Cross-Chapter CVE-Class Vocabulary Reference

178 words · ~1 min read

---

*Cross-chapter shared reference for `vca-sec-101` (Foundations / Defense) + `vca-pen-101` (Penetration Testing) + `vca-adv-101` (Adversarial Capstone) + `vca-adv-102` (AI & Agentic Security) + `vca-re-101` (Reverse Engineering, primary) + `vca-re-011` (RE Foundations) + downstream RE-201 / RE-301. html` CVE-Class Vocabulary Companion sidebar landed firing #315). *

Purpose. Single canonical academy-wide CVE-class taxonomy that cyber-track courses cite by name rather than reproducing in each course's own materials. Belt-2 through Belt-5 graduates need to **recognize the families when they read disclosure write-ups** and **name the primitives when they discuss exploitation strategy with colleagues**, even if they have personally reproduced only one or two CVEs end-to-end through their academy capstone work. This handout supplies the vocabulary scaffold; per-course labs supply the methodology.

Print and pin during SEC-101 Module 4 (Vulnerability Landscape), PEN-101 Week 6 (CVE-driven exploitation set), ADV-101 Week 1 (Belt-5 vocabulary onboarding), ADV-102 Week 2 (CVE landscape adjacent to LLM-OWASP), RE-101 Module 7 (CVE class reading list), RE-011 Module 3 (foundational disclosure-vocabulary onramp).

---

§1: Purpose and scope

This handout is the academy's **vocabulary-tier set** for the modern CVE landscape: a deliberately taxonomic reference that names families, pairs each family with a representative CWE, anchors each family on a canonical publicly-disclosed CVE, and supplies one paragraph of teaching context per row. It is **not** a methodology reference (per-track curricula supply that), **not** an exploit-development handbook (RE-201 + Part-II RE electives + post-academy work like OSEE / GXPN cover that), and **not** a how-to-discover-vulnerabilities reference (the lab-target curricula across the cyber track teach that against intentionally-vulnerable harnesses under `--authorized-by` discipline).

The scope this handout covers. Three families that together account for the majority of disclosed-and-exploited vulnerabilities students will encounter during their academy career and beyond: **memory-corruption + parser bugs**, **injection + deserialization**, and **auth-bypass + supply-chain**. The three families are not exhaustive (race conditions, cryptographic flaws, side channels, and protocol-level design defects each warrant their own future taxonomy handouts), but they cover the classes graduates will most often encounter when reading vendor PSIRT advisories, CISA KEV catalog entries, and conference-talk write-ups during their first decade of professional practice.

The vocabulary-tier vs methodology-tier distinction. A vocabulary-tier handout teaches students to **recognize and name** what they encounter. A methodology-tier handout teaches students to **reproduce, exploit, or discover** the bug class. The two tiers serve different pedagogical purposes and should not be conflated. ADV-101's primary teaching anchor is one CVE (Longenecker SB6141 CSRF) reproduced end-to-end at methodology depth; this handout's ~22 named CVEs and ~5 exploit-dev primitives sit at vocabulary depth so the Belt-5 graduate can hold a competent disclosure-landscape conversation alongside that one deeply-reproduced anchor. Per the discovery-learning approach (`project_discovery-learning approach.md`), vocabulary scaffolding intentionally leaves space for students to discover detail through hands-on lab encounters; the rows below are anchors, not exhaustive treatments.

Worked-example anchor. Section §5 supplies one canonical worked PoC for Heartbleed (CVE-2014-0160) at structural depth: the RFC 6520 wire format, the malformed request bytes, the simplified pre-patch `tls1_process_heartbeat()` function with its missing bounds-check annotated, a synthetic illustrative leaked-memory hexdump, and the literal one-line patch from OpenSSL commit 96db902. The pedagogical thesis is that one worked PoC of this quality (annotated byte sequence + literal vulnerable function + literal one-line fix) buys more career-long retention than many additional taxonomy rows, because the bounds-check rule it grounds generalizes far beyond TLS into every protocol implementation, parser, and deserializer the graduate ever writes.

Citation discipline. Every CVE name in this handout is publicly disclosed, fully patched at time of writing, and referenced in NVD + the original vendor advisory. CWE pairings follow MITRE conventions per `cwe.mitre.org`. Where an academy lab reproduces or simulates a CVE family, the lab name is named alongside (the reproduction always happens against intentionally-vulnerable lab-owned target infrastructure under `--authorized-by` discipline, never against production systems).

---

§2: Family 1: Memory-corruption + parser bugs

The memory-corruption family is the historical center-of-gravity for systems-level exploitation. C and C++ codebases (operating system kernels, cryptographic libraries, browser engines, network parsers, embedded firmware) dominate the production substrate, and the language-level lack of bounds-checking means that every attacker-controlled length, every untrusted pointer arithmetic, every freed-then-reused allocation is a potential exploit primitive. The CWE classes below are the named shapes a Belt-5 graduate should recognize when reading disclosure write-ups, conference talks, and post-mortems. Several of these classes also shape the defensive practices: every memory-corruption mitigation surface (ASLR, KASLR, NX/DEP, stack canaries, CFI, CET) was designed against one or more of these specific bug shapes.

Class	CWE	Anchor CVE	One-line shape
Heap buffer over-read	CWE-126	CVE-2014-0160 (Heartbleed)	Attacker-controlled length field read past allocation; leaks adjacent memory.
Stack buffer overflow	CWE-121	CVE-2003-0352 (RPC DCOM / Blaster)	Unchecked `strcpy` / `memcpy` overruns return address.
Heap overflow	CWE-122	CVE-2017-5638 (Apache Struts / Equifax content-type)	Crafted input expands heap chunk past its allocation; corrupts adjacent metadata.
Use-after-free (UAF)	CWE-416	CVE-2021-1782 (iOS XNU)	Object referenced after `free()`; reclaimed allocation primitives let attacker control a vtable / function pointer.
Double-free	CWE-415	CVE-2019-11707 (Firefox JIT)	Same allocation freed twice; corrupts allocator metadata.
Format string	CWE-134	CVE-2021-3156 (sudo Baron Samedit)	Attacker-controlled format specifier reads/writes arbitrary stack memory.
Integer overflow → buffer overflow	CWE-190	CVE-2018-4407 (XNU ICMP)	Wrapped size arithmetic produces a small allocation, then a large copy.
Type confusion	CWE-843	CVE-2019-5786 (Chrome FileReader)	Allocation treated as a different type than it actually holds; leads to controlled write.
TOCTOU race	CWE-367	CVE-2016-5195 (Dirty COW)	State checked at time T; acted upon at time T+Δ; window large enough to swap state.

Heap buffer over-read (CWE-126), Heartbleed (CVE-2014-0160). The cleanest example in the academy reading list of an attacker-controlled length field driving a copy past the actual record size. OpenSSL's TLS heartbeat handler trusted the wire-format `payload_length` field without validating it against the actual record size, then `memcpy`'d that many bytes back to the attacker, leaking ~64 KB of adjacent server heap memory per request, repeatable until session keys or private-key material happened to land in the leaked window. Section §5 supplies the full worked PoC. The defensive rule generalizes: **every attacker-controlled length must be validated against the**

actual amount of data received before it is trusted for any allocation, copy, or pointer arithmetic. SEC-101's protocol-implementation set and PEN-101's parser-fuzzing module both anchor on this CVE.

Stack buffer overflow (CWE-121), Blaster / RPC DCOM (CVE-2003-0352). The textbook stack-smash that catalyzed Microsoft's Trustworthy Computing initiative and shipped Windows-side mitigations (DEP, ASLR, /GS stack cookies) over the following decade. An unchecked `strcpy` in the RPC DCOM endpoint allowed a remote attacker to overflow a stack buffer, corrupt the saved return address, and redirect execution to attacker-supplied shellcode. The Blaster worm propagated worldwide in August 2003 using this primitive. Pedagogically this is the simplest exploitation primitive in the family: overwrite return address, transfer control. RE-011's stack-frame chapter and PEN-101's binary-exploitation module both reference this CVE as a foundational anchor.

Heap overflow (CWE-122), Apache Struts / Equifax (CVE-2017-5638). Crafted Content-Type header drove an Object-Graph Navigation Language (OGNL) expression-evaluation path that ultimately corrupted heap-allocated structures and reached arbitrary code execution. Better known as the vulnerability behind the Equifax breach (148 million records compromised; 2017). The same CVE appears in the injection family below as a deserialization-class shape; Struts CVEs frequently cross-cut multiple CWE families because the underlying attack chain spans expression-evaluation + memory-corruption + deserialization layers. ADV-101 references this CVE as a textbook chained-primitive disclosure; the Equifax breach itself becomes a case study in patch-management organizational failure.

Use-after-free (UAF, CWE-416), iOS XNU (CVE-2021-1782). A race condition in the XNU kernel's voucher-reference-counting code allowed an object to be `free()`'d while a stale reference remained reachable; the freed allocation could then be reclaimed under attacker control to forge a kernel object with attacker-chosen vtable contents, yielding kernel-mode code execution. Used in the iOS "FORCEDENTRY" / Pegasus chain to deliver in-kernel shellcode without user interaction. UAF is the dominant class in modern browser and kernel exploitation; heap-grooming + Feng Shui techniques (see §6 exploit-dev primitives) are how attackers reliably reclaim the freed allocation. RE-201 + Part-II RE electives anchor on this CVE for kernel-exploit pedagogy.

Double-free (CWE-415), Firefox JIT (CVE-2019-11707). A type-confusion in Firefox's IonMonkey JIT compiler allowed an `Array` object to be `free()`'d twice, corrupting heap allocator metadata in a way that yielded an arbitrary read/write primitive. Used in-the-wild against Coinbase Employees in 2019. Pedagogically: the JIT-compiler attack surface is a recurring source of browser CVEs because JITs must reason

about value-type identity across compilation stages and any inconsistency becomes a memory-corruption primitive. RE-101's browser-internals module and ADV-101's Belt-5 reading list both name this CVE.

Format string (CWE-134), sudo Baron Samedit (CVE-2021-3156). A heap-based buffer overflow in `sudo`'s argument-parsing path triggered by a crafted command-line argument with an unescaped backslash; allowed any local user to escalate to root on virtually every Unix-like system shipping `sudo` for the prior decade. Format-string vulnerabilities are now rare in actively-maintained codebases (compilers warn, static analyzers flag), but legacy codebases and embedded firmware still ship them. RE-011 covers format-string at vocabulary depth as a historical primitive; PEN-101 does not currently include a format-string lab but the Baron Samedit CVE is named in the privilege-escalation reading list.

Integer overflow → buffer overflow (CWE-190), XNU ICMP (CVE-2018-4407). Apple's XNU kernel computed an ICMP packet length using arithmetic that wrapped on attacker-crafted inputs, allocating a buffer smaller than the actual packet then copying the full attacker-controlled payload (a wrapped-size primitive). The class is pedagogically important because integer-overflow bugs are easy to write (size arithmetic looks innocuous) and hard to spot in review (unsigned wraps silently). The defensive rule: **validate size arithmetic against architecture-specific bounds (typically `SIZE_MAX`) before using the result for allocation.** RE-201's kernel-internals module and ADV-101's Belt-5 reading list both name this CVE.

Type confusion (CWE-843), Chrome FileReader (CVE-2019-5786). A Blink-engine bug in Chrome's `FileReader` API caused an object allocated as one type to be later accessed as a different type, yielding an attacker-controlled vtable dereference that the attacker could turn into renderer-process code execution. Used in-the-wild as a Chrome zero-day in 2019. Type-confusion is increasingly the dominant class in modern browser exploitation because language-level type systems fail to fully prevent the cross-allocation aliasing that browser engines internally rely on. RE-101's browser-internals module and Part-II browser-exploitation electives anchor here.

TOCTOU race (CWE-367), Dirty COW (CVE-2016-5195). A classic time-of-check-to-time-of-use race in the Linux kernel's copy-on-write (COW) page-fault handler allowed any local user to write to read-only memory by carefully timing two threads (one mapping a private read-only file, one racing to mark the page dirty before the COW resolved). The window was originally believed too small to exploit reliably, then turned out to be exploitable in a few seconds with the right thread-scheduling pattern. Pedagogically, TOCTOU classes are about **the gap between checking a condition**

and acting on it: file-system races, environment-variable races, signal-handler races, kernel-data-structure races. SEC-101's privilege-boundary chapter and RE-011's kernel-vocabulary module both name this CVE.

---

§3: Family 2: Injection + deserialization

The injection family is what dominates the modern web-application and API-attack surface. Where memory-corruption requires the attacker to bypass language-level bounds checks, injection requires the attacker only to slip user input across a parsing boundary that mistakes data for code. The shape recurs at every parsing boundary in computing: SQL queries, shell invocations, template engines, deserializers, log-evaluators, JNDI lookups, HTML/JS contexts. The defensive rule that generalizes across the whole family: **never concatenate untrusted input into a context that interprets text as code; use the language's parameterized / parser-aware API instead.** This is the family that the OWASP Top 10 has flagged as #1 or #3 every year since 2013, and the family ADV-102 extends into the AI-era prompt-injection register.

Class	CWE	Anchor CVE	One-line shape
SQL injection	CWE-89	CVE-2023-34362 (MOVEit Transfer / Cl0p)	User input concatenated into SQL; alters query semantics.
OS command injection	CWE-78	CVE-2024-3094 (XZ-Utils backdoor; adjacent shape)	User input concatenated into shell invocation; alters command boundary.
Server-side template injection (SSTI)	CWE-1336	CVE-2025-65106 (LangChain Jinja2, ADV-102 anchor)	User input rendered as template source; engine evaluates it as code.
Insecure deserialization	CWE-502	CVE-2017-5638 (Apache Struts / Equifax)	Untrusted bytes deserialized; gadget chain reaches arbitrary execution.
JNDI lookup injection	CWE-917	CVE-2021-44228 (Log4Shell)	Logger evaluates attacker-supplied JNDI URL; fetches and executes remote class.
XSS (reflected/ stored/DOM)	CWE-79	CVE-2018-0114 (Cisco web-UI)	User input echoed into HTML/JS context without encoding.
Cross-site request forgery	CWE-352	Longenecker SB6141 (ADV-101 primary)	Browser auto-attaches credentials to attacker-crafted cross-origin request.

SQL injection (CWE-89), MOVEit / Cl0p (CVE-2023-34362). Progress Software's MOVEit Transfer file-transfer appliance contained a SQLi vulnerability in its web-front-end that the Cl0p ransomware group used to compromise hundreds of organizations through the second half of 2023, including major US federal contractors and state agencies. SQLi has been ranked as the #1 web-application vulnerability for two decades; despite parameterized-query APIs being available in every major language since the 1990s, codebases continue to concatenate user input into SQL queries, especially in legacy enterprise software and admin interfaces. PEN-101 Week 4 reproduces SQLi against an intentionally-vulnerable lab harness (DVWA + sqlmap); SEC-101's parameterized-query practices pairs against this CVE as a defensive case study.

OS command injection (CWE-78), XZ-Utils backdoor (CVE-2024-3094). The XZ-Utils backdoor is technically a supply-chain compromise (it's named again in §4), but the in-binary shape was a command-injection primitive: attacker-supplied SSH session metadata, decoded inside the modified `liblzma`, decoded into a payload that the SSH

daemon process executed in its own context. The pedagogical lesson is that command-injection vulnerabilities don't need a `system()` call to be present in the source: `popen()`, `exec()`, `subprocess.call(..., shell=True)`, and any number of string-interpolated shell invocations create the same primitive. PEN-101 Week 5 reproduces command-injection against intentionally-vulnerable lab harnesses; the XZ-Utils story is named as a real-world supply-chain-meets-injection cross-cut.

Server-side template injection (SSTI, CWE-1336), LangChain Jinja2 (CVE-2025-65106). LangChain's Python framework, used to build LLM-powered applications, had a default code path where user-supplied prompt templates were rendered through Jinja2's `Environment.from_string()` without sandboxing, meaning a prompt template containing `{{config.__class__.__init__.__globals__['os'].popen('id').read()}}` (or any of the well-known Jinja2 sandbox-escape patterns) yielded server-side code execution. SSTI as a class predates AI by a decade (Flask + Jinja, Django, Twig, Velocity, and FreeMarker have all shipped famous SSTI CVEs), but the AI ecosystem has reintroduced the class at scale because LLM-application frameworks need to render dynamic templates from user input. ADV-102 Week 3 anchors on this CVE as the central intersection of OWASP LLM Top 10 (LLM01: Prompt Injection) with classical web-vulnerability taxonomy.

Insecure deserialization (CWE-502), Apache Struts / Equifax (CVE-2017-5638). The Equifax CVE again, this time read through the deserialization lens: the OGNL expression-evaluation that ultimately yielded arbitrary code execution started as a Content-Type header parser that deserialized attacker-controlled bytes through a Java object-graph deserializer (without the strict allow-list that ysoserial-era hardening recommended). The "gadget chain" register (chains of method calls reachable through Java's serialization-callback machinery that ultimately reach `Runtime.getRuntime().exec()`) is its own substantial vocabulary; tools like `ysoserial` enumerated dozens of public chains across common Java libraries. Pedagogically: **never deserialize untrusted input through a deserializer that can instantiate arbitrary types.** ADV-101 references this CVE in Week 6 as the textbook case for the cohort-rotation Belt-5 midterm.

JNDI lookup injection (CWE-917), Log4Shell (CVE-2021-44228). Apache Log4j 2's `Logger.error(message)` API recursively evaluated `${jndi:...}` substrings in the logged message, fetching and executing a remote Java class on attacker demand, turning every Java application that logged user-controllable strings (which is virtually all of them) into a remote-code-execution target. The November-December 2021 Log4Shell incident is the largest single-vulnerability fire-drill the industry has had since

Heartbleed; SEC-101's incident-response register and ADV-101's reading list both anchor here as a case study in how a single supply-chain dependency can ship the vulnerability into thousands of organizations simultaneously. The defensive lesson: **logger libraries should not evaluate substitution syntax in user-supplied strings.**

XSS (CWE-79), Cisco web-UI (CVE-2018-0114). Cisco shipped a stored-XSS vulnerability in the web-administration UI of multiple network-device product lines; an attacker-controlled string injected into a configuration field would later execute as JavaScript in the browser of any administrator who viewed the affected page. XSS is the most common web-application vulnerability in absolute count (millions of CVEs over the past two decades) but has shifted in severity: modern browser content-security-policy (CSP) enforcement, cookie `HttpOnly` + `SameSite` flags, and framework-level auto-encoding (React's JSX, Angular's binding model, Vue's templates) have substantially reduced practical impact. PEN-101 Week 4 reproduces XSS against intentionally-vulnerable lab harnesses; the Cisco CVE is named as a real-world case where the affected target is a network-management tool rather than a public-facing web app.

Cross-site request forgery (CSRF, CWE-352), Longenecker SB6141 (ADV-101 primary). The SB6141-CSRF disclosure (Chris Longenecker, 2018, against the Motorola SB6141 cable modem's web-administration UI) is **the academy's primary worked-CVE for ADV-101's Belt-5 capstone**: the modem's admin UI lacked CSRF tokens, meaning any malicious page the user visited could submit cross-origin POST requests to `192.168.100.1` (the SB6141's default LAN-side admin address) to reconfigure the modem. ADV-101 students reproduce the disclosure end-to-end against lab-owned SB6141 hardware under `--authorized-by` discipline, write a CERT/CC-grade disclosure report, score the CVSS, and ship a defensive `--dry-run + --authorized-by`-equipped tool against the vulnerability. CSRF as a class has been substantially mitigated by the modern web platform (cookies default to `SameSite=Lax`; framework auto-tokens); the SB6141 anchor demonstrates how **embedded device firmware lags the modern web platform by a decade or more**.

---

§4: Family 3: Auth-bypass + supply-chain

The auth-bypass + supply-chain family covers the failure modes where the attacker does not need to corrupt memory or inject code. They simply need to **convince the system that they are someone (or something) they are not**, or **inject themselves into a trusted distribution channel**. The family is pedagogically

distinct from memory-corruption and injection because the bugs are usually not language-level safety failures; they are protocol-level, configuration-level, or organizational-level design mistakes. The defensive practices here is also organizational, not just technical: SBOMs, build-reproducibility, signing-verification chains, and dependency review are the controls that address supply-chain compromise. SolarWinds + XZ-Utils together established that this family is now a strategic-level threat for any organization that ships software, not just a technical-level concern for individual developers.

Class	CWE	Anchor CVE	One-line shape
Hardcoded credentials	CWE-798	ARRIS / Broadcom modem family (multiple)	Vendor-fixed credential in firmware; trivially recovered, universally usable.
Authentication bypass	CWE-287	CVE-2023-4966 (Citrix Bleed)	Session-state handling lets attacker assume an authenticated identity.
Path traversal	CWE-22	CVE-2024-1709 (ConnectWise ScreenConnect)	`..` sequences resolve to files outside intended scope.
SSRF	CWE-918	CVE-2021-26855 (Exchange ProxyLogon)	Server fetches an attacker-controlled URL; accesses internal services.
Build-time supply-chain compromise	CWE-506	CVE-2024-3094 (XZ-Utils)	Backdoor injected via build process of widely-deployed dependency.
Software supply-chain compromise	CWE-1357	SolarWinds SUNBURST (CVE-2020-10148)	Trusted update channel ships attacker-modified binary to thousands of victims.
Insecure model serialization (AI-era)	CWE-502	CVE-2024-3568 (Hugging Face transformers `pickle`)	ML weight file deserialized as Python pickle; arbitrary code on load.

Hardcoded credentials (CWE-798), ARRIS / Broadcom modem family. Cable-modem and consumer-router firmware has shipped vendor-fixed credentials embedded directly in firmware images for the better part of two decades; the credentials are recoverable via firmware extraction (the central skill of RE-101) and, because the same firmware ships across millions of identical devices, a single discovered credential is

universally usable across the deployed fleet. The ARRIS / Broadcom family (TM82x, SB6121, SB6141, SB8200, and many others) is the academy's named anchor because it overlaps the SB6141 lab target. The Belt-5 cohort-rotation midterm (per ADV-101 Week 6) explicitly uses a different ARRIS / Broadcom variant than the SB6141 anchor for assessment, validating that the methodology transfers across firmware versions. RE-101's firmware-extraction module + ADV-101's Belt-5 capstone both anchor on this family.

Authentication bypass (CWE-287), Citrix Bleed (CVE-2023-4966). Citrix NetScaler ADC and Gateway products contained a buffer-over-read vulnerability that leaked session tokens from server memory; an attacker could replay the leaked tokens to assume any authenticated user's session, bypassing multi-factor authentication entirely because MFA had already happened earlier in the session lifecycle. Citrix Bleed was actively exploited by ransomware groups (LockBit, Medusa) against enterprise targets through late 2023 and into 2024. The class spans all the ways session-state handling can fail: token-prediction, session-fixation, replay-after-leak, broken-MFA-state. SEC-101 Module 5 (Authentication architectures) and PEN-101 Week 7 (Auth bypass register) both anchor on this CVE.

Path traversal (CWE-22), ConnectWise ScreenConnect (CVE-2024-1709). A path-traversal in ScreenConnect's setup wizard allowed unauthenticated attackers to access setup-only resources from a fully-installed system, then chain to a separate authentication-bypass (CVE-2024-1708) for full administrative compromise. ScreenConnect is widely deployed by managed-service-providers (MSPs) for endpoint-management, which made the CVE pair a high-value target for supply-chain-attack-style exploitation in February 2024. Path-traversal as a class has been around since web servers existed; the modern register includes URL-encoded variants (`%2e%2e%2f`), Windows-vs-POSIX path-separator differences, and Unicode-normalization bypasses. PEN-101 Week 4 reproduces path-traversal against intentionally-vulnerable lab harnesses; SEC-101 covers the defensive practices (canonicalization, allow-list, chroot/jail).

SSRF (CWE-918), Exchange ProxyLogon (CVE-2021-26855). Microsoft Exchange Server's Client-Access frontend contained a server-side request forgery (SSRF) vulnerability that allowed an unauthenticated attacker to craft requests that the Exchange server would issue on the attacker's behalf to internal services, including Exchange's own backend, which trusted the frontend's identity. Combined with three other Exchange vulnerabilities (CVE-2021-26857, -26858, -27065), ProxyLogon yielded full Exchange-server compromise; HAFNIUM and other actors exploited it widely through the first quarter of 2021. SSRF has become especially dangerous in cloud-native

architectures because cloud-metadata endpoints (AWS IMDSv1's `169.254.169.254` is the textbook example) trusted any local request, meaning a single SSRF could exfiltrate IAM credentials. ADV-101's cohort-rotation midterm option 4 reproduces an SSRF + cloud-metadata-exposure shape against a containerized harness.

Build-time supply-chain compromise (CWE-506), XZ-Utils (CVE-2024-3094). The XZ-Utils backdoor (March 2024) is the most consequential supply-chain compromise of the modern era to be detected before mass exploitation: a sophisticated multi-year campaign, attributed to a sock-puppet maintainer ("Jia Tan") who gained commit access through ordinary open-source contribution patterns, injected obfuscated payload bytes into `xz/liblzma`'s release tarballs (not the git repository, only the tarballs) such that any system linking against `liblzma` while running `sshd` would expose an authenticated SSH backdoor to a holder of the attacker's signing key. The vulnerability was discovered by Andres Freund (a Microsoft Postgres developer) noticing a 500ms latency anomaly during `sshd` logins on a Debian sid system, an extraordinary catch that prevented mass deployment. SEC-101 Module 7 (Supply-chain defense) and ADV-101's reading list both anchor on this CVE; the case study is forward-looking pedagogy for the SBOM + reproducible-build practices.

Software supply-chain compromise (CWE-1357), SolarWinds SUNBURST (CVE-2020-10148). SolarWinds Orion (a network-monitoring product widely deployed by US federal agencies and Fortune 500 enterprises) shipped a trojanized update in March-June 2020 that included the SUNBURST backdoor, the result of attacker (APT29) compromise of SolarWinds's build infrastructure. Approximately 18,000 organizations downloaded the trojanized update; a smaller number (~100) had the second-stage payload activated. SUNBURST established that **trusted-update-channel compromise** is a viable strategic-level attack vector, and catalyzed industry-wide investment in supply-chain controls (CISA's BOD 22-01, NIST SP 800-161, the EO 14028 SBOM mandate, sigstore/cosign, SLSA framework). SEC-101 Module 7 and ADV-101 Belt-5 reading list both anchor on this case as the worked example of strategic-level supply-chain risk.

Insecure model serialization (AI-era, CWE-502), Hugging Face transformers `pickle` (CVE-2024-3568). A Python pickle file masquerading as an ML model weight file, when loaded by `torch.load()` or `transformers.AutoModel.from_pretrained()` without `weights_only=True`, yields arbitrary Python code execution at load time, because Python `pickle` deserialization can instantiate arbitrary classes and call arbitrary methods. The class predates AI by decades (Python's `pickle` documentation has warned against deserializing untrusted input since the 1990s) but the AI ecosystem reintroduced it at

scale because model-distribution platforms (Hugging Face Hub, model marketplaces) routinely host attacker-uploadable artifacts. ADV-102 Week 4 anchors on this CVE as the AI-era central case for OWASP LLM Top 10 (LLM05: Supply Chain) intersecting with classical insecure-deserialization. ADV-101's cohort-rotation midterm option 3 reproduces this shape against a containerized older-version transformers harness.

---

§5: Worked PoC anchor: Heartbleed (CVE-2014-0160) as an information-leak primitive

> **Educational framing.** Heartbleed is a 12-year-old, fully-patched, widely-published vulnerability covered in every undergraduate security course. The annotated structures and simplified function below are illustrative, not runnable exploit code; the academy's doctrine is that any reproduction work happens against intentionally-vulnerable test infrastructure (a local OpenSSL 1.0.1f container in `fwlab`) under `--authorized-by` per the same discipline ADV-101 enforces for the SB6141 CSRF. The pedagogical value is making the bounds-check requirement *concrete*: students who have seen the literal shape of the bug write better defensive code for the rest of their careers.

Heartbleed is the canonical worked example of an **information-leak primitive**. It is also the canonical worked example of **CWE-126 (out-of-bounds read driven by attacker-supplied length)**. Understanding the literal byte-level shape of the request is what makes the bounds-check rule (*"never trust an attacker-supplied length without validating against the actual record size"*) memorable. The fix is one line of C; the lesson lasts a career.

§5.1: The TLS Heartbeat Request record (RFC 6520)

```
/* RFC 6520 §4 - heartbeat_message structure
 * The wire format the protocol defines, as the server expects. */
struct {
    HeartbeatMessageType type;       /* 1 = heartbeat_request */
    uint16               payload_length;
    opaque               payload[HeartbeatMessage.payload_length];
    opaque               padding[padding_length];   /* >= 16 bytes */
} HeartbeatMessage;
```

The attacker's observation: `payload_length` is a 16-bit field on the wire, attacker-controlled, with no protocol-level requirement that the actual `payload` field be that long. A vulnerable implementation (OpenSSL 1.0.1 through 1.0.1f) trusted the attacker-supplied length when copying the payload into the response buffer.

§5.2: The malformed request, claim 65535-byte payload, send 1 byte

```
/* Bytes the attacker puts on the wire after TLS handshake completes.
 * Annotated with TLS record-layer + heartbeat structure. */
unsigned char heartbeat_request[] = {
    /* TLS record header */
    0x18,                /* ContentType: heartbeat (24) */
    0x03, 0x02,          /* ProtocolVersion: TLS 1.1 */
    0x00, 0x03,          /* record length: 3 bytes follow */

    /* HeartbeatMessage starts here */
    0x01,                /* type: heartbeat_request */
    0xff, 0xff           /* payload_length: 65535 - THE LIE */
    /* No actual payload bytes, no padding - the record ended at byte 3. */
};
```

§5.3: What pre-patch OpenSSL did, simplified

```
/* Pre-patch ssl/t1_lib.c tls1_process_heartbeat(), simplified for teaching.
 * Real source: openssl-1.0.1f/ssl/t1_lib.c lines ~2586-2625.
 * The actual fix landed in commit 96db902 (Bodo Möller / Adam Langley). */

int tls1_process_heartbeat_VULNERABLE(SSL *s) {
    unsigned char *p = &s->s3->rrec.data[0];   /* inbound record */
    unsigned short payload;                    /* attacker-controlled length */
    unsigned char *buffer, *bp;

    /* Read the attacker-supplied length off the wire. */
    n2s(p, payload);                           /* payload = 0xFFFF (65535) */

    /* MISSING CHECK: nothing here verifies that
     *   payload + padding + 1 + 2 <= s->s3->rrec.length
     * The record was 3 bytes total; payload claims 65535. */

    /* Allocate a response buffer sized to the LIE, not the truth. */
    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;
    *bp++ = TLS1_HB_RESPONSE;
    s2n(payload, bp);                          /* echo the lie back */

    /* THE BUG: copy `payload` (65535) bytes starting at `p`, which
     * points to a single attacker-supplied byte (here, none at all).
     * Bytes p[1..65534] come from whatever happens to live in heap
     * memory after the inbound record buffer: session keys, server
     * private-key material, plaintext POST bodies from other users,
     * cached TLS session state, anything adjacent to this allocation. */
    memcpy(bp, p, payload);                    /* OUT-OF-BOUNDS READ */

    /* Send the response - the attacker now receives ~64KB of server
     * heap memory back in a heartbeat_response record. */
    return dtls1_write_bytes(s, TLS1_RT_HEARTBEAT,
                             buffer, 3 + payload + padding);
}
```

§5.4: What the leaked memory looks like to the attacker

```
# A heartbeat_response received by the attacker after sending the
# malformed request above. Real captures from 2014 disclosure showed
# patterns like the following - server heap memory adjacent to the
# inbound TLS buffer, sometimes including credential material.

00000000  02 ff ff 00 00 00 00 00  00 00 00 00 00 00 00 00   ........  ........
00000010  43 6f 6f 6b 69 65 3a 20  73 65 73 73 69 6f 6e 3d   Cookie:   session=
00000020  61 62 63 31 32 33 64 65  66 34 35 36 0d 0a 41 75   abc123de  f456..Au
00000030  74 68 6f 72 69 7a 61 74  69 6f 6e 3a 20 42 65 61   thorizat  ion: Bea
00000040  72 65 72 20 65 79 4a 68  62 47 63 69 4f 69 4a 49   rer eyJh  bGciOiJI
...
0000ff00  2d 2d 2d 2d 2d 42 45 47  49 4e 20 52 53 41 20 50   -----BEG  IN RSA P
0000ff10  52 49 56 41 54 45 20 4b  45 59 2d 2d 2d 2d 2d 0a   RIVATE K  EY-----.
...
# The attacker repeats the heartbeat hundreds of times; each response
# samples a different region of the server's heap. Over time, session
# tokens, private keys, and plaintext request bodies accumulate.
```

§5.5: The fix, one bounds check

```
/* The patch (OpenSSL commit 96db902, 2014-04-07): validate the
 * attacker-supplied length against the actual record size BEFORE
 * trusting it for the memcpy. */
if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0;   /* silently drop malformed heartbeat */
```

§5.6: Pedagogical takeaway and where this leads

The pedagogical takeaway. Every attacker-controlled length field is a potential Heartbleed. The defensive rule generalizes far beyond TLS: when input arrives with a length-prefix, the receiver must validate the length against the actual amount of data received before using the length for any allocation, copy, or pointer arithmetic. Students who have read the literal four-line `memcpy` and the literal one-line fix carry this rule into every protocol implementation, parser, and deserializer they ever write.

Where this leads. A "Heartbleed-class" finding on a modern target requires the disclosure-coordination discipline ADV-101 teaches against the Longenecker SB6141 anchor: lab-owned target, written authorization, CERT/CC-grade report, CVSS scoring, vendor coordination per ISO/IEC 29147. The reproduction technique is different; the engagement discipline is identical. **Attribution.** Heartbleed was discovered independently by Neel Mehta (Google) and by a Codenomicon team (Riku, Antti, Matti); the OpenSSL fix landed in commit 96db902 (Bodo Möller and Adam Langley, 2014-04-07). The disclosure timeline (notification on 2014-04-01, public disclosure on 2014-04-07) became a case study in coordinated multi-vendor disclosure under embargo. A future supplementary handout (`adv-101-heartbleed-deep-walk.md`, per §9 supplements) expands this section to ~600-800 lines: full RFC 6520 walkthrough, OpenSSL source-tree archaeology, the historical disclosure timeline, the LibreSSL fork, and the Core Infrastructure Initiative funding response.

---

§6: Tool-defense catalog (responsible-adversary engineering primitives)

The Tool v0.1 → v1.0 sequence in ADV-101 Labs 4/5/7 introduces three primitives (`--authorized-by` + `--dry-run` + structured logging) that distinguish a responsible-adversary capstone tool from a script-kiddie one-off. Production-grade adversarial tools commonly carry the additional five primitives below. The Belt-5 graduate should **recognize** all five even if the academy's capstone tool implements only the first three; reading red-team tool source in the field requires the full vocabulary.

Scope-limit network primitive. Explicit `--target-cidr` allowlist; tool refuses to send any packet to an address outside the allowlist regardless of arguments. The engineering motivation: humans typo. A misconfigured `nmap` scan against `0.0.0.0/0` is a career-ending event; a misconfigured exploit-PoC against a production address is a federal-law-violation event. The defensive primitive is **explicit allowlist over implicit denylist**, the tool fails closed by refusing to send to anything not pre-authorized. Implementation pattern: argument-parser validates the target against the allowlist before any socket is opened; raw-socket primitives wrap a final-check that drops outbound packets to non-allowlist destinations as a defense-in-depth.

Pre-flight target fingerprint with refusal-on-mismatch. Tool runs a fingerprint check against the target before issuing any active probe and refuses to continue if the fingerprint does not match the expected target family. ADV-101's Tool v0.1 is exactly this primitive at minimum viable depth: the SB6141-CSRF tool first issues a benign HTTP

GET to confirm the target's `Server:` header matches the SB6141 family before issuing any CSRF-shaped POST. The engineering motivation: **keeps the tool from running against an arbitrary target by accident**. A graduate testing the tool from a coffee-shop network does not accidentally probe the coffee shop's router. Implementation pattern: signature library matches Server / WWW-Authenticate / TLS-cert-CN / SNMP sysDescr / banner against a known-target signature; mismatch → exit 1 with diagnostic output.

Kill-switch / heartbeat / time-limit. Tool exits if it cannot reach an instructor-controlled heartbeat endpoint, or after `--max-runtime-seconds` elapses. The engineering motivation: **bounded blast radius**. A tool that can keep running forever is a tool that an adversary can repurpose; a tool that requires periodic check-in to a controlled endpoint is one that can be revoked centrally. Implementation pattern: background thread polls the heartbeat URL every N seconds; on first failure or timeout-expiry, tool sets a "drain" flag that prevents new operations and exits cleanly within bounded time. The pattern parallels production-system circuit-breakers and auto-scale-down policies.

Forensic-friendly output. Structured JSON-Lines log (one event per line, machine-parseable) that downstream forensic and SIEM tools can ingest; explicit `invocation-id` per run for log-correlation; PCAP-companion pointer for any network primitive used. The engineering motivation: a responsible-adversary engagement produces evidence that the customer's blue-team can review and audit; a script-kiddie tool produces output that is unreadable two days later. Implementation pattern: each emit-event call produces one JSON object per line with fields `{ts, invocation_id, event, args, result}`; a separate `--log-pcap` flag captures the full network exchange as a `.pcap` alongside the JSONL. RE-101's tool-engineering practices pairs against this primitive as a baseline expectation for tool releases.

Sandbox / VM detection refusal (defensive variant). Tool refuses to run if it detects it is being executed inside a malware-analysis sandbox (Cuckoo, ANY.RUN, CIS Sandbox) on the assumption that a legitimate engagement does not run there. The academy capstone does **not** implement this primitive; it is named here so graduates recognize the technique when reading red-team tool source in the field. The defensive use-case: prevents the tool from leaking its TTPs to a malware-analysis vendor's classifier. The dark-side use-case: malware authors use the same primitive to evade analysis. The pedagogical lesson is that the technique is the same; the legitimacy is determined by **who is running the tool against whom under what authorization**, not by the technique itself. ADV-101 Week 8 reading list names this primitive in the context of evasion ethics.

---

§7: Research-conference list

Belt-5 graduates pursuing vuln-research careers should track the conferences where the discipline ADV-101 teaches is **performed and judged in public**. The register below names the venues, what each is known for, and the kind of work that gets accepted.

Pwn2Own. Trend Micro Zero Day Initiative's biannual exploit-demonstration competition (Vancouver in spring; Tokyo / Toronto / Ireland rotation in fall; an automotive-focused edition in Tokyo and an industrial-focused edition recently added). Sets the discipline of **one-shot exploit + responsible disclosure under live audience**. Researchers register against named target software, receive a fixed time window to demonstrate exploitation against a fully-patched version, and the vulnerabilities are immediately disclosed to the affected vendor. Browser, kernel, virtualization, and embedded categories. Cash prizes and "Master of Pwn" recognition. Belt-5 graduates with browser or kernel exploitation skills target Pwn2Own as the discipline's most public proving ground.

DEFCON + Black Hat. Annual Las Vegas pair (early August; "Hacker Summer Camp"). DEFCON is the larger and more practitioner-flavored event with named "villages" (Car Hacking, ICS, Aviation, Bio, IoT, RF, Lock-Picking, Voting, Social Engineering) and the famous DEFCON CTF; Black Hat is the more enterprise-flavored counterpart with formal Briefings tracks. Talks span every track in the academy curriculum; the DEFCON CTF qualifier is one of the academy's named cohort-stretch goals. Belt-5 graduates submit talk proposals to both via the annual CFP cycles.

OffensiveCon. Berlin, annual (typically May). Deep-technical exploit-development conference with a tight selection bias toward original-research talks and live exploit demonstrations. Belt-5 graduates pursuing browser, kernel, or iOS exploitation read the proceedings; OSEE (Offensive Security Exploitation Expert) certification holders disproportionately publish here.

REcon. Montreal, annual (typically June). Reverse-engineering conference; cross-cuts ADV-101 with RE-101 + RE-201. Talks span malware analysis, firmware archaeology, hardware reverse-engineering, vintage computing, and tool development. The Hex-Rays plug-in contest (separately listed below) is REcon-adjacent culturally. Belt-5 graduates with RE skills target REcon as a community-fit conference.

USENIX Security + IEEE S&P ("Oakland") + NDSS + ACM CCS. The four "tier-1" academic security venues. Annual. Original-research papers (~14-page, peer-reviewed, formal proceedings). RE-101 and RE-201 already point here. ADV-101 graduates publishing original findings target the practitioner tracks (USENIX Security's "Security

Notes" or the workshop tracks at all four venues); the main-conference tracks expect a substantial novel-research contribution and typically take a graduate-school or industry-research lab to produce.

Hex-Rays plug-in contest. Tool-engineering venue for the IDA Pro / Ghidra / radare2 ecosystem. Annual contest; Hex-Rays publishes winning plug-ins. The kind of public artifact ADV-101's capstone tool would extend toward; demonstrates that **building tools the community uses** is itself a research-track output, not just a precondition for research.

Discovery + intelligence references (companion catalog). MITRE CWE (`cwe.mitre.org`), the *class* of weakness; pairs with CVE the way SI-units pair with measurements. **CISA Known Exploited Vulnerabilities (KEV) catalog**, the industry signal for "not just disclosed, actually exploited in the wild." Federal civilian agencies are required to remediate KEV entries on a deadline; private-sector defenders use it as a triage signal. **EPSS (Exploit Prediction Scoring System)**, FIRST.org-maintained probabilistic score complementing CVSS. **NVD + vendor PSIRTs** (MSRC, Apple Security, Google VRP, Cisco PSIRT, Broadcom Security), the disclosure substrate.

---

§8: Cross-track reference table

The following table is a **placeholder reference register** capturing each cyber-track course's expected pickup points for this handout. Section numbers are best-effort estimates from the existing course-page outlines as of 2026-05-03; the next-round spec-curriculum refinement will replace placeholders with exact section/chapter/lab numbers as each course's detailed module structure stabilizes.

Course	Module / week / lab	Pickup purpose
`vca-sec-101`	Module 4 (Vulnerability Landscape). Primary	All 3 family tables read as the foundational vocabulary set; Heartbleed worked PoC anchors the bounds-check defensive rule for the protocol-implementation set.
`vca-sec-101`	Module 5 (Authentication architectures)	§4 auth-bypass family + Citrix Bleed anchor for session-state failure modes.
`vca-sec-101`	Module 7 (Supply-chain defense)	§4 supply-chain rows (XZ-Utils + SolarWinds) anchor the SBOM + reproducible-build practices.
`vca-pen-101`	Week 4 (Web-application exploitation)	§3 injection family + SQLi / XSS / path-traversal rows; PEN-101 reproduces these against intentionally-vulnerable lab harnesses (DVWA + sqlmap + custom containers).
`vca-pen-101`	Week 5 (Command injection + privilege escalation)	§3 OS command injection row + §2 sudo Baron Samedit format-string row for local-priv-esc.
`vca-pen-101`	Week 6 (CVE-driven exploitation set)	This handout cited as the Belt-3 vocabulary baseline; students self-test which families they can name without reference.
`vca-pen-101`	Week 7 (Auth-bypass register)	§4 Citrix Bleed + ConnectWise ScreenConnect rows.
`vca-adv-101`	Week 1 (Belt-5 vocabulary onboarding)	This handout read end-to-end as the Belt-5 entry-baseline; written self-assessment confirms vocabulary-tier mastery before methodology work begins.
`vca-adv-101`	Week 6 (Cohort-rotation midterm practical)	§3 Insecure deserialization + §4 Hardcoded credentials + §4 SSRF + §4 Insecure model serialization rows are the source for the 5 cohort-rotation midterm options.
`vca-adv-101`	Belt-5 capstone (Week 8 → 12)	§5 Heartbleed worked PoC as the structural template for the student's own CERT/CC-grade disclosure write-up against the SB6141 CSRF anchor; §6 tool-defense catalog as the engineering-quality ladder.
`vca-adv-102`	Week 2 (CVE landscape adjacent to LLM-OWASP)	This handout cited as the classical-CVE baseline against which the LLM-OWASP Top 10 register is contrasted.
`vca-adv-102`	Week 3 (Prompt injection + SSTI cross-cut)	§3 SSTI row + LangChain Jinja2 anchor as the central intersection with LLM01 (Prompt Injection).
	Week 4 (AI supply chain)	

Course	Module / week / lab	Pickup purpose
`vca-adv-102`		§4 Insecure model serialization row + Hugging Face transformers `pickle` anchor as the central intersection with LLM05 (Supply Chain).
`vca-re-101`	Module 3 (Disclosure-vocabulary onramp)	§2 + §3 + §4 family tables read as the vocabulary substrate for reading vendor PSIRT advisories.
`vca-re-101`	Module 7 (CVE class reading list)	This handout cited as the canonical CVE-vocabulary anchor for the reading list; per-class anchors map into the firmware-reverse-engineering lab register.
`vca-re-101`	Module 9 (Embedded firmware exploitation)	§4 Hardcoded credentials family + ARRIS / Broadcom anchor (overlaps SB6141 lab target).
`vca-re-011`	Module 3 (Foundational disclosure-vocabulary onramp)	§2 family table read as the foundational systems-vocabulary set; CVEs are named without yet reproducing them. Methodology comes in RE-101.
`vca-re-011`	Kernel-vocabulary module	§2 Dirty COW + Use-after-free rows for kernel-class vocabulary anchors.

Cross-link discipline. Each cyber-track course page (`vca-*.html`) should add a one-sentence pointer to this handout in its course-overview section, in catalog tone, of the form: *"For the academy-wide CVE-class vocabulary set cited in this course's modules, see `handouts/cross-chapter-cve-class-vocabulary-reference.md`."* The handout itself is **not** inlined into any public catalog page (per the catalog-vs-classroom doctrine).

---

§9: Decisions / Pedagogy / Supplements

Decisions

1. **Three-family taxonomy chosen over five-family.** The `sonnet-led-cyber-track-eval-2026-05-03.md` §6 supplement proposal listed "memory-corruption + injection + auth-bypass + supply-chain + AI-era" as five families. This handout collapses them into three (memory-corruption + injection-deserialization + auth-bypass-supply-chain) by treating supply-chain as a sub-cluster of auth-bypass (both are "the attacker convinces the system they're someone they're not" failures at different scales) and treating AI-era CVEs as instances of existing classes (insecure-deserialization for pickle CVEs, SSTI for LangChain CVEs) rather than a separate family. Rationale: three families is the canonical cognitive load for vocabulary

mastery; collapsing avoids the false-precision of an arbitrary five-way split. AI-era anchors are still named in the existing classes' rows (CVE-2024-3568, CVE-2025-65106) so coverage is not lost.

2. **Heartbleed retained as the single worked PoC anchor.** The probe-doc §6 pedagogy #2 finding ("the Heartbleed worked PoC is the pedagogically high-value addition") carries the handout's structure; this handout transplants the PoC verbatim from the sidebar rather than re-authoring it. A future supplementary handout (`adv-101-heartbleed-deep-walk.md`) expands the PoC to ~600-800 lines per the related topics list. Other CVE families (deserialization, SSTI, SQLi) deserve their own worked-PoC handouts at similar depth; this is named under §9 supplements as a forward-stretch round.
3. **Vocabulary-tier-only register; no exploit-development implementation depth.** Per the catalog-vs-classroom doctrine, even handouts that are not public-catalog-tier should respect the tier they target. This handout is a Belt-2-through-Belt-5 vocabulary scaffold, not an exploit-dev register; the §6 tool-defense primitives are named at vocabulary depth (one paragraph each), not at code-template depth. A separate `adv-101-tool-defense-engineering-register.md` (per §9 supplements) carries the implementation-depth treatment.
4. **Cross-track table is placeholder-grade in v1.** Section §8 names courses + estimated module/week/lab pickups but does not yet have exact section numbers for every cell. Rationale: this handout's authoring lane is moving in parallel with `spec-frontend`'s public-page distillation lane and `spec-curriculum`'s downstream cross-track linkage refinement. Forcing exact section numbers into v1 would block the round on coordination overhead; placeholders enable v1 to ship and the next-round refinement to fill in exact numbers as each course's module structure stabilizes.
5. **Discovery + intelligence references inlined into §7 rather than promoted to a top-level section.** The `adv-101.html` sidebar gave them a top-level section; this handout subordinates them to §7 (research-conferences) because they are companion infrastructure to the research set, not a parallel taxonomy. CISA-KEV + EPSS + NVD + vendor PSIRTs + MITRE-CWE pair as the *intelligence layer* researchers consume; conferences are the *publication layer* researchers contribute to. Subordinating intelligence to research preserves the conceptual coherence.

Pedagogy

1. **Vocabulary-tier handouts decouple recognition from reproduction.** A Belt-5 graduate doesn't need to have personally reproduced 22 CVEs end-to-end. They need to recognize the families when they read disclosure write-ups and name the primitives when they discuss exploitation strategy with colleagues. This handout makes the recognition-tier scaffold explicit and reproducible; per-track curricula supply the methodology-tier reproduction work against intentionally-vulnerable lab targets. The two tiers are pedagogically distinct. Confusing them produces curricula that are either under-rigorous (vocabulary without methodology) or unscaleable (every student reproduces every CVE end-to-end).
2. **One worked PoC at structural depth beats many taxonomy rows.** The Heartbleed §5 expansion is the pedagogically high-value addition. Three CVE-class tables are useful as taxonomy. The worked PoC (annotated byte sequence + literal vulnerable function + literal one-line fix) is what makes the bounds-check rule *memorable*. Students who have read the four-line `memcpy` and the one-line patch carry this rule into every protocol implementation, parser, and deserializer they ever write for the rest of their careers. The pedagogical value of one worked PoC of this quality exceeds the value of many additional table rows. Future supplements (§9) should preserve this principle: each cyber-track family deserves at least one worked PoC at structural depth before second-and-third worked PoCs are added at the same depth.
3. **The discovery-learning approach applies to vocabulary scaffolding.** Per `project_discovery-learning approach.md`, vocabulary scaffolding intentionally leaves space for students to discover detail through hands-on lab encounters. This handout deliberately does not exhaust each family. Race conditions, cryptographic flaws, side channels, and protocol-level design defects are each named as future-handout supplements rather than crammed in. The student who encounters Spectre/Meltdown for the first time during a CSA-201 micro-architecture lab brings real curiosity to the side-channel vocabulary; the student who memorized Spectre/Meltdown in Week 1 of SEC-101 brings memorization fatigue.
4. **The cross-track table makes the handout's central role explicit.** Section §8's per-course-per-module pickup register turns this handout from "one more reference document" into a *coordination artifact*: every cyber-track course points back to it, every cyber-track student reads it at known points in their academy progression, and the academy's CVE-vocabulary coverage is auditable as a single matrix. This is the canonical-anchor pattern: one document carries the vocabulary, many courses cite it,

and divergence between courses is detectable as cross-reference drift. The pattern parallels `cross-chapter-stdlib-service-reference.md` for the CSA-101 → Virtus-OS coordination lane.

5. **Tool-defense primitives are an "engineering-quality ladder" not a "checklist."** The §6 register names five primitives a Belt-5 graduate should *recognize*, but the academy's capstone tool implements only the first three. The pedagogical implication is that **engineering quality is a ladder, not a binary**. Students learn to read tool source by recognizing what could be present even when it isn't, and to design their own tool's roadmap by naming the next rung up. This frames tool development as a craft trajectory rather than a one-shot exercise; graduates carry the engineering-quality ladder mental model into post-academy work.

Supplements

1. **`handouts/cross-chapter-mitre-attack-academy-tour.md`, second-priority follow-on.** Walks the 14 MITRE ATT&CK Tactics (Recon → Impact) with one named technique per Tactic plus academy-lab pointer. Becomes a primary reference for PEN-101 + ADV-101 + SEC-101. Cross-references this handout's CVE families at the technique-detail level (e.g., T1190 Exploit Public-Facing Application points to §3 injection family + §4 auth-bypass family). Estimated authoring LoE: ~3-4 hr Opus on `munsonj`.
2. **`handouts/adv-101-heartbleed-deep-walk.md`. Supplementary reference.** Standalone expansion of §5 to ~600-800 lines: full RFC 6520 walkthrough, OpenSSL source-tree archaeology (which files / which versions), the historical disclosure timeline (April 2014 events), the broader defensive lessons from the post-Heartbleed scrutiny of OpenSSL (the LibreSSL fork, the Core Infrastructure Initiative funding, the OpenBSD audit). Becomes ADV-101 + SEC-101 supplementary reading. Estimated LoE: ~4-5 hr Opus on `munsonj`.
3. **`handouts/adv-101-tool-defense-engineering-register.md`. Supplementary reference.** Standalone expansion of §6: each of the five primitives expands into engineering-motivation + worked-code-sketch + test-pattern + documented-failure-mode-the-primitive-prevents. Becomes ADV-101 capstone reading; cross-references CSA-201 (where the tool implementation is built) and AI strand (ADV-102's responsible-AI-tool register). Estimated LoE: ~4-6 hr Opus on `munsonj`.
4. **`handouts/cross-chapter-research-conference-register.md`, academy-career-scaffold supplement.** Standalone expansion of §7: each conference gets its tradition, its publication venue (proceedings / talks / videos), how a Belt-5 graduate would target

it (paper submission / CFP timeline / sponsoring travel), and historical anchor talks every academy-track-graduate should be familiar with. Becomes a primary reference for the academy's career-pathway documentation. Estimated LoE: ~3-4 hr Opus on `munsonj`.

5. **`handouts/cross-chapter-owasp-llm-top10-cve-map.md`, ADV-102 anchor handout.** Single table mapping each LLM01-10 item to 1-2 real CVEs + academy lab/handout pointer; cross-references this handout's CVE-vocabulary set for the classical-CVE comparison axis. Becomes a primary reference for ADV-102 + AI-201 + AI-301. Estimated LoE: ~3-4 hr Opus on `munsonj` or `munsonj2.0` (depending on prompt-injection-depth boundary results).
6. **`handouts/re-101-embedded-iot-cve-family-taxonomy.md`, RE-101 anchor handout.** Standalone expansion of the RE-101-relevant rows from this handout (especially §4 Hardcoded credentials → ARRIS / Broadcom family) into a full embedded-IoT CVE family taxonomy: Ripple20 / URGENT/11 / BusyBox / Boa / Realtek-SDK / Broadcom-Wi-Fi / ARRIS / MediaTek SDK lineage. Cross-references this handout for the broader CVE-class context. Estimated LoE: ~4-5 hr Opus on `munsonj`.
7. **`handouts/cross-chapter-race-conditions-and-side-channels.md`. Future fourth-family supplement.** Race conditions (TOCTOU expansion + signal-handler races + thread-safety bugs), cryptographic flaws (timing oracles + padding oracles + RNG failures), and side channels (Spectre / Meltdown / power-analysis / EM emanations). This handout names Dirty COW (CWE-367) under §2 but the broader race-condition and side-channel topics deserve their own taxonomy at vocabulary depth; future handout. Estimated LoE: ~5-6 hr Opus on `munsonj`.
8. **Per-course CVE-vocabulary band declarations.** Each cyber-track course should declare in a comment-block which CVE-vocabulary depth it covers (Bronze: 0-2 CVEs / Silver: 3-7 / Gold: 8+). This handout's existence makes Gold achievable for any course that cites it (the citation imports vocabulary access for the student); per-course bands track adoption. Audit-pattern: the cross-track linkage table in §8 doubles as the band-declaration reference.
