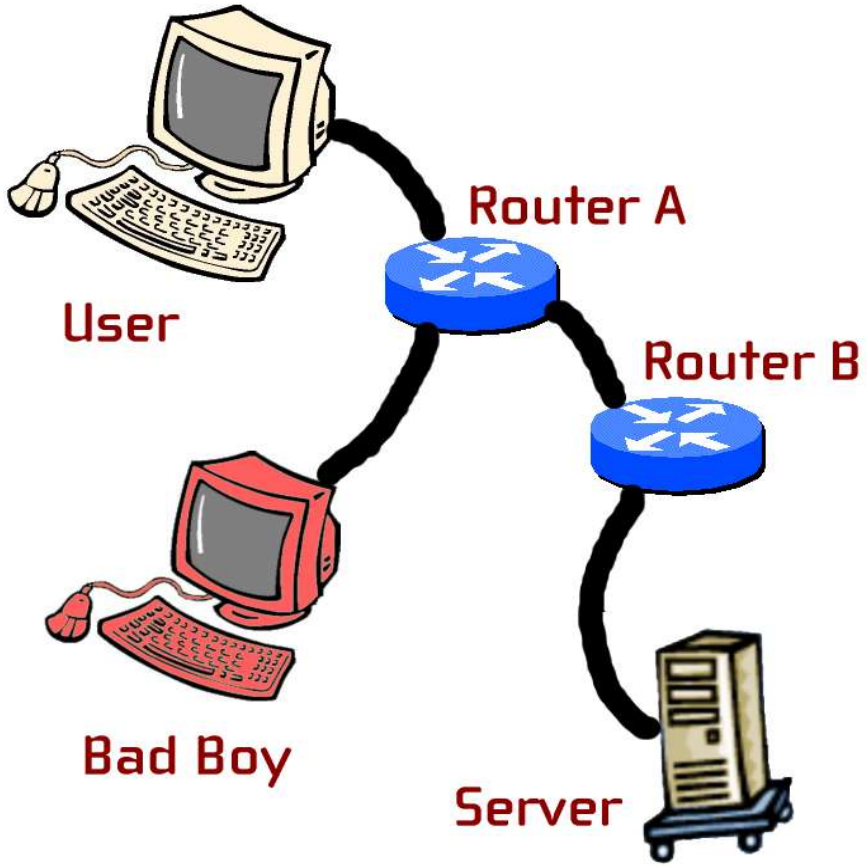
Secure networking

Michal Ludvig
SUSE Labs

mludvig@suse.cz

1/30/04

Computer network



- Communication between **User** and **Server**:
  - In the form of **packets**.
  - Traverse several **Routers**.
  - Can be intercepted by a **BadBoy**.

BadBoy listens ...

```
14:34:50.808982 user.domain.com.35558 > www.porn.com.80: tcp 137 (DF)
0x0000   4500 00bd 524e 4000 4006 0079 ac10 0002        E...RN@.@..y....
0x0010   d882 d8d6 8ae6 0050 d683 1771 9f37 920d        .......P...q.7..
0x0020   8018 16d0 bb19 0000 0101 080a 0639 453d        .............9E=
0x0030   1858 219f 4745 5420 2f70 6f72 6e2f 696e        .X!.GET./porn/in
0x0040   6465 782e 6874 6d6c 2048 5454 502f 312e        dex.html.HTTP/1.
0x0050   300d 0a55 7365 722d 4167 656e 743a 2057        0..User-Agent:.W
0x0060   6765 742f 312e 382e 320d 0a48 6f73 743a        get/1.8.2..Host:
0x0070   2077 7777 2e70 6f72 6e2e 636f 6d0d 0a41        .www.porn.com..A
0x0080   6363 6570 743a 202a 2f2a 0d0a 4175 7468        ccept:.*/*..Auth
0x0090   6f72 697a 6174 696f 6e3a 2042 6173 6963        orization:.Basic
0x00a0   2062 4739 7663 3256 794f 6c42 6c5a 4739        .bG9vc2VyOlBlZG9
0x00b0   7761 476c 735a 513d 3d0d 0a0d 0a               waGlsZQ==....
```

- What a **BadBoy** can see in this **packet**...
  - It's a **cleartext** request for a webpage
  - www.porn.com – server name
  - /porn/index.html – document name
  - Authorization – encoded username and password

Disclosing the password

- The **Authorization** string contains an encoded username and password.

```
Authorization: Basic bG9vc2VyOlBlZG9waGlsZQ==
```

- Type Basic means Base64 encoding.

```
$ echo bG9vc2VyOlBlZG9waGlsZQ== | openssl base64 -d
looser:Pedophile
```

- Here we are ... username **looser**, password **Pedophile** :-)

Cleartext fun

- Majority of the Internet traffic is in **cleartext** (non-encrypted).
- Vulnerable protocols
  - Telnet, POP3, IMAP, HTTP, SMTP and almost every other...
- Possible risks
  - Passive – eavesdropping
    - Gathering passwords and other sensitive information.
  - Active - „man-in-the-middle“
    - Replay attack – resending a piece of informtion many times.
    - Packet modifications – changing the information on the way between user and server.

Cryptography comes ...

- ... and brings privacy and confidentiality

Encrypt ... but how?

- **On the application level**
  - Application to application security
  - SSH, Kerberos, PGP, ...
  - Generic SSL/TLS protocol used with many services:
    - HTTPs, IMAPs, POP3s, ...
- **On the network level**
  - Host to host or even Network to network security
  - No need to modify the applications
  - IPsec, CIPE, proprietary VPN solutions

CIPE vs. IPsec

CIPE

- Linux and Windows only.
- Preshared keys only.
- Uses UDP packets.
- Doesn't work with Linux kernel 2.6.
- Only two ciphers available.

IPsec

- Widely supported standard.
- Preshared keys, RSA keys, X.509 cetrificates, Kerberos tickets.
- Uses ESP and/or AH protocol. Could be encapsulated in UDP.
- Two implementations for kernel 2.6 available.
- Currently 8 ciphers and 6 digests to choose from.

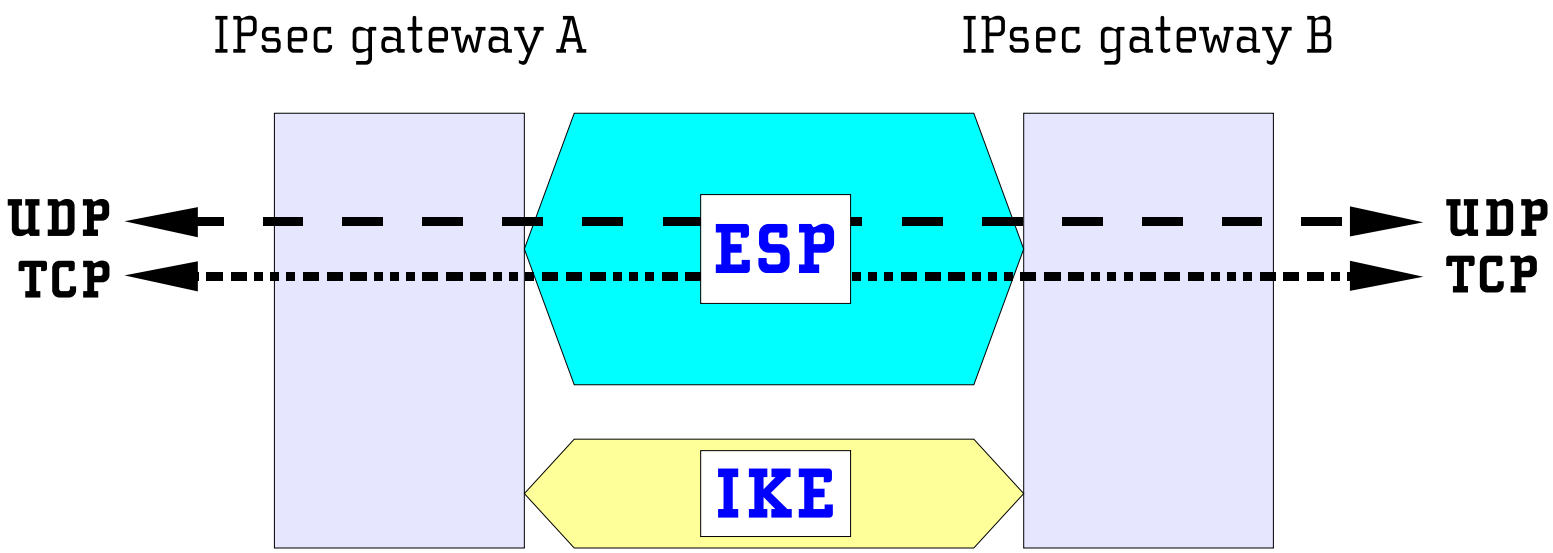
IPsec architecture

- **Kernel**
  - Keeps the symmetric encryption keys and security policies
  - Encrypts outgoing packets
  - Decrypts and verifies incoming packets
  - Uses ESP and/or AH protocol (IP protocols).
- **Userspace daemon**
  - Negotiates the keys with the other party
  - Pushes the keys and security policies into the kernel
  - Uses IKE protocol (UDP protocol).

IPsec in Linux

- **Kernel**
  - Two implementations
  - FreeS/WAN
    - Available since 2.0.x kernel
    - Requires FreeS/WAN userspace.
    - More mature
  - Native IPsec stack
    - Available since 2.6.0
    - RFC-based API
    - Still has some problems
    - Used in SUSE Linux' kernel
- **Userspace**
  - Three choices
  - (Super)FreeS/WAN
    - Most widely used
    - Many extensions
    - Shipped in SUSE Linux
  - IPsec-tools
    - Ported from NetBSD
    - Only for native 2.6 IPsec
  - ISAKMPd
    - Ported from OpenBSD
    - Only for native 2.6 IPsec

IPsec traffic outline



- Only two types of the traffic on the wire:
  - **IKE** – Key exchange and connection control
  - **ESP** – Encrypted traffic

ESP – Encapsulating Security Payload

- IP protocol #50
- Provides **encryption**
  - Prevents eavesdropping
  - **BadBoys** can still see the traffic, but can't understand it.
- Provides **authentication**
  - Prevents “man-in-the-middle” attacks
  - Ensures that the packet wasn't modified since it was sent.
  - Prevents “replay attack” - every packet is accepted only once.

AH – Authentication header

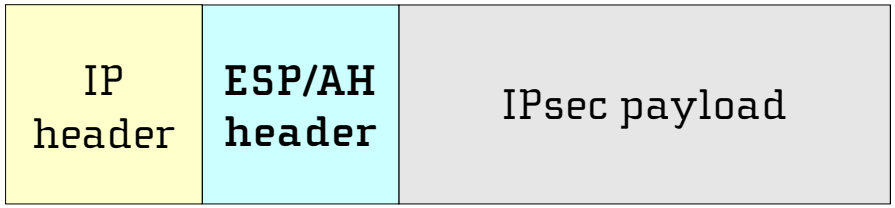
- IP protocol #49
- Provides only **authentication**, no encryption
  - Not really usefull with IPv4
  - **ESP** provides similar functionality

IKE – Internet Key Exchange

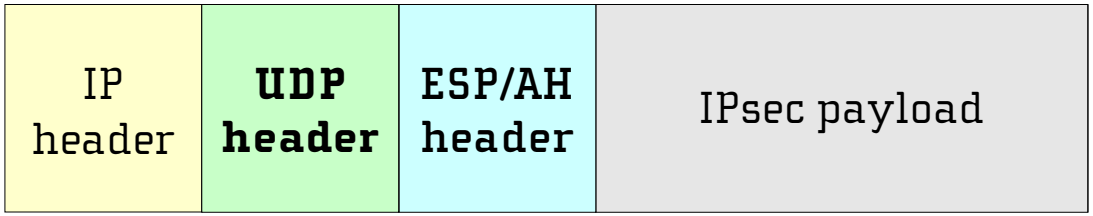
- UDP protocol, port 500
- Implemented in userspace.
- Can use different methods for peer's authentication:
  - Preshared keys
  - X.509 certificates
  - RSA keys
  - Kerberos tickets
- Result is a key used for **ESP**/**AH** protocols.

NAT Traversal

- **NAT Traversal** (NAT-T)
  - Encapsulates IPsec traffic into UDP packets for passing through NAT gateways and firewalls.
  - Native IPsec

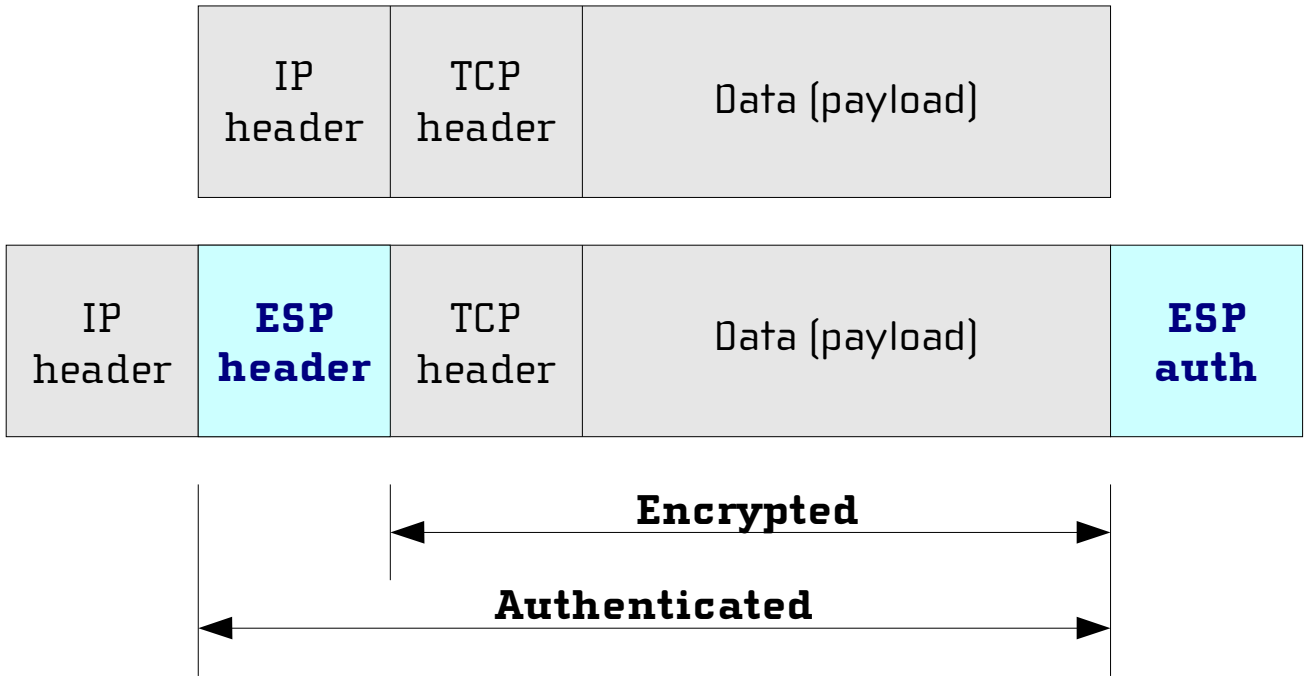


  - With NAT-T



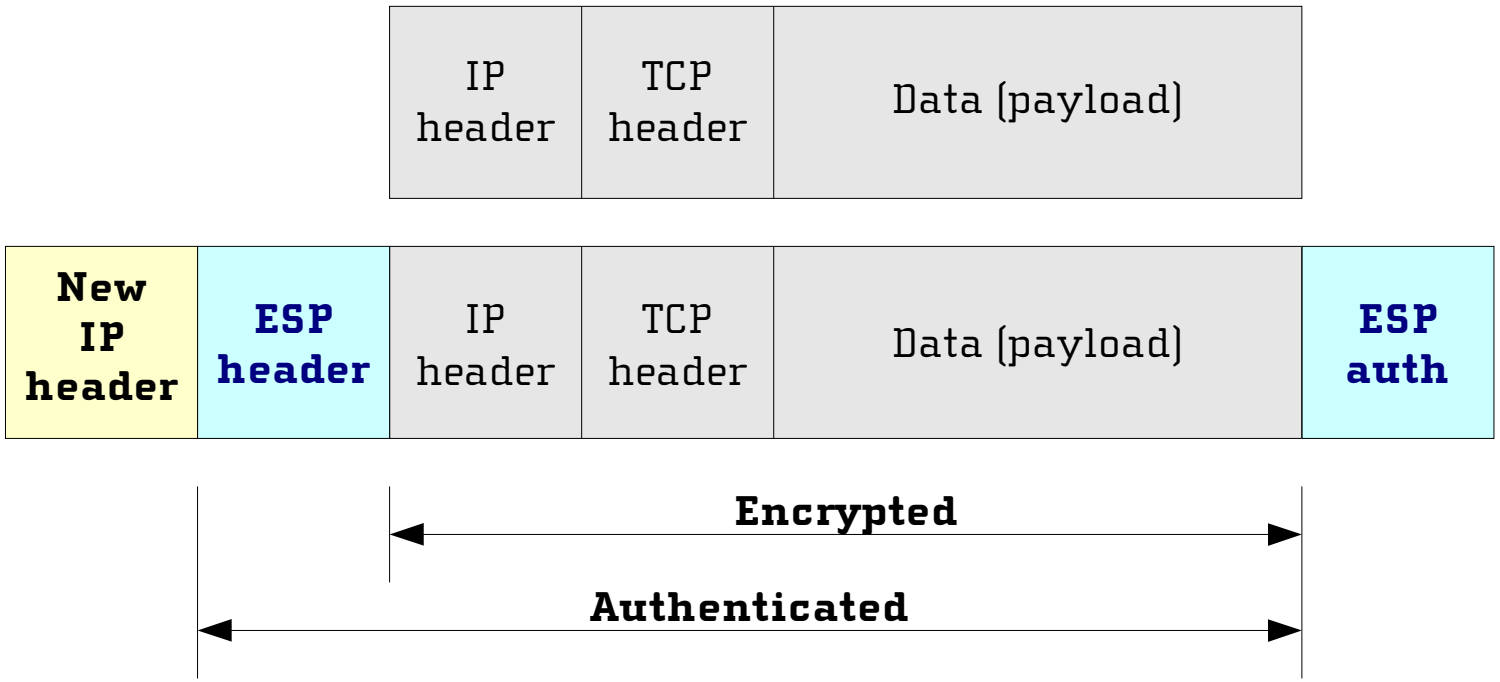
Transport mode

- Encapsulates only the transported data, not the IP headers.
- Used for **host to host** connections.



Tunnel mode

- The whole original packet is encapsulated.
- Used for **network to network** connections.



Firewalls

- Firewalling the **encrypted traffic**
  - Only ESP protocol and UDP port 500 required:

```
# iptables -A INPUT -p udp –dport 500 -j ACCEPT
# iptables -A INPUT -p esp -j ACCEPT
```

- Firewalling the **tunelled traffic**
  - Done on device ipsec0 (only with FreeS/WAN)
  - Example: blocking outgoing ICMP over IPsec tunnel

```
# iptables -A FORWARD -p icmp -o ipsec0 -j DROP
```

Read more ...

- Advanced topics for interested audience
  - **SPDB** – Security Policy Database
    - Contains IPsec policies for different *From<->To* traffic.
  - **SADB** - Security Association Database
    - Holds cryptographic keys for use with ESP/AH.
  - **PF_KEYv2**
    - Kernel interface for managing SADB and SPDB entries.
  - Read more in appropriate RFCs :-)

Questions?

No?

Thank you!