NET201 - DEFENDING .NET FRAMEWORK

Course Learning Objectives

Recognize insecure coding practices from web applications vulnerabilities found in the OWASP Top 10. Implement defensive coding techniques in .NET 4.5 and learn about common frameworks and tools to help support secure coding in .NET. Contrast between insecure and secure coding practices through examples taken from our vulnerable .NET web application.

Description

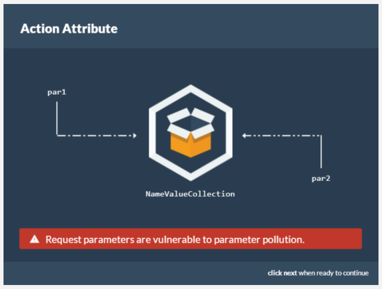
Understand Microsoft .NET 4.5 vulnerabilities common to the OWASP top 10, and see how these vulnerabilities affect .NET web applications. Students will learn to define and identify secure code, differentiate between secure coding methods, employ secure code in practice, and design and judge effectiveness of secure coding practices. This course will build upon high-level concepts in the OWASP Top 10 by deep diving into each concept from a developer's perspective and demonstrating insecure vs. secure code.

Audience

.NET developers
.NET architects

Time Required

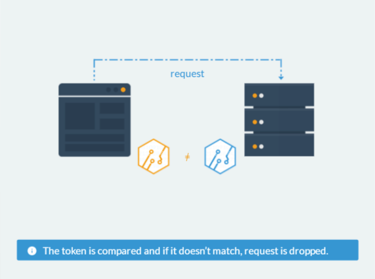
Tailored learning - 60 minutes total



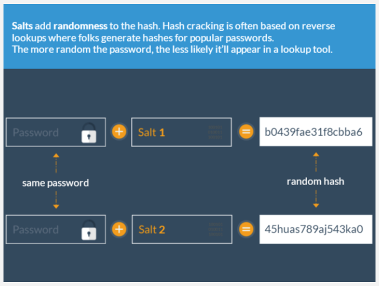
Action Attribute

⚠ Request parameters are vulnerable to parameter pollution.

click next when ready to continue



The token is compared and if it doesn't match, request is dropped.



Salts add **randomness** to the hash. Hash cracking is often based on reverse lookups where folks generate hashes for popular passwords.
The more random the password, the less likely it'll appear in a lookup tool.

NET201 - DEFENDING .NET FRAMEWORK

Course Outline

1. SQL injection

- About
- Retrieve data from DB
- Listing customers
- Query database customers
- SQL injection
- Bind Parameters
- How the query was changed
- Solution

2. Cross-site Scripting (XSS)

- About
- Normal redirection
- Code: The form
- XSS
- XSS exploited
- Escaping
- AntiXSS Encoder
- Importance of context
- Set default encoder
- Code: Identifying the output
- Solution

3. Authorization and session management

- About
- Container Based Authorization
- .NET Membership provider
- Code: Container based web.config
- By role
- By user
- Solution
- Session timeout
- Auth token timeout
- Solution
- Disable autocomplete
- Auth token timeout

4. Forced browsing

- Setting up the problem
- Downloading the file
- Creating download links
- Sending requested file
- Forced browsing
- Copying the URL
- Modifying the URL
- Indirect Access Maps
- Defense in action
- Create mapping for reports
- Get file from map
- Sending the file to user
- Indirect Mapping

5. Cross-site Request Forgery

- Problem: Admin message board
- Leave a message
- The message form
- The malicious form
- CSRF form
- Exploiting CSRF
- Anti-CSRF Tokens
- Use of tokens
- The form
- Generate AntiCSRF token
- Validate AntiCSRF tokens
- Solution

6. Insecure storage

- Problem: Storing passwords
- Problem: Encrypting data
- Salts
- Register an account
- Problem: Weak hashing
- Code: Weak MD5 hash
- Cracking the hash
- Create salt value
- Adding salt to hash
- Solution
- Password Based Key Derivation
- About PBKD
- Generate key from password
- Solution
- Encrypting data
- Encrypt data with passphrase
- Generating Key and IV
- Using AES
- Decryption

7. Unchecked redirects

- About
- Code: Message
- Code: Redirect
- Unchecked Redirect
- URL Mapping
- Using a whitelist
- Create URL mapping
- Redirecting from GUID value
- Solution

8. Security misconfiguration

- HTTP parameter pollution
- Debugging code and files
- Backdoor code
- Information leakage
- Action Attribute
- Don’t use Request.Params
- Input validation
- Disable and remove debug data
- ASP.NET Tracing
- ASP.NET Debug Mode

Security Compass