[PENETRATION TESTING & APPSEC FUNDAMENTALS]

OWASP Foundation
* Open Worldwide Application Security Project
* nonprofit foundation that works to improve software security

Resources
* OWASP Cheat Sheet Series
  cheatsheets.owasp.org
* OWASP Web Security Testing Guide (WSTG)
  owasp.org/www-project-web-security-testing-guide/latest/
* OWASP App. Sec. Verification Std. (ASVS)
  owasp.org/www-project-application-security-verification-standard/
* OWASP Mobile App. Security Project (MASVS, MASTG) mas.owasp.org
* OWASP Projects owasp.org/projects
* OWASP Global & Regional Events
  owasp.org/events/
* OWASP Top Ten owasp.org/Top10/
  ↳ ≠ a methodology, but for awareness abt web vulns.

github.com/OWASP/DevGuide/blob/master/02-Design/01-Principles%20of%20Security%20Engineering

Secure Design Principles

Defence-in-Depth
If 1 ctrl would be reasonable, more ctrls that approach risks in diff. fashions are better
* If used in depth, ctrls can make severe vulns. extraordinarily difficult to exploit ⇒ vulns. unlikely to occur
ex., VPN, strong pswd policy, MFA

Fail Safe
default to a secure state during design or implementation failure
* secure state: "Fail Safe Defaults"
* unless subject given explicit access to object, deny access
ex. implementation:
↳ exception handling

Economy of Mechanisms
keep sys. design as simple & small as possible
* KISS principle
* avoid using complex architectures for simpler approaches

Least Privilege
person\process given only min. level of access rights (ie., privileges) necessary for assigned op.
* limits damage in case of exploited vuln.
* proper granularity of privileges & permissions should be established

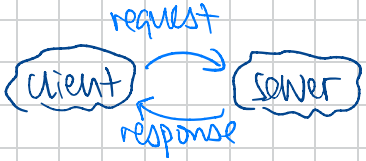
Separation of Duties
authorization concept; require >1 person to complete task
* maker (create transaction)
  checker (authorize transaction)
* key fraud & error ctrl
ex., maker\checker roles in order sys. as part of Role Based Access Ctrl

Penetration Testing
security exercise where cybersec expert attempts to find & exploit vulns. in comsys
* purpose: identify weak spots in system's defenses which attkers can exploit
* types
  ↳ white box: testing sys. w full knowledge & access to all src code & arch. docs.
  ↳ grey box: testing sys. w some knowledge of sys. internals (login creds. given)
  ↳ black box: testing sys. w/out knowledge of internal workings of sys., src code, arch, or login creds.

Burp Suite "swiss army knife" for app. pen testing
* Burp Repeater
  ↳ enables modding & sending HTTP msgs repeatedly
  ↳ purposes:
  → send request w varying param. vals. to test for input-based vulns.
  → send series of HTTP requests in specific seq. to test for vulns. in multi-step processes or vulns. relying on manipulating conn. state
* Burp Intruder
  ↳ tool for automating customized atks against web apps
  ↳ enables configuring atks that send same HTTP requests repeatedly, inserting diff. payloads into predefined positions each time
  ↳ purposes:
  → [fuzzing] for input vulns.
    ↳ automated software testing method that injects invalid, malformed, or unexpected inputs into sys. to reveal software defects & vulns.
  → perform [brute-force atks] → involves "guessing" uname & pswds to gain unauthorized access to sys.
  → enumerate valid identifiers & other inputs
    ↳ enumeration: identify info abt in-scope assets
  → info gathering (harvesting) of useful data

HyperText Transfer Protocol (HTTP)

* stateless protocol
* uses port 80 (HTTP – plain text) or port 443 (HTTPS – encrypted)
* communication btwn host & client occurs via request/response pair



Request Methods → TRACE, OPTIONS, etc.

GET Request

retrieve data fm webserver to view it

```
GET / HTTP/1.1
Host: www.ntu.edu.sg
User-Agent: Firefox/97.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Request Line; Request Headers; Request Message Header

POST Request

sends data to webserver to process (ex., value keyed into form)

```
POST / HTTP/1.1
Host: www.ntu.edu.sg
User-Agent: Firefox/97.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

name=foo&city=bar
```

Request Line; Request Headers; Request Message Header; A blank line separates headers and body; Request Message Body

Response Methods

∀ HTTP request, a response is sent fm server to user (browser)

```
HTTP/1.1 200 OK
Date: Sun, 17 Sep 2022 02:37:43 GMT
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Server: gws
…
Content-Length: 202101

<!doctype html><html …
```

Status Line; Response Headers; Response Message Header; A blank line separates headers and body; Response Message Body (e.g. HTML that is being rendered in the browser)

* 5 Classes:
  - ↳ Informational Responses (100–199)
  - ↳ Successful Responses (200–299)
  - ↳ Redirection Msgs (300–399)
  - ↳ Client Error Responses (400–499)
  - ↳ Server Error Responses (500–599)

Website Types

- ↳ static website
  webpages delivered exactly as they are stored in ∅ real-time content changes
- ↳ dynamic website
  generates content in real-time, typically using db languages & scripting languages to provide interactivity & personalized XP

Ways to Attack Dynamic Websites

* injection attks (SQLi, XSS, HTML, etc.)
* "spamming" by sending requests multiple times & polluting db
* unauthenticated access
* vuln. in lib(s) being used
& more, depending on implementation & business logic!

Same-Origin Policy (SOP)

URLs have same origin if same → protocol, port (if specified), host

Checks against the URL http://www.example.com/dir/page.html

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same protocol, host and port
http://www.example.com/dir2/other.html	Success	Same protocol, host and port
http://username:password@**www.example.com**/dir2/other.html	Success	Same protocol, host and port
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)
http://www.example.com:**80**/dir/other.html	Depends	Port explicit. Depends on implementation in browser.

Protection Against Username/Password Enumeration

* multifactor authentication (MFA)
* login throttling
* CAPTCHA
* acct. lockout
* strong pswd policy

Mitigations (Client Side)

* usage of HTTPS (HTTP via TLS)
* block "mixed content" (HTTP & HTTPS on same webpage)
* identify weak signature algos. of certs
* same-origin policy (SOP)