Authentication

IT443 – Network Security Administration
Instructor: Bo Sheng

Outline

• Authentication Mechanisms
• Key Distribution Center and Certificate Authorities
• Session Key

Authentication

• Authentication is the process of reliably verifying certain information.
• Examples
  – User authentication
    • Allow a user to prove his/her identity to another entity (e.g., a system, a device).
  – Message authentication
    • Verify that a message has not been altered without proper authorization.

Authentication Mechanisms

• Address-based authentication
  – Assume the identity of the source can be inferred based on the network address from which packets arrive.
  – Adopted early in UNIX and VMS
• Berkeley rtools (rsh, rlogin, etc)
  – /etc/hosts.equiv file: List of computers
  – Per user .rhosts file: List of <computer, account>
  – Ubuntu: /etc/hosts.allow, /etc/hosts.deny, man hosts_access

• Threat
  – Breaking into an account on one machine leads to breaking into other machines accounts
  – Spoof of network address

Authentication Mechanisms

• Password-based authentication
  – Use a secret quantity (the password) that the prover states to prove he/she knows it.
  – Threat
    • Eavesdropping
    • Password guessing/dictionary attack

Alice: I'm Alice, the password is fiddlesticks

Authentication Mechanisms

• Cryptographic authentication protocols
  – Basic idea:
    • A prover proves some information by performing a cryptographic operation on a quantity that the verifier supplies.
    • Usually reduced to the knowledge of a secret value
      • A symmetric key
      • The private key of a public/private key pair
Passwords as Crypto Keys

- Symmetric key systems:
  - Hash the password to derive a 64/128 bits key

- Public key systems:
  - Difficult to generate an RSA private key from a password
  - Usual solution:
    - Encrypt the private key with the users password and store the encrypted result (e.g., using a directory service)

Eavesdropping & Server Database Reading

- If public key crypto is not available, protection against both eavesdropping and server database reading is difficult:
  - Hash => subject to eavesdropping
  - Challenge requires Bob to store Alice’s secret in a database

*I’m Alice, \( H(K_{Alice-Bob}) \) \* \*Bob

Alice

\[ H(K_{Alice-Bob}, R) \]

Bob

Eavesdropping & Server Database Reading

- Example of basic authentication using public keys:
  - Bob challenges Alice to decrypt a message with her public key

\begin{align*}
\text{Alice} & \quad \cdots \quad \text{Bob} \\
\text{I’m Alice} & \quad \cdots \quad \text{R} \\
\text{Sig}_{K_{Alice-Bob}}(R) & \quad \cdots \quad \text{Bob}
\end{align*}

Outline

- Authentication Mechanisms
  - Key Distribution Center and Certificate Authorities
  - Session Key

Key Distribution Center

- New nodes are configured with a key to the KDC
  - e.g., \( K_A \) for node \( A \)
- If node \( A \) wants to communicate with node \( B \)
  - \( A \) sends a request to the KDC
  - The KDC securely sends to \( A \):
    \( E_{K_{AB}}(R_{AB}) \) and \( E_{K_{BA}}(R_{AB}, A) \)

Advantages:
- Single location for updates, single key to be remembered

Drawbacks:
- If the KDC is compromised!
- Single point of failure/performance bottleneck => multiple KDC?

Certification Authorities

- How do you know the public key of a node?
- Typical solution:
  - Use a trusted node as a certification authority (CA)
    - E.g., VeriSign, GoDaddy
  - Everybody needs to know the CA public key
  - The CA generates certificates: Signed(A, public key, validity information)
  - Certificates can be stored in a directory service or exchanged during the authentication process

Advantages (over KDC):
- The CA doesn’t have to be online => more physical protection
- Not a performance bottleneck, not a single point of failure
- Certificates are not security sensitive: only threat is DoS
- A compromised CA cannot decrypt conversation but can lead to impersonation
Certificate Revocation

- What if:
  - Employer left/fired
  - Private key is compromised

- Solution: similar to credit cards
  - Validity time interval (‘not before’, ‘not after’)
  - Use a Certificate Revocation List (CRL): X.509
    - E.g., lists all revoked and unexpired certificates

Lab 1

Question 1
Type in the following command and answer the questions $tracert -d yahoo.com
- How many hops is your machine away from yahoo.com? (Attach the output in the lab report)
- Execute the same command again. Is the output the same as the first time? (Hint: no) Which hops are changed? Observe and compare the difference, and explain the reason.

Lab 1

Question 2
- Use your CS account to log in linux1.cs.umb.edu and compare the following two commands in the shell and explain the difference between the outputs.
- What is the IP address of cs.umb.edu? Assume CS department’s network uses a 23-bit IP prefix, how many IP addresses the department can support?
  158.121.104.2: 158.121.0.104, 158.121.0.0, 0.00000010
- Write a script to find all the IP addresses assigned in CS department that have globally recognized domain names. (Consider the same assumption of 23-bit prefix)
  158.121.104.0-158.121.104.255, 158.121.105.0-158.121.105.255

Lab 1

Question 3
- The encryption algorithm aes-128-ecb is a 128-bit block cipher. Design an experiment to verify it.
- How much information can you recover? Please explain why.

Electronic Code Book (ECB)

Cipher Block Chaining (CBC)
Outline

- Authentication Mechanisms
- Key Distribution Center and Certificate Authorities
- Session Key

Session Key Establishment

- Authentication is not everything
  - What could happen after authentication?
    - E.g., connection hijacking, message modification, replay, etc.
  - Solution use crypto → need a share key between communicating entities because public encryption/decryption is expensive
  - Practically authentication leads to the establishment of a shared key for the session
    - A new key for each session:
      - The more data an attacker has on a key the easier to break
      - Replay between sessions
      - Give a relatively "untrusted" software the session key but not the long-term key
      - Good authentication protocol can establish session keys that provide forward secrecy

Password-Based User Authentication

- User demonstrates knowledge of a secret value to authenticate
  - most common method of user authentication

  ![Password-Based User Authentication Diagram](image)

- Threats to password-based authentication?

Some Issues for Password Systems

- A password should be easy to remember but hard to guess
  - that’s difficult to achieve!
- Some questions
  - what makes a good password?
  - where is the password stored, and in what form?
  - how is knowledge of the password verified?

Password Guessing

- Online
  - Try passwords until accepted
    - Limit number of trials and lock account: e.g., ATM machine
      - DoS problem: lock all accounts
    - Increase minimum time between trials
    - Prevent automated trials: Turing tests
    - Long passwords: pass phrases, initials of sentences, reject easy passwords
    - What is the protection used by Yahoo? Hotmail? Gmail? Facebook? Your banks?

- Offline
  - Attacker captures \( X = f(password) \)
    - Dictionary attack: try to guess the password value offline
    - The secret space should be large
    - Strong password
Password Length

- Online attacks:
  - Can 4/6 digits be sufficient if a user is given only three trials?

- Offline attacks:
  - Need at least: 64 random bits = 20 digits
  - Or 11 characters from a-z, A-Z, 0-9, and punctuation marks
  - Too long to remember by a human
  - Or 16 characters pronounceable password (a vowel every two characters)
  - Conclusion:
    A secret a person is willing to remember and type will not be as good
    as a 64-bit random number

Attacks on Passwords

- Suppose passwords could be up to 9 characters long

- This would produce $10^{18}$ possible passwords; 320,000 years to try them all at 10 million a second!

- Unfortunately, not all passwords are equally likely to be used

Example

- In a sample of over 3000 passwords:
  - 500 were easily guessed versions of dictionary words or first name / last name
  - 86% of passwords were easily guessed

<table>
<thead>
<tr>
<th>Length in characters</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of passwords</td>
<td>15</td>
<td>72</td>
<td>464</td>
<td>477</td>
<td>706</td>
<td>605 (lower case only)</td>
</tr>
</tbody>
</table>

Dictionary Attacks

- Attack 1 (online):
  - Create a dictionary of common words and names and their simple transformations
  - Use these to guess the password

- Attack 2 (offline):
  - Usually $F$ is public and so is the password file
    - In Unix, $F$ is crypt, and the password file is /etc/passwd.
    - Compute $F$ (word) for each word in the dictionary
    - A match gives the password
Dictionary Attacks

• Attack 3 (offline):
  – To speed up search, pre-compute $F$ (dictionary)
  – A simple look up gives the password

```
Eagle
Wine
Rose
...
Pre-computed
Dictionary

Password file

Password: YkPT

KED!

Look up

Password file
```

Password Guidelines

1. Initial passwords are system-generated, have to be changed by user on first login
2. User must change passwords periodically
3. Passwords vulnerable to a dictionary attack are rejected
4. User should not use same password on multiple sites
5. etc. etc.

Authentication with Shared Secret

```
Alice
I'm Alice
A challenge $R$
$J_K_{Alice-Bob}(R)$

Bob
```

• Weaknesses
  – Authentication is not mutual; Trudy can convince Alice that she is Bob
  – Trudy can hijack the conversation after the initial exchange
  – If the shared key is derived from a password, Trudy can mount an off-line password guessing attack
  – Trudy may compromise Bob's database and later impersonate Alice

Password Salt

• To make the dictionary attack a bit more difficult
• Salt is a $n$-bit number between 0 and $2^n$
• Derived from, for example, the system clock and the process identifier

```
Password + Salt

H

Username, Salt, $H(Password + Salt)$
```

Outline

• Login authentication protocol
• Mutual authentication protocol

Authentication with Shared Secret

```
Alice
I'm Alice
$K_{Alice-Bob}(R)$

Bob
```

• A variation
  – Requires reversible cryptography
  – $R$ must be limited lifetime
• Weaknesses
  – All the previous weaknesses remain
  – Trudy doesn't have to see $R$ to mount off-line password guessing
    if $R$ has certain patterns (e.g., the timestamp)
  • Trudy sends a message to Bob, pretending to be Alice
Authentication with Public Key

- Bob’s database is less risky
- Weaknesses
  - Trudy can trick Alice into signing something
    - Use different private key for authentication

Outline

- Login authentication protocol
- Mutual authentication protocol

Mutual Authentication

- Reflection Attack
  - Lesson: Don’t have Alice and Bob do exactly the same thing
    - Different keys
      - Totally different keys
        - $K_{Alice-Bob} = K_{Bob-Alice} + 1$
    - Different Challenges
      - The initiator should be the first to prove its identity
        - Assumption: initiator is more likely to be the bad guy

Reflection Attack

Countermeasure
Mutual Authentication

- Public keys
  - Authentication of public keys is a critical issue

![Diagram of mutual authentication with public keys]

Mutual Authentication

- Mutual authentication with timestamps
  - Require synchronized clocks
  - Alice and Bob have to encrypt different timestamps

![Diagram of mutual authentication with timestamps]

Lab 2

- Certificate Authorities

![Diagram of certificate authorities]