# ECE 473/573
# Cloud Computing and Cloud Native Systems
# Lecture 25 Cloud Security

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

November 18, 2024

# Outline

TCP/IP Network Security

Web Security

# Reading Assignment

- This lecture: Cloud Security
- Next lecture: please watch the video on cloud security architecture from RSA Conference 2019 `https://www.youtube.com/watch?v=4TxvqZFMaoA`
  - We will not have an in-person class or a Zoom session.

# Outline

TCP/IP Network Security

Web Security

# Security in TCP/IP Network

- TCP/IP network is created to address availability concerns.
  - Confidentiality and integrity are expected to be addressed through a layered approach.
- How to protect TCP/IP communications?
  - For efficiency reasons, many widely used TCP/IP protocols like HTTP do not address confidentiality and integrity by default.
  - The attacker may see packets, requests, responses and etc., and modify them to inject malicious code.
- Compromised systems communicating via TCP/IP further complicate the security issuses
  - How to monitor and control TCP/IP communications?
- How to achieve security without requiring substantial changes to existing infrastructures?

# Internet Protocol Security (IPsec)

- A secure communication protocol at IP layer.
  - Any other service on top of IP, like TCP, obtains the same security guarantees automatically.
  - Encapsulate IP packets to be protected in AH and ESP IP packets – literally, no change is needed to route them.
- Authentication: host
- Modes of operation
  - Transport mode: protect host to host communication, e.g. among a group of servers within a data center.
  - Tunnel mode: protect router to router communication, e.g. a VPN across Internet that interconnects groups of servers at different geographic locations.
- Widely available, but usages are limited to professionals or specific applications like VPN due to its complexity.

# IPSec: Internet Key Exchange (IKE)

- Service running on IPSec hosts that establishes security associations (SA) among communicating parties.
  - Similar to key establishment.
  - Also include negotiation of ciphers, hash algorithms, and other security properties like lifetime.
- Authenticate hosts and establish session key by
  - Manual configurations of pre-shared keys or public keys.
  - Certificates signed by a trusted CA.
  - Delegation to other protocols like Kerberos.

# IPSec: AH and ESP

- Authentication Headers (AH)
  - IP protocol number 51
  - Provide integrity/message authentication
  - Optionally support sequence number to resist replay attacks.
- Encapsulating Security Payload (ESP)
  - IP protocol number 50
  - Provide confidentiality only (not recommended), or confidentiality and integrity (recommended).
  - Also resist replay attacks.
- Note that you can't hide/encrypt destination IP addresses; otherwise intermediate routers don't know where to route the packets.

# Transport Layer Security (TLS)

- Successor of Secure Sockets Layer (SSL)
  - SSL has been deprecated because of security concerns.
  - However, the name 'SSL' remains in use, e.g. when mentioning TLS as TLS/SSL, or using Java API.
  - You should use TLS 1.1 or above, and avoid SSL 1.0,2.0,3.0, as well as TLS 1.0 .
- Provide confidentiality and integrity over TCP connections.
  - Client connects to server via TCP, then negotiates via a handshaking procedure to determine cipher parameters and to perform authentication and key establishment.
  - Finally the byte streams are protected by authenticated encryption and sent over the TCP transport.

# TLS Authentication

- Via Public-Key Infrastructures (PKI).
- Server authentication
  - Server provides its certificate.
  - Client verifies the server certificate using the corresponding CA's public key.
- Client authentication
  - Server provides a list of CAs that it would trust.
  - Client provides one of its certificates that is signed by one of server's CAs.
  - Server verifies the client certificate using the corresponding CA's public key.
- Usually server authentication only.

# TLS: Certificates Management

- CA certificates (public key) distribution.
  - Usually as part of your OS installation.
  - Can be updated manually.
  - Only install OS from legitimate sources and be careful to provide others with root accesses to servers.
- Certificate revocation list (CRL)
  - Each certificate has an expiration date. An expired certificate won't be accepted.
    - Could attackers change that expiration date?
  - CAs will provide a list of all revoked certificates that are not expired, which should be refered when verifying certificates.
  - Clients and servers need to get this list on a timely basis.

# Firewalls

- Monitor and control network traffic with predefined rules.
  - Deny or allow network traffics based on source and destination addresses, protocol, content, etc.
  - Redirect packets by transforming their headers.
- Connectionless rules
  - Stateless: evaluate packets independently
  - Simple to implement efficiently.
  - E.g. to prevent all packets from a host B to reach a host A.
- Connection-based rules
  - Stateful: track packets within a stateful protocol like TCP.
  - Require resources to track protocol states.
  - E.g. to prevent a host B to initiate a communication to a host A while allowing A to initiate a communication with B.

# Outline

TCP/IP Network Security

Web Security

# HyperText Transfer Protocol Secure (HTTPS)

- A.k.a. HTTP over SSL or HTTP over TLS.
  - HTTP communication entirely on top of TLS (over TCP), usually use port 443.
  - Provide confidentiality and integrity.
  - Usually server authentication only, but client authentication could also be added.
- Domain name authentication
  - HTTPS server certificates need to include matching domain names and/or ip addresses for the connection to be considered secure by browsers.
  - Provide protection against IP address spoofing and DNS spoofing.
  - CA certificates can also be included with new browser installations – don't install browser from unknown sources!

# Authentication for RESTful Services

- Usually, RESTful requests are protected by HTTPS with server authentication only, and clients are authenticated with other means like username and password.
- It is not practical and not secure to send those information for every RESTful request.
  - Users only expect to input those information once.
  - Keeping those information in the memory of user's computer increases the risk of them been stolen by other malicious processes.
- How can we authenticate the user once and preserve the authenticated user across multiple RESTful requests?

# Cookie Based Authentication

- Cookie: a HTTP mechanism that allows HTTP servers to pass information back to HTTP clients.
  - As key-value pairs via the Set-Cookie HTTP response header.
  - Clients are required to send the cookie back for the following requests.
- Cookie based authentication
  - The first RESTful request would be a login request that contains user account name and password.
  - The server backend will authenticate the user and send back a cookie containing an randomly generated string, which is usually known as the access token or the session key.
  - All following requests will contain that access token to identify the user.

# Threats for Cookie Based Authentication

- Quality of access token
  - Attackers may trick the server backend to use a known token.
  - Attackers may successfully guess the access token.
  - The access token should be generated randomly after each successful login and be long enough.
- Attackers may read the access token from HTTP packets.
  - HTTPS should be used to protect all HTTP communications.
- Attackers may steal the access token from user's computer.
  - The server backend should mark cookies as session cookies for browsers to make better protection.
  - The server backend could further check for unusual uses of access tokens, e.g. unusual ip addresses.

# Cross-Site Scripting (XSS)

- Assume that there is a bug in a bank website that returns a web page containing whatever included in the HTTP request.
  - Actual bugs may be more subtle.
- An attacker, aware of the bug, create a HTTP request to the website containing a short JavaScript code.
  - It reads the access token and sends it back to a server controlled by the attacker.
- The attacker may trick a normal user to send the request.
- The browser then displays the returned web page and runs the malicious script.
  - The malicious script runs in a web page generated by the legitimate bank website so has full control over the access token.

# XSS Mitigation

- Same-origin policy: scripts (running in a browser) are only allowed to access resources in the same website.
  - So the malicious script cannot simply send the stolen access token back to a different server controlled by the attacker.
- RESTful services: separate code and data
  - Server-side scriptings are more likely to have XSS issues as HTTP requests and responses are usually interpreted directly.
  - It will be less risky if browsers do not interpret RESTful responses as runnable scripts – still, efforts are required on both backend and frontend.

# Summary

- TCP/IP network is not secure.
- But we can protect it with proper system setup and choice of protocols.
  - Without breaking existing network infrastructure and applications.