Moving Target Defense

Mihail Plesa
Simona David

Table of contents

Introduction

- The question is not if a company will get hacked.
- The question is when a company will get hacked.

- How long can the attackers study the environment?
- How long can the attackers go undetected?

Introduction

- 280 days

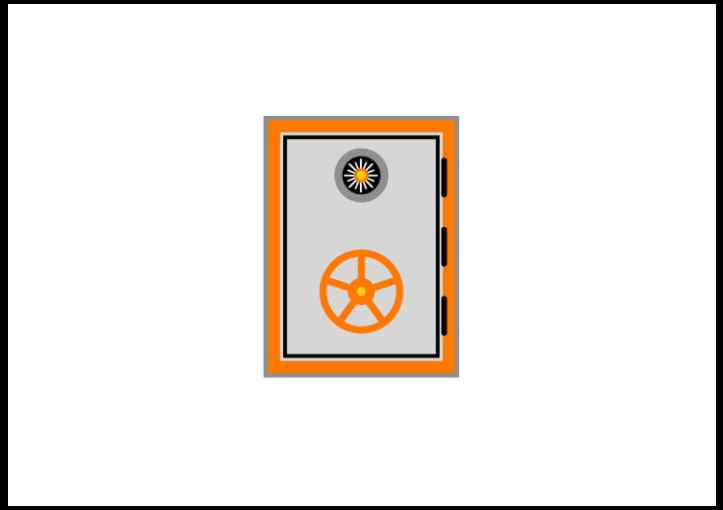
Attacks:

- Targeted
- Un-targeted

Conventional approach

Secure systems by:

- patching all vulnerabilities
- keeping application up-to-date
- staying alert



There is no such thing as perfect security.

Moving Target Defense (MTD)

- 2009 National Cyber Leap Year Summit.

There is no such thing as perfect security, so the focus should be on building alternative defensible systems.

Moving Target Defense (MTD)

Purpose: balance the asymmetry between attackers and targets

How: changing or reducing the attack surface

Attack surface: any property of the system that can be used for an attack

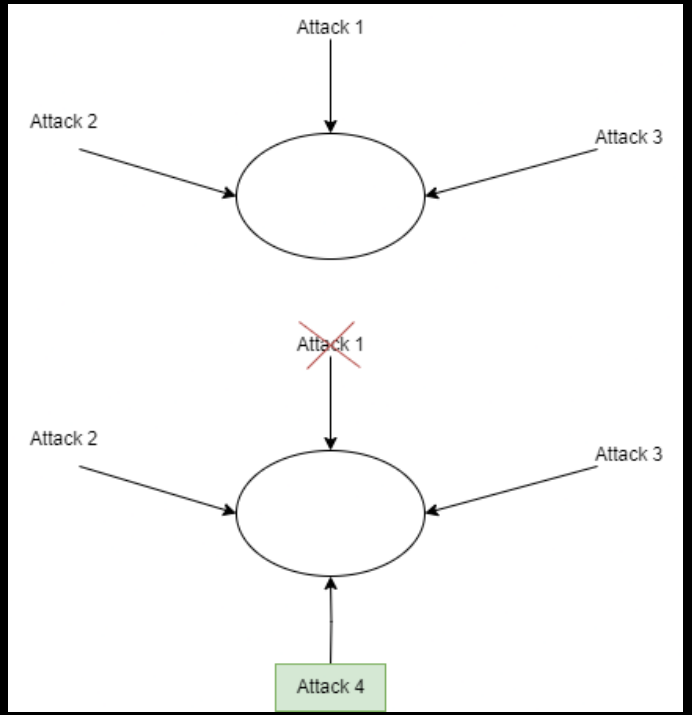
Examples:

- Vulnerabilities
- IP
- Port
- Anything that constitutes a pattern

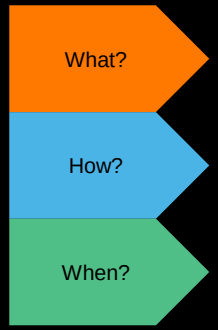
Introduction

Attackers thrive on patterns!

MTD is designed to increase the time and effort an attacker needs to make the attack successful.



Strategy



What parameter to change?

How will it be changed?

When will it be changed (periodically, after a certain event)?

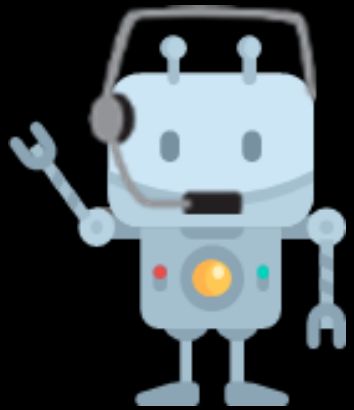
Strategy

If changed periodically, what time interval to use?

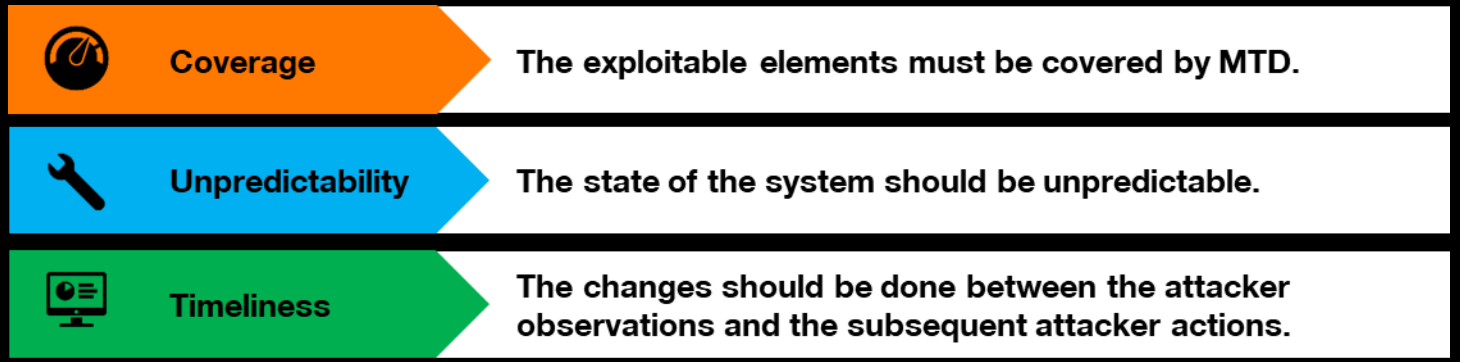
Too often -> impacts system's performance.

Too rare -> it may not help.

If changed after a certain event, what event will be the trigger?
Do we cover events on every part of the attack surface?



Challenges


Coverage
The exploitable elements must be covered by MTD.
Unpredictability
The state of the system should be unpredictable.
Timeliness
The changes should be done between the attacker observations and the subsequent attacker actions.

Examples

- Instruction Set Randomization (ISR)
- NOP insertion
- Address Space Layout Randomization (ASLR)
- IP shuffling
- Honeypot
- Etc

The moving part can be done at these levels:

- Network
- Application
- Software

The shell game

In the “shell game”, an operator places a target such as a pea under one of the three identical face-down shells and shuffles them quickly for many times. When stopped, the player who can correctly identify which shell contains the pea, wins.

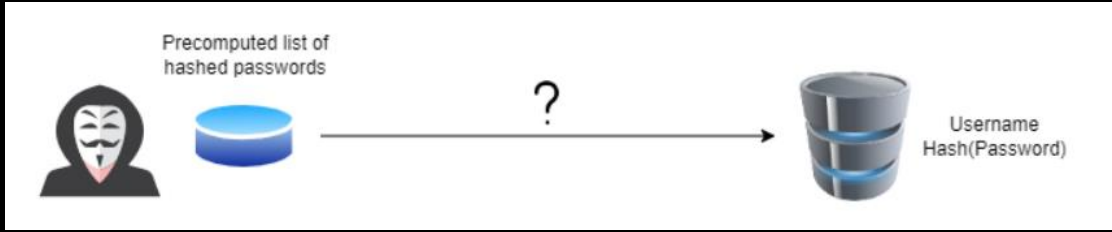
Sufficient movement of the shells will lead to confusion => Moving Target Defense

MTD in cryptography

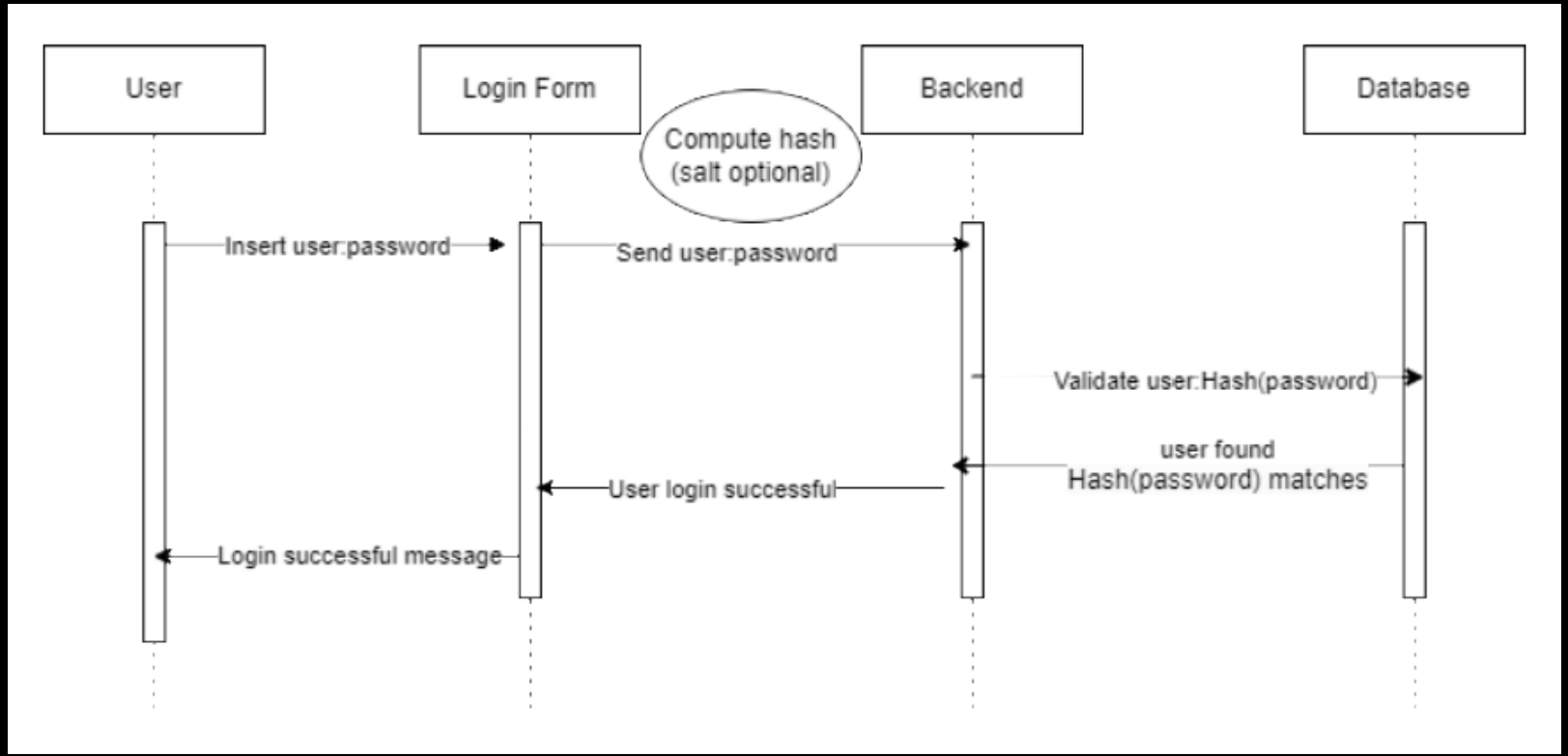
- changing the encryption key periodically
- switching between multiple cryptosystems

MTD in storing secrets

When attackers gain access to a database containing passwords, they will try to find the plaintext passwords associated to the values in the database.



MTD in storing secrets



MTD in storing secrets

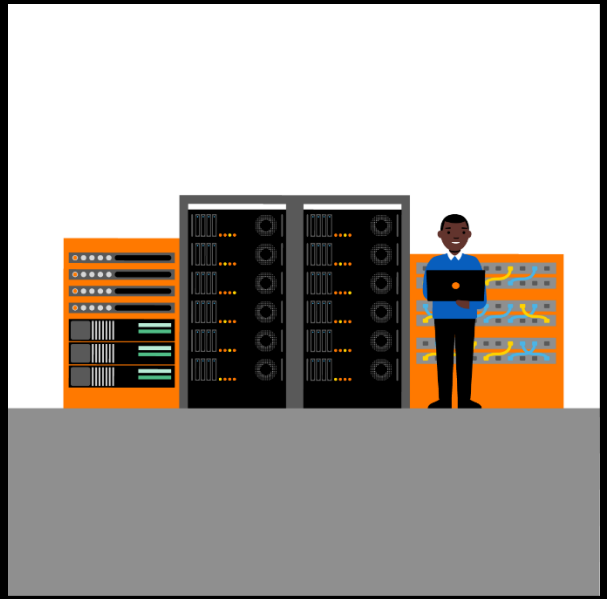
This provides an attacker with a lot of info on the algorithm used

MD5 – 32 chars

SHA1 – 40 chars

SHA256 – 64 chars

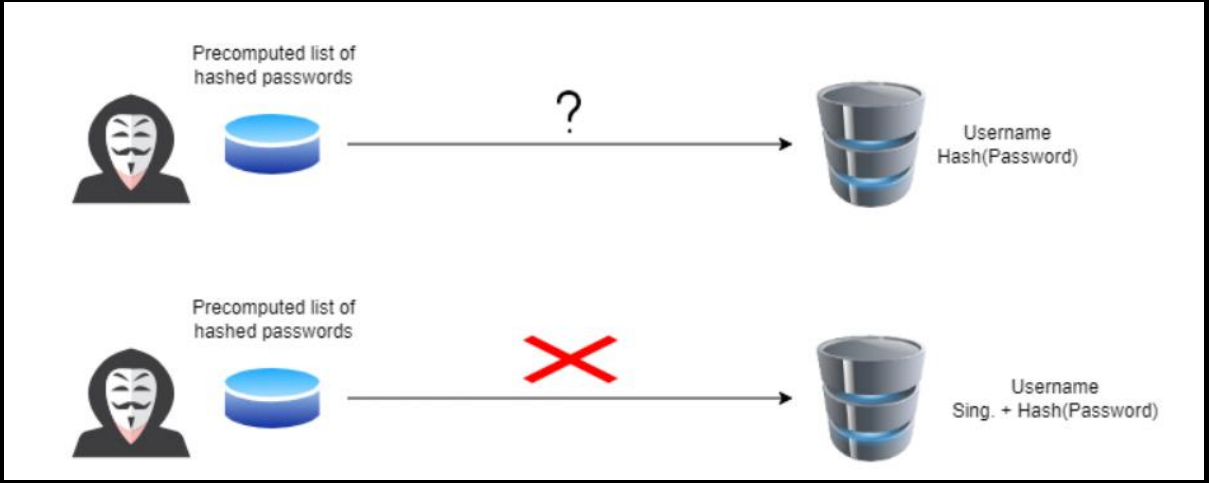
SHA 512 – 128 chars



MTD in storing secrets

Is there any way to make the dictionary attack unusable?

Cost + effort > results



Next step

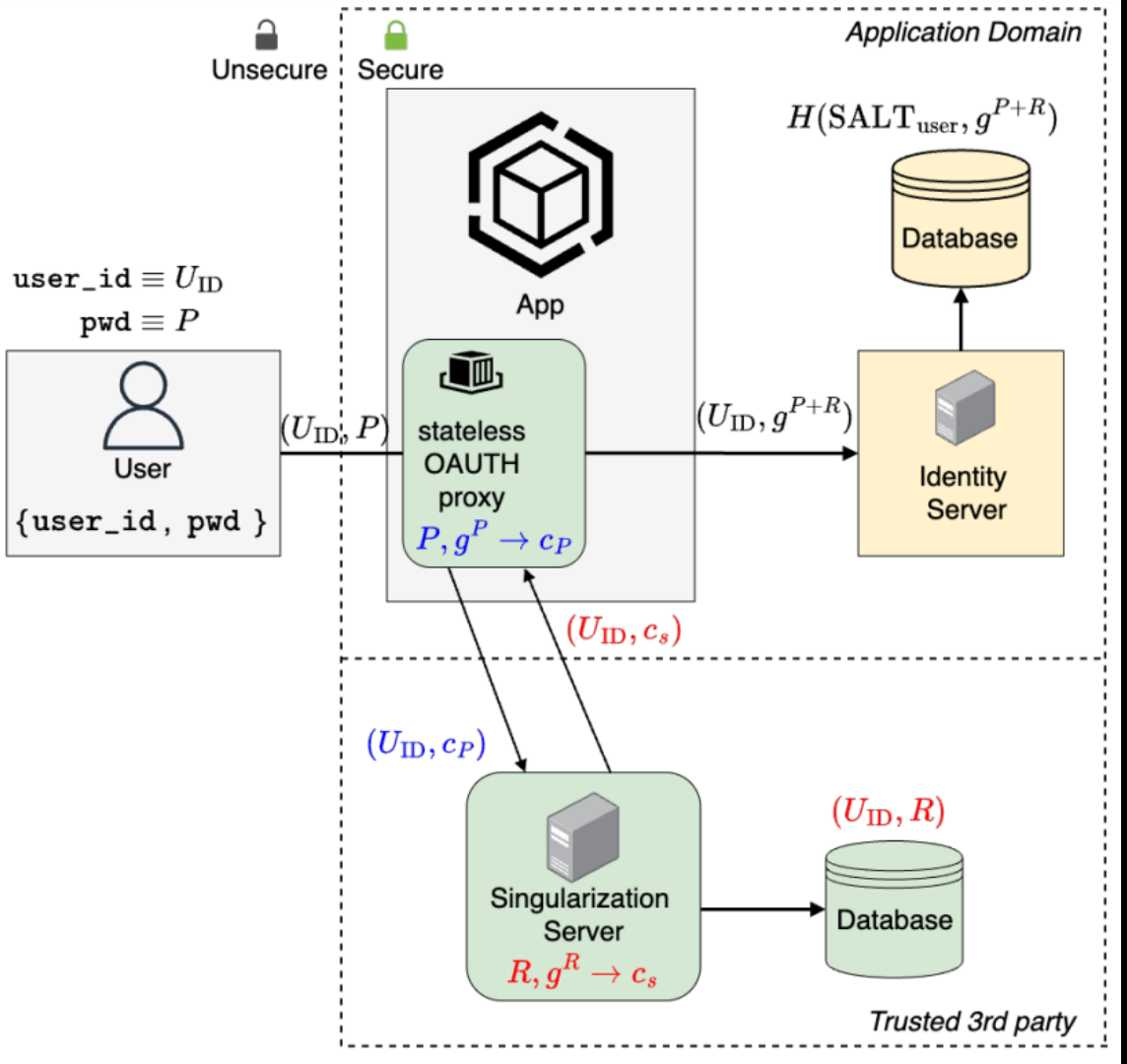
- Password-based authentication remains essential in enterprise security.
- Increasing vulnerabilities due to low-entropy passwords.
- Traditional storage methods (hashing + salt) are insufficient against modern threats.
- Need for enhanced, privacy-preserving solutions.
- Offline attacks targeting password databases.
- Risks during data breaches exposing user credentials.
- Limitations of current methods:
- Hashing with salt: low entropy passwords are vulnerable.
- Password reuse and weak passwords.
- Privacy concerns with third-party authentication.

Related Work

- Password managers: generate high-entropy passwords, but pose single points of failure.
- Multi-factor authentication (MFA): adds security but increases complexity.
- Third-party services (OAuth, SAML, OpenID): streamline access but raise privacy issues.
- Emerging trends:
- Decentralized identity systems (blockchain).
- Privacy-preserving authentication protocols.

Singularization Concept

- Dynamic defense strategy: each system instance is unique.
- Enhances security without extensive infrastructure changes.
- Integrates with existing identity servers.
- Core idea: increase password entropy via a dedicated service.


Unsecure
Secure
Application Domain
$H(\mathrm{SALT}_{\mathrm{user}}, g^{P+R})$
Database
App
$\mathtt{user\_id} \equiv U_{\mathrm{ID}}$
$\mathtt{pwd} \equiv P$
User
{user_id, pwd }
(U_{ID}, P)
stateless
OAUTH
proxy
$P, g^P \rightarrow c_P$
$(U_{\mathrm{ID}}, g^{P+R})$
Identity
Server
(U_{ID}, c_s)
(U_{ID}, c_P)
Singularization
Server
$R, g^R \rightarrow c_s$
(U_{ID}, R)
Database
Trusted 3rd party

Singularization Concept - Advantages

- Increased password entropy (≥128 bits).
- Brute-force attack complexity becomes infeasible.
- Separation of databases:
  - Singularization Service and ID server are independent.
- Homomorphic encryption ensures confidentiality.
- Stateless proxy:
  - No key storage, reducing attack surface.
- Seamless integration with existing systems.
- Privacy-preserving: no plaintext passwords or user data shared.
- Minimal infrastructure modifications.
- Resistant to offline brute-force attacks.

Questions?


Thank you!
orange™