

# **FSI LEI 2025/2026**

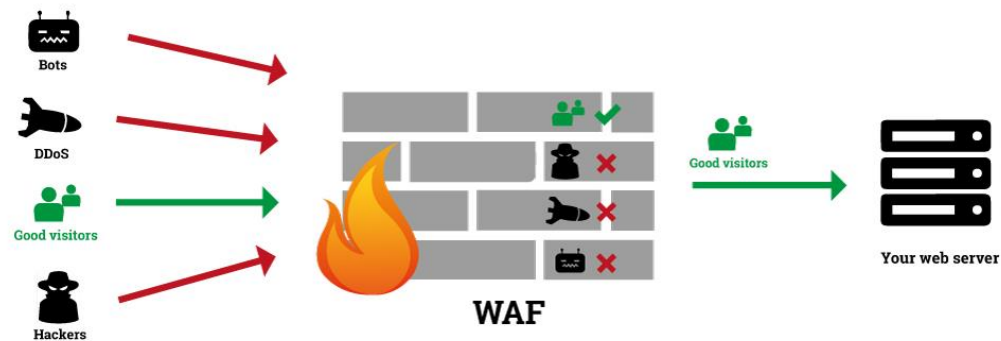
---

## **Practical class #8**

- **Web application firewall with ModSecurity**

# Web Application Firewall (WAF)

- A web application firewall (or WAF) filters, monitors, and blocks HTTP traffic to and from a web application
- A WAF is able to filter the content of specific web applications while regular firewalls serve as a safety gate between servers
- By inspecting HTTP traffic, it can prevent attacks stemming from web application security flaws, such as SQL injection, cross-site scripting (XSS), file inclusion, and security misconfigurations



# Web Application Security Consortium Project (WASC)

- WASC produces open source and widely agreed upon best-practice security standards for the World Wide Web
- Main WASC Projects:
  - Web Security Articles
  - The Web Hacking Incidents Database (WHID)
  - Web Application Security Scanner Evaluation Criteria
  - Distributed Open Proxy Honeypots
  - The Script Mapping Project
  - Web Security Glossary
  - WASC Threat Classification v2
  - Web Application Firewall Evaluation Criteria
  - Web Application Security Statistics

# WASC Threat Classification

- The Threat Classification is an effort to classify the weaknesses, and attacks that can lead to the compromise of a website, its data, or its users

Vulnerability	Design	Implementation	Deployment
<a href="#">Abuse of Functionality</a>	X		
<a href="#">Application Misconfiguration</a>		X	X
<a href="#">Brute Force</a>	X	X	
<a href="#">Buffer Overflow</a>		X	
<a href="#">Content Spoofing</a>		X	
<a href="#">Credential/Session Prediction</a>		X	
<a href="#">Cross-Site Scripting</a>		X	
<a href="#">Cross-Site Request Forgery</a>	X	X	
<a href="#">Denial of Service</a>	X	X	
<a href="#">Directory Indexing</a>			X
<a href="#">Format String</a>		X	
<a href="#">HTTP Response Smuggling</a>		X	
<a href="#">HTTP Response Splitting</a>		X	
<a href="#">HTTP Request Smuggling</a>		X	
<a href="#">HTTP Request Splitting</a>		X	
<a href="#">Integer Overflows</a>		X	
<a href="#">Improper Filesystem Permissions</a>		X	X
<a href="#">Improper Input Handling</a>		X	
<a href="#">Improper Output Handling</a>		X	
<a href="#">Information Leakage</a>	X	X	X
<a href="#">Insecure Indexing</a>		X	X
<a href="#">Insufficient Anti-automation</a>	X	X	
<a href="#">Insufficient Authentication</a>	X	X	
<a href="#">Insufficient Authorization</a>	X	X	

<a href="#">Insufficient Password Recovery</a>	X	X	
<a href="#">Insufficient Process Validation</a>	X	X	
<a href="#">Insufficient Session Expiration</a>	X	X	X
<a href="#">Insufficient Transport Layer Protection</a>	X	X	X
<a href="#">LDAP Injection</a>		X	
<a href="#">Mail Command Injection</a>		X	
<a href="#">Null Byte Injection</a>		X	
<a href="#">OS Commanding</a>		X	
<a href="#">Path Traversal</a>		X	
<a href="#">Predictable Resource Location</a>		X	X
<a href="#">Remote File Inclusion (RFI)</a>		X	X
<a href="#">Routing Detour</a>			X
<a href="#">Server Misconfiguration</a>			X
<a href="#">Session Fixation</a>		X	X
<a href="#">SQL Injection</a>		X	
<a href="#">URL Redirector Abuse</a>	X	X	
<a href="#">XPath Injection</a>		X	
<a href="#">XML Attribute Blowup</a>		X	
<a href="#">XML External Entities</a>		X	
<a href="#">XML Entity Expansion</a>		X	
<a href="#">XML Injection</a>		X	
<a href="#">XQuery Injection</a>		X	

# Web Application Firewall Evaluation Criteria (WAFEC)

- The purpose of WAFEC is to draw one's attention to the features that are of *potential importance* to a given project.
- Comprehensive list should be used as basis to form a much shorter list of features that are *required*
- Develop:
  - A set of web application firewall evaluation criteria
  - A testing methodology that can be used by any reasonably skilled technician to independently assess the quality of a WAF solution
- Categories:
  - Deployment Architecture
  - HTTP Support
  - Detection Techniques
  - Protection Techniques
  - Logging
  - Reporting
  - Management
  - Performance
  - XML

# Protection Techniques

- Brute Force Attacks Mitigation
- Cookie Protections Measures
- Session Attacks Mitigation
- Hidden Form Fields Protection
- Cryptographic URL and Parameter Protection
- Strict Request Flow Enforcement
- WASC Threat Classification v2.0:
  - <http://projects.webappsec.org/w/page/13246978/Threat%20Classification>

# WAF examples

## Appliance

- Barracuda Networks WAF
- Citrix Netscaler Application Firewall
- F5 BIG-IP Advanced WAF (formerly known as ASM)
- Fortinet FortiWeb
- Imperva SecureSphere
- Qualys WAF
- Radware AppWall
- Rohde & Schwarz Cybersecurity WAF

## Open-source

- **ModSecurity**

## Cloud

- Akamai Technologies Kona
- Alibaba Cloud
- Amazon Web Services AWS WAF
- Barracuda Networks CloudGen WAF and WAF-as-a-Service
- Cloudbric
- Cloudflare
- Fortinet FortiWeb
- F5 Silverline
- Fastly
- IBM Cloud Internet Services WAF
- Imperva Incapsula
- Microsoft Azure Application Gateway with WAF
- Oracle Cloud Infrastructure WAF
- Qualys WAF
- Radware
- Rohde & Schwarz Cybersecurity WAF
- Sucuri Firewall
- VMware NSX Advanced Load Balancer (formerly Avi Vantage)

# ModSecurity



- ModSecurity is a toolkit for real-time web application monitoring, logging, and access control
- Real-time application security monitoring and access control
- Full HTTP traffic logging
- Continuous passive security assessment
- Web application hardening

<https://github.com/SpiderLabs/ModSecurity>



# Deployment

- Options:
  - Embedded
  - Reverse proxy
- Modes:
  - Real-Time Monitoring and Attack Detection
  - Attack Prevention and Virtual Patching

# Phases for Rules

Phase number	Phase name	Phase occurs
1	REQUEST_HEADERS	Right after Apache has read the headers of the HTTP request.
2	REQUEST_BODY	After the request body has been read. Most ModSecurity rules are written to be processed in this phase.
3	RESPONSE_HEADERS	Right before the response headers are sent back to the client.
4	RESPONSE_BODY	Before the response body is sent back to client. Any processing of the response body to inspect for example data leaks should take place in this phase.
5	LOGGING	Right before logging takes place. At this point requests can no longer be blocked—all you can do is affect how logging is done.

Source: Brian Rectanus, *ModSecurity 2.5*, 2009

- Most common is phase 2

# OWASP ModSecurity Core Rule Set (CRS) Project



- ModSecurity is a web application firewall engine that provides very little protection on its own, it must be configured with rules
- OWASP ModSecurity Core Rule Set (CRS) Project is a set of generic attack detection rules for use with ModSecurity or compatible web application firewalls
- CRS aims to protect web applications from a wide range of attacks, including the OWASP Top Ten
- CRS provides protection against many common attack categories, including SQL Injection, Cross Site Scripting, Locale File Inclusion, etc.

<https://owasp.org/www-project-modsecurity-core-rule-set/>