Identification and Authentication Failures

Alexander Tushinsky

Cybersecurity & Software Development Consultant

@ltmodcs alextushinsky.com

Overview

- What are Identification and Authentication Failures?
- Examples
- Remediation

Identification and Authentication Failures

- Allows automated attacks such as Credential Stuffing or Brute-Force attacks
- Poor password policy
- Sessions that do not expire in a timely manner

Credential Stuffing

- Compromised list of usernames and passwords
- Attempts to use credentials against target site

Brute-Force

- Random list of usernames and passwords
- Sometimes the username is known
- Attempts to use random passwords against target

Poor Passwords

- Length of password
- Complexity requirements
- Password update requirements
- Verification via multifactor authentication
- Clear-text storage at the database level

Sessions

- Do not expire in timely manner
- Session Identifiers do not change
- Session Identifiers exposed via the URL
- Lack of TLS Use (HTTPS)

Demo

Identification and Authentication Failures

- Brute-Force attack
    - Password theft
    - Account compromise

Managing Identity

Passwords

- Use ASP.NET Core Identity
  - Enable 2FA
  - Configure password complexity
  - Enable lock-out
  - Confirm user email
- Avoid custom solutions
- Salted and hashed password storage

```
builder.Services.AddDefaultIdentity<IdentityUser>(options =>
   {
       options.User.RequireUniqueEmail = true;
       options.User.AllowedUserNameCharacters =
                    "abcdefghijklmnopqrstuvwxyz1234567890@-.";

       options.SignIn.RequireConfirmedAccount = true;
       options.SignIn.RequireConfirmedEmail = true;
       options.SignIn.RequireConfirmedPhoneNumber = false;

       options.Password.RequiredLength = 12;
       options.Password.RequireLowercase = true;
       options.Password.RequireUppercase = true;
       options.Password.RequireDigit = true;
       options.Password.RequireNonAlphanumeric = true;
       options.Password.RequiredUniqueChars = 10;

       options.Lockout.AllowedForNewUsers = true;
       options.Lockout.DefaultLockoutTimeSpan =
                                 TimeSpan.FromHours(1);
       options.Lockout.MaxFailedAccessAttempts = 5;
   });
```

- Configure User, SignIn, Password, and Lockout settings for Microsoft.AspNetCore.Identity
- Require unique email, and control the set of characters that can be used in usernames
- Configure sign-in requirements
- Configure password length and complexity
- In particular, the use of RequiredUniqueChars enforces unique passwords
- Set up lockout features

Sessions and Cookies

- Avoid Session Fixation
  - Clear Session object
  - Expire Session cookies
- Set Cookie expiration policies
- Store only secure cookies

```
 builder.Services.AddDistributedMemoryCache();

 builder.Services.AddSession(options =>
    {
        options.IdleTimeout = TimeSpan.FromMinutes(20);
        options.Cookie.HttpOnly = true;
        options.Cookie.IsEssential = true;
    });
```

◀ Enable Session expiration

◀ Enable HttpOnly for Sessions

```
public IActionResult Logout()
{
    HttpContext.Session.Clear();

    return View();
}
```

◀ Clear the Session object upon user log out

```
app.UseCookiePolicy(
    new CookiePolicyOptions
    {
        Secure = CookieSecurePolicy.Always
    });
```

- Enable secure cookies in the Startup.cs Configure() method.

Demo

Remediation

- Review the fixes implemented

Summary

Identification and Authentication Failures

- Identified problems leading to account compromise
- Looked at several solutions that can be implemented to help make our applications more secure

Up Next:
Security Logging and Monitoring Failures