STUDENT HANDOUT · WEEK 5

Vulnerability Types Across the Enterprise

SY0-701 objectives 2.3. Notes, exercises, review questions, and the lab.

Name	
Date	

THIS WEEK YOU WILL

1. Explain memory injection, buffer overflow, race conditions, and malicious updates.
2. Distinguish SQL injection from cross-site scripting and name the main fix for each.
3. Explain why end-of-life, legacy, and firmware weaknesses persist and how to contain them.
4. Describe VM escape, resource reuse, and common cloud-specific vulnerabilities.
5. Identify supply chain, cryptographic, and misconfiguration weaknesses in a scenario.
6. Explain side loading, jailbreaking, and zero-day vulnerabilities and their controls.
7. Prioritize findings using CVE, CVSS, and real exposure.

KEY IDEAS

Application flaws make code misbehave

Memory injection	Attacker code is written into a running process's memory. (DLL injection)
Buffer overflow	Input larger than its buffer overwrites nearby memory. (Overwritten return address)
Race condition	State changes between the time of check and time of use. (TOC/TOU)
Malicious update	A trusted update channel delivers attacker code. (SolarWinds Orion, 2020)

SQL injection and XSS trust user input

Flaw	What goes wrong	Main fix
SQL injection	User input runs as part of a database query	Parameterized queries
Stored XSS	Script saved on the site runs for every visitor	Output encoding, input validation
Reflected XSS	Script in a link is echoed back in the response	Output encoding, Content Security Policy
DOM-based XSS	The page's own JavaScript writes input unsafely	Safe DOM methods, no innerHTML with input

Old systems and firmware outlive their patches

Operating system	Hardware
Unpatched kernel and service flaws	Firmware flaws in BIOS, UEFI, and device controllers
Privilege escalation bugs turn a user into an admin	End-of-life: the vendor stops selling, then supporting
Unneeded services left listening	Legacy: old gear still in use, often unpatchable
Fixed by vendor patches on a regular cycle	Fixed by firmware updates, isolation, or replacement

Shared hardware creates shared weaknesses

Vulnerability	What happens	Defense
VM escape	Code in a guest breaks out to the hypervisor	Patch hypervisors, limit guest features
Resource reuse	Memory or disk from one VM is given to another	Clear memory and storage before reuse
Cloud misconfiguration	Storage or services exposed to the internet	Configuration scanning, least privilege
Weak cloud identity	Overprivileged keys, accounts without MFA	MFA, short-lived keys, access reviews
Insecure APIs	Management API lacks auth or rate limits	Authenticate, authorize, and log each call

You inherit the weaknesses of your suppliers

Service provider	An MSP or vendor with remote access is breached first. (Kaseya VSA, 2021)
Hardware provider	Counterfeit or tampered parts ship with flaws built in. (Gray-market network gear)

Software provider	Vendor code or a library you depend on is flawed or poisoned. (Log4Shell in Log4j, 2021)

Weak crypto and bad settings undo good tools

Cryptographic	Misconfiguration
Deprecated algorithms: DES, RC4, MD5, SHA-1	Default usernames and passwords left in place
Short keys or old protocols such as SSL and TLS 1.0	Storage buckets or shares open to everyone
Keys stored next to the data they protect	Unneeded ports, services, and sample files
Flawed libraries, such as Heartbleed in OpenSSL	Excess permissions and verbose error pages

Mobile devices and flaws no one has patched

Vulnerability	What it means	Control
Side loading	Installing apps from outside the official store	MDM policy that blocks unknown sources
Jailbreaking	Removing iOS restrictions to gain root access	MDM jailbreak detection, deny access
Rooting	The Android equivalent of jailbreaking	Compliance check before email access
Zero-day	A flaw exploited before a vendor patch exists	Defense in depth, behavior-based detection

Rank findings by severity and exposure

9.0 to 10	CVSS critical range
7.0 to 8.9	CVSS high range
CVE	One ID per public flaw
KEV	CISA list of exploited flaws
	Score is not priority: a 7.5 on an internet-facing server can outrank a 9.8 on an isolated lab box.

EXERCISE 1: NAME THE VULNERABILITY

8 MINUTES · PAIRS

For each scenario, name the vulnerability type from objective 2.3 and one control that would reduce it.

#	Prompt	Your answer
1	A login form accepts ' OR 1=1 -- and logs the user in as admin	
2	Two transfers sent at once both pass the balance check	
3	A forum post runs a script in every reader's browser	
4	A clinic X-ray PC runs an OS the vendor no longer patches	
5	A new VM finds fragments of another tenant's data on its disk	
6	A signed vendor update installs a backdoor at 300 customers	
7	A staff phone installs an app from a website, not the store	
8	Attackers exploit a flaw the vendor has never heard of	

EXERCISE 2: FIND THE CLINIC'S VULNERABILITIES

7 MINUTES · SMALL GROUPS

For each Lakeside item, name the vulnerability type, rate it high, medium, or low, and pick the first fix.

#	Prompt	Your answer
1	Front desk PCs still on Windows 10, past end of support	
2	Patient portal search box that builds SQL from user text	
3	Wi-Fi router with the admin password printed on its label	
4	Backups encrypted with a DES key hardcoded in a script	
5	Billing vendor with an always-on remote access tool	
6	Physician's jailbroken phone that syncs clinic email	
7	Ultrasound cart the maker says can never be patched	

REVIEW QUESTIONS

1. A program checks a file's permissions, then opens it, and an attacker swaps the file in between. What is this?

2. What is the main fix for SQL injection?

3. Why is an end-of-life system a vulnerability even if it has no known flaws today?

4. What is resource reuse in a virtualized or cloud environment?

5. How does a vulnerability's priority differ from its CVSS severity?

KEY TAKEAWAYS

- **Name the type, find the fix.** SQLi points to code, a default password to settings, EOL to the budget.
- **Input is not code.** Parameterized queries stop SQLi; output encoding stops XSS.
- **Contain what you cannot patch.** Isolate EOL, legacy, and firmware-bound devices and watch them.
- **Severity is not priority.** Add exposure, asset value, and KEV status to the CVSS score.

LAB 5: DISCOVERY AND VULNERABILITY SCANNING WITH NMAP AND GREENBONE

60 MINUTES

You need the week 1 lab plus a scanner VM. Before class, build a Kali Linux VM on the NAT adapter, run sudo apt update && sudo apt install -y gvm and then sudo gvm-setup (the feed download can take an hour or more), then move it to your host-only network; your week 1 Ubuntu and Windows VMs are the targets. Scan only hosts on your own lab network. Scanning systems you do not own, or lack written permission to test, can break the law, including the Computer Fraud and Abuse Act, and school and employer networks are off limits.

Steps

1. Confirm isolation: set every lab VM to the host-only network in your hypervisor, then run ip addr on the scanner VM and write down its address and subnet, for example 192.168.56.0/24.
2. Discover live hosts with a ping sweep: nmap -sn 192.168.56.0/24. Check that every address belongs to a VM you own, and stop if anything unknown appears.
3. Scan one target for open ports, service versions, and the OS: sudo nmap -sS -sV -O 192.168.56.101 -oN target1.txt (use your target's address).
4. Scan every TCP port on the same host: sudo nmap -p- 192.168.56.101. Note any open port the first scan missed, since a default scan checks only the 1,000 most common ports.
5. Run Nmap's vulnerability scripts against your own target: sudo nmap -sV --script vuln 192.168.56.101. Record each CVE it reports.
6. Start Greenbone (sudo gvm-start on Kali), sign in at https://127.0.0.1:9392, and wait until the feed status shows the feeds are current. Create a target for your lab hosts and a task that uses the Full and fast scan configuration.
7. Run the task, open the report, and sort by severity. For the top three findings, record the name, CVSS score, port, CVE if listed, and the suggested solution.
8. Map each of the three findings to a vulnerability type from this week, and mark any you suspect is a false positive, with your reason.

Hand in

A two to three page lab report with screenshots of the Nmap discovery and service scans and the Greenbone report summary, a table of your top three findings with their CVSS scores and vulnerability types, and answers to the lab questions.

Questions to answer in your report

1. Which open services did both Nmap and Greenbone flag, and which did only one tool report? Why might they differ?
2. Greenbone ran an unauthenticated scan. What could an authenticated scan with credentials see that this one could not?

3. Pick one finding. Given its CVSS score and exposure, would you rank it higher or lower if this host were Lakeside's patient portal server? Explain.
4. Which finding is most likely a false positive, and how would you confirm or rule it out?