Information Security Lab Project

Offline and Online Brute-Force Attacks on Password-Based Authentication Systems

Course: Information Security

Instructor: Sania Umer

University: Comsats University Islamabad, Wah Campus

Submission Date: 19-12-2025

Submitted By (Group of 4 Students):

1. Aymen Ayesha (SP24-BCS-107)

2. Faisal Ali (SP24-BCS-068)

3. M. Huzaifa Khan (SP24-BCS-114)

4. Ehtesham Faisal (SP24-BCS-055)

1. Introduction

This report presents a detailed study of offline and online brute-force attacks on password-based authentication systems. The objective is to demonstrate how weak hashing practices can be exploited offline and how live systems are targeted online, along with the effectiveness of modern security defenses.

- **Importance of Password Hashing:**
  Password hashing is important because it protects users' passwords by converting them into unreadable formats before storage. Even if a database is compromised, attackers cannot easily recover the original passwords. Hashing also helps prevent password reuse attacks across multiple services. Using strong hashing algorithms enhances overall system security and user trust.

- **Difference between Online and Offline Brute Force Attack:**

Feature	Online Brute Force Attack	Offline Brute Force Attack
• Attack method	• Tries passwords by logging in through the system	• Cracks passwords using stolen hash files
• Speed	• Slow due to login limits and lockouts	• Fast, limited only by hardware power
• Detection	• Easily detected by security systems	• Hard to detect once data is stolen
• Requirement	• Needs live access to the target system	• Needs access to password hashes

2. Part A – Offline Brute-Force Attack

Offline brute-force attacks were performed using Hashcat against MD5, SHA-1, and SHA-256 hashed passwords using the rockyou.txt dictionary.

Table of Crack hashes of MD5:

Hashes	Cracked	df53ca268240ca76670c8566ee54568a	computer
936b58ef16bb5c6235f8ef7be420ad21	phishing	312f91285e048e09bb4aefef23627994	laptop
f3395cd54cf857ddf8f2056768ff49ae	router	10238d6114894bb5edc39981835bbec9	university
b36eb6a54154f7301f004e1e61c87ce8	switch	e91e6348157868de9dd8b25c81aebfb9	security
23eeeb4347bdd26bfc6b7ee9a3b755dd	python	ed469618898d75b149e5c7c4b6a1c415	algorithm

Uncracked: f3f0c6e992b7562598d9865b6fe8b3a6

Table of Crack hashes of SHA1:

Hashes	Cracked	c60266a8adad2f8ee67d793b4fd3fd0ffd73cc61	computer
57024125b8d8c3f6d69f99d7b42e63a71b0bb3f8	phishing	e068381bbd9eec031347912c57dac0f67479ba23	laptop
77eb1db6cb81b3cb088d36ab7aae8f230dcfaa28	router	f9f914060ccb1e10d551ad49016b1a6658d6edec	university
01ba7992f85de477e8e630428eb5ed14769f9155	switch	9a53b0819e7b65cebdd2f94ad43011ac678d3fc3	algorithm
4235227b51436ad86d07c7cf5d69bda2644984de	python	8eec7bc461808e0b8a28783d0bec1a3a22eb0821	security

Uncracked: 316ca0099385ebe6d7fcb9d5e0785deafedfe791

Table of Crack hashes of SHA256:

Hashes	Cracked
2ce4ce947f548d483889bf6b6c4731ebd50a8f8e48fc7b84373c5cc4cf1d35b3	phishing
74c95604043427f0bee1d0e16bfa53afd537f736ad0073c4cc4e1ccb3a82b5dc	router
78b49fb2cc2d2ed6c1bb8383b3d267b3bc623a8a0fb4aff4aa5d3db74c3b4967	switch
11a4a60b518bf24989d481468076e5d5982884626aed9faeb35b8576fcd223e1	python
aa97302150fce811425cd84537028a5afbe37e3f1362ad45a51d467e17afdc9c	computer
5eec0dc419aa8337bf725f026fda9c78c1cb1c642eeaff9d6e1112f37783e942	laptop
5d2d3ceb7abe552344276d47d36a8175b7aeb250a9bf0bf00e850cd23ecf2e43	security
adba6c0ec8a8d89efb03de642427a09302fa4f7474989ecf33fd545a30d2ff5b	university
b1eb2ec8ac9f31ff7918231e67f96e6deda83a9ff33ed2c67443f1df81e5ed14	algorithm

Difficulty level of common hash algorithms (against attacks):

1. **MD5**
   - Very low difficulty level.
   - Vulnerable to collisions and fast brute-force attacks.
   - Considered broken and not safe for password storage.

2. **SHA-1**

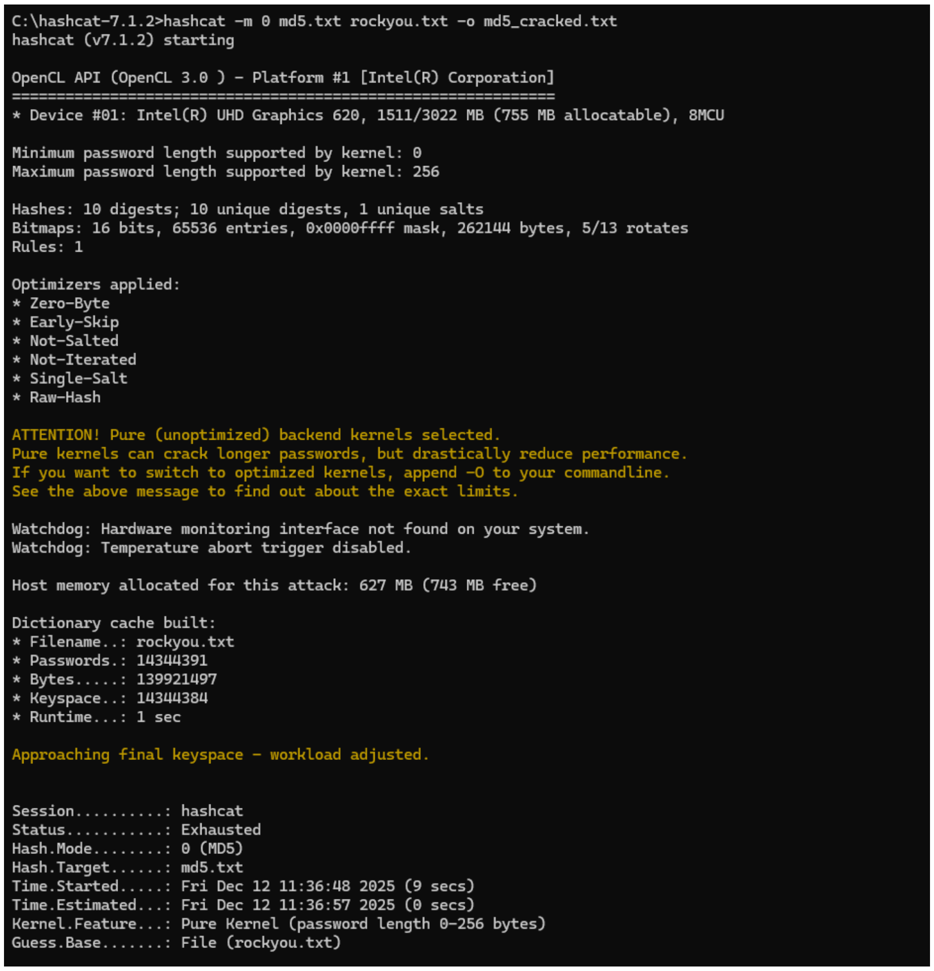
    - Low difficulty level.
    - Collision attacks are practical, making it insecure.
    - Not recommended for security-critical applications.

3. **SHA-256**

    - High difficulty level.
    - No practical collision attacks known so far.
    - Widely used and considered secure when implemented properly.

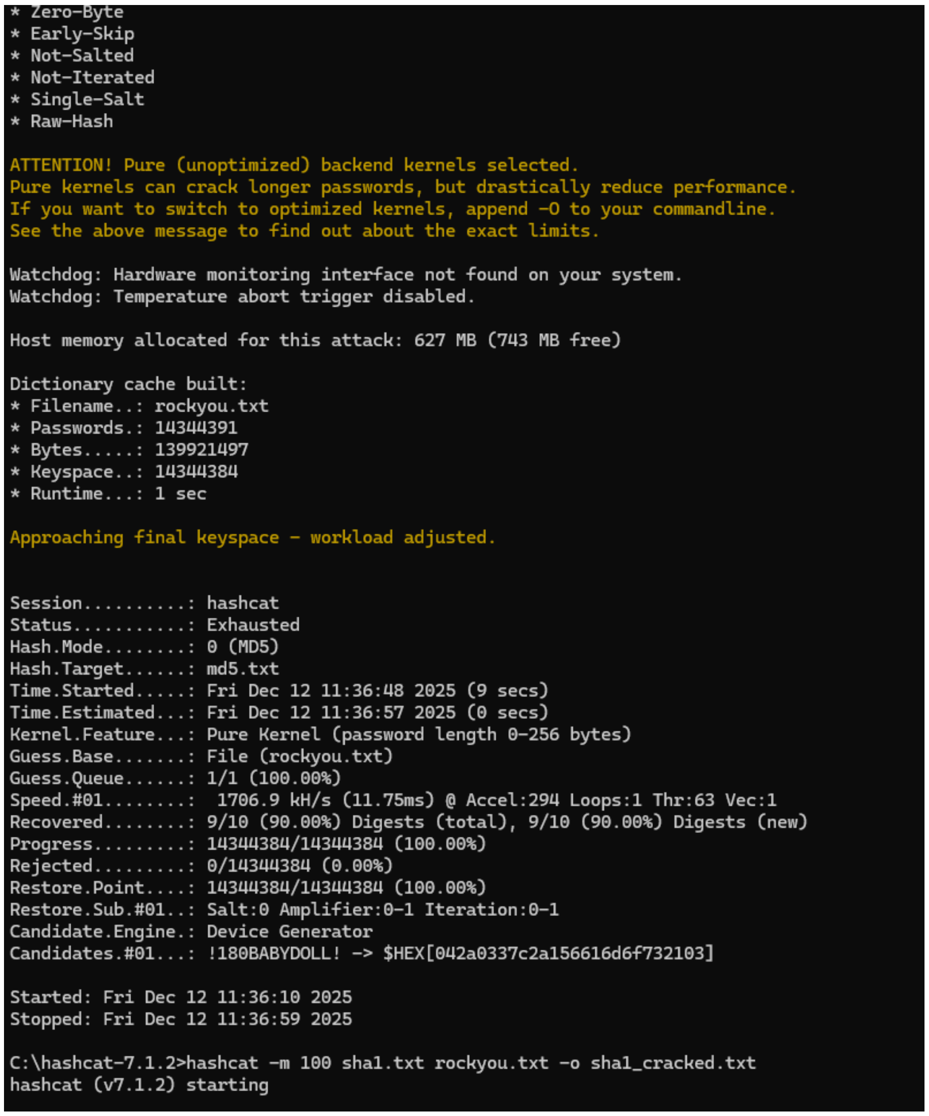
Screenshots of tools used

MD5:



```
C:\hashcat-7.1.2>hashcat -m 0 md5.txt rockyou.txt -o md5_cracked.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 ) - Platform #1 [Intel(R) Corporation]
=============================================================
* Device #01: Intel(R) UHD Graphics 620, 1511/3022 MB (755 MB allocatable), 8MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 10 digests; 10 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory allocated for this attack: 627 MB (743 MB free)

Dictionary cache built:
* Filename..: rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 1 sec

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: md5.txt
Time.Started.....: Fri Dec 12 11:36:48 2025 (9 secs)
Time.Estimated...: Fri Dec 12 11:36:57 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.......: File (rockyou.txt)
```

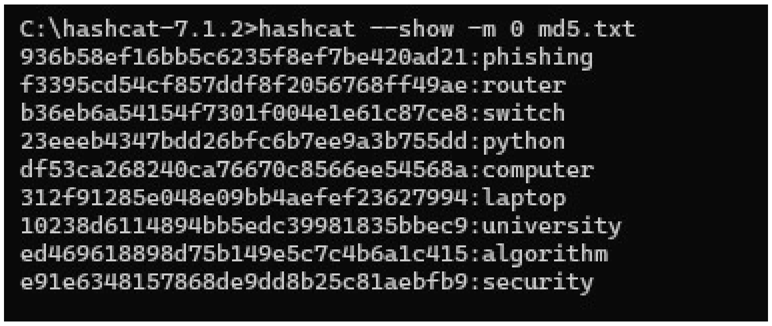
Figure 1: MD5 Command Execution



```
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory allocated for this attack: 627 MB (743 MB free)

Dictionary cache built:
* Filename..: rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 1 sec

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: md5.txt
Time.Started.....: Fri Dec 12 11:36:48 2025 (9 secs)
Time.Estimated...: Fri Dec 12 11:36:57 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#01........:  1706.9 kH/s (11.75ms) @ Accel:294 Loops:1 Thr:63 Vec:1
Recovered........: 9/10 (90.00%) Digests (total), 9/10 (90.00%) Digests (new)
Progress.........: 14344384/14344384 (100.00%)
Rejected.........: 0/14344384 (0.00%)
Restore.Point....: 14344384/14344384 (100.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: !180BABYDOLL! -> $HEX[042a0337c2a156616d6f732103]

Started: Fri Dec 12 11:36:10 2025
Stopped: Fri Dec 12 11:36:59 2025

C:\hashcat-7.1.2>hashcat -m 100 sha1.txt rockyou.txt -o sha1_cracked.txt
hashcat (v7.1.2) starting
```

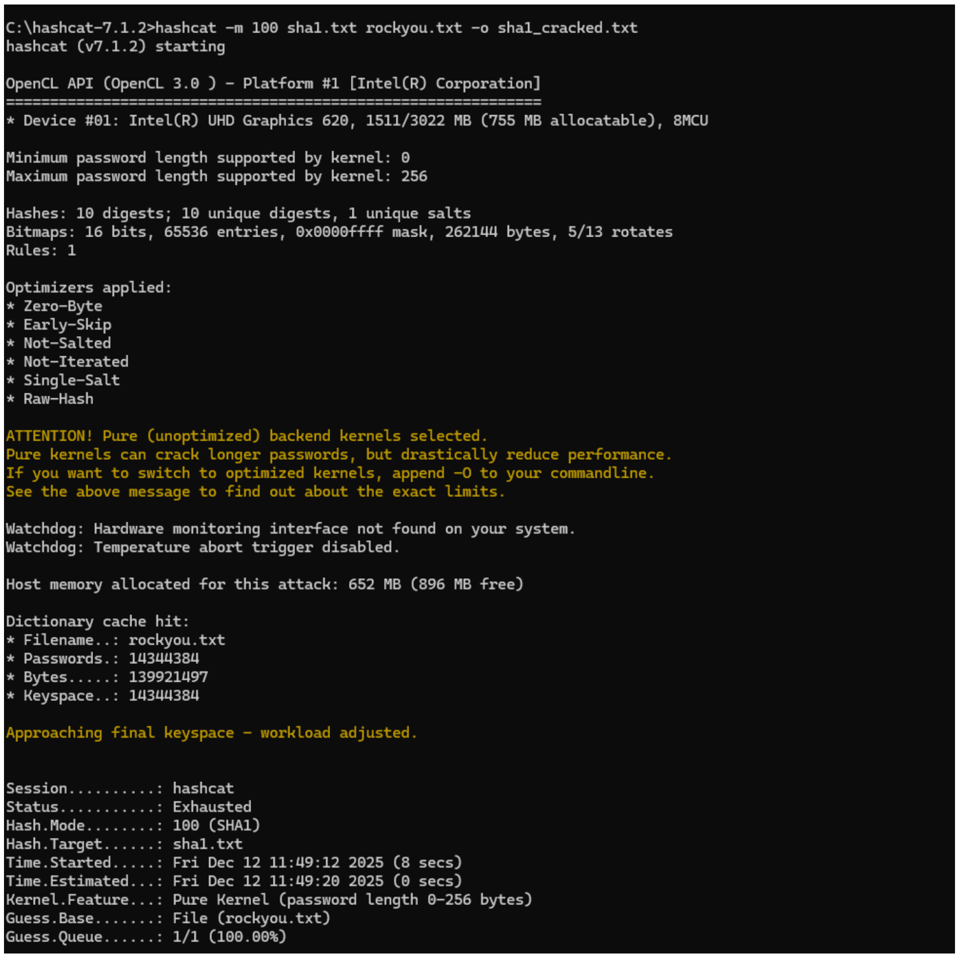
Figure 2: MD5 Output

To Show Cracked MD5.text:



```
C:\hashcat-7.1.2>hashcat --show -m 0 md5.txt
936b58ef16bb5c6235f8ef7be420ad21:phishing
f3395cd54cf857ddf8f2056768ff49ae:router
b36eb6a54154f7301f004e1e61c87ce8:switch
23eeeb4347bdd26bfc6b7ee9a3b755dd:python
df53ca268240ca76670c8566ee54568a:computer
312f91285e048e09bb4aefef23627994:laptop
10238d6114894bb5edc39981835bbec9:university
ed469618898d75b149e5c7c4b6a1c415:algorithm
e91e6348157868de9dd8b25c81aebfb9:security
```

SHA1:



```
C:\hashcat-7.1.2>hashcat -m 100 sha1.txt rockyou.txt -o sha1_cracked.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 ) - Platform #1 [Intel(R) Corporation]
=============================================================
* Device #01: Intel(R) UHD Graphics 620, 1511/3022 MB (755 MB allocatable), 8MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 10 digests; 10 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory allocated for this attack: 652 MB (896 MB free)

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 100 (SHA1)
Hash.Target......: sha1.txt
Time.Started.....: Fri Dec 12 11:49:12 2025 (8 secs)
Time.Estimated...: Fri Dec 12 11:49:20 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
```

Figure 3: SHA-1 Command Execution

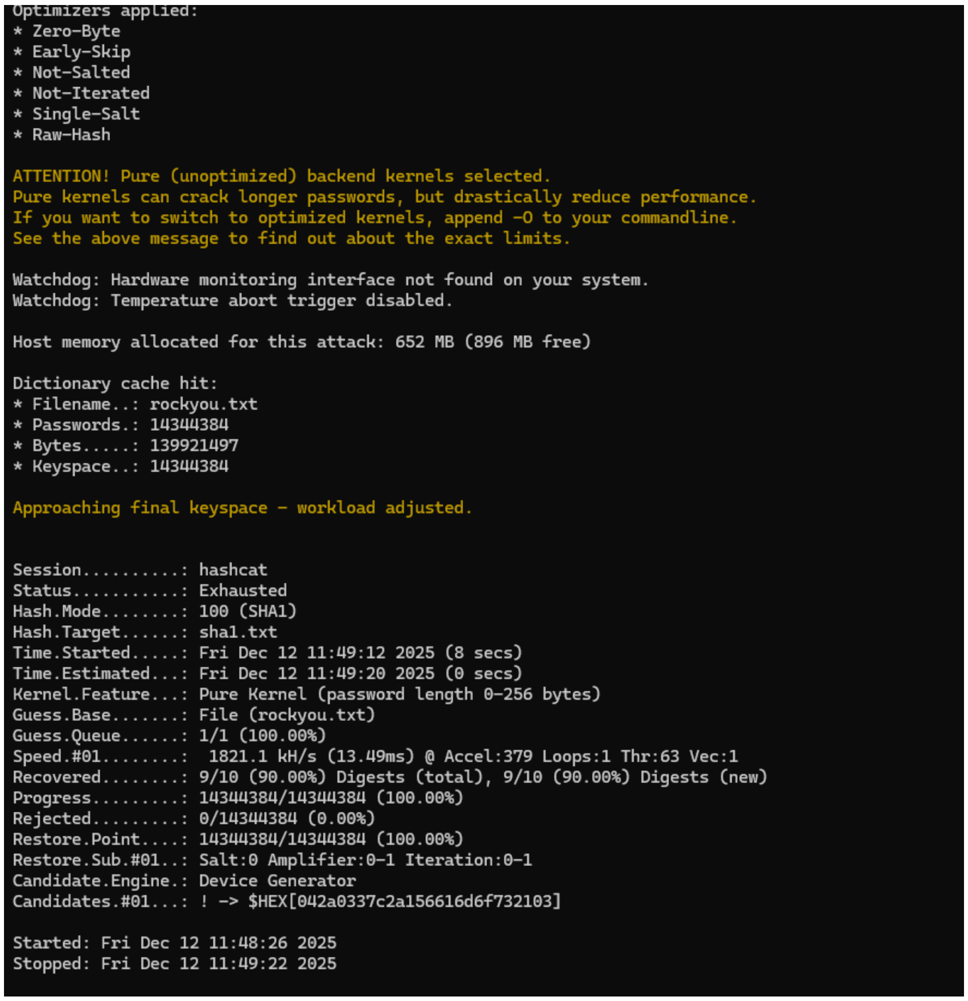
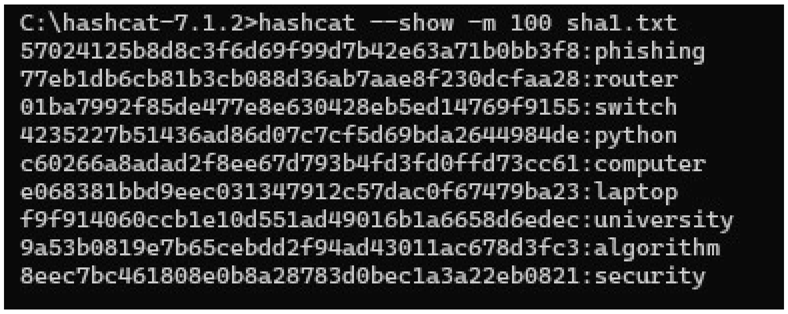
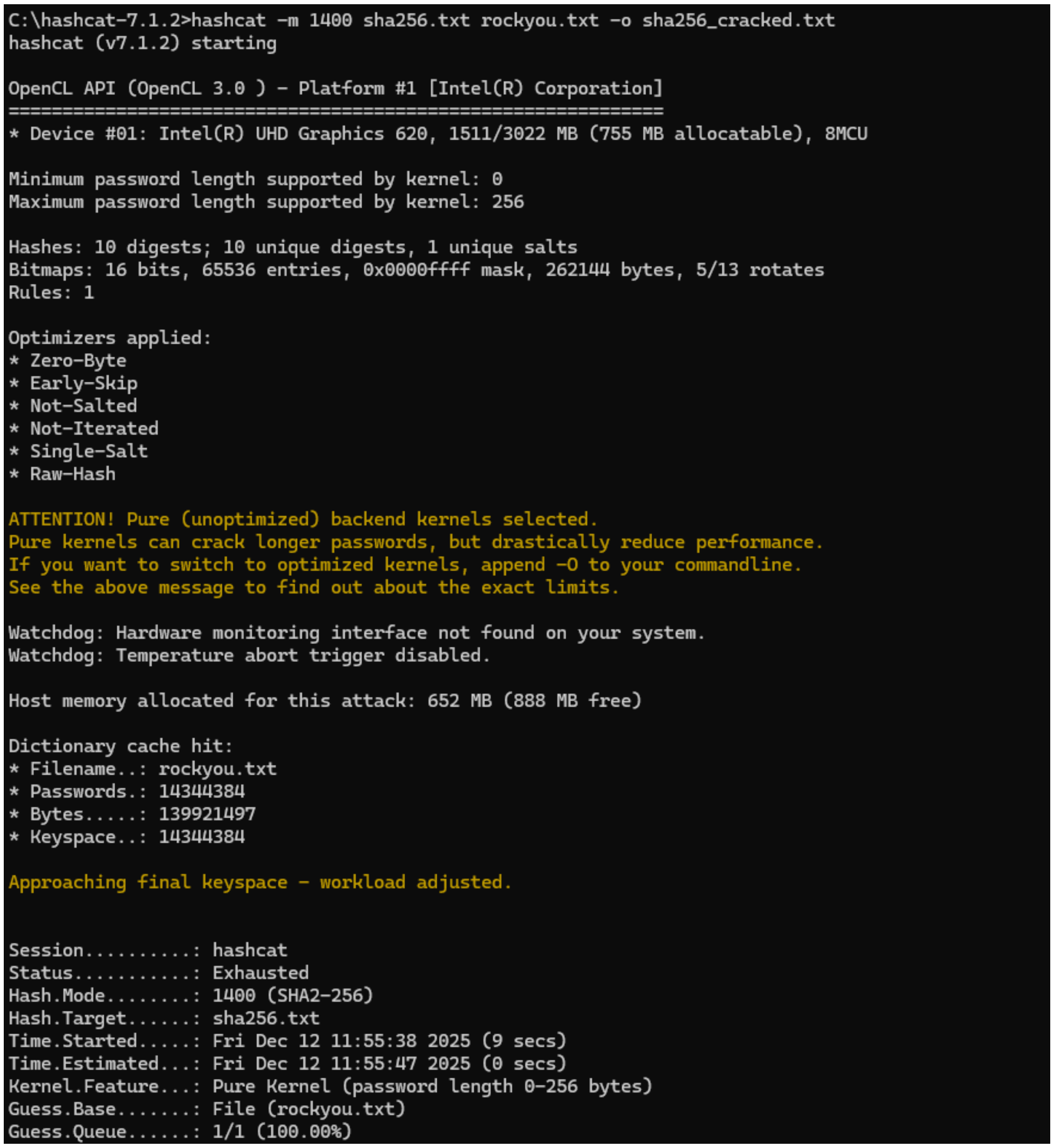

```
Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory allocated for this attack: 652 MB (896 MB free)

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 100 (SHA1)
Hash.Target......: sha1.txt
Time.Started.....: Fri Dec 12 11:49:12 2025 (8 secs)
Time.Estimated...: Fri Dec 12 11:49:20 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#01........:  1821.1 kH/s (13.49ms) @ Accel:379 Loops:1 Thr:63 Vec:1
Recovered........: 9/10 (90.00%) Digests (total), 9/10 (90.00%) Digests (new)
Progress.........: 14344384/14344384 (100.00%)
Rejected.........: 0/14344384 (0.00%)
Restore.Point....: 14344384/14344384 (100.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: ! -> $HEX[042a0337c2a156616d6f732103]

Started: Fri Dec 12 11:48:26 2025
Stopped: Fri Dec 12 11:49:22 2025
```

Figure 4: SHA-1 Output

To Show Cracked SHA1.text:

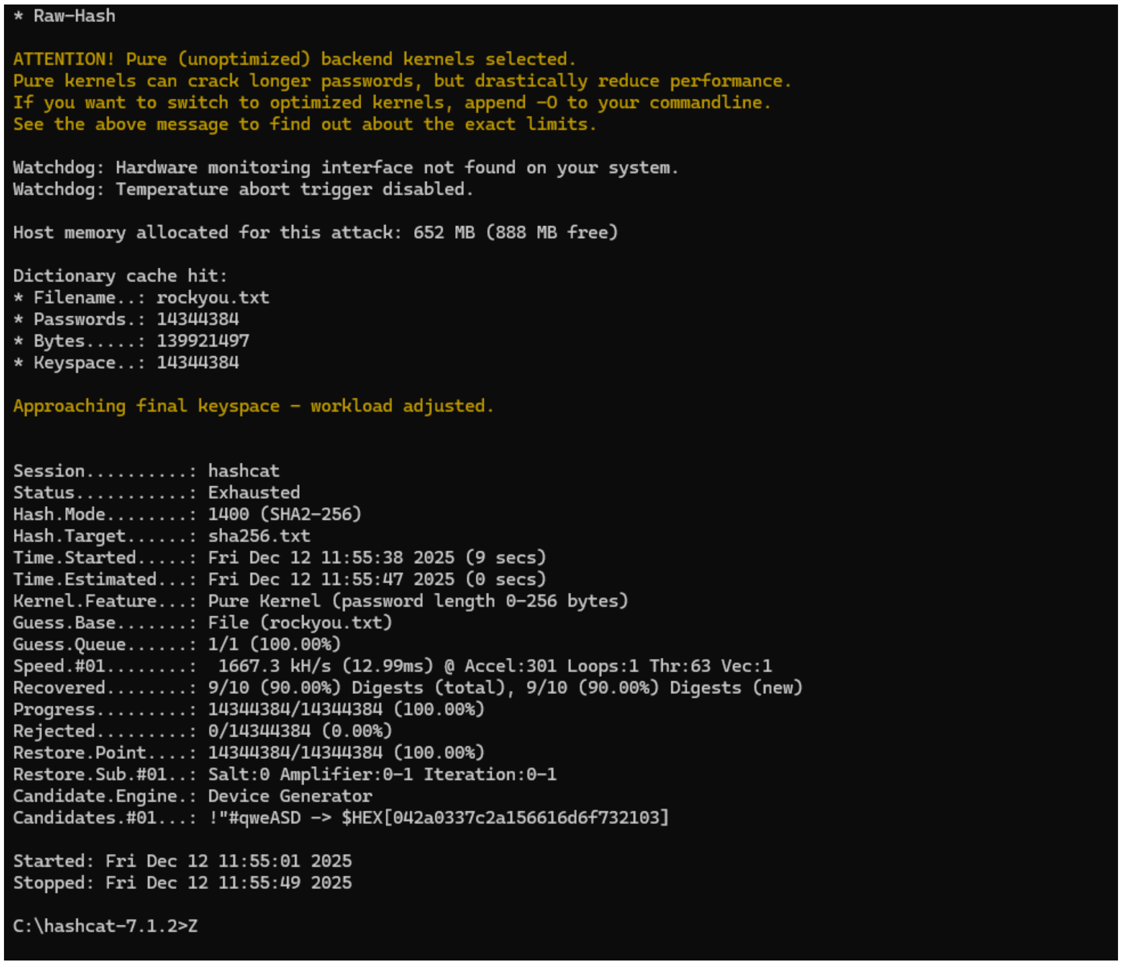

```
C:\hashcat-7.1.2>hashcat --show -m 100 sha1.txt
57024125b8d8c3f6d69f99d7b42e63a71b0bb3f8:phishing
77eb1db6cb81b3cb088d36ab7aae8f230dcfaa28:router
01ba7992f85de477e8e630428eb5ed14769f9155:switch
4235227b51436ad86d07c7cf5d69bda2644984de:python
c60266a8adad2f8ee67d793b4fd3fd0ffd73cc61:computer
e068381bbd9eec031347912c57dac0f67479ba23:laptop
f9f914060ccb1e10d551ad49016b1a6658d6edec:university
9a53b0819e7b65cebdd2f94ad43011ac678d3fc3:algorithm
8eec7bc461808e0b8a28783d0bec1a3a22eb0821:security
```

SHA256:



```
C:\hashcat-7.1.2>hashcat -m 1400 sha256.txt rockyou.txt -o sha256_cracked.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 ) - Platform #1 [Intel(R) Corporation]
=============================================================
* Device #01: Intel(R) UHD Graphics 620, 1511/3022 MB (755 MB allocatable), 8MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 10 digests; 10 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory allocated for this attack: 652 MB (888 MB free)

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: sha256.txt
Time.Started.....: Fri Dec 12 11:55:38 2025 (9 secs)
Time.Estimated...: Fri Dec 12 11:55:47 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
```

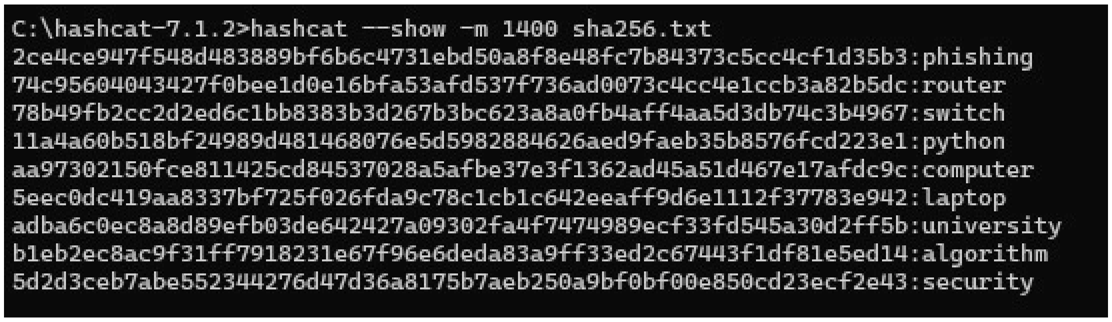
Figure 5: SHA-256 Command Execution



```
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory allocated for this attack: 652 MB (888 MB free)

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: sha256.txt
Time.Started.....: Fri Dec 12 11:55:38 2025 (9 secs)
Time.Estimated...: Fri Dec 12 11:55:47 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#01........:  1667.3 kH/s (12.99ms) @ Accel:301 Loops:1 Thr:63 Vec:1
Recovered........: 9/10 (90.00%) Digests (total), 9/10 (90.00%) Digests (new)
Progress.........: 14344384/14344384 (100.00%)
Rejected.........: 0/14344384 (0.00%)
Restore.Point....: 14344384/14344384 (100.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: !"#qweASD -> $HEX[042a0337c2a156616d6f732103]

Started: Fri Dec 12 11:55:01 2025
Stopped: Fri Dec 12 11:55:49 2025

C:\hashcat-7.1.2>Z
```

Figure 6: SHA-256 Output

To Show Cracked SHA256.text:



```
C:\hashcat-7.1.2>hashcat --show -m 1400 sha256.txt
2ce4ce947f548d483889bf6b6c4731ebd50a8f8e48fc7b84373c5cc4cf1d35b3:phishing
74c95604043427f0bee1d0e16bfa53afd537f736ad0073c4cc4e1ccb3a82b5dc:router
78b49fb2cc2d2ed6c1bb8383b3d267b3bc623a8a0fb4aff4aa5d3db74c3b4967:switch
11a4a60b518bf24989d481468076e5d5982884626aed9faeb35b8576fcd223e1:python
aa97302150fce811425cd84537028a5afbe37e3f1362ad45a51d467e17afdc9c:computer
5eec0dc419aa8337bf725f026fda9c78c1cb1c642eeaff9d6e1112f37783e942:laptop
adba6c0ec8a8d89efb03de642427a09302fa4f7474989ecf33fd545a30d2ff5b:university
b1eb2ec8ac9f31ff7918231e67f96e6deda83a9ff33ed2c67443f1df81e5ed14:algorithm
5d2d3ceb7abe552344276d47d36a8175b7aeb250a9bf0bf00e850cd23ecf2e43:security
```

3. Part B – Online Brute-Force Attack

A secure login system was developed and subjected to controlled online brute-force and dictionary attacks. The system was then hardened using multiple defensive mechanisms.

Architecture of your login system:

- **Frontend:** HTML, CSS and JavaScript
- **Backend:** Server-side logic (Node.js + Express)
- **Database:** MySQL— Where data is stored.
- **Security:** How data is protected (encryption, hashing, Captcha , OTP Verification)

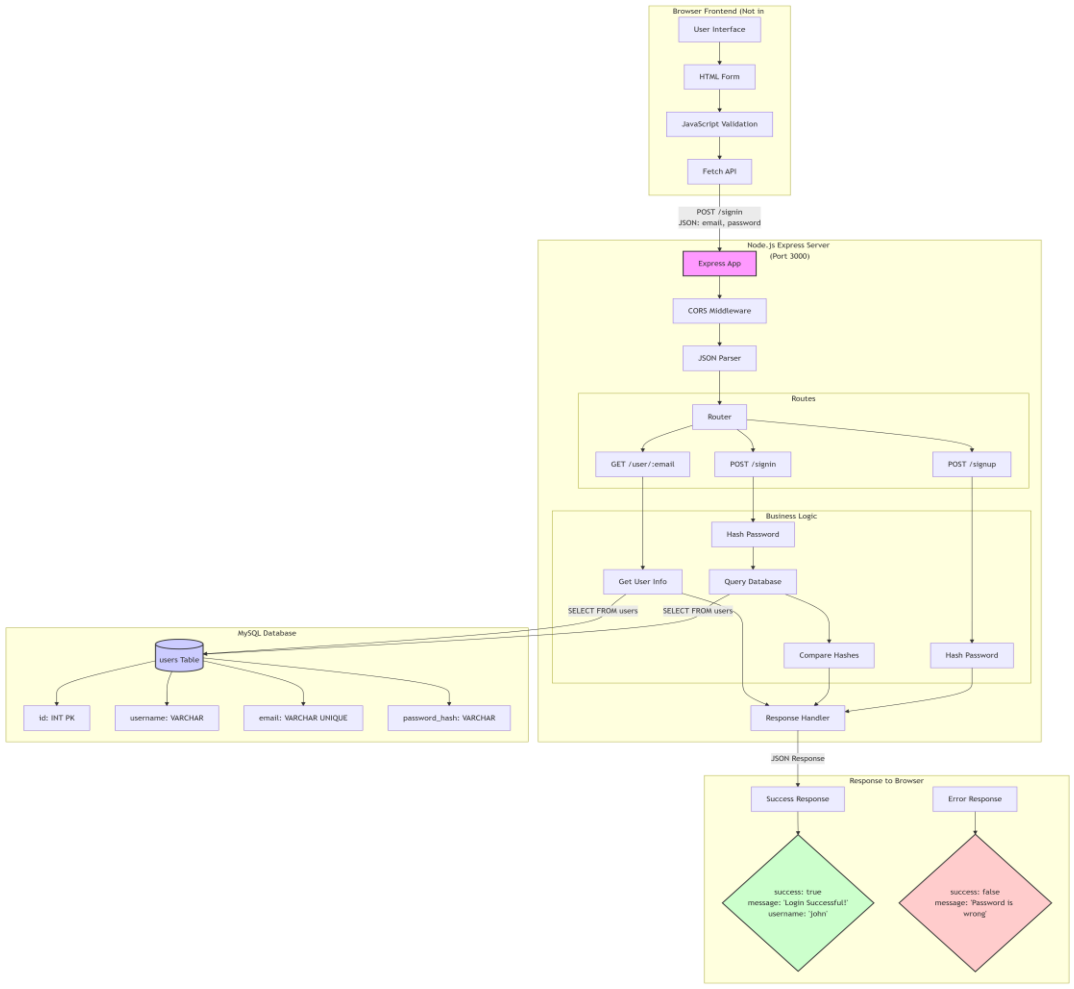


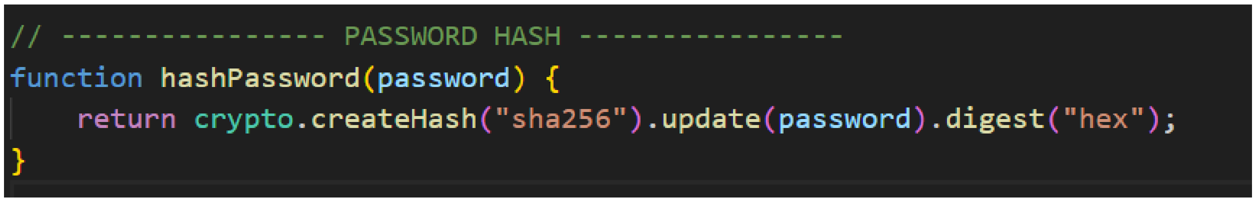
Figure: Architecture of the System

Code snippets for hashing logic:

```
function hashPassword(password) {
```

    return crypto.createHash("sha256").update(password).digest("hex");

}



```
// ---------------- PASSWORD HASH ----------------
function hashPassword(password) {
    return crypto.createHash("sha256").update(password).digest("hex");
}
```

Attack methodology:

There are two attack methodology that I used in this project .

- Try a small dictionary attack.
- Try a password brute-force attempt.

1) **Before Defense :**

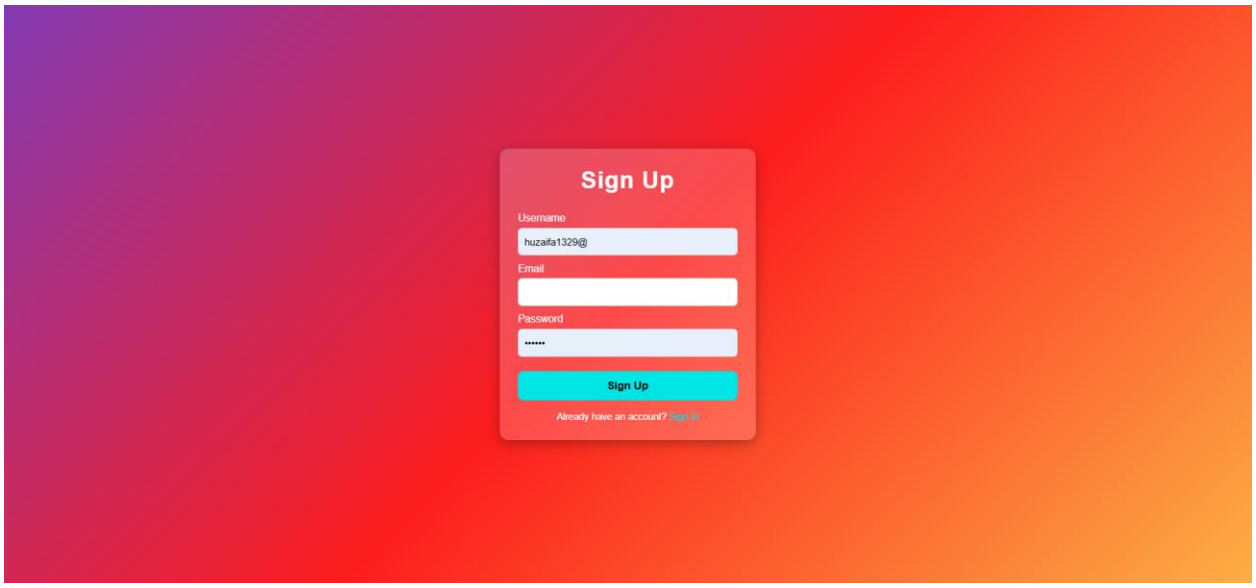
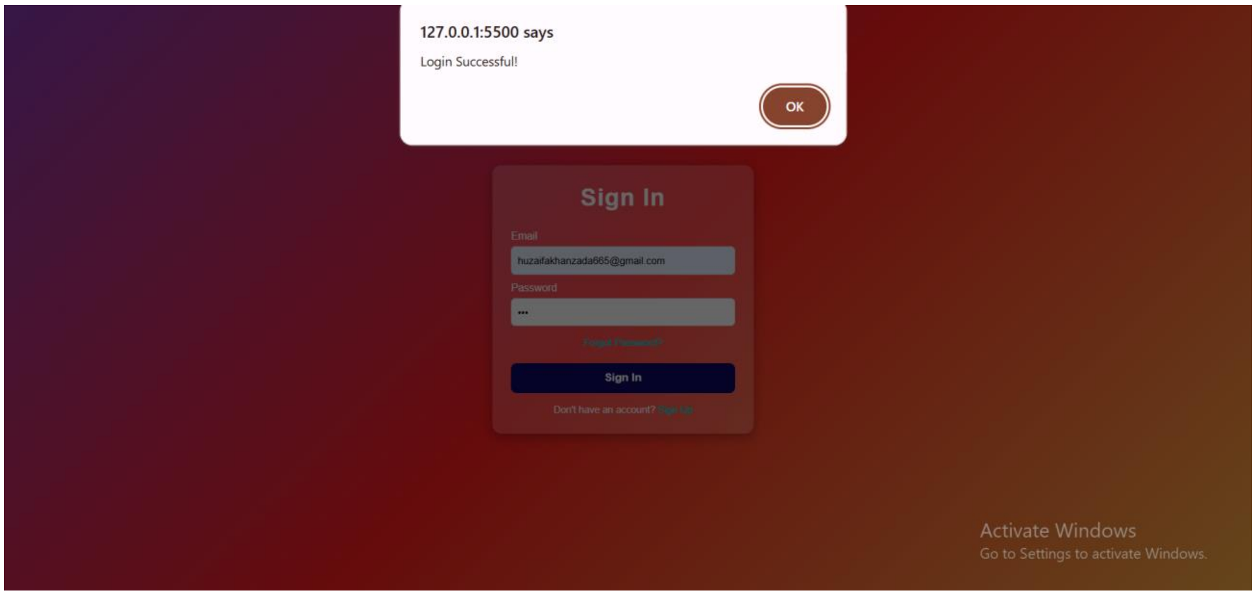


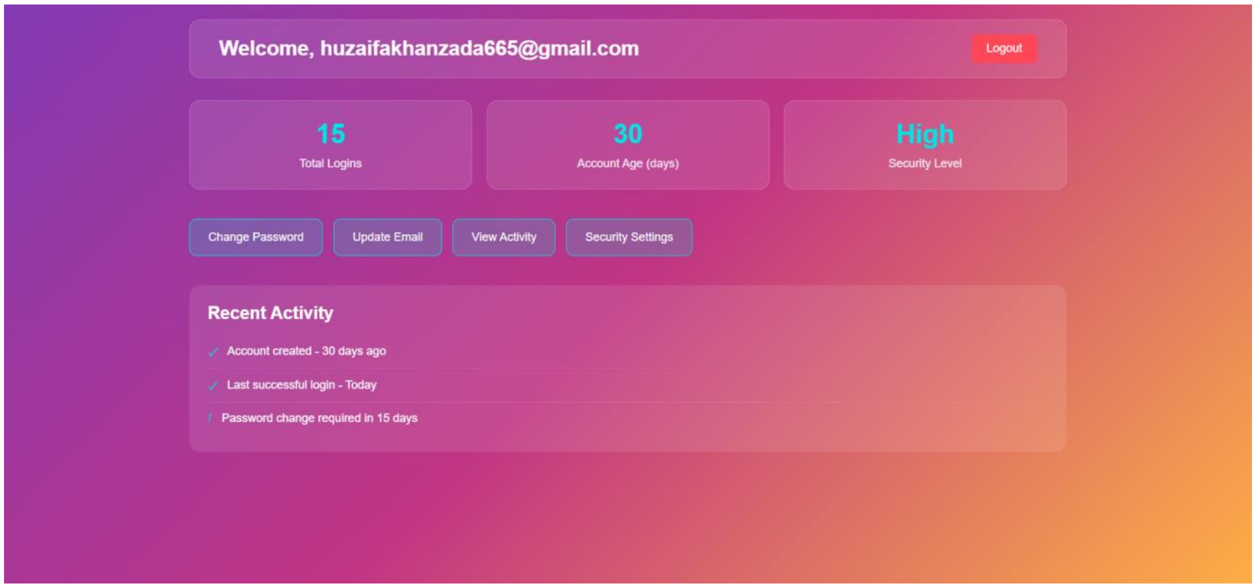
Figure1: SignUp Page before defence


127.0.0.1:5500 says

Login Successful!

OK
Sign In
Email
huzaifakhanzada665@gmail.com
Password
Forgot Password?
Sign In
Don't have an account? Sign Up
Activate Windows
Go to Settings to activate Windows.

Figure2: SignIn Page Before Defense


Welcome, huzaifakhanzada665@gmail.com

15 Total Logins

30 Account Age (days)

High Security Level

Recent Activity

✓ Account created - 30 days ago

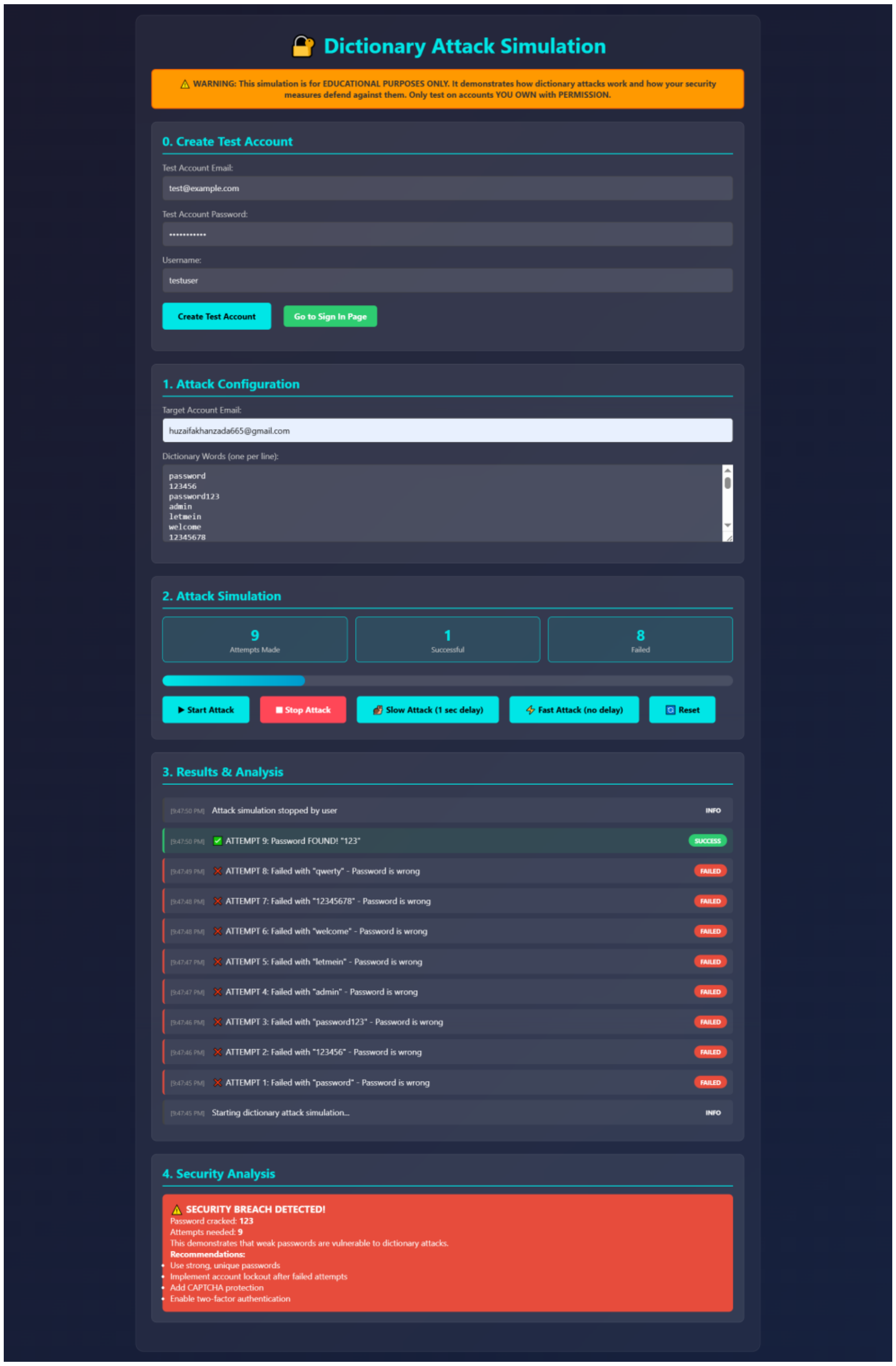
✓ Last successful login - Today

! Password change required in 15 days

Figure3: Home Page

Dictionary Attack Before Defense:

Correct Password found and account is unlocked after 9 attempts as you can see in the below screenshot.



⚠ WARNING: This simulation is for EDUCATIONAL PURPOSES ONLY. It demonstrates how dictionary attacks work and how your security measures defend against them. Only test on accounts YOU OWN with PERMISSION.

0. Create Test Account

Test Account Email: test@example.com

Test Account Password: •••••••••••

Username: testuser

1. Attack Configuration

Target Account Email: huzaifakhanzada665@gmail.com

Dictionary Words (one per line):

```
password
123456
password123
admin
letmein
welcome
12345678
```

2. Attack Simulation

9	1	8
Attempts Made	Successful	Failed

3. Results & Analysis

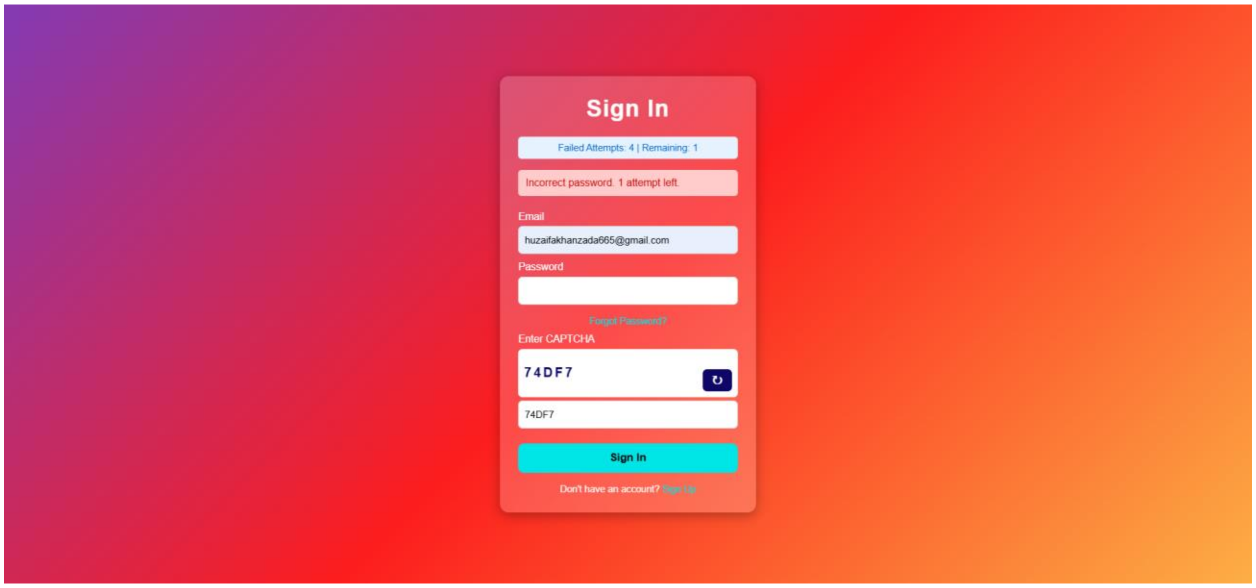
```
[9:47:50 PM]  Attack simulation stopped by user                              INFO
[9:47:50 PM]  ✅ ATTEMPT 9: Password FOUND! "123"                             SUCCESS
[9:47:49 PM]  ❌ ATTEMPT 8: Failed with "qwerty" - Password is wrong          FAILED
[9:47:48 PM]  ❌ ATTEMPT 7: Failed with "12345678" - Password is wrong        FAILED
[9:47:48 PM]  ❌ ATTEMPT 6: Failed with "welcome" - Password is wrong         FAILED
[9:47:47 PM]  ❌ ATTEMPT 5: Failed with "letmein" - Password is wrong         FAILED
[9:47:47 PM]  ❌ ATTEMPT 4: Failed with "admin" - Password is wrong           FAILED
[9:47:46 PM]  ❌ ATTEMPT 3: Failed with "password123" - Password is wrong     FAILED
[9:47:46 PM]  ❌ ATTEMPT 2: Failed with "123456" - Password is wrong          FAILED
[9:47:45 PM]  ❌ ATTEMPT 1: Failed with "password" - Password is wrong        FAILED
[9:47:45 PM]  Starting dictionary attack simulation...                       INFO
```

4. Security Analysis

⚠ **SECURITY BREACH DETECTED!**
Password cracked: **123**
Attempts needed: **9**
This demonstrates that weak passwords are vulnerable to dictionary attacks.
Recommendations:
- Use strong, unique passwords
- Implement account lockout after failed attempts
- Add CAPTCHA protection
- Enable two-factor authentication

Figure4: Dictionary Attack before Defense system apply

2) After Defense:


Sign In

Failed Attempts: 4 | Remaining: 1

Incorrect password. 1 attempt left.

Email

huzaifakhanzada665@gmail.com

Password

Forgot Password?

Enter CAPTCHA

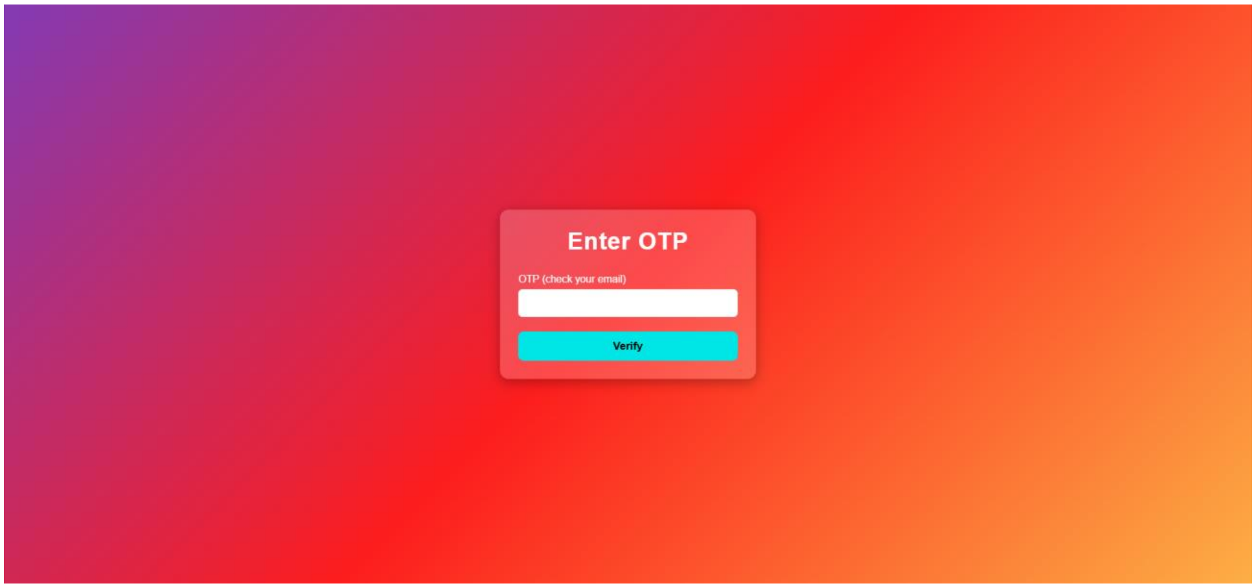
74DF7

74DF7

Sign In

Don't have an account? Sign Up

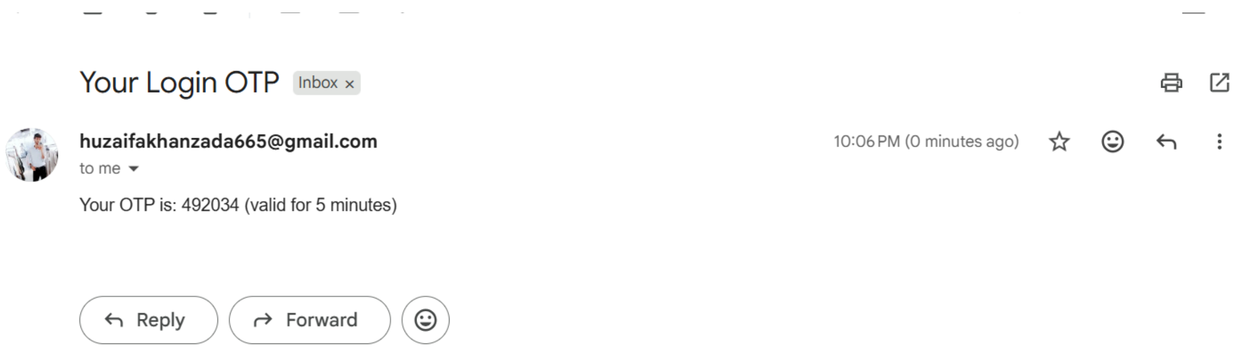
Figure5: SignIn Page After Defense


Enter OTP

OTP (check your email)

Verify

Figure6: Verify OTP Page After Defense


Your Login OTP

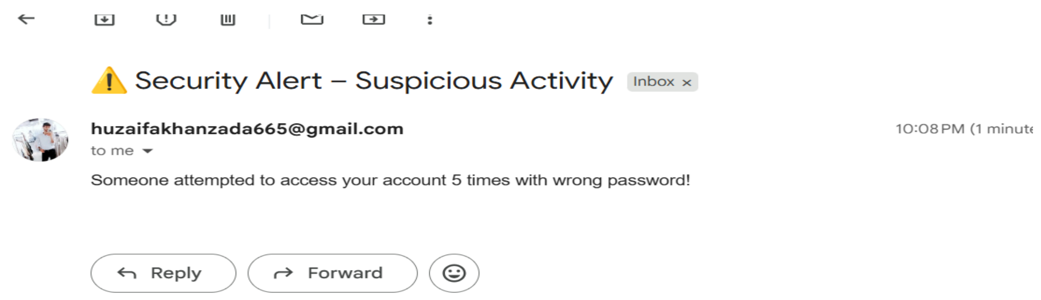
huzaifakhanzada665@gmail.com

to me

10:06 PM (0 minutes ago)

Your OTP is: 492034 (valid for 5 minutes)

Figure7: Successfully OTP sends on email


⚠️ Security Alert – Suspicious Activity

huzaifakhanzada665@gmail.com

to me

10:08 PM (1 minute

Someone attempted to access your account 5 times with wrong password!

Figure8: Alert message on email when someone apply wrong password to access account

Defense techniques implemented:

There are 4 defense techniques that are used in this project to secure the system from any attack (like brute force Attack, dictionary attack).

- Account lockout after 5 failed attempts.
- CAPTCHA
- Email OTP verification
- Email Alert message for malicious activity when someone try to access account by attempting 5 or more times wrong password.

3.1 Defense Effectiveness

Defense Mechanism	Purpose	Observed Result
Account Lockout	Limit login attempts	Attack stopped after multiple failures
Time-based Lock	Delay further attempts	Automated attacks blocked
CAPTCHA	Prevent bots	Scripts failed
OTP / MFA	Second authentication factor	Attack unsuccessful even with correct password

4. Comparison

- **Offline vs Online Attack:**

  a) **Online brute force** tries passwords directly on a live system and is slow due to rate limits and account lockouts.
  b) **Offline brute force** attacks stolen password hashes and is much faster since there are no login restrictions.

- **Why Offline Cracking Is Faster?**
  Attackers can test millions of passwords per second using powerful hardware. No alerts, delays, or account lockouts limit the attack speed.

- **How Hashing and Salting Help Mitigate Attacks?**

a) **Hashing** stores passwords in unreadable form, protecting them if data is leaked.
b) **Salting** adds random data to each password, preventing rainbow table and reuse attacks.

5. Conclusion

This project conclusively shows that unsalted password hashes are vulnerable to offline attacks and that robust, layered defenses are essential for securing live authentication systems.