■ Authentication Security Lab

Project Explanation Guide

Generated on: January 20, 2026

■ Project Overview

> *"This project demonstrates why storing passwords securely is critical, and how modern systems can upgrade from weak to strong security without disrupting users."*

■ The Problem (Security Crisis)

When websites store passwords, they don't store them as plain text. They convert them into a 'hash' - a scrambled version. But here's the problem: **old algorithms like MD5 are too fast**. An attacker with a modern graphics card can try 200 billion password guesses per second. That means a simple password like 'password123' can be cracked in microseconds.

Algorithm	Hash Rate (RTX 4090)	Time to crack "password123"
MD5 (Broken)	200 billion/sec	0.002 seconds
SHA-1 (Weak)	100 billion/sec	0.004 seconds
Argon2 (Secure)	1,000/sec	56 hours

■ The Solution (Modern Security)

Modern algorithms like **Argon2** solve this by being intentionally slow. They require a lot of memory - 64 megabytes per attempt. This means even a powerful GPU can only try 1,000 guesses per second instead of 200 billion. Same password, same attacker, but now it takes years instead of seconds.

■ What Our System Does

Registration: Users can create accounts. We let them choose MD5 (weak) or Argon2 (strong) to show the difference.

Breach Time Estimator: When you type a password, it calculates how long an attacker would need to crack it using different algorithms. You see in real-time that Argon2 takes millions of times longer.

Transparent Migration: This is the clever part. When an old MD5 user logs in, we verify their password, then immediately re-hash it with Argon2 behind the scenes. The user doesn't notice anything, but their account is now secure.

Rate Limiting: We also protect against brute force attacks. After 5 failed login attempts, the account locks for 15 minutes.

■■ Technical Flow

Here's how a login works:

1. User sends username and password

2. We check if they're rate-limited (too many failed attempts)

3. We find their account and check which algorithm they use

4. If MD5: we hash the password with MD5 and compare

5. If the password matches AND they're using MD5, we automatically upgrade them to Argon2

6. Next time they login, they'll use the secure algorithm

This is how real companies like Facebook and Google migrated billions of users to better security.

■ Why Argon2 is Special

Argon2 won the Password Hashing Competition in 2015. It has three key properties:

Memory-hard: Requires 64MB of RAM per hash, so GPUs can't parallelize

Time-cost: We make it run 3 iterations, slowing it down

Built-in salt: Every hash is unique, even for identical passwords

The result: a password that takes 1 second to crack with MD5 would take 200 million seconds with Argon2.

■ Security Comparison

Password	MD5 (Broken)	Argon2 (Secure)	Improvement Factor
pass	Instant	7.6 minutes	∞
password123	0.002 seconds	56 hours	100 million×
P@ssw0rd!	38 seconds	1,200 years	1 billion×
MySecureP@ss123!	38 years	7.6 billion years	200 million×

■ Key Takeaway

*"The main lesson is: **never use MD5 or SHA-1 for passwords**. Always use Argon2, bcrypt, or scrypt. And if you have legacy users, implement transparent migration - it upgrades security without forcing password resets."*

■ Real-World Impact

In 2012, LinkedIn was breached and 117 million passwords were stolen. They used unsalted SHA-1. Within days, most passwords were cracked. If they had used Argon2, those same passwords would still be safe today - it would take centuries to crack them.

■ Quick Glossary

Term	Simple Explanation
Hash	One-way scrambling of a password
Salt	Random data added to make each hash unique
MD5	Old, fast, broken algorithm (1992)
Argon2	Modern, slow, secure algorithm (2015)
Brute Force	Trying every possible password
Rate Limiting	Blocking after too many failed attempts
Migration	Upgrading old hashes to new algorithm

■ End of Document

Good luck with your explanation! ■