Internet Technology

Security

Michael Welzl michael.welzl@uibk.ac.at

DPS NSG Team http://dps.uibk.ac.at/nsg
Institute of Computer Science
University of Innsbruck, Austria

Scope

- Note: only interested in communication related attacks!
  - not: exploitation of OS vulnerabilities (software flaws)!
    ⇒ assumption: software bug-free :)
- Examples of attacks based on software flaws:
  - viruses (flaw in email tool, ..), worms (flaw in web servers, ..), rlogin, ..
- Very common attack (related to network programming): **Buffer Overflow**
  - Assumption 1: (e.g., C) program writes into buffer without proper checks
    data source: Internet packet content
  - Assumption 2: knowledge of OS, compilers, .. ⇒ memory layout
  - Idea: write malicious code into buffer, overwrite function return address
    ⇒ make system execute desired code (e.g., shell with root rights)

...thus, **be careful with memory operations!**

Typical scenario

Carlos Alice Bob

What could Trudy do?

- eavesdrop
- claim to be Alice (to hear Bob's answer)
- change message
(e.g. have Bob call Carlos on the phone)
- deny the service (break the telephone)

Trudy

Considerations for Alice and Bob

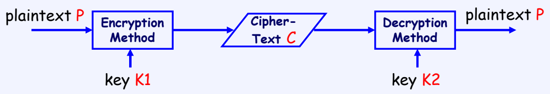
- Confidentiality
  - encryption / decryption using private or public keys
  - prevent eavesdropping: only sender and receiver should understand
- Authentication
  - ensure correct identity of sender and receiver
- Message integrity and nonrepudiation
  - malicious third person should not have a chance to change the content!
  - should be possible to prove that message X was sent by sender Y.
- Availability and access control
  - Common Denial-of-service (DoS) attacks make a system unavailable

Security and layers

Should? Consider e2e arguments...

- Confidentiality
  - Layer-independent; can be implemented at a very high layer!
  - Consider: packet sniffing - common link layer threat (WLAN)
- Authentication
  - relevant at all layers!
  - Consider: IP spoofing (fake source IP address), playback attack (resend sniffed data), man-in-the-middle attack (transparently introduce intermediate system: from A ⇔ B to A ⇔ X ⇔ B - X acts like A to B and like B to A) - common network layer threats
- Message integrity and nonrepudiation
  - Changing content occurs in transit - thus, ideally: network/transport layers
- Availability and access control
  - Layer-independent

Cryptology



- Symmetric (private) key system:
  - K1 = K2; known only to sender and receiver
  - e.g. Caesar cipher (shift letters by fixed amount): not so hard to crack; e.g. when word is known to occur or letter occurrence frequency is known
  - Data Encryption Standard (DES): 56 bit common, but failed in a "challenge"
  - Remaining question: how to distribute K?
- Asymmetric (public) key system:
  - K1 public (but associated with receiver, e.g. contained in Bob's email signature)
  - K2 secret (known only to Bob)

Public key encryption/decryption: RSA

RSA (Rivest, Shamir, Adleman):
- choose two large primes, p and q (> 10^{100})
- compute n = p x q, z = (p-1) x (q-1)
- choose e relatively prime to z (i.e. e and z have no common factors)
- find d such that e x d mod z = 1

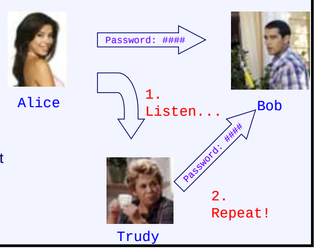
Simple example:
- p = 3, q = 11 ⇒ n = 33, z = 20
- then e.g., d = 7 (rel. prime to z = 2x2x5)
- then e.g., e = 3 (3 x 7 = 21, 21 mod 20 = 1)

Encryption of P: C = P^e (mod n) ⇒ public key: (n, e)
Decryption of C: P = C^d (mod n) ⇒ private key: (n, d)

Intruder must factor n into p, q: said to take 100^{25} years for 500-digit n, while n is only a few hundred bytes.

Authentication

- Who am I communicating with?
  - phone: recognizing voice helps
  - letter: authentication done via signature
- Need a signature for digital communication!
- Common: password
- Problem: eavesdropping
  - even encryption cannot prevent playback attack!



Authentication /2

- Obvious solution: require password to change with every message
  - e.g., number of cycling passwords, change passwords according to a rule
- Nonce: random number from Bob, must be used in Alice's answer
  - Similar to TCP connection setup (reflected seqno prevents server from mistaking old SYN)
  - e.g., with RSA: Alice could encrypt nonce with her private key, Bob could then decrypt it with her public key; If result correct, sender is Alice (only she knows her private key)
  - Requires Bob to retrieve Alice's public key
  - Can be intercepted by Trudy; thus, whole process is only as secure as key exchange
- Can only be solved by adding a trusted intermediary which distributes keys
  - Certification Authority (CA) certifies that public key belongs to an entity (person)
  - Key Distribution Center (KDC): used for symmetric key systems
    - stores per-person key (e.g. manually configured)
    - Alice uses it to retrieve a one-time session key ("I want to talk to Bob")
    - Well known example: Kerberos authentication service
- CA, KDC must be trustworthy - e.g., governmental

Integrity

- Public key encryption for every message is not convenient
  - Problems with RSA method: large result, computationally expensive
  - Desirable: less computational overhead, small fixed size result
- RSA recovers complete message from signature; unnecessary!
  - Proof of sender could just as well use RSA over message checksum only
  - Or calculate a checksum, for that matter...
- Thus, better solution: digital signature
  digital equivalent of actual signature; uniquely identifies a person
- Goal of checksum is to find errors; goal of signature is to be unique!
  - Solution: message digest, e.g. MD5 (128 bit); quite similar to checksum
  - Note: checksums, message digests are hash functions

Security in practice

Example systems

Pretty Good Privacy (PGP)

- Email security solution, invented by Phil Zimmerman 1991
  - famous criminal investigation case by the US government
  - after 3 years, case dropped in 1996
- PGP does it all:
  - symmetric key cryptography
  - public key cryptography
  - digital signature
- Flexible: choice of algorithms
- Public keys commonly distributed online (sig-file, website)

  Also: PGP signing parties, e.g. at IETF meetings

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Bob:My husband is out of town
    tonight.Passionately yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7
    G6m5Gw2
---END PGP SIGNATURE---
```


Secure Socket Layer (SSL)

- Developed by Netscape
- Layered on top of TCP, yet application independent
  - selected by using a specific port; e.g., standard port 443 for HTTP
  - HTTP which uses SSL = HTTPS
- Security services:
  - server authentication (e.g. via predefined trusted CAs in browser)
  - data encryption
    - Browser generates symmetric key
    - encrypts it with server's public key from CA
    - server decrypts symmetric key with private key
    - then, symmetric key is used
  - client authentication (optional, uses client certificates)
- IETF successor: "Transport Layer Security (TLS)"

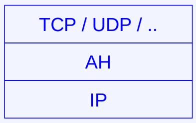
IPsec

- IPSec = protocol *suite* (not a single protocol)
  - provides framework for new encryption or authentication algorithms
    ⇒ can survive if algorithm is broken!
- Network layer security:
  automatically affects the whole TCP/IP stack (TCP, UDP, ICMP, SNMP, ..)
- Authentication + data integrity
  - Authentication Header (AH) protocol
- ... + confidentiality
  - Encapsulation Security Payload (ESP) protocol
  - More complicated (requiring more processing) than AH
- For both AH and ESP, source, destination handshake:
  - create Service Agreement (SA): *unidirectional network-layer logical channel*
  - uniquely identified by: protocol (AH or ESP), source IP address, Security Parameter Index (SPI) (32-bit connection ID)

Not connectionless anymore!

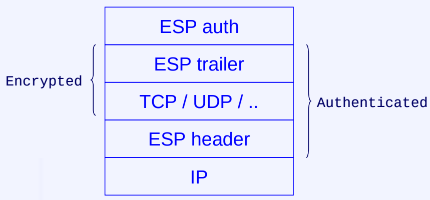
Authentication Header (AH) Protocol

- AH header inserted between IP and transport header (TCP/UDP)
- Fields:
  - Next Header - similar to "Protocol" field in IP header
  - Security Parameter Index (SPI) - 32-bit connection ID
  - Sequence Number - used to prevent playback and man-in-the-middle attacks
  - Authentication Data - variable length field containing a digital signature, computed using the algorithm specified by the SA
- AH authenticates complete packet (also IP header except TTL)



Encapsulation Security Payload (ESP) Protocol

- Fields (similar to AH, but different position):
  - ESP header: Security Parameter Index (SPI), Sequence Number - similar to AH
  - ESP trailer: Next Header - encrypted!
  - ESP auth: Authentication Data



More IPsec facts

- Internet Key Exchange (IKE) algorithm
  - default key management protocol for IPsec
- Internet Security Association and Key Management Protocol (ISKMP)
  - definition of procedures for SA setup/teardown
- Tunnel mode
  - *transparently* deploy IPsec (security gateway machines / firewalls)
    possibility: bundle TCP connections to hide communicating peers
  - encapsulate / decapsulate complete packet (also IP header)
- IPsec works with IPv4 and IPv6 (AH is extension header in IPv6)
- AH = (roughly) subset of ESP, kept for historical / compatibility reasons
  - note: AH checks IP header!
- Several additional complex issues: NAT, PMTUD + tunnel mode

802.11 security

- Well-known problem: war driving, parking lot attacks
- Wired Equivalent Privacy (WEP) protocol uses symmetric key to
  - authenticate (128-bit nonce per frame)
  - encrypt (RC4 algorithm; works well iff key is never used more than once!)

  between host and wireless access point
- Does not define key distribution
- Known to be insecure - e.g., WEP key changes too often
- Solution: 802.11i, also called WPA2 (Wireless Protected Access)
  - defines key management using RADIUS authentication servers

Some other problems

DoS attacks

- DoS: prevent a system from operating properly
- Logic attacks
  - exploit software flaws
  - examples: Ping-of-Death, WinNuke, ..
  - Prevention: upgrade / repair software

Problem = OS
⇒ less interesting for the 'net

- Flooding attacks
  - overwhelm CPU, memory, network resources
  - Prevention: very difficult
    (how to distinguish „good“ from „bad“ requests?)
  - Typically small packets
    (most network resources limited by CPU, not bandwidth)
  - Examples: TCP SYN, TCP ACK, IP fragment, DNS request, ..

Idea: cause additional processing overhead

DoS attacks /2

- TCP SYN (and similar) attacks:
  - remember: per-flow state not scalable
  - TCP needs per-flow state (connection state, address, port numbers, ..)
  - 1 SYN packet: search through existing connections + allocate memory
  - TCP SYN attack exploits TCP scalability problem!
- Distributed attacks:
  - Install remote controlled daemon on "zombie" hosts
  - Use more network resources to increase the amount of packets
- IP spoofing:
  - use wrong IP source address
  - Variant: "reflector attack":
    - source address = innocent 3rd party, 3rd party replies (adds traffic)
    - amplified by broadcast addresses! Examples: smurf, fraggle

Fighting the SYN problem: Cookies

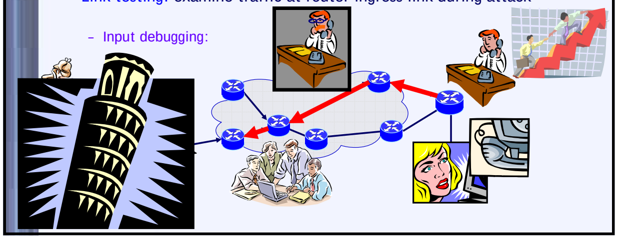
- SCTP: Association establishment - 4-way handshake
  - Host A sends INIT chunk to Host B
  - Host B returns INIT-ACK containing a cookie
    - information that only Host B can verify
    - No memory is allocated at this point!
  - Host A replies with COOKIE-ECHO chunk; may contain A's first data.
  - Host B checks validity of cookie; association is established
- TCP:
  - Sequence number negotiated at connection setup
  - Idea:
    - do not maintain state after SYN at server
    - encode cipher in sequence number from server to client
    - Client must reflect it ⇒ check integrity; if okay, generate state from ACK
  - Only requires changes at the server
  - See http://cr.yp.to/syncookies.html for further details (how to activate this in Linux, ..)

DoS identification

- Assumption: spoofed source addresses are chosen randomly
  (true for several known attack tools)
  - Victim's responses: equi-probably distributed across the entire Internet address space ("backscatter")
  - Probability of receiving a response: *n*m/2^32*
    (n=number of monitored hosts, m = number of flooding packets)
- Samples contain: victim address, kind of attack (port numbers, packet type), timestamp (⇒ calculate duration), lower limit of attack rate
  (*rate >= backscatter rate * 2^32/n*)
- Conservative result from monitoring a LAN ingress link:
  - 12805 attacks in 1 week
  - more than 5000 victim IP addresses in more than 2000 domains
  - 50% of attacks with more than 350 packets / s
  - 50 % of attacks from invalid TCP packets (probably TCP SYN)

DoS defence: traceback

- IP Traceback: find the source although the address is spoofed
  - problems: false computer accounts, call forwarding, reflector attacks
- Link testing: examine traffic at router ingress link during attack
  - Input debugging:



DoS defence: traceback /2

- Problem with Input Debugging: management overhead
- Controlled Flooding:
  - Flood links, observe DoS traffic perturbations
  - requires participating flooding hosts, good topological Internet map
  - requires no support from network operators!
  - problem: counter a DoS attack with a DoS attack?
- Logging:
  - log all traffic, detect path of flood packets via data mining
  - problem: resource requirements
  - advantage: can be used after attack
- Random marking schemes / ICMP Traceback:
  - very seldom: mark packets / generate packets with path information
  - victim can reconstruct path after the attack

Firewall trouble

- Typical configuration: block ICMP packets
- Path MTU Discovery
  - set IP "don't fragment" flag
  - start with big packets
  - [*gradually*] decrease size upon ICMP Destination Unreachable [- *Fragmentation Needed*] reply
- layer 3 functionality - may be initiated from layer 4
  - TCP problem with arbitrary packet drops
- Path MTU Discovery Black Hole Detection problem:
  No ICMP messages from unresponsive routers or filtered by firewalls
  ……hard to detect and solve!

NAT for security

- Actual IETF name: NAPT (Network Address / Port Translator)
  - also known as: masquerading, IP forwarding
- Map local ip addr. / (tcp or udp) port no. pair to globally unique ip address / port no.
  - single globally unique ip address can be used by several local hosts at once
- Some disadvantages (there are more!):
  - Problems with specific port numbers
  - Hard to set up a server behind a NAT (IP address not visible to the outside)
  - Architecturally critical; problems with many Internet mechanisms (e.g., mobility)
- One disadvantage can also be an advantage:
  Not visible to the outside = not an easy target for attacks!
  - e.g., problematic for Troyans

Conclusion: security and layers, again

- Security makes sense and may be required in many layers
- Advantage of security in lower layers:
  automatically provide security to everything on top
- Advantage of security in upper layers:
  specific security tied to application
- General question: what is tied to what?
  - e.g., WLAN authentication can only bind users to MAC addresses
  - IPSec authentication can only bind users to IP addresses
  - Similarly, SSL cannot solve an ECN security problem

References

- DoS:
  David Moore, Geoffrey M. Voelker & Stefan Savage, "Inferring Internet Denial-of-Service Activity", Proc. 2001 USENIX Security Symposium

  Stefan Savage, David Wetherall, Anna Karlin & Tom Anderson, "Network Support for IP Traceback", IEEE/ACM Transactions on Networking Vol. 9, No. 3, June 2001
- Path MTU Discovery / Firewalls:
  RFC 1191, RFC 2923, RFC 2979 (firewall)
- Everything else: any of the three books that were recommended for the "computer networks" lecture