Chinese Washing Machine

A case study of an observed attack on IKEv1

Roman Hoog Antink and James Hulka*

May 5, 2014

In the Autumn of 2010 we at Open Systems AG in Zurich, Switzerland experienced IKE daemon software crashes exclusively at two locations even though we were operating more than 1500 VPN devices with several thousand IPsec tunnels between them on a global scale. Because we were using auto-generated configurations, our suspicions were raised that something interesting was going on. This paper describes the phenomenon and presents a working theory of the intentions of the attackers. We also provide a detailed description of how the manipulation of IKE packets that took place can be simulated, hence triggering the same software crash in a controlled environment.

*rha@open.ch, jah@open.ch

Revision History

Revision	Date	Author(s)	Description
1.0	15.04.2014	RHA, JAH	published
1.1	05.05.2014	RHA	minor corrections

Contents

1. Introduction

In the Autumn of 2010 we were running $\sim$ 1500 VPN[1] hosts in more than 150 countries. At that time we started migrating from Openswan[1] to ipsec-tools[2] because we were unhappy with the stability of pluto, the Internet Key Exchange (IKE) daemon used by Openswan. We were switching to Racoon, the IKE daemon used in ipsec-tools.

1.1. How We Use IPsec

Our requirements of high availability on continuously running systems motivated us to contribute a number of patches that went into the final ipsec-tools version 0.8.0, fixing various memory leaks related to reloading the configuration and a few crashes. We even fixed Racoon so it would not delete too many existing SAs[2] during configuration reload.

Even now, Racoon is limited to IKE version 1. This means throughout this document by *IKE* we mean IKEv1, unless stated differently.

We use IPsec[3] in transport mode and by adding GRE[4] encapsulation we achieve the same functionality as IPsec in tunnel mode. In contrary to tunnel mode however, our setup allows us to send multicast traffic over IPsec, and thus run OSPF[3]. This is a well known trick in the Linux networking community.

We were running the Linux kernel version 2.6.32.22 with grsecurity[4] and a couple of other patches applied.

We generate the configuration for each VPN host automatically based on VPN connections defined in a relational database. The database does not contain any IKE or IPsec parameters, except for the RSA[5] keys. The only thing the database tells the configuration scripts, is which interfaces of which hosts are connected by a VPN tunnel. All VPN tunnels use exactly the same settings. Only the RSA keys differ. This means that if there was a configuration error breaking anything, it would affect all locations in the same way.

1.2. IKEv1

Let us explain, in a very simplified form and in the form that we had used it, how IKEv1 works. If you are interested in an in depth explanation read through the material provided in the links in [6] and [7].

IKEv1 consists of two Diffie-Hellman[8] key exchanges in which keys are derived to be used to encrypt/authenticate IPsec traffic. This gives IPsec perfect forward secrecy,

[1] Virtual Private Network

[2] Security Association: a temporary set of cryptographic algorithms and keys negotiated between two VPN peers.

[3] Internet Protocol Security: a set of standards and protocols to build VPNs.

[4] Generic Router Encapsulation: a protocol that allows encapsulating IP in IP.

that is to say the keys that are used for the IPsec traffic are temporary, change with each new key exchange and if compromised **do not** compromise the permanent keys used to negotiate them.

Phase 1 In IKEv1 phase 1 (main mode) the authentication method, hash algorithm, encryption algorithm and keying material for IKE phase 2 is exchanged. An authentication and encryption key is created forming the phase 1 security association. The two peers authenticate against each other, ensuring they did not negotiate with an impostor. As illustrated in figure 1, the last two packets are encrypted.

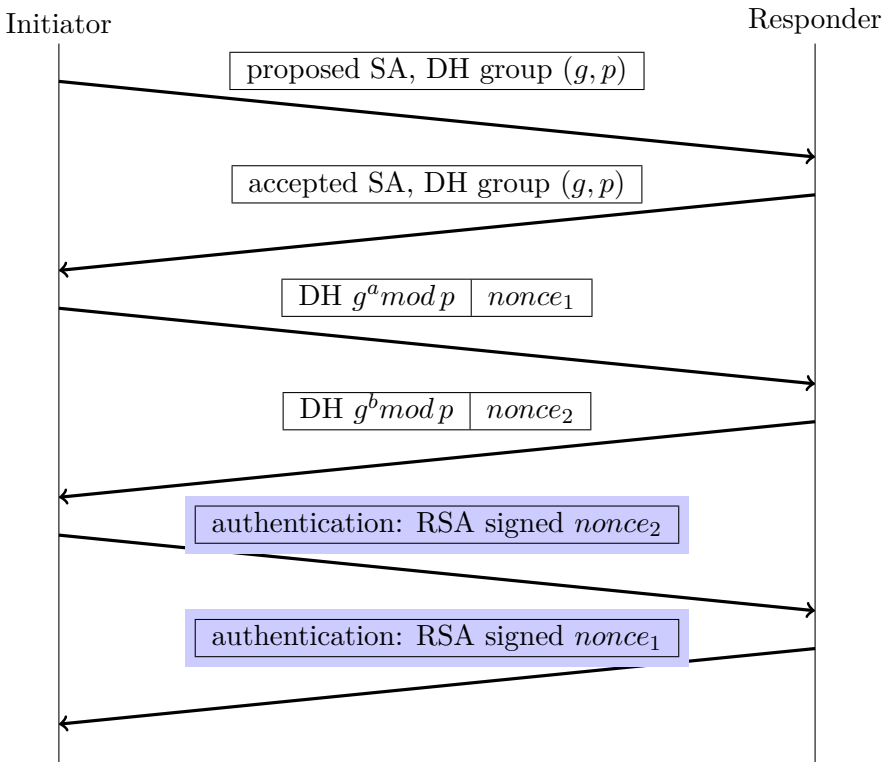

Figure 1: Packets exchanged during phase 1 of a IKEv1 key exchange

Phase 2 IKEv1 phase 2 (quick mode) is negotiated using the secure channel created in phase 1. In phase 2 the protocol (AH[5] or ESP[6]), hash algorithm, encapsulation mode, encryption algorithm, lifetime of the security association and the final keys used for the secure channel are determined. The security association produced by phase 2 is used to secure the IPsec traffic. Figure 2 illustrates the corresponding three encrypted packets.

[5]Authentication Header: used to ensure the authenticity of data transmitted over a unsecured channel without encryption.

[6]Encapsulated Security Payload: the protocol that transports the encrypted and authenticated user data of a VPN.

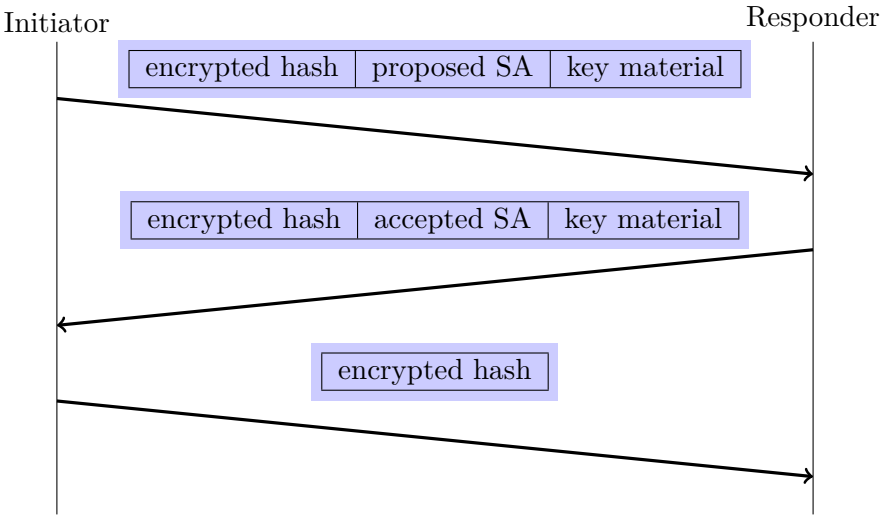

Figure 2: Packets exchanged during phase 2 of a IKEv1 key exchange

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initiator_cookie															
responder_cookie															
next_payload								major_version				minor_version			
exchange_type								flags							
message_id															
length															

Figure 3: IKEv1 header; the last bit in the *flags* field indicates content is encrypted

2. The Chinese Washing Machine

As we were migrating more and more systems from Openswan to Racoon, the first reports from our operations front rolled in. Most of them were tiny things we could fix in the init script or monitoring scripts. Among those were one or two software crashes. These rare errors are not the ones you concentrate on while dealing with hundreds of systems. So it took a while before we realized a concentration of these crashes had occurred and we started looking into it.

2.1. The Symptom

All five affected hosts were located in Shanghai and Guangzhou, two major cities, one on the southern and one on the eastern Chinese coast. Each host experienced up to three crashes per day, at least 6 to 8 hours apart. The crashes occurred during IKE key exchanges. Our biggest concern at the time were our customers, who due to the crashes were experiencing VPN outages. As we provide services operating 24/7, the pressure was high to prevent these crashes altogether. So we focused on correcting the null pointer buried deep in the code without really understanding the cause.

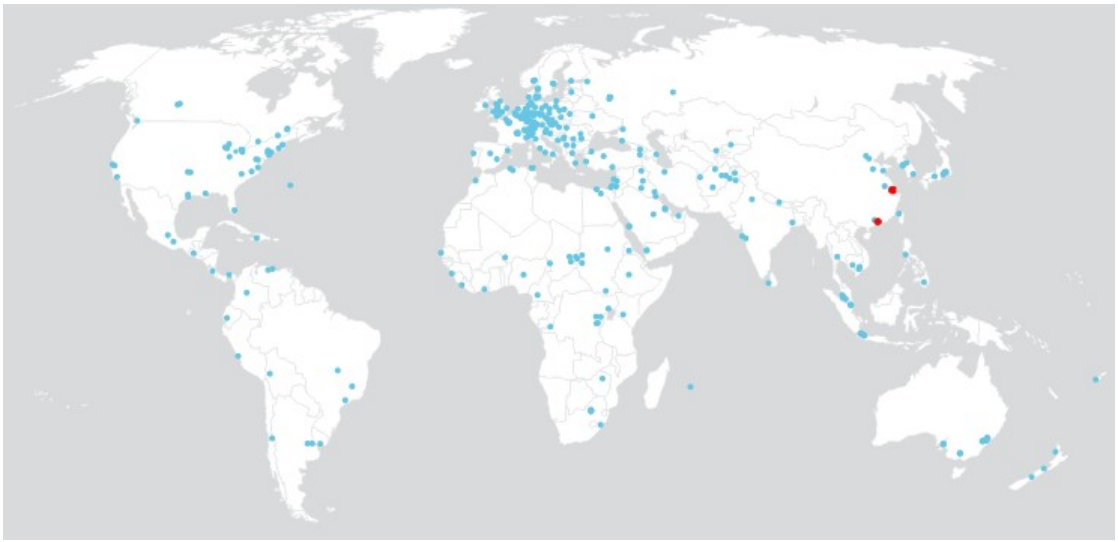

Figure 4: World map of our hosts; Shanghai and Guangzhou marked red

2.2. The Fix

The fix looks pretty simple. It is just this single diff[7] chunk, shown from the NetBSD source repository [9] in listing 1.

[7] A common UNIX tool to extract differences from text files.

```
1  diff -u -p -r1.45 -r1.46
2  --- src/crypto/dist/ipsec-tools/src/racoon/ipsec_doi.c 2010/10/11 14:16:30
       1.45
3  +++ src/crypto/dist/ipsec-tools/src/racoon/ipsec_doi.c 2010/12/14 17:57:31
       1.46
4  @@ -4166,9 +4166,14 @@ ipsecdoi_id2sockaddr(buf, saddr, prefixl
5     u_int8_t *prefixlen;
6     u_int16_t *ul_proto;
7   {
8  - struct ipsecdoi_id_b *id_b = (struct ipsecdoi_id_b *)buf->v;
9  + struct ipsecdoi_id_b *id_b = NULL;
10    u_int plen = 0;
11
12 + if (buf == NULL)
13 +    return ISAKMP_INTERNAL_ERROR;
14 +
15 + id_b = (struct ipsecdoi_id_b *)buf->v;
16 +
17    /*
18     * When a ID payload of subnet type with a IP address of full bit
19     * masked, it has to be processed as host address.
```

Listing 1: Patch fixing the null pointer exception

The service routine **ipsecdoi_id2sockaddr** accessed a null pointer in *buf* in line 8, while the two peers were in the process of an IKE key exchange.

At the time we did not bother nor have the time to investigate the why and the what. We were discouraged from digging deeper by several things.

- In order to comprehend the IKE protocol version 1, we had to read multiple RFCs[8] that contain many vague phrasings, making it very hard to imagine what a real world packet would look like.
- The RFCs offer so many options and features making it very hard to understand their security implications.
- In 2003 Niels Ferguson and Bruce Schneier wrote in their paper [10] about IPsec:

  > We feel very strongly that the resulting system is well beyond the level of complexity that can be analysed or properly implemented with current methodologies.

That was our experience as mere mortals who had not implemented an IKE daemon on our own. We simply did not have the time to investigate a complex protocol and its implementation.

2.3. Affected VPN Tunnels

Looking at what systems experienced crashes and which did not, a clear pattern emerged. For a host to be affected, it had to run VPN tunnels to another location

[8] Request For Comments, a collection of Internet standards. In our case at least RFC 2408 and RFC 2409

within China *and* at the same time be connected to one of a certain group of ISPs[9].

Tunnels from the two cities mentioned before and originating at the ISPs and ASs[10] listed in table 2 would show the effect, whereas VPN tunnels originating at the ISPs listed in table 3 were not affected.

ISP	**AS**
China Telecom	4134, 58461
China Unicom	17622
Shanghai Telecommunication Company	4812

Table 2: ISPs and ASs were the anomaly occurred

ISP	**AS**
Vanco	10212
China Network Communications (CNC)	17621

Table 3: ISPs and ASs were the anomaly did not occur

All other ISPs available in those two cities where either not used by our customers or our hosts connected to them had no VPN tunnels within China. Thus the list above is bound to our perspective and may be incomplete. It is remarkable, how the AS numbers match most of those found to be involved in active scans against Tor nodes by Philipp Winter and Stefan Lindskog in 2012 [12].

Keep in mind that this listing does not imply the ISPs did produce the effect by themselves. On the contrary, it is far more probable they are connected to the same Internet backbone by coincidence and the effect was caused by a third unknown entity outside of the networks of those ISPs.

2.4. Attack, Network Issue Or Configuration Issue

The first question that came to our mind was "Is it a configuration issue, a network issue or an attack?"

Configuration Issue As stated in section 1.1, our VPN configuration is identical in every detail on every single host in the world, except for the RSA keys and IP addresses. This pretty much eliminates the configuration as cause for a local phenomenon.

[9] Internet Service Provider

[10] Autonomous System - the designator for a collection of networks.
We present AS numbers based on historical data from Team Cymru [11]

Network Issue We face all kinds of mysterious network phenomenons every day, as we provide our services in more than 170 countries in the world. These include many countries without a reliable power grid, not to speak of their internet infrastructure.

The typical cases of network issues we face are routers with an old and buggy operating system, operated by ISPs. We suspect another big part of issues is caused by miseducated network administrators trying hard to eliminate all kinds of IP fragments in their networks.

While at the time we could not exclude network issues for sure, they did not seem very likely. The crashes occurred only sporadically without any obvious time pattern. Racoon, running a complex protocol, crashed from time to time and not on every single key exchange. Once Racoon was restarted key negotiations worked fine for a span of 6 to 8 hours, after which the next sporadic crash would occur.

Directed or broadly applied Attack A sporadic software crash is definitely something an attacker does not want to cause, not even as a side effect of his endeavour. This consideration made an attack targeting our installations directly or even Racoon in general very unlikely.

From what we knew at the time (see also section 2.5), an attack against a VPN product used by most people was more probable. We suspected some widely deployed VPN product might be vulnerable to downgrade attacks[11] by manipulating a specific packet of the IKE key exchange.

Theoretically the effect could have been caused by two types of interference. Either by dropping specific IKE packets as they traveled from one VPN peer to the other, or by manipulating certain portions of IKE packets. The latter seemed far more practicable, as it gives the attacker more possibilities. In addition packet loss is a common phenomenon on IP networks and is being handled well by network software in general.

Considering all the details of the phenomenon explained so far, we were not able to pinpoint the nature of the beast. You can find our detailed considerations whether this was an attack in section 4 on page 17.

2.5. Looking Around

In 2010 we were already seeing several unrelated indications of what was going on, in terms of informational surveillance.

- 2010: China had built their third super computer [13], which was the fastest at the time. The two Chinese super computers built previously are located in Shanghai and Shenzhen, which is next to Guangzhou. Maybe a mere coincidence.

  Even for a simple packet manipulation you would need lots of computing power when dealing with the IPsec traffic of a large Internet backbone. On the other

[11] see section 4.2 page 18.

hand, IKE traffic in general produces little amount of data even with a short re-keying interval, so the effort might not be so big after all.

But once an adversary would manage to decrypt IPsec traffic on a broad scale, he would definitely require adequate computing power to process the huge amount of data he made accessible.

Other countries might have done it in the mean time as well, but no details of their actual methods have been published [14]. We still believe that the cryptography has not been broken, while subtle implementation errors might offer shortcuts for an attacker.

- 2011: the German Federal Office for the Protection of the Constitution issued a warning[15] for business travelers to high risk countries like China and Russia. They advised against bringing electronic equipment with any personal or critical data.

- A comprehensive article on security.uri.edu [16] from 2012 gives insight on recommended behaviour when traveling to countries like China and Russia.

There was little information available about actual attack techniques against IKE. Eventually we had to go back to daily business. So we let the subject rest for three years, without a single further incident. We had fixed memory allocation and memory access issues in all execution paths we used within Racoon. Until we switched to Strongswan and IKEv2, the software worked very reliably.

3. Simulation Of The Attack

After laying dormant for more than three years the idea of further analyzing what was known about the Racoon crashes was re-raised. As the original software crashes had been isolated and fixed with the help of core dumps, and recognition of the anomaly and collection of data relating to the issue was done posthumously, there were no traffic captures available to analyze for traffic manipulation. We tried collecting traffic captures in 2011, a few months after we fixed Racoon, with the hope of the same anomaly still being present. Unfortunately the anomaly had disappeared.

3.1. Where To Start

Determining the source of the null pointer would require reversing the path of all calls to ipsecdoi_id2sockaddr (see section 2.2), the function in which the null pointer crash occured. The following is a tree of functions that eventually call ipsecdoi_id2sockaddr (the tree has been simplified for brevity).

ipsecdoi_id2sockaddr	← delete_spd		
	← quick_i2recv*		
	← quick_r1recv		
← delete_spd	← purge_remote		
	← isakmp_ph1delete		
← purge_remote	← isakmp_info_recv_d*		
	← isakmp_info_send_r_u*		
← isakmp_ph1delete	← purge_isakmp_spi*		
	← isakmp_ph1expire	← isakmp_info_recv_d*	
	← remove_ph1	← revalidate_ph12*	
	← migrate_ph1_ike_addresses	← pk_recvmigrate*	
← quick_r1recv	← get_proposal_r*		

Table 4: Code entry points (marked with *) for ipsecdoi_id2sockaddr

Even in its simplified form this table shows eight different entry points called or defined in at least six different source files[12] from which ipsecdoi_id2sockaddr can be called. It was soon clear that analyzing the logic for each of these paths was going to be very time consuming and we decided rather to start a targeted fuzzing of the software during key exchanges.

Considering we were dealing with a null pointer and were still assuming that this was induced due to missing or erroneous data being received in an IKE phase 1 or phase 2

[12] isakmp.c, isakmp_quick.c, isakmp_inf.c, handler.c, session.c, pfkey.c

payload, the first logical step appeared to be to start removing payload data to see how the software would react.

3.2. Boys And Their Toys

Our laboratory setup consisted of two hosts with a version of Racoon not containing the patch that fixed the null pointer (see section 2.2) and a configuration nearly identical to the originally affected hosts, with the exception that the security association lifetimes were much shorter to allow a faster test life-cycle. We were trying to produce the null pointer crash on Alice (figure 5) and her IPsec peer Bob (figure 5). Between Alice and Bob we used a Debian installation as a router (IKEv1 Proxy, figure 5) allowing us to see and more importantly manipulate all traffic between the hosts.

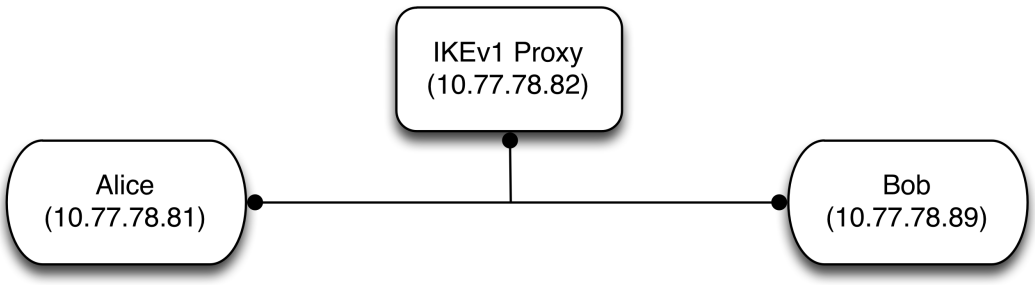

Figure 5: Test network; Alice and Bob had the IKEv1 Proxy set as their default gateway

As part of analyzing the packet structure used by IKE we wrote a simple IKE parser in python using the popular dpkt[17] packet creation/parsing library. Leveraging this parser we then wrote an IKEv1 proxy using nfqueue[18] and its python library[19] to intercept the packets we wanted to play with on the Debian router.

The Racoon version used as well as the proxy code can be found on github [20].

3.3. IKE Mangle

Because IKE phase 1 contains some packets that are not encrypted we started by manipulating phase 1 traffic. We setup our IKEv1 proxy to remove payloads from phase 1 packets hoping that the missing data would be carried through to the call of ipsecdoi_id2sockaddr. Although this prevented the IPsec tunnel from rekeying it did not crash the software.

As we had seen a large amount of Racoon log messages stating packets had been rejected or ignored because the encryption bit was not set on IPsec peers of the hosts experiencing the null pointer crashes, our next step was to try changing the encryption

bit in the flag section of the IKEv1 header (figure 3) to 0 during a phase 1 exchange. This manipulation produced the same log messages, however once again it only prevented the IPsec tunnel from being created and did not crash the software.

It was clear from the beginning that if a complex crypto attack had been attempted we would not be able to recreate the parameters needed to cause the crash. Our hope was not to perfectly mimic the attack, as this would have required more information (such as traffic captures). Instead we wanted to try to isolate what packets may have been manipulated in a possible attack scenario. With this in mind we moved on to mangling phase 2 packets in the same way we had mangled phase 1 packets.

The phase 2 tests once again prevented the IPsec tunnel from rekeying but it did not cause the software to crash. We continued with the phase 2 tests having a look through the logs and monitoring the kernel IPsec events to see if we could find any anomalies in Racoon's behaviour, at the same time only manipulating packets from Bob (IKEv1 responder) to Alice (IKEv1 initiator). During one such test in a stroke of luck we turned off our IKE mangle proxy just a few seconds into a phase 2 exchange, and much to our surprise the Racoon instance on one of the test hosts hit the exact null pointer we were looking for.

3.4. Time Bomb

After producing the result we were looking for, we started dissecting the conditions that led to the software crash and our best clues came from the kernel IPsec events and log messages we added to the software before each ipsecdoi_id2sockaddr call. In the IPsec events we could see that a phase 2 security association was deleted even though Racoon was still unsuccessfully trying to renegotiate phase 2 with mangled packets. Once the IKEv1 proxy was turned off Racoon started receiving valid phase 2 packets again, however the first packet received was handled by quick_i2recv (racoon/isakmp_quick.c) which performs two identity checks using ipsecdoi_id2sockaddr. The problem was that the first identity check used the local security association which had already been deleted and in turn triggered the null pointer crash.

Breaking down the time-line we got the following on Alice:

1. phase 2 security associations for IPsec tunnel are deleted

```
Timestamp:  Thu Mar 27 20:18:22 2014 519654 usec
Deleted src 10.77.78.81 dst 10.77.78.89
Timestamp:  Thu Mar 27 20:18:22 2014 520516 usec
Deleted src 10.77.78.89 dst 10.77.78.81
```

2. Racoon (Alice) sends initial phase 2 renegotiation packet (packet 1 figure 2)

```
<1e>Mar 27 20:18:23 racoon:  INFO: initiate new phase 2 negotiation:
10.77.78.81[500]<=>10.77.78.89[500]
```

3. Racoon (Alice) receives the reply packet (quick_i2recv) for the packet found in point 2 but does not process it (packet 2 figure 2)

```
<1e>Mar 27 20:18:23 racoon:  [10.77.78.89] ERROR: Packet wasn't encrypted.
<1e>Mar 27 20:18:23 racoon:  [10.77.78.89] ERROR: failed to pre-process
ph2 packet (side:  0, status:  5).
```

4. Racoon (Alice) receives an initial phase 2 packet but does not process it (packet 1 figure 2)

```
<1e>Mar 27 20:18:24 racoon:  [10.77.78.89] ERROR: Packet wasn't encrypted.
<1e>Mar 27 20:18:24 racoon:  [10.77.78.89] ERROR: failed to pre-process
ph2 packet (side:  1, status:  1).
```

5. we turn off the proxy and Racoon (Alice) receives a reply packet for the packet found in point 2 again, this time producing the null pointer (packet 2 figure 2)

```
<06>Mar 27 20:18:33 kernel:  racoon[5942]:  segfault at 8 ip 00000000004385e2
sp 00007fffb87784b0 error 4 in racoon[400000+93000]
```

The following packet capture shows the series of packets sent back and forth leading up to the null pointer.

Time	SRC IP	DST IP	Protocol	Size	Description
20:18:23	10.77.78.81	10.77.78.89	ISAKMP	438	Quick Mode
20:18:23	10.77.78.89	10.77.78.81	ISAKMP	406	Quick Mode[Malformed Packet]
20:18:24	10.77.78.89	10.77.78.81	ISAKMP	438	Quick Mode[Malformed Packet]
20:18:24	10.77.78.81	10.77.78.89	ISAKMP	118	Informational
20:18:33	10.77.78.81	10.77.78.89	ISAKMP	438	Quick Mode
20:18:33	10.77.78.89	10.77.78.81	ISAKMP	406	Quick Mode
20:18:33	10.77.78.89	10.77.78.81	ISAKMP	406	Quick Mode

Table 5: Packet capture leading up to the Racoon crash

3.5. No Room For Error

In order to trigger the null pointer crash, the IKE packet manipulation had to occur within a very specific time frame. If the proxy was turned off before the 'failed to pre-process ph2 packet' messages appeared in the log (see section 3.4), the phase 2 exchange

proceeded without interruption. However, if the proxy was left running the phase 2 exchange timed out (quick_timeover in racoon/isakmp_quick.c) after approximately 45 seconds. This means the manipulation lasted less than 45 seconds once the phase 2 exchange had started, otherwise it would not have crashed the software.

Despite attempting to replicate the same conditions multiple times (easily hundreds of times in a series of days and weeks), we were not able to produce the same crash by manipulating phase 1 packets, which in the end makes sense considering the entry point leading to the crash is part of the phase 2 negotiation.

To either disprove or further strengthen our suspicion we tried to reproduce the crash by using the proxy to drop phase 2 packets for varying lengths of time. We were unsuccessful in all cases, despite running these tests much longer than required to produce the effect by manipulating phase 2 packets. Of course this does not completely prove that it was not caused by a network anomaly.

The only way we could find to reproduce the crashes, was to flip the encryption bit in the IKE header of a certain packet in transit. The 16 bit UDP[13] checksum also needed to be recalculated or zeroed out for the packet to enter the IP stack of its destination.

[13]User Datagram Protocol: the transport protocol used for IKE.

4. What's Bringing You Down?

4.1. Will Someone Please Think Of The Network

The fact that the crashes were occurring at irregular intervals had already made the idea that a network anomaly was causing the null pointer an unlikely possibility. Combine this with the fact that inducing the crash required a very specific set of packets to be deemed unusable by Racoon, as well as the fact that the crashes were geographically isolated while running the same key exchanges world wide, it is in our opinion highly to extremely unlikely that a network anomaly caused the null pointer crash.

If this was just a phenomenon caused by transmission errors, the UDP checksum would have had to be changed to the exact matching new value by accident. In addition, the very bit that indicates whether the IKE payload is encrypted in a 438 byte packet, would have had to be flipped from 1 to 0. Please refer to appendix A on page 20 for an in depth analysis of what the odds are.

We cannot completely rule out that the anomaly leading to the observed crashes was a networking issue. The possibility, for the reasons stated above, that the crashes were induced by an attack on the IKE protocol is in our opinion substantially higher.

4.2. Attack Types

An attack on IKE makes the most sense in three scenarios:

1. **Denial of Service** (DoS) - prevent the target from being able to establish a secure channel

2. **Man in the middle** (MITM) - become a relay in the target's secure channel, being able to intercept all communication in clear text

3. **Downgrade** - reduce the security parameters used in the target's secure channel, making it easier or even trivial to subvert the encryption

DoS Although a DoS was observed this appears to have been a very obscure side effect. We would even argue that the alleged perpetrator of the attack did not intend for this to happen as a DoS is only really effective if it is constant enough to cause a serious gap in the service. A DoS against VPN only happening a few times daily is extremely ineffective, as it is likely to be noticed quickly after which the service can be restarted.

MITM This is the sexiest of the three possibilities, it is however also the most difficult to substantiate as at this point there are no known crypto attacks against IKEv1. Mightier minds have wrestled with this topic [10] and their general conclusion appears to be that implementation weaknesses are more likely due to the complexity of the protocol.

Downgrade Implementation weaknesses leads us directly into the possibility of a downgrade attack. If an implementation of IKEv1 were to accept unencrypted phase 2 packets it would be possible for an attacker to choose NULL encryption and/or a weak hash algorithm. NULL encryption allows IPsec to be used without any encryption at all, this would allow anyone along the path between the two IPsec peers to read all traffic in the IPsec tunnel.

A weaker hash algorithm, theoretically, could allow an attacker with enough computing power to change the contents of an IPsec tunnel on the fly and recalculate the authenticating hashes. A serious issue with this theory, however is that the phase 2 packets dealing with negotiating these attributes contain an encrypted hash which among other things verifies the integrity of the attributes.

4.3. Teeth And Fingernails But No Corpse

In short we do not know how the attack was structured, we do know that it involved IKEv1 phase 2 (quick mode) and that it was most likely meant to subvert the cryptographic integrity of the IPsec tunnel. Based on the infrequency and the very exact timing/number of packets manipulated required to crash Racoon (see section 3.4), the attack appears to either have been conducted by a live operator or an advanced automated system designed with stealth in mind.

5. Conclusion

Based on what we have observed there seem to be more questions rather than answers. Future research possibilities based on this paper could include testing popular (including closed source) implementations of IKEv1 using the same setup outlined in section 3. It would also be interesting to manipulate IKEv2 traffic to see how various implementations react to unexpected changes. We would also be very interested to know if anyone else has observed similar and/or strange IKEv1 or IKEv2 behaviour, especially hard evidence of IKE traffic being manipulated by a third party.

While we have no absolute proof the crashes were caused by a malicious attack, it appears more than likely. We assume the attacker cleared the IKE encryption flag during quick mode and tried to manipulate the negotiation of encryption and hash algorithms for ESP.

It should be noted that the IKEv1 implementation in Racoon was not susceptible to the manipulation and Racoon itself was easily patched to prevent the null pointer crash.

A. Probability Of Random Bit Errors

Because we have no absolute proof the crashes were caused by a deliberate attack, it is interesting to estimate how unlikely random transmission errors could lead to the same effect.

A.1. A Simplified Model

Let's use a simplified model to estimate the probability causing the effect with random bit errors as they occur during signal transmission over the Internet. We are establishing a best case estimation. As long as the probability produced by this model is **more likely** than the real probability that would result from taking into account every detail to its full complexity, we still make our point, regardless how different from reality our result might be.

A.2. Which Bits To Flip

Packet Layout

Looking at the layout of an IKE packet, there are two regions of interest regarding our case:

1. The encryption flag in the IKE header.[14]
2. The checksum field in the UDP header.[15]

The crash was caused by clearing the IKE encryption flag[14]. Due to the way UDP checksums are calculated[15], this requires adapting the UDP checksum, because a single bit flip cannot lead to a UDP checksum collision. So it is safe to say the checksum must change by at least one bit for the packet to be accepted by its destination.

Assume The Checksum Changes By Just One Bit

According to RFC 768[15] UDP checksums are optional however, because we assume no attack in this chapter, clearing the UDP checksum is not part of the model.

Considering the similar case where the UDP checksum is cleared by coincidence is not necessary, as it would require even more bits to change which is far less probable. Remember, we may keep our model simpler than the reality as long as the model's outcome is more probable.

In the simplest case, flipping a bit of the content of a packet will flip one single bit in the 16 bit UDP checksum. In practice a single bit flip potentially could affect all bits of the checksum at once. For the reason stated above, the less probable cases are neglected where more than one bit of the checksum need to adapt.

[14] The encryption flag is part of the ISAKMP header. Please refer to RFC 2408 [21] ISAKMP page 22

[15] The basic element of the UDP checksum is the 1-complement addition of 16 bit words. This kind of checksum would miss if one or more entire words changed their position, whereas it can not ignore a single bit flip. See also RFC 768 [22].

A.3. What Are The Odds

Taking all considerations and simplifications of the model into account, the question how likely the crash was caused by transmission errors, boils down to the probability of two specific bits of an entire IKE exchange being flipped while all other bits remain unchanged.

If this reduced case turns out to be very unprobable, the simplifications of the model appear to be justified, as they would make it even less probable. And thus our assumption that this was a malicious attack would be affirmed substantially.

Doing The Math

We know from observation with a packet capture tool that a full IKE key exchange consists of 1672 bytes (6 packets main mode for phase 1 and 3 packets quick mode for phase 2). A rekeying exchange uses the existing phase 1 SA and therefore requires only 488 bytes (3 packets for phase 2) of transmitted data, thus 3904 bits. Let's use l for the number of bits.

This includes the IP header [23], but not any link layer header such as Ethernet [24], since the IP header and everything on top comprises the portion of the packet that travels all the way from source to destination. Neglecting the full exchange and concentrating on the rekeying case again is a valid simplification, as it requires fewer bits to follow the rules.

The probability of a bit error p is assumed to be $1 \cdot 10^{-7}$, which is one bit out of ten million. This is just a rough guess, based on satellite links [25]. In reality we were using ordinary land line connections, so again we bend the model in favor of the doubt.

$$P = p^2 \cdot (1-p)^{l-2} \tag{1}$$

From the formula (1) it is easy to see that the bit error rate p has little impact on the calculation, whereas the length of the transmission is very decisive, which is known precisely. As denoted before, formula (1) calculates the likeliness of two specific bits being flipped during an IKE rekeying exchange. Using the values $p = 1 \cdot 10^{-7}$ and $l = 3904$ as explained, this results in a probability of

$$P = 1 \cdot 10^{-14} \tag{2}$$

How Small Is Too Small To Imagine

Numbers so small are hard to classify let alone to imagine. We can compare it with ordinary things putting it into perspective.

Let us assume an Internet connection of very bad quality - again we are reducing the odds in favor of the doubt - where the rekeying would take place more often than the usual three times a day. For the sake of the argument we assume 300 rekeyings per day, which means rekeying every four minutes. This is a best case estimation far from realistic, thus theoretically reducing our chances of proving anything.

Even with this bad expectation we would encounter the crash only every 900 million years, which is one tenth of the expected lifespan of our solar system[16].

[16]The solar system is expected to exist for about 9 billion years in total, as stated in [26].

References

[1] *Openswan*. 2014. URL: https://www.openswan.org/.

[2] *IPsec-Tools*. 2014. URL: http://ipsec-tools.sourceforge.net/.

[3] *OSPF*. 2014. URL: http://en.wikipedia.org/wiki/Ospf.

[4] *grsec*. 2014. URL: http://grsecurity.net/.

[5] *RSA*. 2014. URL: http://en.wikipedia.org/wiki/RSA_%28cryptosystem%29.

[6] IBM. *IKEv1*. 2014. URL: http://pic.dhe.ibm.com/infocenter/zos/v1r12/index.jsp?topic=%2Fcom.ibm.zos.r12.hald001%2Ff1a1c5a0219.htm.

[7] D. Harkins and D. Carrel. *RFC 2409 IKE*. 1998. URL: http://tools.ietf.org/html/rfc2409.

[8] *Diffie-Hellman Key Exchange*. 2014. URL: http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange.

[9] *NetBSD source repository*. 2014. URL: http://cvsweb.netbsd.org/bsdweb.cgi/src/crypto/dist/ipsec-tools/src/racoon/ipsec_doi.c.diff?r1=1.45&r2=1.46&only_with_tag=MAIN.

[10] Niels Ferguson and Bruce Schneier. *A Cryptographic Evaluation of IPsec*. 2003. URL: https://www.schneier.com/paper-ipsec.html.

[11] Team Cymru. *Historical Whois Database*. 2014. URL: http://www.team-cymru.org/Services/ip-to-asn.html.

[12] Philipp Winter and Stefan Lindskog. *How the Great Firewall of China is Blocking Tor*. 2012. URL: http://www.cs.kau.se/philwint/static/gfc/.

[13] *Super Computer Centers In China*. 2014. URL: http://en.wikipedia.org/wiki/Supercomputer_centers_in_China.

[14] *Hammerstein Turbine Turbulence*. 2014. URL: http://arstechnica.com/information-technology/2014/03/nsas-automated-hacking-engine-offers-hands-free-pwning-of-the-world/.

[15] Heise Zeitschriften Verlag. *Vorsicht bei Reisen in "spionageträchtige" Länder*. Aug. 2011. URL: http://heise.de/-1165782.

[16] University of Rhode Island. *Travel to China or Russia*. 2012. URL: http://security.uri.edu/travel/travel-to-china-or-russia/.

[17] *Python packet creation / parsing library*. 2014. URL: http://code.google.com/p/dpkt.

[18] *libnetfilter_queue*. 2014. URL: http://www.netfilter.org/projects/libnetfilter_queue/doxygen/.

[19] *Python Netfilter Queue*. 2014. URL: https://packages.debian.org/wheezy/python-nfqueue.

[20] James Hulka. *Github Repository*. 2014. URL: https://github.com/open-ch/chinese-washing-machine.

[21] D. Maughan et al. *RFC 2408, ISAKMP*. Nov. 1998. URL: http://tools.ietf.org/html/rfc2408#page-22.

[22] J. Postel. *RFC 768*. Aug. 1980. URL: http://tools.ietf.org/html/rfc768#page-2.

[23] *IPv4 Header*. 2014. URL: http://en.wikipedia.org/wiki/IPv4_header#Header.

[24] *Ethernet*. 2014. URL: http://en.wikipedia.org/wiki/Ethernet.

[25] M. Allman, D. Glover, and L. Sanchez. *RFC 2488*. Jan. 1999. URL: http://tools.ietf.org/html/rfc2488#page-3.

[26] *Sun, Life Phases*. 2014. URL: http://en.wikipedia.org/wiki/Sun#Life_phases.