# Experiments in making IPsec scale

Presentation given during FSCONS 2012, 2012-11-11.

# About MC

Michael Cardell Widerkrantz

Consulting software developer

Specialising in computer network software development.

Malmö, Sweden

mc@hack.org

http://hack.org/mc/

# IPsec

IPsec

- is a security extension to IP.
- works on the IP layer.
- protects layers above: UDP, TCP, et cetera.
- is typically implemented in the kernel as part of the IP stack.

Between two hosts IPsec provides:

- Confidentiality.
- Integrity.
- Replay protection.
- Peer authentication.

# IPsec terminology

- Security Association (SA): The shared security attributes between two peers. One SA per direction.
- Security Policy (SP): Rules about what traffic to protect and how.
- ESP: Encapsulating Security Payload. What IPsec looks like on the wire.
- Transport Mode: Direct transfer between two nodes.
- Tunnel Mode: Tunnel between two networks through two security gateways.

# Typical use of IPsec today

- Virtual private network (VPN) tunnels.
- Road warriors, typically a laptop user on an untrusted network who needs to reach the office LAN.

# The vision

*All* nodes on the entire Internet authenticated and encrypted.

# The problem - key distribution

# Key distribution and management

- X.509 certificates – Certificate trees, Certificate Authorities.
- Raw public keys – Out of band distribution. How?
- Pre-shared symmetric keys – Out of band distribution. The military solution: Armed escort.
- Kerberos – A trusted third party. Centralized control.

# Internet key exchange protocol (IKE)

IKE is the most common key exchange protocol. Typically implemented as a userland server.

- Automatically authenticates peers and creates Security Associations.
- In some implementations the IKE dialogue is triggered by Security Policies in the kernel.
- Two versions defined: IKEv1 and IKEv2.
- IKEv1 unnecessary complex. Many configuration possibilities. Hard to get a compatible setup. IKEv2 the answer.

# How IKE works

Warning: Simplified.

- Establish IKE Security Association.
- Exchange CERT payloads (X.509 or raw RSA public key).
- Identify: Both peers send an identification, for example an FQDN such as “alice.example.org”.
- Authenticate: Challenge to find out if the peer knows private key.
- Diffie-Hellman handskake to create a session key.
- Create Child SAs for traffic: One per direction.
- (Define Security Policy to protect traffic.) Might be done outside of IKE server.

# Key distribution using X.509 certificates

- CERT payload X.509. Signed by common Certificate Authority.
- The CA’s public key (or certificate chain leading to CA) needs to be distributed to all nodes in advance.
- What CA do we use? In a corporate environment we use the IT department’s CA.
- How do we scale up?
- Compare the situation with HTTPS. A mess! *Many* CA keys pre-loaded in each browser.

# Key distribution by DNS

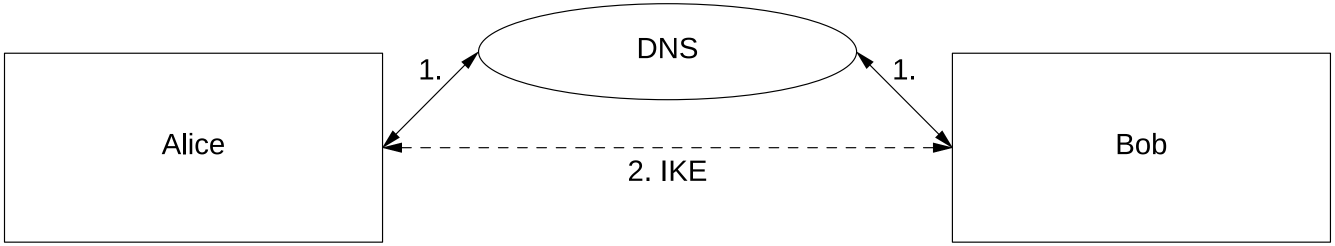
- Existing infrastructure with ’trusted’ roots.
- We can use existing DNS Resource Record, IPSECKEY (RFC 4025).
- Can verify published keys with DNSSEC.

# Earlier attempts at using DNS

- Didn’t FreeS/WAN support keys in DNS?
- Yes, but FreeS/WAN used reverse zones (IP adress to name, in-addr.arpa).
- Many not authorised to change their reverse.
- Nodes moves around much. Dynamic IP addresses.
- Much easier to control forward zones (names -> IP address). Many operating systems supports DNS Update out of the box.
- Code removed from current FreeS/WAN forks.

# Forward DNS key distribution scenarios

# Endpoints known by name



1. Alice and Bob already knows each other’s name. They query DNS for each other’s address *and* public keys. When received they load the public key and sets up a Security Policy for the peer’s address.

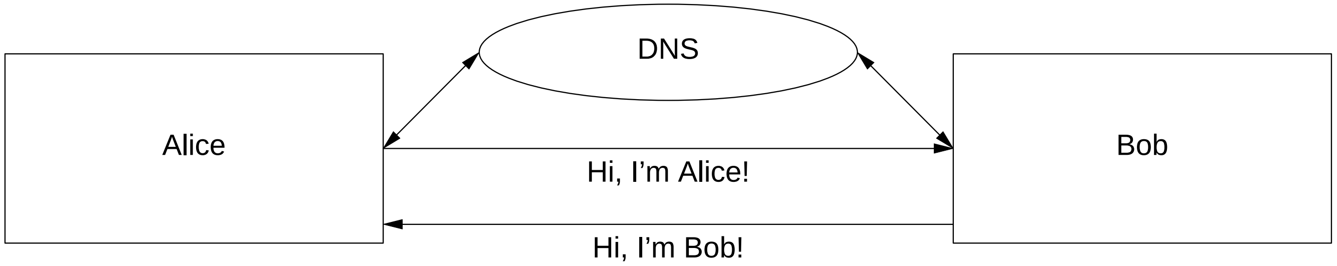
2. When/if the Security Policy is triggered an IKE dialogue is started automatically.

This scenario provides strong authentication both ways.

# Realistic example

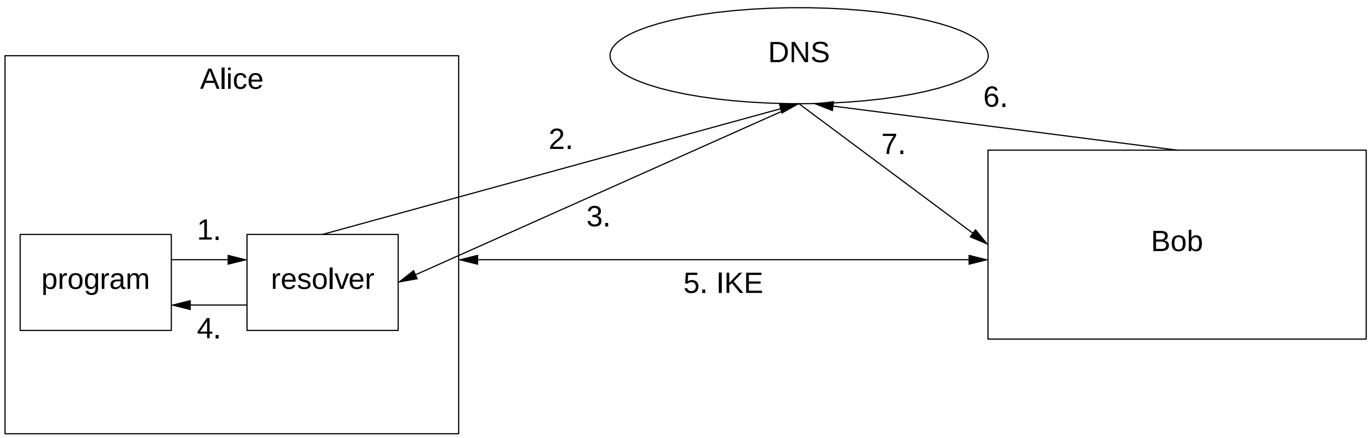
- Customer IT guy says to MC: The VPN concentrator is vpn.example.com.
- MC to IT guy: My laptop is brain.hack.org.
- That’t it!

# IKE server queries DNS



- The IKE daemon on both ends loads keys from DNS when receiving the peer’s name.
- Open for Man in the Middle if we don’t know names in advance, but “better than nothing”.
- Secure if we know names in advance.

# Capturing resolver



1. A program asks a local resolver for Bob’s address.

2. Resolver queries DNS for Bob’s address *and* Bob’s public key.

3. DNS replies with address and the public key, if available. The resolver now loads the key and sets up policy.

4. The resolver tells the program Bob’s address.

5. Alice says “Hi, I’m Alice” to Bob through IKE.

6. Bob queries DNS for Alice’s key.

7. Bob gets Alice’s key and can authenticate Alice through IKE.

Might be open for a man in the middle attack in one direction.

# Patches to racoon

About a year ago I hacked on the old *racoon* IKEv1 server to support these three scenarios. My patches support:

- loading raw RSA public keys into a running racoon.
- doing DNS queries for IPSECKEY, that is the “IKE server queries DNS” scenario.

# Helper scripts

I wrote two helper scripts in Perl:

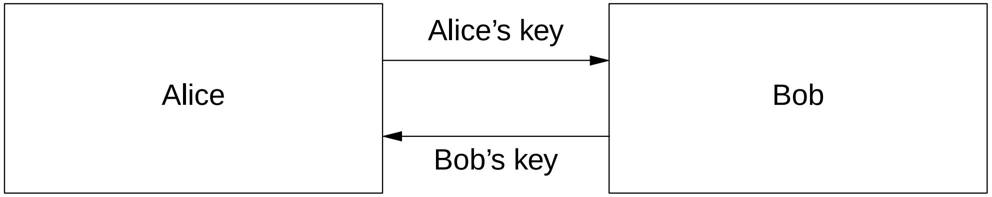
- *autosp.pl:* Scenario “Endpoints known by name”. Queries DNS for keys, loads them and automatically sets security policy for peer.
- *ns.pl:* Scenario “Capturing resolver”.

These can be used together with a DNSSEC validating resolver.

# Experiences using racoon and DNS key distribution

- Most promising scenario: “Endpoints known by name”. Provides strong authentication in both directions.
- Weakest point in above scenario: DNS update authentication.
- Problems with large DNS records (IPSECKEY) in the field. Also problems with DNSSEC. TCP sometimes filtered.
- No gain from using DNS in “IKE server queries DNS” scenario if names not known in advance. CERT payload instead?
- Better to use a key fingerprint rather than whole key in DNS?
- Much easier to find compatible configurations with IKEv2 than IKEv1.
- Need more modern code base.

# Better-than-nothing security



- BTNS is IPsec with anonymous keys.
- Peer’s public key is sent in IKE dialogue.
- Peer’s key *can* be validated with a stored fingerprint but wildcard allowed.
- We can accepts peer’s key without validation and go on to (anonymous) authentication.
- Defined in RFC 5386 & RFC 5387, November 2008.
- No available implementation!

# Why use BTNS?

- Protecting layers above.
- Provides what IPsec provides: Confidentiality, integrity and replay protection.
- But there’s no authentication! It’s open for a Man in the Middle!?
- Yes, but we get continuity of association — we know we are still speaking to the same party we started the communication with.
- Protects against passive surveillance.
- Importantant spoof protection in long lived sessions, e.g. BGP.
- Perhaps use as fallback when DNS keys not available?
- Better than nothing!
- Compare current mail delivery practice: SMTP + STARTTLS w/ self-signed certs.

# OpenBSD’s iked

- Supports IKEv2.
- Modern code base.
- Standard C89.
- Uses privelege separation.

# Patches to iked

I have patched *iked* to support:

- authentication with raw RSA keys.
- a “btns” keyword in policies to allow BTNS.
- fingerprint search for trusted anonymous keys.
- a special BTNS wildcard to allow *any* keys.

# Experiences with BTNS

- A BTNS wildcard currently affects *all* BTNS policies. Perhaps settable per policy?

# Current limitations

Current limitations in OpenBSD and/or *iked:*

- *iked* doesn’t support Transport Mode (although OpenBSD does). Not clear if this is needed.
- No automatic triggering of IKE dialogue from kernel Security Policy.
- Peer has to be exact adress if we *initiate* IKE dialogue, not a network.

# Potential future work

- Port *iked* to FreeBSD and Linux.
- Change DNS helper scripts to work with *iked.*
- Support for initiating IKE sessions on demand in *iked.*

# More information

The projects:

- http://hack.org/mc/projects/ipsec/
- http://hack.org/mc/projects/btns/

The author:

- Michael Cardell Widerkrantz <mc@hack.org>
- http://hack.org/mc/

The sponsor:

- Stiftelsen för Internetinfrastruktur (.SE) http://iis.se/

CONTENTS