Network Security

Encrypting Network Communication

Radboud University, The Netherlands

Spring 2019

Acknowledgement

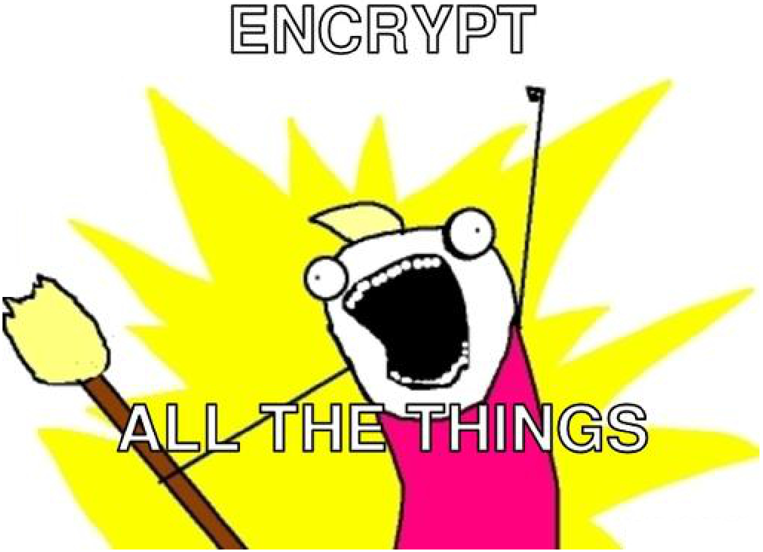
Slides (in particular pictures) are based on lecture slides by Ruben Niederhagen (http://polycephaly.org)

A short recap

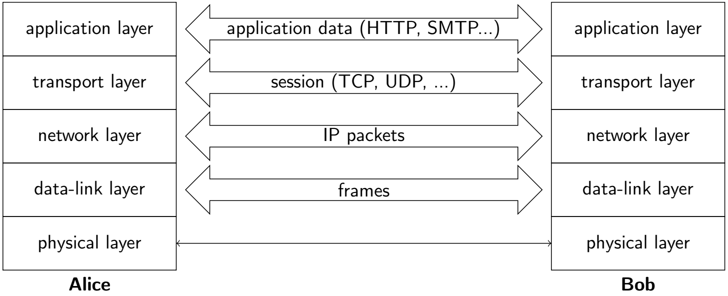
- Hostname resolution in the Internet uses DNS
- Two kinds of servers: authoritative and caching
- Two kinds of requests: iterative and recursive
- DNS tunneling:
  - Encode (SSH) traffic in DNS requests to authoritative server
  - Special authoritative server extracts and handles SSH data
- DNS DDOS amplification:
  - Send DNS request with spoofed target IP address
  - Much larger reply launched onto target
- DNS spoofing/cache poisoning: provide wrong DNS data
- Blind spoofing: cannot see (but trigger) request
- Countermeasure against blind spoofing: randomization
- Most powerful attack: sniffing DNS spoofing
- Countermeasures: Use crypto to protect DNS
  - DNSSEC (with various problems)
  - Alternative: DNSCurve
  - Other alternative: DNS over HTTPS (DoH)

A longer recap

- So far in this lecture: various attacks (often MitM):
  - ARP spoofing
  - Routing attacks
  - DNS Attacks
- Conclusion: sniffing (and modifying) network traffic is not dark arts
- It's doable for 2nd-year Bachelor students
- It's even easier for administrators of routers
- So far, relatively little on countermeasures... so, what now?

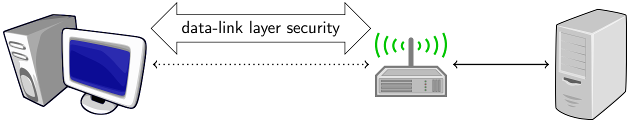

ENCRYPT
ALL THE THINGS

Cryptography in the TCP/IP stack



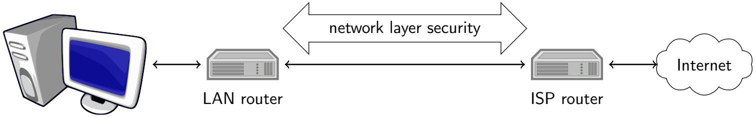
- Application-layer security (e.g., PGP, S/MIME, OTR)
- Transport-layer security (e.g., TLS/SSL)
- Network-layer security (e.g., IPsec)
- Link-layer security (e.g., WEP, WPA, WPA2)

Link-layer security



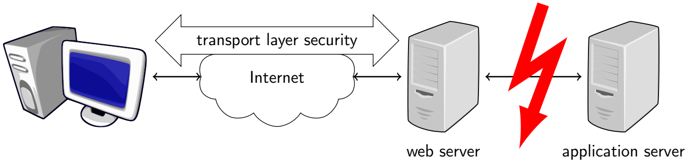
- Encrypt all network packets between network links, e.g., WPA2
- Point-to-point security between network interfaces
- "Encrypt to a MAC address"
- Careful if all hosts in a network use the same key (PSK):
  Protection only against *outsiders*

Network-layer security



- Encrypt IP packets, standard protocol: IPsec
- Point-to-point security between entities identified by IP addresses, typically routers or firewalls
- Routers encrypt and decrypt unnoticed by higher layers
- "Encrypt to an IP address"

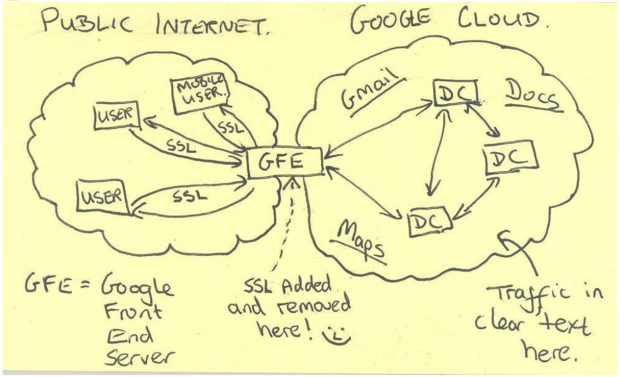
Transport-layer security



- Encrypt sessions and messages, e.g. TLS/SSL
- communication between web browser and server, or email clients and servers
- entities identified by connections, port numbers
- “Encrypt to a server process”
- part of the communication might still be unprotected (to application server or between mail servers)

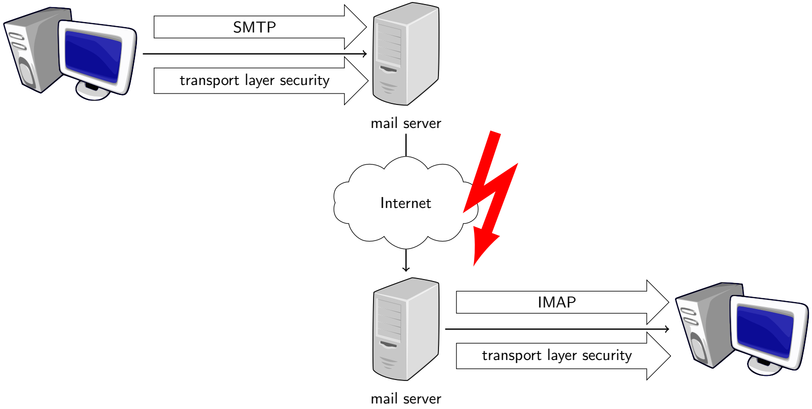
TOP SECRET//SI//NOFORN

Current Efforts - Google

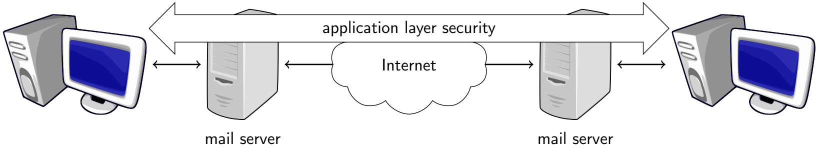

PUBLIC INTERNET.
GOOGLE CLOUD.
USER
MOBILE USER.
USER
SSL
SSL
SSL
GFE
Gmail
DC
Docs
DC
DC
Maps
GFE = Google Front End Server
SSL Added and removed here! ☺
Traffic in clear text here.

TOP SECRET//SI//NOFORN

Transport-layer security



Application-layer security



- Add security to standard message formats
- For email: entire link between two user mail clients is protected
- More prominent: secure messaging (e.g., Signal, Wire)
- authentication of sender and data
- end users have control over their keys
  (but need to know what they are doing, what keys to trust)
- end-to-end security (“encrypt to an e-mail address or ID”)

IPsec

- Obvious first reflex: we want end-to-end security
- How many people here regularly encrypt e-mail?
- How many people here already did before first-semester “Security” lecture?
- Problem with application-level security: users
  - Need to rewrite every single application
  - Need users to switch to secured applications
  - Need users to take care of keys
- Not impossible. . . who is using WhatsApp or Signal?
- But tricky. Who checked the fingerprints of their contacts?
- Transport-layer security needs applications to be modified to use secure transport layer
- Idea of network-layer security: No need to change applications (or user behavior)
- IPsec’s promise: network security happening without you even noticing

IPsec overview (simplified)

IPSec is a protocol *suite*

- Authentication header (AH) protocol
  - Transport mode
  - Tunnel mode
- Encapsulating Security Payloads (ESP) protocol
  - Transport mode
  - Tunnel mode
- Security Association (SA) protocol

IPsec – Security Associations

- Concept to formalize unidirectional security relationships between two parties
- Security Association Database (SADB) contains list of active security associations (SA)

SA parameters:

- sequence number, sequence number overflow
- anti-replay window
- AH information: authentication algorithm, key, key lifetime, etc.
- ESP information: encryption algorithm, key, key lifetime, etc.
- lifetime of the SA
- IPsec protocol mode (tunnel or transport)
- maximal packet size

IPsec – Modes of Operation

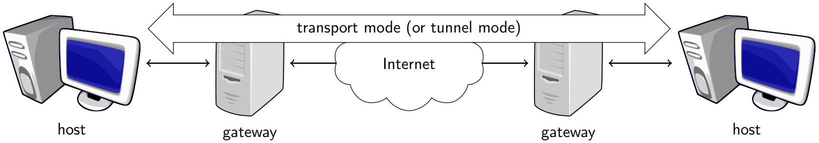
Transport mode:

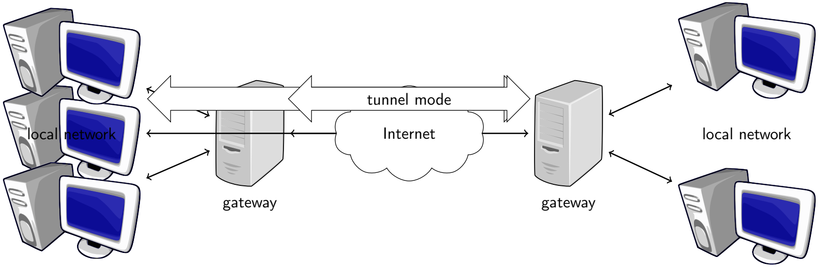
- Only the payload of the IP packet is protected
- Data is protected from source to destination
- Header information is completely in the clear
- Used only between hosts

Tunnel mode:

- Entire IP packet is protected (i.e. IP header and data)
- Becomes the payload of a new IP packet
- May contain different source and destination addresses
- Can be used between hosts, gateways, or host-gateway

IPsec – Modes of Operation





IPsec – Authentication Header

The Authentication Header provides

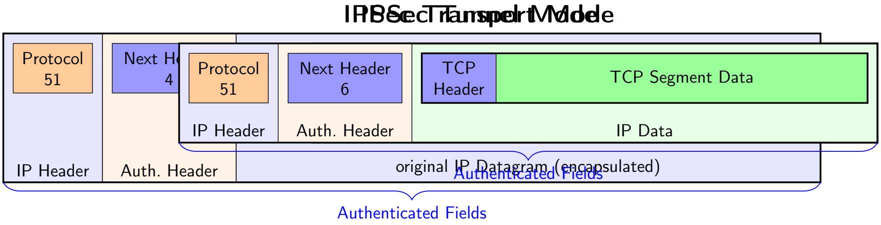
- data integrity,
- authentication of IP packets,
- protection against replay attacks.

First two by use of a Message Authentication Code (MAC), e.g. HMAC-SHA1-96.

IP packet is expanded with an AH that contains items such as:

- next header — type of the header following this header,
- payload length — length of AH,
- Security Parameter Index (SPI) — identifies an SA,
- sequence number,
- authentication data — contains the MAC of the packet, also called Integrity Check Value (ICV).

IPsec – Authentication Header



ICV (truncated HMAC) is computed over:

- immutable IP header fields (fields that do not change in transit), e.g., source address, IP header length,
- Auth. Header (except authentication data field),
- IP data.

Excluded fields are set to zero for HMAC computation.

IPsec – Authentication Header

Anti-replay protection prevents resending copies of authenticated packets.

- Uses sequence number field.
- For each new SA, sequence counter set to 0.
- Keep track of overflow (sequence number is 32 bits), negotiate new SA when counter reaches $2^{32} - 1$.
- Check whether counter is in window of fixed size.
- Right edge $=$ highest sequence number so far received (with valid authentication).
- Mark numbers of received packets with valid authentication.
- Advance window if new sequence number falls to the right of window and packet authenticates.
- Discard packet if number falls to the left of window or packet does not authenticate.

IPsec – Encapsulating Security Payload (ESP)

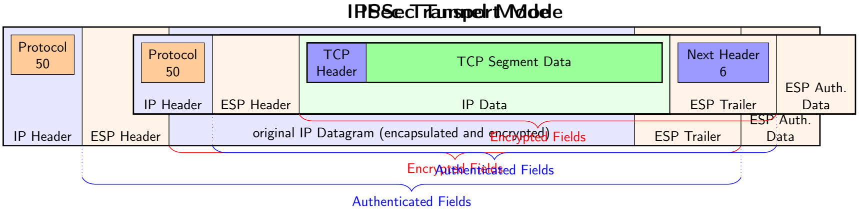
The Encapsulating Security Payload provides:

- confidentiality, i.e. encryption with block cipher in CBC mode, e.g. AES-CBC,
- functionality as in AH-like authentication, anti-replay (optional).

ESP adds an ESP header, encrypts the payload and adds an ESP trailer. An ESP packet contains:

- security parameter index (SPI),
- sequence number,
- payload data (encrypted),
- padding – to achieve data length a multiple of 32 bits (encrypted),
- padding length (encrypted),
- next header (encrypted),
- (optional) authentication data.

IPsec – Encapsulating Security Payload



- In transport mode, only data is encrypted, i.e. source and destination are in the clear
- In tunnel mode, the whole package is encrypted, i.e. real source and destination addresses are hidden
- Authentication not over IP header fields, only ESP header and data

IPsec - crypto algorithms (until 2014)

See RFC 4835 (now obsolete)

- Encryption: block ciphers in Cipher Block Chaining (CBC) mode
  Must have:
  - NULL encryption (RFC 2410)
  - AES-CBC with 128-bit keys
  - TripleDES-CBC (168-bit keys)
- Message authentication/integrity: Hash-based Message Authentication Code (HMAC),
  Must have:
  - HMAC-SHA1-96

  May have:
  - HMAC-MD5-96
- These are symmetric algorithms, need a pre-shared secret key
- Different options for key-agreement protocols: PSK, Internet Key Exchange (IKE, IKE2), Kerberos (KINK), IPSECKEY DNS records

IPsec - crypto algorithms (since 2014)

See RFC 7321

Old Requirement	**New Requirement**	Algorithm
MAY	SHOULD+	AES-GCM with a 16 octet ICV
MAY	SHOULD+	AES-GMAC with AES-128
MUST-	MAY	TripleDES-CBC
SHOULD NOT	MUST NOT	DES-CBC
SHOULD+	SHOULD	AES-XCBC-MAC-96
SHOULD	MAY	AES-CTR

IPsec problems

- Crypto of IPsec is not really state of the art
- IPsec ESP allows (in principle) encryption without authentication
- Attack by Degabriele and Paterson, 2007
- Consequence: don't use encrypt-only!
- IPsec AH authenticates IP header (incl. source and dest.)
- NAT changes IP header (source or dest.)
- Possible to get IPsec through NAT, but needs effort (RFC 3715)
- Most important problem: **It's complicated!**

"The first two generations of these documents (principally RFCs 1825–1829, published in 1995, and 2401–2412, published in 1998) are really only intended to provide a guide for implementors and are notoriously complex, difficult to interpret and lacking in overall structure.

...

The third and latest incarnation of the core IPsec standards were published as RFCs 4301–4309 in December 2005, and are somewhat more accessible.

...

Another quote. . .

“We are of two minds about IPsec. On the one hand, IPsec is far better than any IP security protocol that has come before: Microsoft PPTP, L2TP, etc. On the other hand, we do not believe that it will ever result in a secure operational system. It is far too complex, and the complexity has lead to a large number of ambiguities, contradictions, inefficiencies, and weaknesses. It has been very hard work to perform any kind of security analysis; we do not feel that we fully understand the system, let alone have fully analyzed it.” —Ferguson, Schneier, 2003

Userspace VPN

- Sort-of alternative to IPsec tunnel: `sshuttle` (“poor-man’s VPN”)
- Disadvantages:
  - You need SSH access to the target
  - Need `iptables` rules to redirect traffic
- Generalize this idea: *user-space VPN*
- Software that authenticates users and tunnels traffic
- Examples: SSH, OpenVPN, WireGuard
- Question: How does the software get the traffic to tunnel (preferably without `iptables`)

TUN interfaces

- Linux provides TUN (tunneling) “software network interface”
- For routing, this acts like any other interface
- Output *IP* packets are fed into software that reads from file `/dev/net/tun`
- Use this mechanism to set up VPN between `tyrion` and `arya` with SSH:

```
tyrion # echo 1 > /proc/sys/net/ipv4/ip_forward
tyrion # ip tuntap add dev tun3 mode tun
tyrion # ip addr add dev tun3 10.0.5.1/24
tyrion # ip l set dev tun3 up

arya # echo 1 > /proc/sys/net/ipv4/ip_forward
arya # ip tuntap add dev tun5 mode tun
arya # ip addr add dev tun5 10.0.5.2/24
arya # ip l set dev tun5 up

tyrion # ssh -o Tunnel=point-to-point -w 3:5 arya
```

- Now try:

```
tyrion # ping 10.0.5.2
```

TAP interfaces

- TUN interfaces input/output IP packets
- Alternative: TAP interfaces that input/output ethernet frames
- Example (again with SSH)

```
tyrion # echo 1 > /proc/sys/net/ipv4/ip_forward
tyrion # ip tuntap add dev tap3 mode tap
tyrion # ip addr add dev tap3 10.0.5.1/24
tyrion # ip l set dev tap3 up

arya # echo 1 > /proc/sys/net/ipv4/ip_forward
arya # ip tuntap add dev tap5 mode tap
arya # ip addr add dev tap5 10.0.5.2/24
arya # ip l set dev tap5 up

tyrion # ssh -o Tunnel=ethernet -w 3:5 arya
```

- Now try:

```
tyrion # ping 10.0.5.2
```

- You receive ARP packets through TAP
- The hosts are logically connected on the link layer
- They are in the same broadcast domain

SSL/TLS

Secure Sockets Layer (SSL) and Transport Layer Security (TLS):

- TLS is a variant of SSLv3
- Today use TLS 1.2 or 1.3
- SSL originally designed for web environment by Netscape
- Design goals: security of web traffic, email, etc.
- Had to work well with HTTP
- Provides transparency for higher layers

SSL/TLS provides a secure channel between server and client:

- Confidentiality
- Server (and client) authentication
- Message integrity

SSL/TLS

SSL/TLS runs on top of TCP:

- Transparent for application-layer protocols
- SSL/TLS connection acts like a secured TCP connection
- Most protocols running over TCP can be run over SSL/TLS instead
  e.g., HTTP $\rightarrow$ HTTPS, SMTP $\rightarrow$ SMTPS, ...

Protocols in SSL/TLS:

- Handshake Protocol: initiate session,
  Authenticate server/client, establish keys
- Record Protocol: data transfer,
  Compute MAC for integrity, encrypt MAC and data
- Alert Protocol: alert the other side of exceptional conditions,
  e.g., errors and warnings.

SSL/TLS (< 1.3) Handshake

- Client $\rightarrow$ Server: ClientHello
  - ClientRandom: random number,
  - Session ID (when resuming a session),
  - List of available CipherSuites:
    pk key exchange, pk auth, sym encryption, hash alg.

    Example: `TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256`

ECDH	Elliptic curve Diffie Hellman key exchange.
ECDSA	Elliptic curve digital signature algorithm.
AES_128_CBC	AES with 128-bit key in CBC mode.
SHA256	SHA with 256-bit output for HMAC.

SSL/TLS (< 1.3) Handshake (cont.)

- Server $\rightarrow$ Client: ServerHello
  - ServerRandom: random number,
  - Session ID: implementation specific, random number
  - Chosen CipherSuite.
- Server $\rightarrow$ Client: Certificate
  - Server sends server certificate to client,
    client obtains server's public key and verifies certificate.
- Server $\rightarrow$ Client: ServerKeyExchange

for DHE:	P^a, random a,
for ECDHE:	$[a]P$, random a,
for RSA:	–

- Server $\rightarrow$ Client: ServerHelloDone
  - Message marks end of server messages.

SSL/TLS Handshake (cont.)

- Client $\rightarrow$ Server: ClientKeyExchange

for DHE:	P^b for a random b,
for ECDHE:	$[b]P$ for a random b,
for RSA:	random value encrypted with server's public key.

- Client $\rightarrow$ Server: ChangeCipherSpec
  - Notify that client switched to new CipherSuite.
- Client $\rightarrow$ Server: Finished
  - Encrypted Finished message containing hash over the previous handshake messages.

- For DHE and ECDHE, client and server compute joint session key.

SSL/TLS Handshake (cont.)

- Server $\rightarrow$ Client: ChangeCipherSpec
  - Notify that server switched to new CipherSuite.
- Server $\rightarrow$ Client: Finished
  - Encrypted Finished message containing hash over the previous handshake messages.

Interrupted session can be resumed:

- Server and client are supposed to store session ID and MasterSecret,
- client sends session ID in ClientHello,
- reduced protocol: Hello, ChangeCipherSpec and Finished messages,
- new keying data is exchanged,
- new session keys are derived.

SSL/TLS Record Protocol

Record protocol to exchange encrypted and authenticated data:

- Payload data is split into fragments
  which are protected and transmitted independently;
  when received, fragments are decrypted and verified independently.
- Each fragment is authenticated with a MAC which is appended;
  MAC is over a sequence number (anti-replay) and the content.
- Data fragment and MAC are encrypted.
- A record header is attached to the encrypted data,
  containing information necessary for interpreting the record
  such as type of data (e.g. Handshake or ApplicationData),
  length, and SSL version.
- (header || encrypted fragment and MAC) is sent.

What SSL/TLS Cipher Suites to use?

- Earlier protocol versions support NULL and EXPORT ciphers
- Also, earlier versions included RC4, DES, MD5, SHA1. . .
- Major overhaul in TLS 1.3: only support reasonable crypto
- Still, some algorithms are hard to implement securely (e.g., AES-GCM) or prone to randomness attacks (e.g., DSA/ECDSA)

Who do you trust?

- HTTPS (HTTP over SSL/TLS) uses pre-installed root certificates in the browser
- Operating systems come with various pre-installed certificates
- Authenticating a communication partner means: follow chain of trust to root CA
- Compromise one root CA and all browsers are compromised
- Forge a root CA's certificate and all browsers are compromised
- Rogue CA certificate from MD5 vulnerabilities, 2008: http://www.win.tue.nl/hashclash/rogue-ca/
- DigiNotar compromised in 2011: >300,000 Iranian Gmail users compromised

SSLstrip

- Marlinspike, 2009: `sslstrip`
- Possible for an active attacker to “avoid” HTTPS
- Idea: rewrite links from HTTPS to HTTP
- Requires that client does not enforce HTTPS
- More details:
  - Erik’s lecture on Web Security
  - `http://www.thoughtcrime.org/software/sslstrip/`
  - `https://www.youtube.com/watch?v=MFol6IMbZ7Y`
- Last homework assigment…

Practice exam

https://cryptojedi.org/peter/teaching/netsec2019/practice-exam.pdf

- No solutions
- Q&A in the lecture and lab session next week
- More questions by e-mail to me