Middleboxes, Cryptography, & Secure Protocols

CS640, 2014-04-30

Announcements
● Assign #5 due Thursday, May 7 @ 11pm

Outline
● Middleboxes
● Crypto algorithms
● Session keys
● Public key infrastructure
● Secure protocols

Middleboxes
● Systems in the “middle” of the network that examine and block packets and flows
  ○ Middle = on the path between pairs of communicating hosts
  ○ Packets forced to pass through middlebox -- based on physical topology, or using SDN
● Basic firewalls
  ○ Apply simple rules to decide whether to forward or block packets
  ○ Rules are based on fields in packet headers -- e.g., source/destination IPs, transport layer protocol, source/destination ports
  ○ Default rule to either forward or block if no other rules match
● Advanced firewalls
  ○ Maintain some state about active connections
    ■ E.g., current state of TCP connection -- SYN sent, SYN+ACK sent, established, FIN sent, etc.
    ■ Rules based on both packet headers and current state
  ○ Application-aware
    ■ Extra protection for services that should not be blocked (e.g., HTTP)
    ■ E.g., check if HTTP POST method is allowed
    ■ E.g., check if client is requesting a web page from a domain that is on a blacklist because it is known to host malware
    ■ Often acts as a proxy -- terminates TCP connection from client and establishes separate TCP connection to server
● Intrusion detection/prevention systems
  ○ Performs deep packet inspection
    ■ Look at payload of packet, not just headers, to decide if it should be blocked
    ■ Knows the format of packets for many different transport and application protocols -- HTTP, SSH, SSL/TLS, FTP, NFS, FTP, NTP, etc.
    ■ Cannot perform deep packet inspection on traffic that is encrypted!
  ○ Maintains state about active connections
    ■ Connection info such as src/dst IP, src/dst port, and TCP connection state
    ■ Reassembled payloads -- e.g., HTTP reply may be split among multiple packets, so the packets are reassembled into a single memory region that contains the entire reply
Uses a set of signatures (i.e., rules) to detect malicious traffic
- Specific sequences of packets -- e.g., TCP SYN+ACK after TCP FIN
- Keywords in payloads -- e.g., “root”
- MD5 sum of payload -- compare against database of MD5 sums for known malware
- Large numbers of packets to one host in a short period of time
  - DoS
  - Port scan -- look for hosts which have sockets listening on a particular port and will accept connections; host may be running an outdated version of software that has a known vulnerability
- IDS just raises alerts, while IPS raises alerts and blocks traffic
- Speed of signature matching can significantly impact latency and throughput
  - Want efficient pattern matching algorithms
  - Sometimes use custom hardware
  - Can be easily parallelized

### Basic Cryptography

- Encrypt/decrypt algorithm should be
  - Public -- inventing algorithms is hard, so we don’t want to have to develop a new one if something is leaked
  - Easy to compute with the key -- efficient in software and hardware, and on mobile devices which have fewer resources
  - Hard to compute without the key -- computers keep getting more powerful
- Key should be
  - Secret
  - Long -- length of key often determines “level” of security
- Types of functions
  - Cryptographic hash -- no keys
  - Symmetric/secret key -- one shared key
  - Asymmetric/public key -- pair of keys: one public & one private

### Cryptographic Hash
- Also known as “cryptographic checksum” -- used to detect if message has been tampered
- Take message $m$ of any length, and produce smaller message $h(m)$
- Properties
  - Pre-image resistance -- hard to find $m$ given $h(m)$; “one-way” function
  - Second pre-image resistance -- given a message $m$, it is hard to find a message $m'$ that hashes to the same $h(m)$
  - Collision resistance -- hard to find any two messages $m$ and $m'$ such that $h(m) = h(m')$
• How many bits for the hash?
  ○ If hash is \( n \) bits, it takes \( 2^n \) tries to find a collision
  ○ MD5 -- uses 128 bits; weakness known for a while
  ○ SHA-1 -- uses 160 bits; also not recommended for use
  ○ SHA-2 -- collection of six different hash functions; use 224, 256, 384 or 512 bits
  ○ SHA-3 -- emerging standard
• Example: self-certifying names
  ○ File-sharing software (e.g., BitTorrent) names files with \( h(\text{file-data}) \)
  ○ Verify \( h(\text{downloaded-data}) = \text{name of file} \)

**Symmetric/Secret Key**
• Sender and receiver share a common key
• None of the original structure of the plaintext should exist in the ciphertext
  ○ Otherwise, attackers could look for patterns -- e.g., commonly used letters in the English language; HTTP request starts with method (GET, POST, etc.)
• Variants
  ○ Data Encryption Standard (DES) -- 64-bit keys (8-bits are parity); easy to recover a key given today's processing power
  ○ Triple DES (3DES) -- 168-bit keys; encrypt using first 56-bits, decrypt using middle 56-bits, encrypt using last 56-bits (inverse for decryption); slow to implement in software
  ○ Advanced Encryption Standard (AES) -- 128, 192, or 256-bit keys; fast implementations in hardware or software and low memory footprint
• Challenge: key distribution
  ○ Physically deliver key -- not practical
  ○ Use existing key to deliver new key -- need unique key pair for each pair of endpoints
    \( n*(n-1)/2 \) total keys for \( n \) endpoints
  ○ Use key distribution center (KDC) -- KDC generates session keys and distributes them to pairs of endpoints that wants to communicate; need \( n \) master keys if you have \( n \) endpoints

**Asymmetric/Public Key**
• Pair of keys
  ○ Public key given to many senders
  ○ Private key kept a secret by the receiver (i.e., owner)
  ○ The private key cannot be derived from the public key
• Data is encrypted using the public key, but can only be decrypted using the private key
- Data can also be encrypted using the private key and decrypted using the public key
  - Obviously not useful for sending confidential data
  - But, can be used to verify a message came from a specific entity, since only one entity should have the private key and the ability to encrypt a message using that key
- RSA (named after inventors Rivest Shamir, and Adleman)
  - Relies on the high computational cost of factoring large numbers
- Much slower than symmetric key, but key distribution is easier

**Public Key Infrastructure**
- Mechanism for certifying bindings between public keys and identities
  - Identities could be email address, domain name, etc.
  - Binding ensured using a digital signature
- Public key certificate
  - Identity of entity being certified
  - Public key of the entity being certified
  - Identity of the signer
  - Digital signature
  - Compute digest of identifies and public key using cryptographic hash
  - Encrypt digest using signer’s private key
  - Anyone with signer’s public key can decrypt the digest and compare it against a digest they compute
  - Identifier for digital signature algorithm -- Elliptic Curve Digital Signature Algorithm (ECDSA) is the current standard
  - Expiration timestamp
- Chain of trust -- e.g., X certifies public key for Y, and Y certifies public key for Z, then there is a chain of certificates from X to Z
- Certificate Authority (CA)
  - Verifies identities and issues public key certificates
  - Trusted organization (e.g., VeriSign)
  - Web browsers preconfigured with certificates for common CAs
  - Publish certificate revocation list -- digitally signed list of certificates that have been voided