CSE 406: Computer Security Sessional

Department of CSE

Bangladesh University of Engineering and Technology

July 2025

Team Members

Name: **Tamim Hasan Saad**

Student ID: **2005095**

Name: **Habiba Rafique**

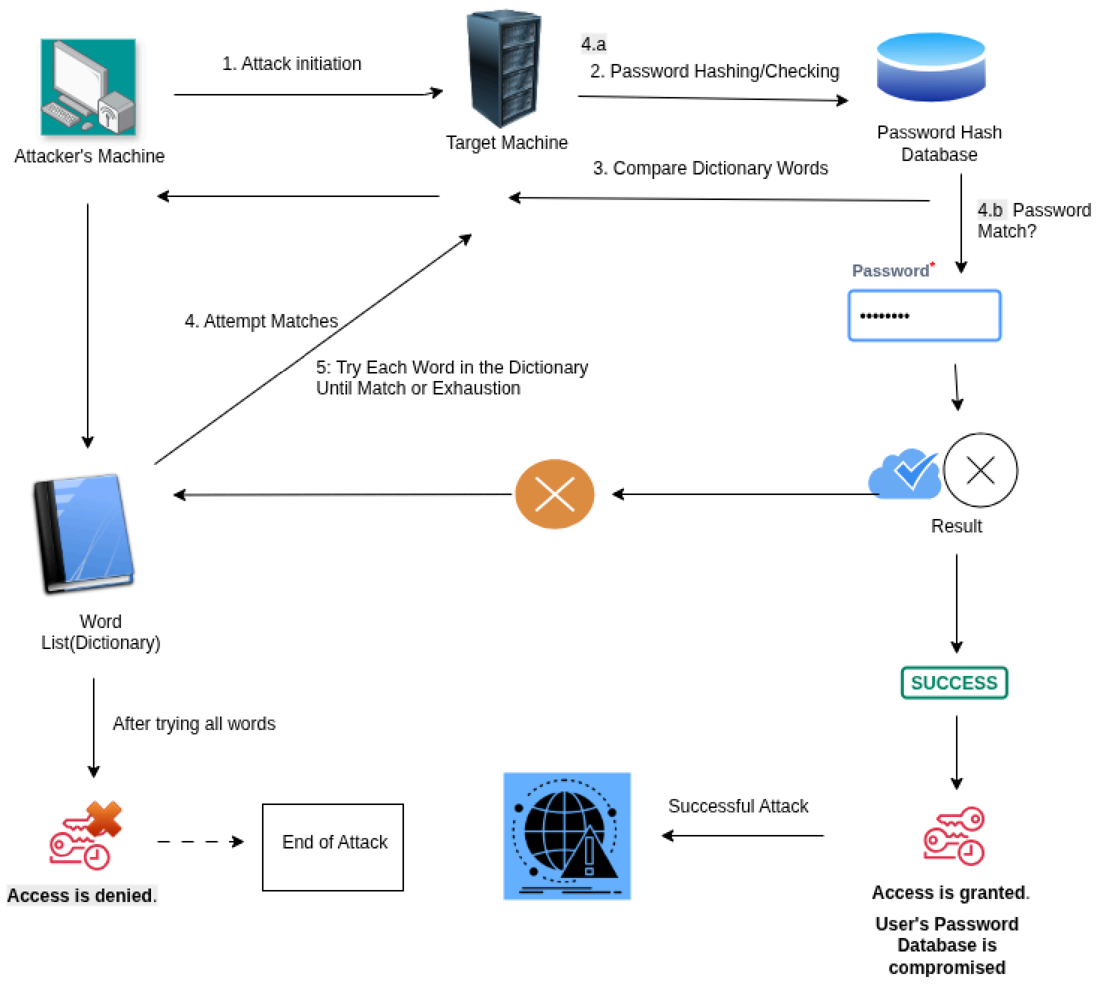
Student ID: **2005096**

Dictionary Attack

Definition:

A Dictionary Attack is an attack vector used by the attacker to break in a system, which is password protected, by putting technically every word in a dictionary as a form of password for that system.
The dictionary can contain words from an English dictionary and also some leaked list of commonly used passwords and when combined with common character replacing with numbers, can sometimes be very effective and fast.
Type:Offline or Online brute-force strategy
Example:Typing Password123,admin2025,etc from a wordlist.

Topology:



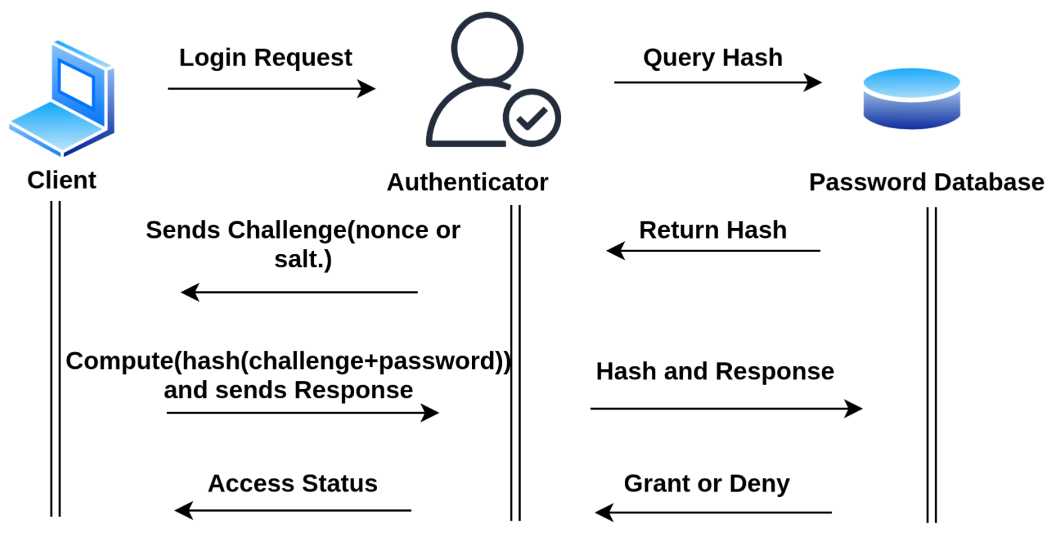
Timing Diagram: Original Authentication Protocol

Entities Involved:

- **Client:** The user attempting to authenticate.
- **Authenticator:** The server or system verifying the user's credentials.
- **Password Database:** Stores the hashed passwords for verification**.**

Sequence of Events:

1. **Client → Authenticator:** The client sends a login request, typically including a username.
2. **Authenticator → Password Database:** The authenticator queries the password database to retrieve the stored hash for the provided username.
3. **Password Database → Authenticator:** The password database returns the stored hash.
4. **Authenticator → Client:** The authenticator sends a challenge to the client, often a nonce or salt.
5. **Client → Authenticator:** The client computes a response (e.g., HMAC) using the challenge and their password, then sends it back to the authenticator.
6. **Authenticator → Password Database:** The authenticator hashes the received response and compares it with the stored hash.
7. **Authenticator → Client:** If the hashes match, the authenticator sends an "access granted" message; otherwise, it sends "access denied."

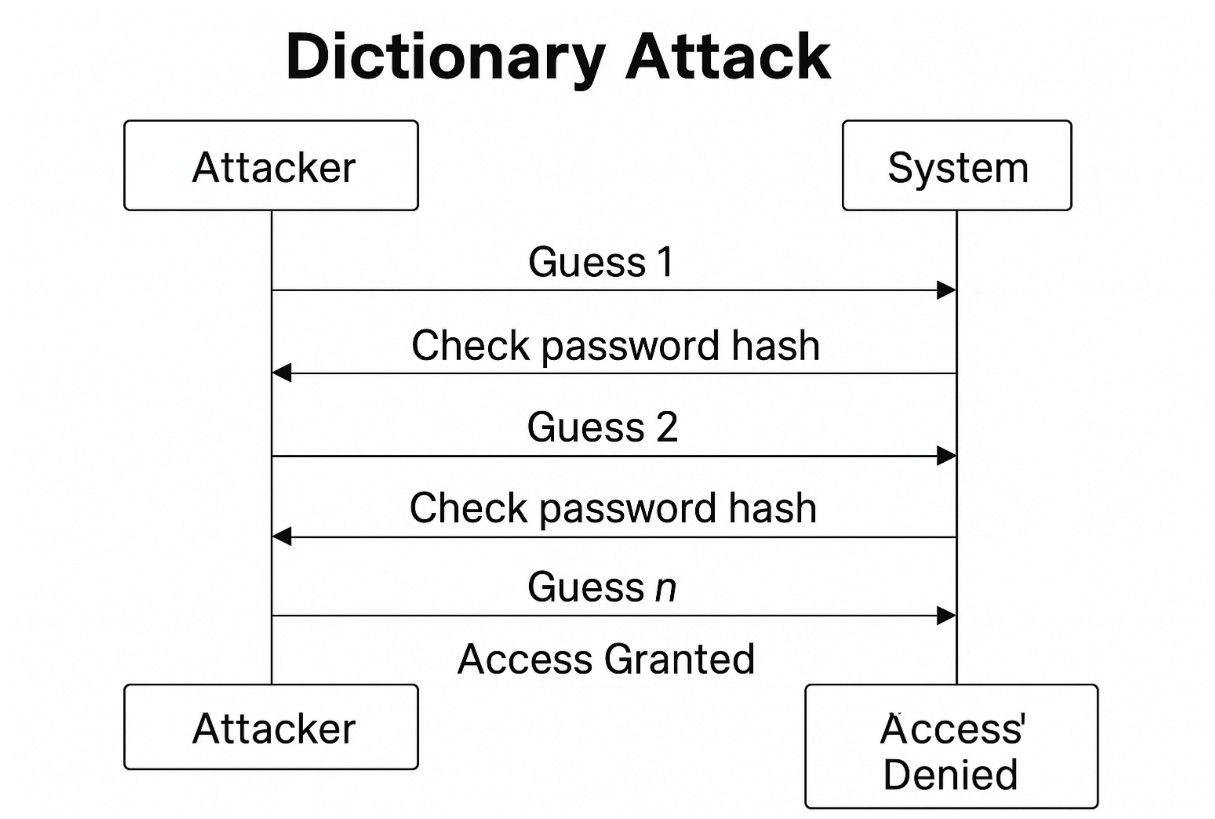


Timing Diagram:Original Protocol

Timing Diagram : Dictionary Attack

Entities Involved:

1. **Attacker:** The entity attempting to guess the password.
2. **Target System:** The system under attack, which compares the provided password with the stored password hash.
3. **Password Hash Database:** Stores the password hash used for comparison with the attacker's guesses.



Sequence of Events:

1. **Attacker → Target System:**

   Time Step: `t1`

   The attacker sends a login request with a guessed password.

2. **Target System → Password Hash Database:**

   Time Step: `t2`

   The target system retrieves the stored password hash for the username.

3. **Password Hash Database → Target System**:

   Time Step: `t3`

   The password hash database returns the stored hash for comparison.

4. **Target System → Attacker:**

Time Step: `t4`

The target system hashes the received guess and compares it with the stored hash.

5. **Target System → Attacker:**

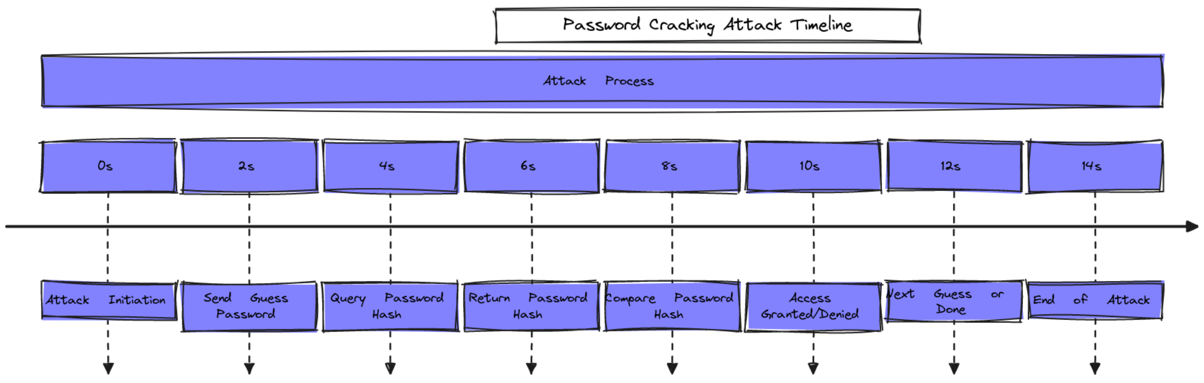
   Time Step: `t5`

   If the hashes match, access is granted, and the target system sends an "Access Granted" message. If there's no match, it sends "Access Denied."

6. **Attacker → Target System:**

   Time Step: `t6`

   The attacker continues to the next guess and repeats the process.



Packet / Frame Details for the Attack and Header Modifications

In this section, we detail the structure of the login request packets used in the dictionary attack, along with modifications made to various protocol layers. These alterations are designed to evade detection, bypass rate-limiting mechanisms, and ensure the attack remains scalable and persistent. The attack leverages standard HTTP/HTTPS protocols to submit credential guesses, with automated tools iterating through a dictionary wordlist.

Login Request Packet Structure

The core of each attack packet is a standard HTTP POST request to the target's login endpoint. Below is an example structure:

```
POST /login HTTP/1.1
Host: target-server.com
Content-Type: application/x-www-form-urlencoded
Content-Length: [length]
User-Agent: [automated tool identifier]

username=[target_username]&password=[dictionary_word]
```

This format mimics legitimate login attempts, ensuring protocol compliance while varying only the password field with each iteration.

Packet/Frame Header Modifications

To enhance stealth and efficiency, modifications are applied across network layers. These changes help distribute the attack, avoid throttling, and blend with normal traffic. The following list outlines key modifications:

- **Layer: Ethernet**
  Field (Original): Src MAC
  Value at Attack Time: Attacker NIC (rotates across VMs)
  Purpose: Evade simple MAC filtering by cycling through virtual machine interfaces.

- **Layer: IP**
  Field (Original): Src IP
  Value at Attack Time: 1. True IP; 2. Cloud/relay pool
  Purpose: Bypass per-IP throttles by distributing requests across multiple sources, such as cloud proxies.

- **Layer: TCP**
  Field (Original): Src Port
  Value at Attack Time: Ephemeral (auto-increment)
  Purpose: Enable parallel sockets for concurrent requests without port exhaustion.

- **Layer: HTTP/S**
  Field (Original): User-Agent
  Value at Attack Time: Legitimate browser strings (rotated)
  Purpose: Blend with normal traffic by mimicking common web browsers.

- **Layer: HTTP/S**
  Field (Original): Method
  Value at Attack Time: POST /login
  Purpose: Standard credential submission to maintain protocol validity.

- **Layer: Body**
  Field (Original): username
  Value at Attack Time: Target account
  Purpose: Constant value to focus on a specific user.

- **Layer: Body**
  Field (Original): password
  Value at Attack Time: Next word-list entry
  Purpose: Changes every request to test dictionary entries sequentially.

- **Layer: Header**
  Field (Original): Cookie
  Value at Attack Time: Re-uses valid session cookie
  Purpose: Avoid new-session overhead by maintaining state across attempts.

- **Layer: Header**
  Field (Original): X-Forwarded-For
  Value at Attack Time: Spoofed /24 block
  Purpose: Defeat naïve rate limits by simulating requests from a subnet range.

- **Layer: Timing**
  Field (Original): Request Interval
  Value at Attack Time: 10-30 ms between packets
  Purpose: Keep just below lockout threshold to sustain the attack without triggering defenses.

Frame Modifications

Additional frame-level adjustments are implemented to further obscure the attack:

- **Modification Type: Automated Tool Signatures**
  Description: Headers modified to appear as legitimate browser requests.
  Impact: Evades detection by security tools that flag anomalous patterns.

- **Modification Type: Session Management**
  Description: Maintains session state across multiple attempts.
  Impact: Reduces server overhead and suspicion by simulating persistent user behavior.

- **Modification Type: Rate Limiting Evasion**
  Description: Distributed across multiple IP addresses and timing variations.
  Impact: Bypasses automated blocking mechanisms like IP-based throttling.

- **Modification Type: Payload Variation**
  Description: Only password field changes while maintaining protocol compliance.
  Impact: Appears as normal failed login attempts, reducing the likelihood of alerting

monitoring systems.

Key Packet Modifications

- **Rapid Sequential Requests**: Automated tools send thousands of login attempts with minimal delays between requests, maximizing throughput.
- **Password Field Variation**: Each packet contains a different password from the dictionary wordlist, enabling systematic guessing.
- **Session Handling**: May include session tokens or cookies to maintain connection state, avoiding repeated session initialization.
- **Rate Limiting Bypass**: Potentially distributed across multiple IP addresses or use of proxy chains to evade per-source restrictions.
- **HTTP Headers**: Standard headers maintained to appear as legitimate login attempts, enhancing stealth.

Response Analysis

Server responses are monitored to detect success or adjust the attack:

- **Success Response**: HTTP 200 with authentication tokens/redirects, indicating a valid password match.
- **Failure Response**: HTTP 401/403 with error messages, prompting the next guess.
- **Rate Limiting**: HTTP 429 or connection timeouts, signaling the need for IP rotation or slower pacing.

These modifications ensure the attack is both effective and difficult to detect in a real-world scenario.

Justification: Why This Design Should Work

This dictionary attack design exploits inherent weaknesses in password-based authentication systems, particularly those lacking robust rate-limiting or multi-factor protections. By automating high-volume guesses while maintaining protocol compliance, the approach achieves a high probability of success against weak credentials. Below, we justify its effectiveness through key attack and technical advantages.

Attack Effectiveness

1. **Server Vulnerability Exploitation**: Most authentication systems validate each password attempt immediately without implementing proper throttling mechanisms, allowing rapid iteration through guesses.

2. **Automated Scale**: Dictionary attacks can test thousands of common passwords in minutes, far exceeding human capability and overwhelming basic defenses.

3. **High Success Probability**: Common passwords and leaked password databases significantly increase success rates, especially against users who reuse credentials.

4. **Resource Asymmetry**: Minimal computational cost for the attacker versus server processing overhead for each validation, enabling sustained operations with low risk.

Technical Advantages

1. **Stateless Nature**: Each attempt is independent, allowing for parallel processing and easy distribution across resources.

2. **Protocol Compliance**: Uses standard HTTP authentication protocols, making detection challenging as traffic appears legitimate.

3. **Scalability**: Can be distributed across multiple attack vectors and IP addresses, adapting to network size and defenses.

4. **Persistence**: Can continue indefinitely until successful or manually stopped, with built-in mechanisms to handle failures like rate limits.

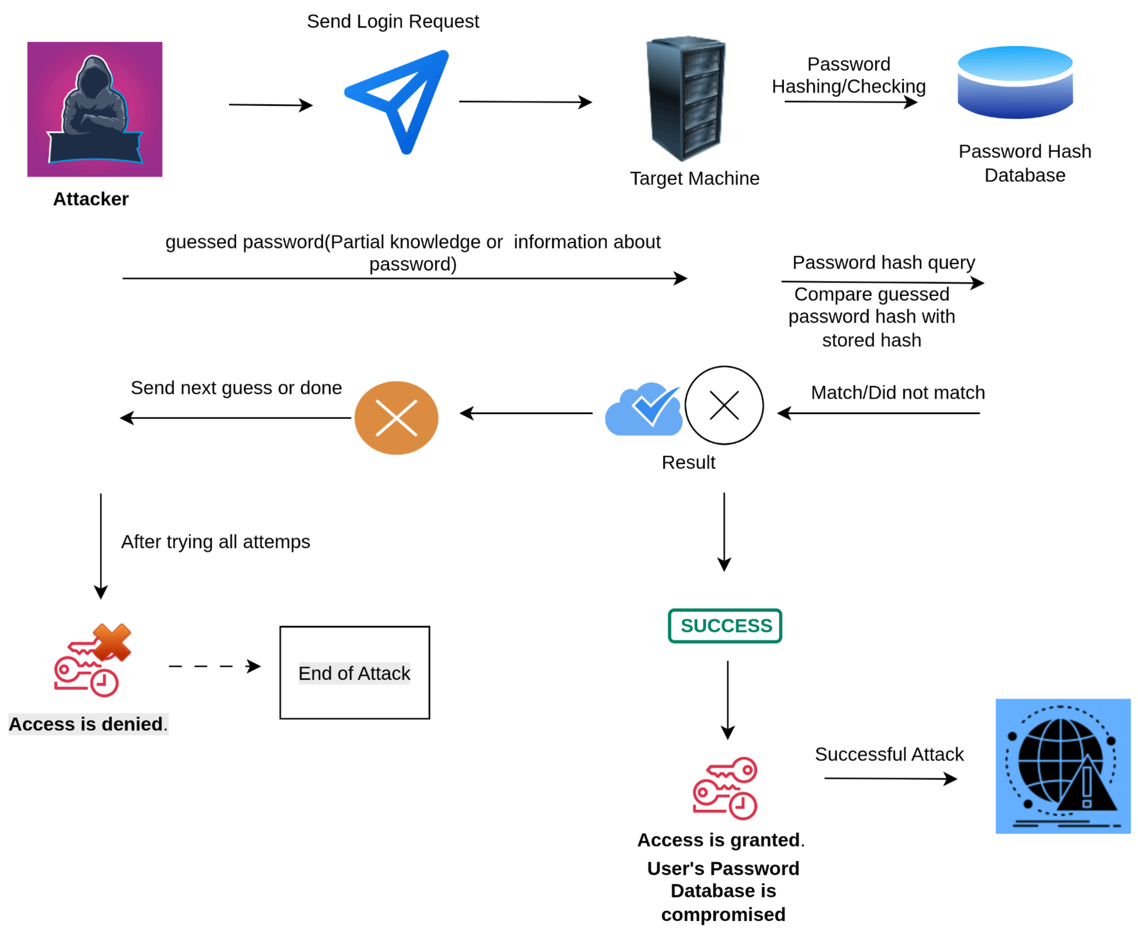
Known Password Attack

Definition:

A Known Password Attack occurs when an attacker has access to information about a password (or part of it) and uses this knowledge to compromise an account or system. The attack typically exploits weak, reused, or predictable passwords.

In a Known Password Attack, the attacker might use personal information (like a person's name, birthday, or something from their social media) to guess the password.

Topology:

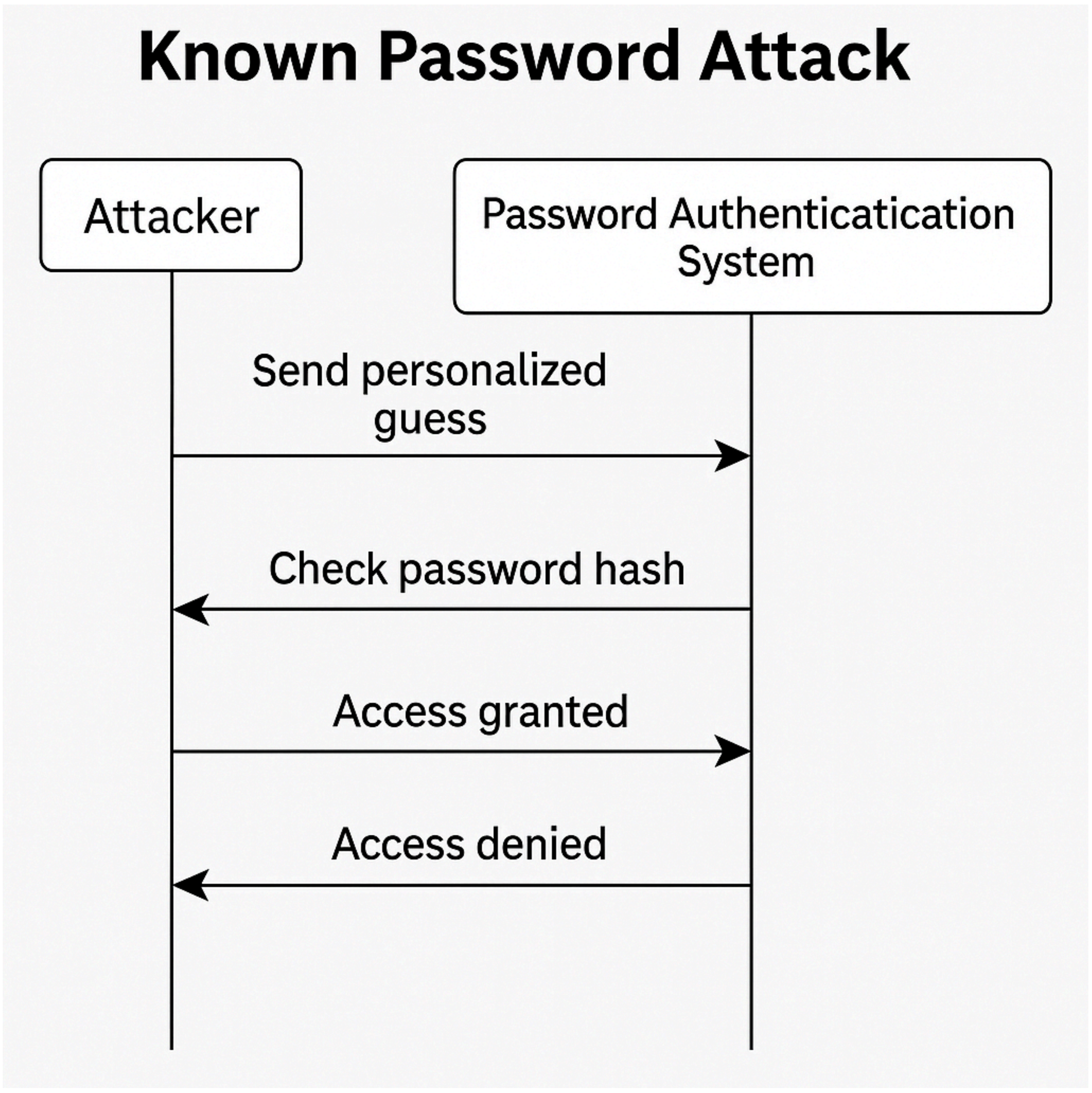


Timing Diagram: Known Password Attack

Entities Involved:

1. **Attacker**:Initiates the attack by sending login requests and password guesses.
2. **Target System**:Receives login requests, queries the password hash database, and compares the guessed password against the stored hash.

3. **Password Hash Database**:Stores hashed passwords used by the target system to verify guesses.



Sequence of Events:

1. **Attacker Sends Login Request**:
    - **Time Step**: t1
    - The attacker initiates the attack by sending a login request to the target system. This request is typically a **username** and a guessed **password** (from known patterns or partial knowledge about the password).

    - The request is sent through the network or system interface to the target system.

2. **Target System Queries Password Hash Database**:
    - **Time Step**: t2
    - Once the system receives the login request, it queries its internal **password hash database** to retrieve the hashed version of the password.
    - The database is responsible for securely storing password hashes, and the system will check if the hashed version of the attempted password matches the stored hash.

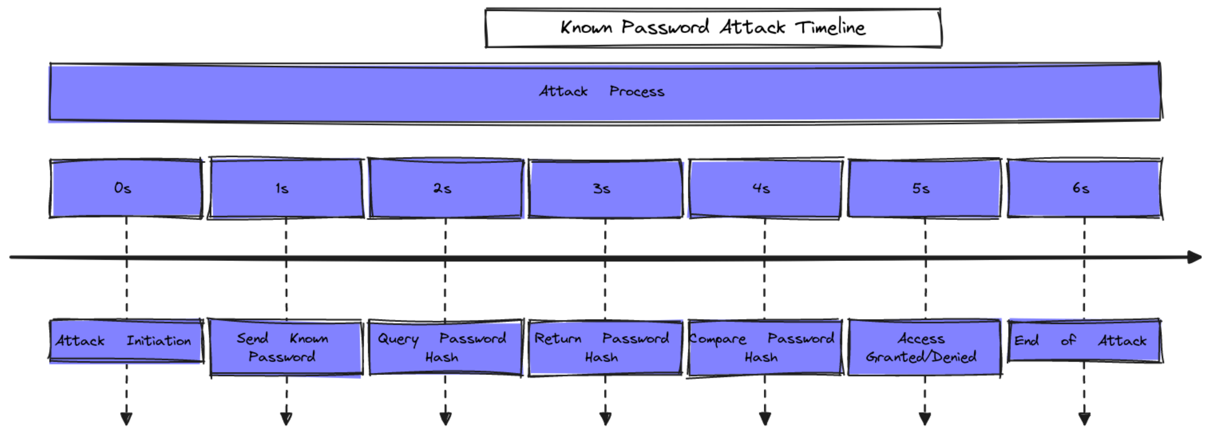
3. **Password Hash Comparison**:
    - **Time Step**: t3
    - The **target system** compares the guessed password hash with the hash retrieved from the **password hash database**.
    - If the hash of the guessed password matches the stored hash, it confirms that the password is correct. If not, it denies access.

4. **Access Granted or Denied**:
    - **Time Step**: t4
    - After the comparison step, the **target system** sends an access response back to the attacker.
        - If the guessed password is correct, the attacker receives an **access granted** response.
        - If the guessed password is incorrect, the system sends an **access denied** response.
    - The response is sent back through the system interface or network.

5. **Next Guess or Stop**:
    - **Time Step**: t5
    - After receiving the access status, the attacker decides whether to continue the attack or stop:

- **Next Guess**: If the password is incorrect, the attacker sends another guess based on further knowledge of the password.
- **Stop**: If the attacker guesses the correct password, they stop the attack.

6. **Repeat Process**:
   - **Time Step**: t6 (and repeats)
   - If the attacker opts to continue, the process repeats from the **"Send Guess Password"** step. Each time the attacker sends a new guess, the system compares it with the stored password hash until the attacker either succeeds or gives up.



Packet / Frame Details for the Attack and Header Modifications

In this section, we outline the structure of the targeted login request packets employed in the known-password attack, along with modifications applied across various protocol layers. These alterations are intended to enhance stealth, mimic legitimate user behavior, and evade detection mechanisms such as anomaly scoring or behavioral alerts. The attack relies on intelligence-driven password guesses derived from open-source intelligence (OSINT), breach data, and pattern recognition, while operating within standard HTTP/HTTPS protocols to submit credential attempts.

Targeted Login Request Packet Structure

The foundation of each attack packet is a standard HTTP POST request directed at the target's login endpoint. Below is an example structure:

```
POST /login HTTP/1.1
Host: target-server.com
Content-Type: application/x-www-form-urlencoded
Content-Length: [length]
User-Agent: [legitimate browser signature]

username=[specific_target]&password=[personalized_guess]
```

This format replicates authentic login attempts, ensuring protocol adherence while varying only the password field with each targeted guess.

Packet/Frame Header Modifications

To improve evasion and realism, modifications are implemented across network layers. These changes facilitate geolocation matching, device fingerprinting, and behavioral mimicry, allowing the attack to blend seamlessly with normal traffic. The following list details key modifications:

- **Layer: Ethernet/IP**
  Field (Original): Src IP
  Value at Attack Time: Residential proxy near victim
  Purpose: Matches usual geo-profile to avoid triggering location-based anomaly detection.

- **Layer: TCP**
  Field (Original): Src Port
  Value at Attack Time: Random
  Purpose: Normal session appearance to simulate typical ephemeral port usage.

- **Layer: HTTP/S**
  Field (Original): Referer
  Value at Attack Time: Legitimate site section
  Purpose: Looks like click-through to enhance the credibility of the request origin.

- **Layer: HTTP/S**
  Field (Original): Method
  Value at Attack Time: POST /login
  Purpose: Standard credential check to maintain protocol validity.

- **Layer: Body**
  Field (Original): username
  Value at Attack Time: Victim e-mail/ID

Purpose: Fixed target identifier to focus on a specific account.

- **Layer: Body**
  Field (Original): password
  Value at Attack Time: Personalized guess (e.g., Fluffy2018!)
  Purpose: Built from OSINT & leaks to test targeted, high-probability credentials.

- **Layer: Header**
  Field (Original): Accept-Language
  Value at Attack Time: Victim's locale
  Purpose: Evades anomaly scoring by aligning with expected user preferences.

- **Layer: Header**
  Field (Original): Time-of-Day
  Value at Attack Time: Sent during victim's usual hours
  Purpose: Avoids behavioral alerts by timing requests to match typical activity patterns.

- **Layer: Header**
  Field (Original): User-Agent
  Value at Attack Time: Victim's known browser/device
  Purpose: Mimics legitimate device to reduce fingerprint-based detection risks.

- **Layer: Timing**
  Field (Original): Request Pattern
  Value at Attack Time: 30-90s gaps, ≤3 attempts/session
  Purpose: Mimics human typos to appear as natural failed login behavior.

Advanced Evasion Techniques

Additional adjustments at the frame level are applied to further conceal the attack:

- **Modification Type: Geolocation Matching**
  Description: Using IP addresses near victim's known locations.
  Impact: Prevents geo-fencing or location-based security triggers.

- **Modification Type: Device Fingerprinting**
  Description: Mimicking victim's known browser and device characteristics.
  Impact: Evades detection systems that profile user agents and headers.

- **Modification Type: Behavioral Timing**
  Description: Aligning attacks with victim's known online activity patterns.
  Impact: Reduces suspicion by simulating routine user sessions.

- **Modification Type: Session Fragmentation**
  Description: Spreading attempts across multiple sessions and time periods.

Impact: Lowers the attack's visibility to rate-limiting or monitoring tools.

Advanced Packet Characteristics

- **Social Engineering Data**: Passwords incorporate personal information (birthdays, names, addresses) to exploit user-specific patterns.
- **Breach Data Integration**: Utilizes previously leaked credentials from other services for informed guessing.
- **Pattern Recognition**: Exploits common password creation patterns (e.g., "Password123", "Company2024") based on observed behaviors.
- **Geographic Targeting**: Incorporates location-specific data (local sports teams, area codes) to refine guesses.
- **Temporal Patterns**: Uses time-based information (current year, seasons, recent events) for contextually relevant attempts.

Header Modifications for Stealth

- **User-Agent Spoofing**: Mimics legitimate browser signatures to blend with expected traffic.
- **Referrer Headers**: Includes believable referrer URLs to simulate natural navigation.
- **Accept Headers**: Standard browser accept headers for content types to maintain authenticity.
- **Connection Timing**: Human-like delays between attempts to avoid detection by automated systems.

Response Analysis

Server responses are observed to refine the attack:

- **Success Response**: HTTP 200 with authentication tokens/redirects, indicating a valid password match.
- **Failure Response**: HTTP 401/403 with error messages, prompting the next targeted guess.

- **Rate Limiting**: HTTP 429 or connection timeouts, signaling the need for session fragmentation or timing adjustments.

These modifications ensure the attack is both targeted and difficult to detect in practical scenarios.

Justification: Why This Design Should Work

This known-password attack design capitalizes on human behavioral patterns and system vulnerabilities in password-based authentication, particularly those lacking robust multi-factor protections. By leveraging personalized guesses while maintaining low-volume, stealthy operations, the approach achieves a high likelihood of success against predictable credentials. Below, we justify its effectiveness through strategic, intelligence-driven, and probabilistic advantages.

Strategic Advantages

1. **Human Psychology Exploitation**: Leverages predictable human password creation behaviors and personal attachment to familiar information, increasing the relevance of guesses.

2. **Reduced Attempt Volume**: Higher success probability per attempt reduces detection likelihood and required attack duration, minimizing exposure.

3. **Personalization Effectiveness**: Custom-tailored password guesses based on target research significantly increase success rates compared to generic methods.

4. **Detection Avoidance**: Lower frequency, targeted attempts appear more like legitimate failed login attempts, evading automated monitoring.

Intelligence-Driven Approach

1. **OSINT Integration**: Open source intelligence provides rich password creation material, enabling informed and context-specific guesses.

2. **Breach Database Correlation**: Previously compromised accounts often reuse similar password patterns, allowing for efficient cross-referencing.

3. **Behavioral Analysis**: Understanding target's online presence and personal interests guides password selection, enhancing accuracy.

4. **Adaptive Strategy**: Can modify approach based on password policy requirements and failed attempt feedback, improving iterative success.

Higher Success Probability

1. **Quality Over Quantity**: Few well-researched attempts often more effective than massive dictionary sweeps, optimizing resource use.

2. **Reduced Defensive Triggers**: Lower volume attacks less likely to trigger automated security responses like lockouts.

3. **Social Context**: Passwords incorporating personal meaning more likely to be actual user choices, exploiting cognitive biases.