CNS Printing Lab moved from A/Z E100 to new Biology Bldg
- Go in and tell them you’re a CS356 student and need to print a poster

**Basic Poster Requirements:**
- Do something interesting to the security community
- Present it in the form of a poster
- Need to provide references (on the front of the front of the poster, or you can turn in a separate printed piece of paper)
  - No specific reference format required
  - Need more than just the link
- Lots of wiggle room

**Presentations:** Thursday CS Building 3rd floor
- As long as you have one group member there the whole time you can come and go (for class/other presentations)

**SSL Handshake Protocol**
- Many different variants
  - Has been designed to work with many different entities
- Very flexible
  - Allows client/server pairs with each using a different language
- Most common
  - Anonymous client paired with authenticated server
- **Four Phases:**
  - **Phase 1:** Establish Security Capabilities
    - Establish security capabilities including protocol version, session ID, cipher suite, compression method, and initial random numbers
  - **Phase 2:** Server Authentication and Key Exchange
    - Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase
  - **Phase 3:** Client Authentication and Key Exchange
    - Client sends certificate if requested. Client sends key exchange client may send certificate verification
  - **Phase 4 (optional):** Finish
    - Change cipher suite and finish handshake protocol
    - Generally occurs when there is a small break in the connection
      - One of two things will happen
        - 1) Recognize that the established parameters are still valid, phase 4 not required in this case
2) Established parameters are no longer valid, this is when phase 4 becomes necessary

- Allows server and client to:
  - Authenticate each other
  - Negotiate encryption & MAC algorithms
  - Negotiate cryptographic keys to be used

**SSL Handshake Protocol - Basic: No Client Authentication**

- **Challenge-data:**
  - 4 byte timestamp, 28 byte random value
  - Used to authenticate server later on; also prevent replay attacks
- **Cipher-sec:**
  - Supported cipher and compression methods
- **Clear-master-key:**
  - Not encrypted. Anybody can eavesdrop on this key
- **\{Secret-master-key\}_{server-public-key}**
  - Secret key encrypted with server's public key
- **Client-master-key:**
  - Used to generate client-write-key, server-write-key, client-write-MAC-secret (used to generate MACS that client sends to the server) and server-write-MAC-secret (used to generate MACS that server sends to the client)

**Change Cipher Spec Protocol**

- Changes specs if too long of a wait occurs

**Diffie-Hellman Key Exchange**

- Can be used to share a secret
- First known example of public key-cryptographic technique
  - Only known public key technique that can be used to generate a secret
- Alpha and p are both known
- Alice chooses a secret random integer value Xa
- Bob chooses a secret random integer value Xb
- Alice computes yA (in polynomial time)
  - Extremely hard for Charlie compute Xa from yA
- Bob computes yB (in polynomial time)
  - Extremely hard for Charlie to compute Xb from yB
- Share yA and yB with each other
- Compute Kab
  - Both will have this same value - use this to share a secret

**Diffie-Hellman Key Exchange - Man in the middle attack**

- Alice tries to share a secret with Bob but does not know that Charlie is in the middle capturing her messages
She thinks she is sending value to Bob, but Charlie sends a different value to Bob

- yCA and yCB are the same value
- Instead of setting up a secure key with each other, Alice and Bob have each set up a secure key with Charlie (unknowingly)
- Works because Alice and Bob do not know if they are actually communicating with each other or not
  - **SOLUTION:** set up some kind of authentication system between Alice and Bob
    - Can be done using public key certificates
    - Digital Signatures can be incorporated
      - **Internet Authentication Protocols**

**Mutual Authentication**

- Both parties need to know that they are communicating with each other
  - Alice knows she communicating with Bob and Bob knows he’s communicating with Alice
- Problem: *how to share the key?*
  - **SOLUTION:** Mediated Authentication
    - **Secret Key Based Protocols:**
      - Needham - Schroeder protocols & kerberos
      - Trusted third party generates and distributes shared secret key
    - **Public Key Based:**
      - X.509 Directory authentication service
      - Trusted third party public key

**Needham-Schroeder Protocol**

- Bob and Alice trust a 3rd party
- Objective: set up mutual communication between Bob and Alice
- In the example below, S is the server, A is Alice, B is Bob, \{K_{XY}\} is a message encrypted with K_{XY}, N_A is a non-repeating value sent by A, N_B is one sent by B, K_AS is the secret key shared between A and S, and K_BS is the key shared between S and B.
  - Step 1: A sends S N_A and a request to open a communication channel to B
  - Step 2: S sends \{N_A, B, K_{AB}, \{K_{AB}, A\}K_BS\}K_AS to A
    - Verification not necessary, because if A can open the message, it was sent from the server and correctly received by her
  - Step 3: A decrypts and forwards \{K_{AB}, A\}K_BS to B
  - Step 4: B decrypts, sends \{N_B\}K_{AB} to A
  - Step 5: A sends \{(N_B)-1\}K_{AB} to B
    - This allows B to ensure that this is actually A

**Kerberos - Improved Needham-Schroeder Protocol**

- Trusted key server system
• Provides centralized private-key third-party authentication in a distributed network
  o Allows users access to services distributed through network without needing to
    trust all workstations
  o Rather, all trust a central authentication server
• Provides Single Sign-On
• Third-party authentication scheme involving:
  o Authentication Server (KDC/AS)
    ■ Users initially negotiate with AS to identify itself
    ■ AS provides a non-corruptible authentication credential (ticket granting
      ticket TGT)
  o Ticket Granting Server (TGS)
    ■ Users subsequently request access to other services from TGS using the
      TGT

**Kerberos Inter Realm Authentication**
• Kerberos “Realm” consists of:
  o A Kerberos server including TGS
  o A number of clients, all registered with server
  o Application servers, sharing keys with server
  o Typically a single administrative domain
• If multiple realms, their Kerberos servers must share keys and trust

**Questions:**
• Is is difficult to update SSL protocol?
  o They are backward compatible but does not necessarily mean that one specific
    implementation will also be backward compatible
• Can you completely trust an SSL certificate?
  o It’s not easy, but the SSL certificates can be spoofed
• Is there a standard SSL implementation?
  o No, only guidelines. The implementation of SSL depends mostly on the
    implementing programmer’s skill
• How should we format citations on our posters?
  o For websites, you can just give a link. For papers, you need to use a valid citation
    style
• What if our poster is mostly interactive and not as visual?
  o The TAs and I will be walking around to talk with you about your projects
• Has Diffie-Hellman have vulnerabilities?
○ If there’s no one in the middle intercepting your exchange of x_a and x_b, the vulnerabilities come from the way it’s implemented

● In the Needham-Schroeder protocol, how did Alice decrypt step 3?
  ○ Key K_AS is shared between Alice and the server. She decrypts the outer envelop.
  ○ The inner envelope is still locked with key K_BS, that’s what she sends to Bob in step 4

● Will there be quizzes over ch. 22 & 23? Will they be on the final?
  ○ No quizzes, but the material will be on the final