

1) CSRF : (WSTG - SECC - 05)

- forces an user to do an action unknowingly or unwantedly.

$\xrightarrow{\text{XSS} \rightarrow \text{dynamic}}$
 $\xrightarrow{\text{CSRF} \rightarrow \text{static}}$

2) XSS vs CSRF:

- XSS = uses script CSRF = uses static content
- XSS ≠ CSS, as CSS = Cascading Style Sheets.

3) CSRF Visualization: I Know.

- Transaction that does not req: dynamic content
- submitted without CSRF tokens / mechanisms
- weak tokens / predictable tokens.

4) ZAP Anti CSRF Test form:

- POC to find CSRF flaws using POST reqs.

→ "Generate Anti CSRF forms"

- Similar to BURP → Generate CSRF POC (PRO only)

→ Day 5: Part 2: CSRF exercise

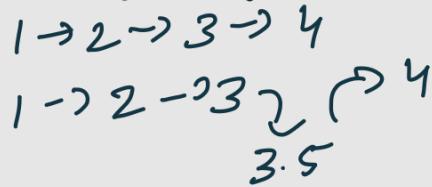
→ Day 5: Part 3: Logic Attacks

- Logic Flaws = Business logic bypass
- Not prominent in industry. (not prominently discussed)

• DIFFICULT TO RESOLVE:

- Every app has many logic flaws and many ways to bypass the logic content
- Depends on application CONTEXT

2) Flow of Logic Flaws:



3) Logic Flaws : Manual Discovery:

- Automated Tool → logic flaw → unnoticed
- ∴ pronounced in auto scanners
- Manual Discovery = prominent + most effective.
- OWASP TG = infinite combos = How to Proceed ?

WSTG - BVSL-01
WSTF - BVSL-02
⋮
09

Day 5 : Part 4: Python for Web App Pentest:

- Python:
 - ↳ Interpreted (GUIDO VAN ROSSUM)
 - 1980's (Monty Python's Flying Circus)
 - (Glue or Scripting stuff)
 - ↳ Public Release → Feb 1991
 - ↳ derived from ABC language
 - (RAD)
 - RAD (Rapid App development)
 - why Python?
 - RAD (Rapid App development)
 - easy
 - Right tool for the job
 - Many libs, many integrations
- But, always be TOOL AGNOSTIC
- Python Availability (python -v) (usually default on Linux / Mac)
 - Python 2 vs Python 3
 - (Legacy)
 - (Future and present of language)
 - Python = simple to use (STUPS)

② Why to use Python when so many tools?

- Customize to you
- use anytime, anywhere as personally needed.

③ Learn Python Basics

④ Python Data Types:

- String
- Boolean
- ↳ int
- ↳ float

• if / elif / else

- ⑤ Python Loops:
- ⑥ Python Lists + Dictionaries:
- ⑦ Python Web Libs:
 - ↳ urllib, urllib2,3
 - ↳ httpLib,2 , Requests
- Eg: of HTTPLibs in Python coding.
- ⑧ POST via Requests
Requests + SSL/TLS (r = requests.get('https://xyz.com'))
- ⑨ Putting (a lot of) Together
(splitlines() removes the newline at end of each line) (./timing.py | sort -n | tail -10)

Day 5: Part 5: Python [EXERCISE]

Day 5: Part 6: WPScan + Exploit DB

- ① WPScan() → word press scanner (automated)
most tools = General or Manual (not specific)
WPScan = tool SPECIFIC for word press
 - ↳ enumerating WP flaws / vulns
 - ↳ WP is very ubiquitous (public / internal) sites
- Check videos for WPScan tool and cmd's.
- ↳ Actively updated; VULN scanner for Wordpress

Day 5: Part 7: WPScan + Exploit DB

[EXERCISE]

Day 5: Part 8: Burp Scanner [PRO]

- Burp Scanner (Automated scanner, only in PRO)
- Caveat Emptor
- Passive: what is present already. No extra req
- Active: makes new req and gathers responses
- Live scanning = Passive + Active (Active scan while traversing)
 - Always stay in scope.
 - Active Scans → can now run multiple scans in parallel (not one MONOLITHIC scan)
 - Allows ↑ efficiency (as it's better than a single large scan, huge)
 - Advanced Scope Control (What(not) to SCAN)
 - Scanning Particular Nodes.
 - Launch separate
 - Add to task and scan (if found a new branch)
 - Burp Scanner confidence levels
 - certain
 - Firm
 - Tentative
 - Info
 - Burp (Retesting + Remediation)
 - open scan
 - check the item to be retested
 - Right click → Audit item (to retest)

Day 5: Part 9: Metasploit (Tool)

- Most popular → exploitation framework
- Made in RUBY (largest RUBY ever)
- Both O/source + commercial dev available.

② MS + Web Testing (N/w well known, but also has WEB functions).

③ Auxiliary → perform functions like scanning, spidering, crawling and querying basic web server. (aux/scanner/http)

> 150 entries in A/scanner/http
→ A/c / nef crawler } 2 scanners
④ MS crawlers : → A/s / http / crawler

crawl / scan is great but can't replace Burp/ZAP

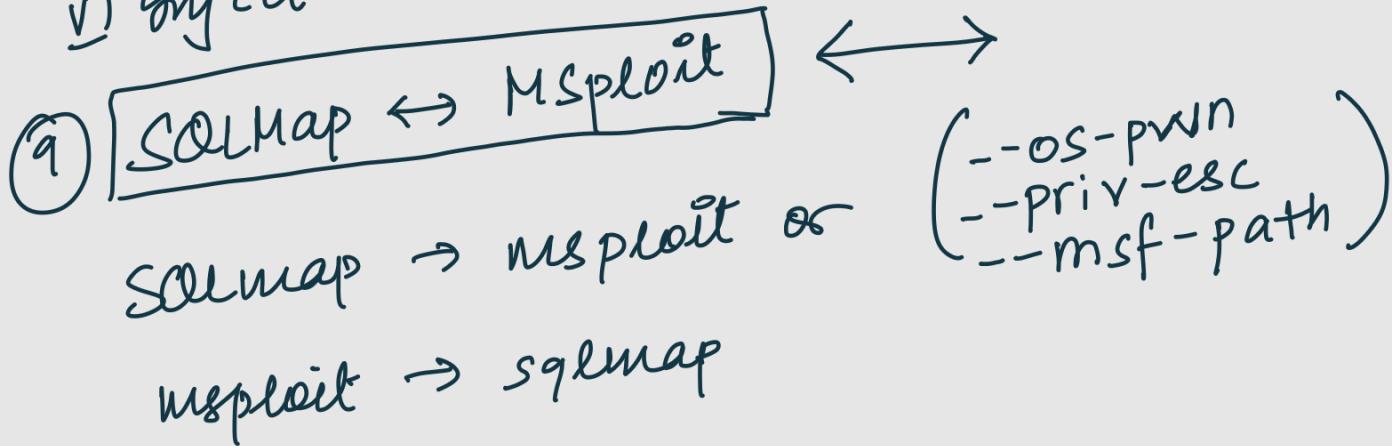
⑤ DB Import : (used to import other tools)

→ most efficient way to prime MS = MS + db-import.
→ Tools output files $\xrightarrow{\text{db-imp}}$ compatible DB for MS
→ hence external tools \Rightarrow MS, due to DB-imp

⑥ MS Integration:

① DB-Import : Any tool with Msplint

- ⑦ Beef + MSexploit:
 hooked Browser + Metasploit = better, faster exploits.
 o expands capabilities for XSS.
- ⑧ Beef + MSexploit: → Lim privs → persistence
- Configure stuff. (beef/config.yaml).
 - In msexploit check for msf\$username + password.
 - i) Update config file
 - ii) Start MS RPC using msfconsole =>
`load msgrpc ServerHost=127.0.0.1 Pass=pass`
 - iii) Update config.yaml in ext/msexploit with details of the RPC started
 - iv) Start BeEF
 - v) inject some MSexploit.



- ⑩ MSexploit + Known Vulns:
- o Main use → Known Vulns
 - o custom app testing = aux modules.
- Examples: CMS, DB's, SQLi, Shellshock, Heartbleed, Drupalgeddon

⑪ Drupal: (like wp) (serve content to end users)

wp and Drupal: Commonly deployed CMS.

Joomla (3rd)

WP Plugins } extend functionality
Drupal Modules }

⑫ Drupalgeddon :

unauth SQLi on all Drupal 7 systems
(Oct 15, 2014) (11pm UTC) (assume all systems
are hacked)

→ unfiltered data access
→ RCE
→ Local Privesc etc.

widespread auto exploits (in hours)

Drupalgeddon Details: (Drupal used Prepstmt() for
SQLi)

Input → expand Arguments() → PDO → Emulate Prep
(array → objects compatible)

stmt → emulated PS allows multiple queries as
1 statement.

∴ any Input → expand Arguments() = EXPLOIT!!

MSploit: multi/http/drupal-drupageddon

e.g.: use exploit/multi/http/drupal-drupageddon.

Day 5: Part 10: MSploit: Exercise

Day 5: Part 11: Drupaleddon 2: exercise

Day 5: Part 12: When Tools Fail

- what to do when tools fail
 - ↳ Diff in server configs
 - ↳ tool quality
 - ↳ some options can/cant
 - ↳ Pentest \Rightarrow Non-Determine

• Taking it to the Next Level: (Abandon exploitation?
NO)

Learn all about MediaWiki

→ Djvu (didn't work; default)

MSploit → PDF images (worked; not default,
specify the file name to MSploit)
(exploit/multi/http/mediawiki-thumb)

Day 5: Part 13: Exercise When Tools Fail

Day 5: Part 14: Business of Pen testing:

Prepping

- ① Skills, Toolkit → Pentest Specifics
(Personal Prep) → Comm'n Plans

Skills → Build skills; Stay informed; Join chapters.
Practices, (Community Pnv)

Toolkit →
 - Tools; Backup Test Results
 - Custom scripts
 - Vuln Scanners

Pre-engagement specifics: (Skills, Toolkit, Backup)

- PIE - engagement

 - Permission ◦ Scope ◦ Rules of Engagement.
 - Source details ◦ Source Code (WBOA) ◦ Filtering
 - Test Accts ◦ Backups ◦ Architecture Diagrams.
 - Green zones
 - Include who all in Emails

③ Commⁿ Plans: → Doc frequencies, date/time of standups meetings

↳ ETA of final report.
(Set a status expectation)

④ Additional commⁿ Plan:

- Begin scheduling final briefing. At when stakeholders can be present for meet.
- What are acceptable commⁿ for sensitive / confid data as such. (email, phone, fileshare etc).

Day 5: Part 15: Post Assessment

- Assessment finished → Results communicated
 - Reports → XL sheets → any format docs
 - RESULTS → XL sheets → Portals (Aocher)

- ① Reports: I know. (well written, informative, Reproducible)
- ↳ Exec Summary
 - ↳ Test Params
 - ↳ Findings.
- } can use 1 or more appendix to incl. addl. info.

- ① EXEC Summary : High level summary for exec's to understand.
- ② Test Params : Project Obj, Scope, Schedule, Targets, Test name, URL, Limitations, Finding summary, Remediation summary.
- ③ Findings : Tech details. What we do. I know.

- ④ Report Appendix:
- ↳ Finding affected many params / URL.
 - ↳ enumerated uname / pass words
 - ↳ Document Authorization Letters
 - ↳ CMD output that's very long.
 - ↳ Data exfil during exploit.

- ⑤ Report Automation: (Templates integrated into WORD)

- Fast ◦ efficient ◦ consistent ↑ Quality
- ↓ time reporting ↑ time hacking.

⑥ Debrief Presentation:

- Debrief PPT = Pentesters can talk to app team and stakeholders (interactive discussion)
(multiple interest)
- As multiple groups = multiple intellects =
multiple presentations by pentest