADF.

#3 - 21.03.2013 18:48 - Tobias Brunner

- Status changed from Resolved to Closed