Web application security in Go

CYDGoWeb3d | 3 days | On-site or online | Hands-on

Your Web application written in Go works as intended, so you are done, right? But did you consider feeding in incorrect values? 16Gbs of data? A null? An apostrophe? Negative numbers, or specifically -1 or -2^31? Because that's what the bad guys will do – and the list is far from complete.

Handling security needs a healthy level of paranoia, and this is what this course provides: a strong emotional engagement by lots of hands-on labs and stories from real life, all to substantially improve code hygiene. Mistakes, consequences, and best practices are our blood, sweat and tears.

The curriculum goes through the common Web application security issues following the OWASP Top Ten but goes far beyond it both in coverage and the details.

So that you are prepared for the forces of the dark side.

So that nothing unexpected happens.

Nothing.

Skills and drills

24 LABS

10 CASE STUDIES

Audience

Go developers working on Web applications

Group size

12 participants

Preparedness

General Go and Web development

Outline

- Cyber security basics
- The OWASP Top Ten 2025
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Managing vulnerabilities in third party components
- Understanding Web application security issues
- Detailed analysis of the OWASP Top Ten elements
- Putting Web application security in the context of Go
- Going beyond the low hanging fruits
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in Go
- Input validation approaches and principles

Table of contents

- Web security configuration issues
  - Content Security Policy
  - Fetch directives
  - Source allowlisting
  - Strict CSP: using nonces and hashes
  - Navigation and reporting directives
  - Document restrictions and sandboxing
  - CSP best practices
- Cookie security
  - Cookie attributes
- Secrets management
  - Hard coded passwords
  - Best practices
- XML entities
  - DTD and the entities
  - Entity expansion
  - External Entity Attack (XXE)
    - File inclusion with external entities
    - Server-Side Request Forgery with external entities
    - *Case study – XXE vulnerability in Ivanti products*

A03 - Software Supply Chain Failures

- Using vulnerable components
- Untrusted functionality import
- Malicious Go packages
- Supply chain security and the Software Bill of Materials (SBOM)
- SBOM examples
- *Case study – The Polyfill.io supply chain attack*
- Vulnerability management
  - *Lab – Finding vulnerabilities in third-party components*
  - DevOps, the CI / CD build process and Software Composition Analysis
  - Dependency checking in Go
  - *Lab – Detecting vulnerable components*

Day 2

› The OWASP Top Ten 2025

A04 - Cryptographic Failures

- Cryptography for developers

- Cryptography basics
- Cryptography in Go
- Elementary algorithms
  - Hashing
  - Hashing basics
  - Hashing in Go
  - *Lab – Hashing in Go*
  - Random number generation
  - Pseudo random number generators (PRNGs)
  - Cryptographically secure PRNGs
  - Weak PRNGs
  - Using random numbers
  - *Lab – Using random numbers in Go*
- Confidentiality protection
  - Symmetric encryption
  - Block ciphers
  - Modes of operation
  - Modes of operation and IV – best practices
  - Symmetric encryption in Go
  - Handling IVs and block padding in Go
  - *Lab – Symmetric encryption in Go*
  - Asymmetric encryption
  - Combining symmetric and asymmetric algorithms

A05 - Injection

- Input validation
  - Input validation principles
  - What to validate – the attack surface
  - Where to validate – defense in depth
  - When to validate – validation vs transformations
- SQL injection
  - SQL injection basics
  - *Lab – SQL injection*
  - SQL injection best practices
    - Input validation
    - Parameterized queries
    - *Lab – Using prepared statements*
    - *Case study – Hacking Fortnite accounts*
- Code injection
  - OS command injection
    - *Lab – Command injection*
    - OS command injection best practices
    - Avoiding command injection with the right APIs
    - *Lab – Command injection best practices*

    - *Case study – Shellshock*
    - *Lab - Shellshock*
    - *Case study – Command injection in Gogs' built-in SSH server*
- HTML injection - Cross-site scripting (XSS)
  - Cross-site scripting basics
  - Cross-site scripting types
    - Persistent cross-site scripting
    - Reflected cross-site scripting
    - Client-side (DOM-based) cross-site scripting
  - *Lab – Stored XSS*
  - *Lab – Reflected XSS*
  - XSS protection best practices
    - Protection principles - escaping
    - *Lab – XSS fix / stored*
    - *Lab – XSS fix / reflected*
    - XSS protection in React

Day 3

› The OWASP Top Ten 2025

A06 - Insecure Design

- The STRIDE model of threats
- Secure design principles of Saltzer and Schroeder
  - Economy of mechanism
  - Fail-safe defaults
  - Complete mediation
  - Open design
  - Separation of privilege
  - Least privilege
  - Least common mechanism
  - Psychological acceptability
- Client-side security
  - Frame sandboxing
    - Cross-Frame Scripting (XFS) attacks
    - *Lab – Clickjacking*
    - Clickjacking beyond hijacking a click
    - Clickjacking protection best practices
    - *Lab – Using CSP to prevent clickjacking*

A07 - Authentication Failures

- Authentication
  - Authentication basics
  - Multi-factor authentication (MFA)
  - *Case study – The InfinityGauntlet attack*
  - Time-based One Time Passwords (TOTP)
- Session handling
  - Session management essentials
  - Why do we protect session IDs – Session hijacking
  - Session fixation
  - Session ID best practices
  - Session handling security considerations in single-page applications
- Password management
  - Storing account passwords
  - Password in transit
  - *Lab – Is just hashing passwords enough?*
  - Dictionary attacks and brute forcing
  - Salting
  - Adaptive hash functions for password storage
  - *Lab – Using adaptive hash functions in Go*
- Password policy
  - NIST authenticator requirements for memorized secrets
  - Password database migration

A08 - Software and Data Integrity Failures

- Integrity protection
  - Message Authentication Code (MAC)
    - Calculating HMAC in Go
    - *Lab – Calculating MAC in Go*
  - Digital signature
    - Digital signature with RSA
    - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
    - Digital signature in Go
    - *Lab – Digital signature with ECDSA in Go*
- Subresource integrity
  - Importing JavaScript
  - *Lab – Importing JavaScript*
  - *Case study – The British Airways data breach*

Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop

And now what?

- Software security sources and further reading
- Go resources