**Module 14: Hacking Web Applications**

**Web Applications**

* Interface between end users and web servers
* Services: An application or software that is deployed over the internet and uses standard messaging protocols such as **SOAP, UDDI, WSDL**, and **REST** to enable communication
* Types of web services
  + **SOAP:** Based on the **XML format** and is used to transfer data between a service provider and requestor
  + **RESTful:** Based on **a set of constraints** using underlying HTTP concepts to improve performance
* Components of web service architecture
  + **UDDI: Universal Description, Discovery, and Integration** is a directory service that lists all the services available
  + **WSDL: Web Services Description Language** is an XML-based language that describes and traces web services
  + **WS-Security**: **Web sercices security** plans an important role in securing web services. It is an extension of SOAP and aims to maintain the integrity and confidentiality of SOAP messages as well as to authenticate users.

**Web Applications Threat**

* OWASP Top10
  + **Injection**
  + **Broken Authentication**
  + **Sensitive Data Exposure**
  + **XML External Entity**
  + **Broken Access Control**
  + **Security Misconfiguration**
  + **Cross-Site Scripting**
  + **Insecure Deserialization**
  + **Using Components with Known Vulnerabilities**
  + **Insufficient Logging and Monitoring**
* Injection
  + Allow untrusted data to be interpreted and executed as part of a cmd or query
  + **SQL injection:** bypass normal security measures and obtain access, executed from the address bar, applications fields…
  + **Cmd injection:** shell injection, html embedding (deface website virtually), file injection
  + **File injection**
  + **LDAP injection**
  + **Server-Side JS injection**
  + **Server-Side Includes Injectoin**
  + **Server-Side template injection**
  + **Log injection**
  + **HTML injection**
  + **CRLF (Carriage return line feed) injection:** inject carriage return (\n) and linefeed (\n) char into user input to trick a web server, web application, or user to terminate the input of a current object and initial a new object
* Broken Authentication
  + Session ID is URLs
  + Password Exploitation
  + Timeout Exploitation
* Sensitive Data Exposure
  + Poorly written encryption code
* XXE
  + A SSRF attack that occur when a misconfigured XML parser allows applications to parse XML input
* Broken Access Control
  + Security Misconfiguration
  + Unvalidated Inputs
  + Parameter/Form tampering
  + Improper error handling
  + Insufficient transport layer protection
* XSS
* Insecure Deserialization
* Using Components with Known Vulnerabilities
* Insufficient Logging and Monitoring
* Other Web application threats
  + **Directory Traversal:** Manipulate variables that reference files with “dot-dot-slash (../)” sequences and its variations.
  + **Unvalidated Redirects and Forwards**
  + **Watering hole attack**
  + **CSRF**
  + **Cookie/Session poisoning**
  + **Web service attacks**
  + **Cookie Snooping**
  + **Hidden Field Manipulation**
  + **Authentication Hijacking**
  + **Obfuscation appliiciation**
  + **Broken session management**
  + **Broken account management**
  + **DoS**
  + **Buffer Oveflow**
  + **CAPTCHA Attacks**
  + **Platform exploits**
  + **Network access attacks**
  + **DMS protocol attacks**
  + **Web-based timing attacks**
  + **MarioNet attack**
  + **RC4 NOMORE attack:** Against RC4 stream cipher
  + **Clickjacking atack**
  + **JS hijacking**
  + **DNS Rebinding attack:** Bypass the same-origin policy’s security constraints and communicate with or make arbitrary requests to local domains through a milicous web page

**Web Application Hacking Methodology**

* Footprint web infrastructure
  + **Server Discovery:** whois, dns interrogation, port scanning
  + **Service Discovery:** nmap, netscantools pro…
  + **Server Identification:** banner grabbing, telnet
  + **Detect web app firewalls and proxies:** trace method, WAFW00F
  + **Hidden Content Discovery:** OWASP ZAP, burpsuite
  + **Load Balancer Detection:** host, dig, lbd, Halberd
* Analyze Web Applications
  + **Identify entry points for user input:** User-agent, referer, accept, accept-language, host headers
  + **identify server-side technologies:** Error message, HTTP headers, HTML code
  + **identify server-side functionality:** GNU Wget, Teleport Pro, BlackWidow
  + **identify files and dirs:** Gobuster, Nmap nse script http-enum
  + **identify web application vulnerabilities:** Vega
  + **map the attack surface**
* Bypass Client-Side controls
  + **Attack Hidden Form Fields**
  + **Attack Browser Extensions**
  + **Perform Source Code Review**
  + **Evade XSS Filters:** Encoding char, Embedding whitespaces, Manipulate tags
* Attack Authentication Mechanism
  + **Username Enumeration**
  + **Password Attacks**
  + **Cookie Exploitation**
  + **Session Attacks**
  + **Bypass authentication:** SAML (Security Assertion Markup Language) messages are encrypted using base64 encoding
* Attack Authorization Schemes
  + URI
  + POST data
  + Query String and Cookie
  + Parameter Tampering
  + HTTP Headers
  + Hidden Tags
* Attack Access Controls
  + Exploiting insecure access controls
* Attack Session Management Mechanism
* Perform Injection Attacks/Input validation attacks
  + Web Scripts injection
  + OS cmd injection
  + SMTP injection
  + SQL injection
  + LDAP injection
  + XPath injection
  + Buffer Overflow
  + File injection
* LFI (Local File Inclusion): Enable attackers to add their own files on a server
* Attack Application Logic Flaws
* Attack Shared Environments
* Attack Database Connectivity
  + **Connection String injection:** Inject para in a connection string by appending them with **semicolon (;)** char
  + **CSPP (Connection String Parameter Pollution) Attacks:** Overwrite para values in the connection string to steal user IDs to hijack web credentials
  + **Connection Pool DoS:** Construct a large malicious SQL query
* Attack Web App Client
  + XSS
  + HTTP header injection
  + Request Forgery Attack
  + Privacy Attack
  + Redirection attacks
  + Frame injection
  + Session fixation
  + ActiveX attacks
* Atack Web Services
  + Probing Attacks
  + SOAP Injection, similar to sql injection
  + **SOAPAction Spoofing**: SOAPActoin is an additional HTTP header used when SOAP messages are transmitted using HTTP. **WS-Attacker** can be used to manipulate the operations included in the SOAPAction headers.
  + **WS-Address Spoofing**: WS-address provides additional routing info in the SOAP meader to support **asynchronous conmmunication**. Attackers send a SOAP message containing fake WS-address info to the server
  + **XML Injection**
  + **Parsing Attacks:** DoS attack or logical errors in web service request processing
  + **Tools:** SoapUI Pro, XMLSpy

**WEB Service APIs**

* **SOAP API:** Enable interactions between applications running on different platforms.
* **REST (Representation State Transfer) API :** An architectural style for web services that serves as a communication medium between various systems on the web
* **RESTful API:** Known as RESTful services, are designed using REST principles and HTTP communication protocols. A collection of resources that use HTTP methods such as PUT, POST, GET, and DELETE
* **XML-RPC**
* **JSON-RPC**

**Webhooks**

* **User-defined HTTP callback** or push APIs that are raised based on events triggered
* Allow applications to **update other appliucations** with the latest info
* Are enrolled along with the **domain registration** via user interface or API to inform clients

**OWASP Top10 API Security Risks**

* Broken Object Level Authorization
* Broken User Authentication
* Excessive Data Exposure
* Lack of Resources and Rate Limiting
* Broken Function Level Authorization
* Mass Assignment
* Security Misconfiguration
* Injection
* Improper Assets Management
* Insufficient Logging and Monitoring

**API Vulnerabilities**

* Enumerated resources
* Sharing resources via Unsigned URLs
* Vulnerabilities in Third-Party Libs
* Improper Use of CORS
* Code Injections
* RBAC (Role-based access control) Privilege Escalation
* No ABAC (Attribute-based access control) Validation
* Business Logic Flaws

**WEB API hacking methodology**

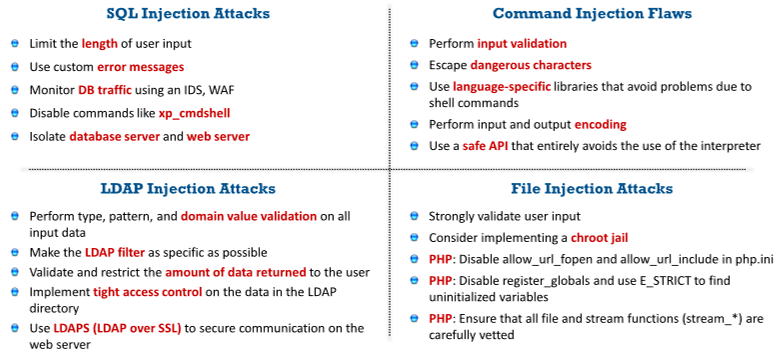
* Identify the target
  + SOAP and REST mostly use HTTP protocols
  + JSON for REST API, XML for SOAP API
* Detect security standards
  + SOAP and REST implement different authentication/authorization standards such as **OpenID Connect, SAML**, OAuth 1.x and 2.x, and WS-Security
  + SSL **only encrypt sensitive user data**
* Identify the attack surface
  + API metadata reveals a lot of technical info such as paths, parameters, and message formats
  + Attacker monitors and records the communication between the API and a client to identify an initial attack surface
* Launch attacks
  + Fuzzing
  + Invalid input attacks
  + Malicious input attacks
  + Injection attacks
  + Insecure SSL configuration
  + IDOR (Insecure direct object references)
  + Insecure session/Authentication handling
  + Login/ Credential Stuffing Attacks
  + API DDOS attacks
  + Authorization attacks on API:
    - OAuth is an authorization protocol that allows a user to **grant limited access** to their resources on a site to a different site without having to expose their credentials
  + reverse engineering
  + user spoofing
  + MITM attack
  + Session replay attacks
* REST API Vulnerabilitiy Scanning: **Astra, OWASP ZAP**
* Bypass IDOR via parameter pollution

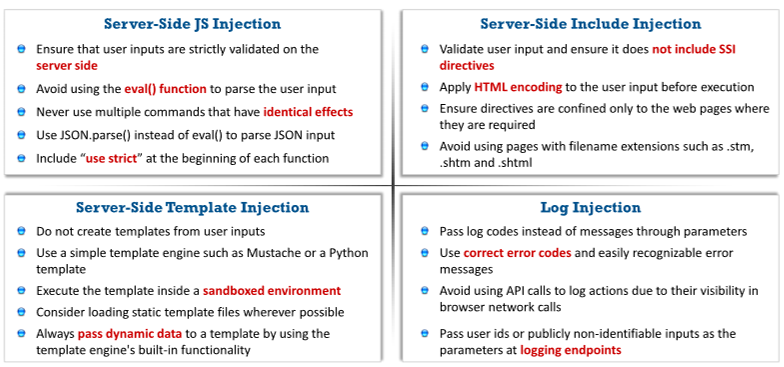
**Web Shells**

* A malicious piece of code or script that is developed using **server-side languages** such as PHP, ASP, PERL, RUBY, and Python and are then installed on a target server
* Attackers **inject malicious script** by exploiting most common vulnerabilities such as remote file inclusion (RFI), local file inclusion (LFI), exposition of administration interfaces, and SQL injection.
* Tools: WSO php webshell
* Gain backdoor access:
  + Attackers exploit **non-validated file uploads** to inject malicious script in a target webserver to gain backdoor access
  + Use tool such as **Weevely** to gain backdoor access to a website without being traced
  + Weevely also helps attackers in performing administrative tasks, maintaining persistence, and spreading backdoors across the target network
* Web Shell Detection Tools: **Web shell detector**

**Web Application Securtity Testing**

* **Manual web app security testing**
* **Automated web app security testing**
* **SAST (Static application security testing):** white-box
* **DAST (Dynamic application security testing):** black-box
* **Fuzz testing:** a black-box testing method. Huge amounts of **random data** will be generated and used against…
* **Source code review**
* **Encoding schemes:** URL encoding, HTML encoding, unicode encoding, base64 encoding, hex encoding
* **Defend against injection**





* RASP for protecting web servers: **Runtime application self protection** can detect runtime attacks.
* Testing tools: Acunetix **WVS, N-Stalker Web App Security Scanner**
* **Firewalls:** dotDefender