Cryptographic Failures

Alexander Tushinsky
Cybersecurity & Software Development Consultant

@ltmodcs alextushinsky.com

Overview

- What are cryptographic failures?
- Examples
- OWASP Recommendations
- Remediation

Cryptographic Failure

- Leads to sensitive data exposure
- May occur when:
  - Data is transmitted in clear text
  - Outdated cryptographic algorithms used
  - Certificates exposed or reused
  - Encryption is not enforced

Sensitive Data Exposure

When it comes to individuals, we can consider any data point that isn't publicly available "sensitive."

Data points such as birthdays, social security numbers, credit card numbers, and even address information fall under PII (Personally Identifiable Information) and are deemed sensitive.

Sensitive Data Exposure

- User has access to data they are not authorized to see
- Application or server disclose application details
- Flaw in code allows users to access someone else’s data

Software Sensitive Data

For software, there are also several sensitive data categories that we should consider.

Connection strings, source code, file paths, API keys, type of operating system, type of web server, and the specifics about our database server should all fall into this classification.

Sensitive Data Exposure

- Server displays detailed error messages which include code, SQL, or application details
- Folders and files that are web accessible that contain sensitive data, including source code
- HTTP Headers that identity the operating system, or technology running the application
- Lack of TLS encryption allows traffic to be captured by 3rd party

Demo

Cryptographic Failures

- HTTP header exposure

Cryptographic Failure Remediation

Cryptographic Failures According to OWASP

Proactive Controls

C1: Define Security Requirements

C3: Secure Database Access

C7: Enforce Access Controls

C8: Protect Data Everywhere

ASVS

V1.6: Cryptographic Architecture

V1.8: Data Protection and Privacy Architecture

V6.1: Data Classification

V6.2: Algorithms

V6.4: Secrets Management

V8.1: General Data Protection

V8.3: Sensitive Private Data

Data Classification

- Organizations should classify data through policy
- Secure procedures for handling sensitive data

User Access

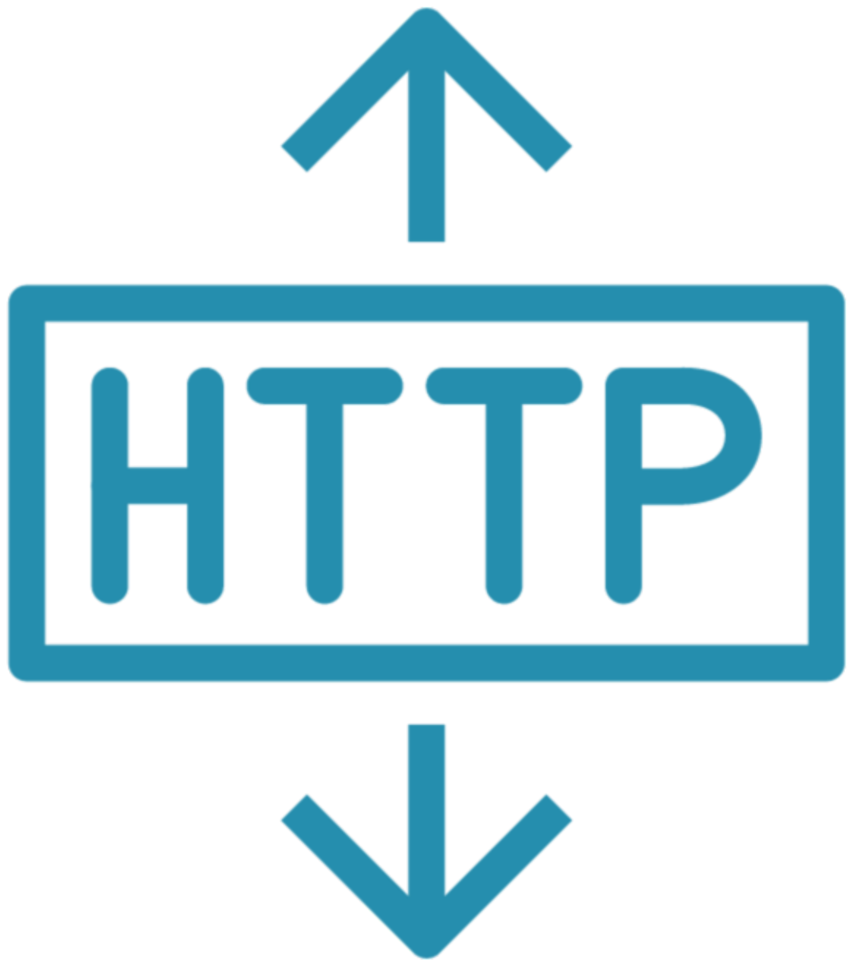
- Filter data to ensure that users only have access to data they are meant to work with
- Ensure that deny by default is implemented
- Assign appropriate roles
- Test your work

Server Considerations

- Avoid text files or ZIP archives with sensitive data in web-accessible folders
- Service accounts set with least privilege access
- Encrypt any sensitive data at rest using table or column level encryption
- Completely segregate development, testing and production environments
- Remove unnecessary components and headers from the environment
- Implement security HTTP headers

Secure HTTP Headers

HTTP Security Headers



- Frequently overlooked
- Can lead to XSS
- May expose server-details
- May bypass TLS encryption

HTTP Headers

- Visit SecurityHeaders.com to identify any issues
- Review the OWASP Secure Header Project
- Use ASP.NET middleware to inject headers into your responses
- Remove un-necessary headers (X-Powered-By)

HTTP Headers

- Strict-Transport-Security
- X-Content-Type-Options
- X-Frame-Options
- Referrer-Policy
- Content-Security-Policy
- Feature-Policy

HTTP Headers

- NWebSec middleware library
- Documentation available: https://docs.nwebsec.com/en/latest/
- Makes it easy to implement secure headers

Header Configuration

Program.cs NWebSec Middleware Library Configuration

```
app.UseHsts(options => options.MaxAge(days: 365)
                                                  .IncludeSubdmains());
app.UseXContentTypeOptions();
app.UseXXssProtection(options => options.EnabledWithBlockMode());
app.UseXfo(options => options.SameOrigin());
app.UseReferrerPolicy(opts => opts.NoReferrer());
app.UseCsp(options => options
     .DefaultSources(s => s.Self())
     .StyleSources(s => s.Self()
        .UnsafeInline()
     )
     .ScriptSources(s => s.Self()
     .UnsafeInline()
     .UnsafeEval())
  );
```

Demo

Remediation

- Add security headers
- Test application using SecurityHeaders.com

Summary

Cryptographic Failures

- Identified what cryptographic failures are
- What OWASP recommends we do
- Implemented security headers, user access controls, and server configurations to mitigate issues

Up Next: Injections