15-744: Computer Networking

L-23 Security

Security

- Denial of service
- IPSec
- Firewalls
- Assigned reading
  - [SWKA00] Practical Network Support for IP Traceback
  - [B89] Security Problems in the TCP/IP Protocol Suite

Overview

- Security holes
- Firewalls
- Denial of service traceback
- Authentication

Basic IP

- End hosts create IP packets and routers process them purely based on destination address alone (not quite in reality)
- Problem – End host may lie about other fields and not affect delivery
  - Source address – host may trick destination into believing that packet is from trusted source
    - Many applications use IP address as a simple authentication method
    - Solution – reverse path forwarding checks, better authentication
  - Fragmentation – can consume memory resources or otherwise trick destination/firewalls
    - Solution – disallow fragments

Routing

- Source routing
  - Destinations are expected to reverse source route for replies
  - Problem – Can force packets to be routed through convenient monitoring point
    - Solution – Disallow source routing – doesn't work well anyway!
- Routing protocol
  - Malicious hosts may advertise routes into network
  - Problem – Bogus routes may enable host to monitor traffic or deny service to others
    - Solutions
      - Use policy mechanisms to only accept routes from or to certain networks/entities
      - In link state routing, can use something like source routing to force packets onto valid route

ICMP

- Reports errors and other conditions from network to end hosts
- End hosts take actions to respond to error
- Problem
  - An entity can easily forge a variety of ICMP error messages
    - Redirect – informs end-hosts that it should be using different first hop route
    - Fragmentation – can confuse path MTU discovery
    - Destination unreachable – can cause transport connections to be dropped

TCP

- Each TCP connection has an agreed upon/negotiated set of associated state
  - Starting sequence numbers, port numbers
  - Knowing these parameters is sometimes used to provide some sense of security
- Problem
  - Easy to guess these values
    - Listening ports #'s are well known and connecting port #'s are typically allocated sequentially
    - Starting sequence number are chosen in predictable way
  - Solution – make sequence number selection more random

Sequence Number Guessing Attack

Attacker → Victim: SYN(ISN_x), SRC=Trusted Host
Victim → Trusted Host: SYN(ISN_s), ACK(ISN_x)
Attacker → Victim: ACK($ISN_{guess\ of\ s}$), SRC=Trusted Host
Attacker → Victim: ACK($ISN_{guess\ of\ s}$), SRC=T, data = "rm -r /"

- Attacker must also make sure that Trusted Host does not respond to SYNACK
- Can repeat until guess is accurate

TCP

- TCP senders assume that receivers behave in certain ways (e.g. when they send acks, etc.)
  - Congestion control is typically done on a "packet" basis while the rest of TCP is based on bytes
- Problem – misbehaving receiver can trick sender into ignoring congestion control
  - Ack every byte in packet!
  - Send extra duplicate acks
  - Ack before the data is received (needs some application level retransmission – e.g. HTTP 1.1 range requests)
  - Solutions
    - Make congestion control byte oriented
    - Add nonces to packets – acks return nonce to truly indicate reception

DNS

- Users/hosts typically trust the host-address mapping provided by DNS
- Problems
  - Zone transfers can provide useful list of target hosts
  - Interception of requests or comprise of DNS servers can result in bogus responses
  - Solution – authenticated requests/responses

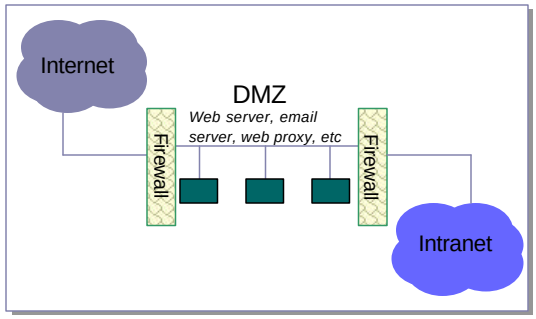
Overview

- Security holes
- Firewalls
- Denial of service traceback
- Authentication

Firewalls

- Basic problem – many network applications and protocols have security problems that are fixed over time
  - Difficult for users to keep up with changes and keep host secure
  - Solution
    - Administrators limit access to end hosts by using a firewall
    - Firewall and limited number of machines at site are kept up-to-date by administrators

Typical Firewall Topology



Types of Firewalls

- Proxy
  - End host connects to proxy and asks it to perform actions on its behalf
    - Policy determines if action is secure or insecure
  - Transport level relays (SOCKS)
    - Ask proxy to create, accept TCP (or UDP) connection
    - Cannot secure against insecure application
  - Application level relays (e.g. HTTP, FTP, telnet, etc.)
    - Ask proxy to perform application action (e.g. HTTP Get, FTP transfer)
    - Can use application action to determine security
  - Requires applications (or dynamically linked libraries) to be modified to use the proxy
  - Considered to be the most secure since it has most information to make decision

Types of Firewalls

- Packet filters
  - Set of filters and associated actions that are used on a packet by packet basis
  - Filters specify fields, masks and values to match against packet contents, input and output interface
  - Actions are typically forward or discard
  - Such systems have difficulty with things like fragments and a variety of attacks
  - Typically a difficult balance between the access given and the ability to run applications
    - E.g. FTP often needs inbound connections on arbitrary port numbers – either make it difficult to use FTP or limit its use

Types of Firewalls

- Stateful packet filters
  - Typically allow richer parsing of each packet (variable length fields, application headers, etc.)
  - Actions can include the addition of new rules and the creation of state to process future packets
    - Often have to parse application payload to determine “intent” and determine security considerations
  - Rules can be based on packet contents and state created by past packets
  - Provides many of the security benefits of proxies but without having to modify applications

Overview

- Security holes
- Firewalls
- Denial of service traceback
- Authentication

Denial of Service

- Objective of attack: make a service unusable, usually by overloading the server or network
- Example: SYN flooding attack
  - Send SYN packets with bogus source address
  - Server responds with SYNACK keeps state about TCP half-open connection
    - Eventually server memory is exhausted with this state
  - Solution: SYN cookies – make the SYNACK contents purely a function of SYN contents, therefore, it can be recomputed on reception of next ACK
- More recent attacks have used bandwidth floods
  - How do we stop these?

Bandwidth DOS Attacks

- Possible solutions
  - Ingress filtering – examine packets to identify bogus source addresses
  - Link testing – how routers either explicitly identify which hops are involved in attack or use controlled flooding and a network map to perturb attack traffic
  - Logging – log packets at key routers and post-process to identify attacker's path
  - ICMP traceback – sample occasional packets and copy path info into special ICMP messages
  - IP traceback

IP Traceback

- Node append (record route) – high computation and space overhead
- Node sampling – each router marks its IP address with some probability *p*
  - P(receiving mark from router d hops away) = $p(1-p)^{d-1}$
  - p > 0.5 prevents any attacker from inserting false router
  - Must infer distance by marking rate → relatively slow
  - Doesn't work well with multiple routers at same distance → I.e. multiple attackers

IP Traceback

- Edge sampling
  - Solve node sampling problems by encoding edges & distance from victim in messages
  - Start router sets "start" field with probability *p* and sets distance to 0
  - If distance is 0, router sets "end" field
  - All routers increment distance
  - As before, P(receiving mark from router d hops away) = $p(1-p)^{d-1}$
- Multiple attackers can be identified since edge identifies splits in reverse path

Edge Sampling

- Major problem – need to add about 72bits (2 address + hop count) of info into packets
- Solution
  - Encode edge as xor of nodes → reduce 64 bits to 32 bits
  - Ship only 8bits at a time and 3bits to indicate offset → 32 bits to 11bits
  - Use only 5 bit for distance → 8bits to 5bits
  - Use IP fragment field to store 16 bits
    - Some backward compatibility issues
    - Fragmentation is rare so not a big problem

Overview

- Security holes
- Firewalls
- Denial of service traceback
- Authentication

Kerberos

- Objective: provide a way for services to authenticate clients
- A client must present a ticket to use service
  - Ticket is obtained from Kerberos system
    - Contains client ID, session key
    - Ticket is encrypted using service's private key known only to service and Kerberos
    - Ensures that only Kerberos could have created ticket
  - Client uses authenticator to prevent replay of tickets
    - Contains client ID, network address, time of day
    - Encrypted using session key
    - Only if recent and Ids match is ticket valid
    - Forces time to be synchronized within few minutes

Kerberos

- Obtaining tickets
  - Client sends message to Kerberos
    - Contains service ID, client ID, time of day
  - Response encrypted with client's private key
    - Contains ticket, session key and timestamp
  - First ticket gotten is for Kerberos Ticket Granting Service
    - All other requests sent to the TGS using the TGS session key
    - Avoids having to provide passwd for each ticket

Kerberos

- To get the TGS ticket, the client contacts the KDS
  - User → workstation (WS): N_{client}
  - WS → Key Distribution Service (KDS): $\{N_{client}, N_{TGS}, T_{current}\}$
  - $Ticket_{TGS} = \{K_{session}, N_{client}, N_{TGS}, T_{current}, WS, Lifetime\}K_{TGS}$
  - KDS → WS: $\{K_{session}, N_{TGS}, Lifetime, T_{current}, Ticket_{TGS}\}K_{client}$
    - Above message is subject to know plaintext attack!
  - User → WS: password → used to decrypt previous message
- To use a service, client creates authenticator:
  - $Authenticator = \{N_{client}, T_{current}, WS\}K_{session}$
  - WS → service: {Authenticator, Ticket}
- Further exchanges are similar to KDS exchange but with TGS using $K_{session}$ instead of K_{client}

Certification Authorities

- Certification authority (CA) binds public key to particular entity
- Entity (person, router, etc.) can register its public key with CA
  - Entity provides "proof of identity" to CA
  - CA creates certificate binding entity to public key
  - Certificate digitally signed by CA

- When Alice wants Bob's public key:
  - Gets Bob's certificate (from Bob or elsewhere).
  - Applies CA's public key to Bob's certificate, get Bob's public key

Overview

- End2end security

IPsec

- Basic idea – add security at the IP layer for use by any upper layer protocol
- Two basic security protocols
  - Identified in IP protocol field
  - Authentication Header (AH)
    - Provides connectionless integrity, data origin authentication, and an optional anti-replay service
  - Encapsulating Security Payload (ESP)
    - Adds confidentiality (encryption) to AH
- Security association (SA)
  - A simplex "connection" between IPsec endpoints
  - Identifies keying info, destination, protocol info
  - Specifies Security Parameters Index (SPI) that is carried in packets

Operating Modes

- Two basic operating modes
- Transport mode
  - A transport mode security protocol header appears immediately after the IP header and any options, and before any higher layer protocols (e.g., TCP or UDP)
- Tunnel mode
  - An "outer" IP header specifies the IPsec processing destination, plus an "inner" IP header specifies the (apparently) ultimate destination for the packet
  - Often used to communicate between gateways to create a virtual private network

Packet Contents (AH)

- Next header – protocol number of contents
- Authentication payload length – length of authentication data
- SPI – Security Parameters Index used to lookup security association
- Sequence number – used for anti-replay
- Authentication data – data used to authenticate packet
- ESP is similar

Modular Design

- Basic protocol to setup security associations
  - Internet Security Association and Key Management Protocol (ISAKMP)
- Many different key exchange protocols, encryption methods and authentication methods

SSL

- Basic idea – add security at the transport layer for use by any application
  - Needs a reliable transport (TCP)
- Handshake
  - Exchange data to determine the strongest shared encryption algorithm
  - Server presents SSL certificate + public key
    - Client checks was issued by a certificate authority trusted by the browser, is not expired, is for the appropriate server
    - Authenticates the server
    - Used to setup key for connection – expensive operation
  - Symmetric key used for encryption
    - Can be used across connections to reduce connection setup cost

Next Lecture: Network Measurements

- How is the Internet holding up?
- Assigned reading
  - [Pax97] End-to-End Internet Packet Dynamics