CVE Snort 3 Rules Reference: Wireshark RCE Quartet (2026-05)

3,559 words · ~16 min read 2026-05-07 v2 (2026-05-07: cyber-use footnote per D7)

Course companion for: vca-mini-wireshark-cves-2026-05 (catalog page) + ADV-101 / SEC-101 / NET-101 **Scope:** Defensive: rule templates, rationale, false-positive considerations, tuning notes **Rule engine:** Snort 3.x (compatible with 3.1.x and later; see §4 for comparison with Suricata 7 sibling) **Parent handout:** `cve-lab-wireshark-rce-quartet-2026-05.md` (full CVE walkthroughs) **Suricata sibling:** `cve-suricata-rules-reference-wireshark-quartet-2026-05.md` **Version:** 2026-05-07 v2 (2026-05-07: cyber-use footnote per D7)

[Authorized under Anthropic acceptable-use cyber-research exception; see handouts/cross-chapter-anthropic-cyber-use-citation.md for policy details and academy provenance.]

Overview

This handout is the Snort 3 sister to `cve-suricata-rules-reference-wireshark-quartet-2026-05.md`. It covers the same four Wireshark CVEs disclosed in May 2026 (CVE-2026-5402, CVE-2026-5403, CVE-2026-5405, CVE-2026-5656), the same eight-section structure, and the same foundational level: rule templates that teach detection shape, not production drop-ins.

Syntax validation. No `snort` binary was available in the authoring environment. All rules below are manually validated against Snort 3.x documentation for keyword existence, header-line semantics, and option ordering. The same manual-validation precedent applies here as in the Suricata sibling.

Lab-harness gating. All hands-on rule testing runs against lab-owned, intentionally-vulnerable Wireshark 4.6.4 or 4.4.14 instances inside the academy `fwlab` container, or against pre-recorded `.pcap` / `.pcapng` files supplied by the instructor. The `--authorized-by` discipline applies unchanged (see §7).

---

§1: What This Handout Covers

For each of the four Wireshark CVEs, this handout provides:

- A one-section Snort 3 rule template with `sid` / `rev` / `gid` / `msg` / `classtype` / `reference` / `metadata` fields filled in
- A detection rationale that walks every option in the rule and explains why it is shaped the way it is
- False-positive considerations naming where each rule will fire on legitimate traffic
- Tuning notes for SOC deployment, including `fast_pattern` placement and `detection_filter` thresholds

Beyond the per-CVE sections, §4 compares the Snort 3 rules directly against the Suricata 7 siblings, §5 provides a cross-bug-class comparison table, and §6 maps each rule to the academy courses where students encounter it.

Every rule in this handout is a **teaching shape**: the goal is that a student reading it understands why each option is present and what invariant it tests. Adapting these templates for production SOC deployment requires environment-specific baseline profiling, false-positive tuning, and change-management approval beyond what this handout describes.

---

§2: The Four Wireshark CVEs at a Glance

CVE	Bug class	Dissector / component	Fixed in
CVE-2026-5402	Heap overflow via integer truncation (CWE-190 -> CWE-122)	TLS dissector, ECH extension parsing	4.6.5
CVE-2026-5403	Heap overflow via loop accounting failure (CWE-122 / CWE-787)	SBC codec plugin, `codec_sbc_decode()`	4.6.5, 4.4.15
CVE-2026-5405	Heap overflow via missing bounds check on uncompressed path (CWE-122 + CWE-120)	RDP dissector, ZGFX `rdp8_decompress_segment()`	4.6.5, 4.4.15
CVE-2026-5656	Path traversal + Lua auto-execution (CWE-22)	Profile import ZIP extraction	4.6.5, 4.4.15

All four CVEs affect Wireshark 4.6.0 through 4.6.4 and (for 5403, 5405, 5656) 4.4.0 through 4.4.14. Advisory references: wnpa-sec-2026-14 (CVE-2026-5402), wnpa-sec-2026-16 (CVE-2026-5403), wnpa-sec-2026-17 (CVE-2026-5405), wnpa-sec-2026-21 (CVE-2026-5656). Full walkthroughs: parent handout `cve-lab-wireshark-rce-quartet-2026-05.md` §1-§4.

---

§3: Per-CVE Snort 3 Rule Templates

§3.1: CVE-2026-5402: TLS Dissector ECH Integer Truncation

§3.1.1: THE BUG

The TLS dissector's ECH transcript-reconstruction loop holds length-field arithmetic in `uint16_t` locals. Attacker-crafted ECH extension length fields exploit truncation at 16-bit boundaries to drive a heap-buffer write beyond the allocated transcript buffer. Full structural walkthrough: parent handout §1.

§3.1.2: WHAT WE WANT TO DETECT

A TLS ClientHello carrying an ECH extension (type `0xFE 0x0D`) whose extension_length field is large enough to cause 16-bit arithmetic overflow in the vulnerable code paths; any value above 16 383 is a candidate for the truncation family.

§3.1.3: SNORT 3 RULE TEMPLATE

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (
    msg:"VCA-ADV CVE-2026-5402 TLS ECH extension oversized length field integer-
truncation shape wnpa-sec-2026-14";
    flow:to_server,established;
    service:tls;
    content:"|fe 0d|"; fast_pattern;
    byte_test:2,>,16383,0,relative,big-endian;
    reference:cve,2026-5402;
    reference:url,www.wireshark.org/security/wnpa-sec-2026-14.html;
    classtype:protocol-command-decode;
    metadata:cve 2026-5402, service tls;
    gid:1; sid:1000001; rev:1;
    # replace SID with site-allocated range per Snort community SID convention
)
```

§3.1.4: DETECTION RATIONALE

`alert tcp $EXTERNAL_NET any -> $HOME_NET any` uses `tcp` as the protocol rather than a TLS-specific protocol label. In Snort 3, TLS is an application-layer service identified by the `service:` keyword rather than the protocol field; the protocol field stays `tcp` for transport.

`flow:to_server,established` limits the rule to the client-to-server direction on an established TCP session. A TLS ClientHello always travels client-to-server; this halves the evaluation surface.

`service:tls` tells Snort 3's preprocessor pipeline to apply the TLS inspector to packets matching this rule. The TLS inspector decodes the TLS record and handshake layers, making the `content:` match operate against the decoded TLS handshake payload rather than raw TCP bytes. This is the Snort 3 equivalent of Suricata's `tls.handshake.type:1` sticky buffer; the `service:` keyword enables inspector-decoded matching without requiring a separate keyword per handshake-message-type.

`content:"|fe 0d|"; fast_pattern;` matches the ECH extension type bytes (0xFE 0x0D) in the ClientHello extensions list. The `fast_pattern` modifier marks this as the primary content match for Snort 3's pattern-matching fast path, improving rule-evaluation throughput by anchoring the first-pass Boyer-Moore search to the most distinctive two-byte sequence.

`byte_test:2,>,16383,0,relative,big-endian` reads the 2 bytes immediately following the ECH extension type (the extension_length field) and tests whether the value exceeds 16 383. The `relative` modifier anchors the offset to the end of the preceding `content:` match. This is the invariant: any ECH extension_length above 16 383 is a candidate for triggering 16-bit arithmetic truncation in the vulnerable code.

`reference:cve,2026-5402` and `reference:url,...` provide triage links in the alert output. Snort 3 uses the `reference:` option with comma-separated type and value; this is distinct from Suricata's `metadata:` field convention for the same information.

`gid:1` is the default Snort 3 generator ID. It is implicit in most deployments but including it explicitly avoids ambiguity in environments with custom preprocessor rules that use other GIDs.

`sid:1000001; rev:1;` assigns a local SID in the site-local range (1000001+). The SID assignment convention for Snort 3 site-local rules starts at 1000001; Snort 3's community ruleset uses SIDs in the 1-999999 range, and Snort 3 Subscriber rules use 1000000-1999999 (TALOS-allocated). Replace with your site's allocated range.

§3.1.5: FALSE-POSITIVE CONSIDERATIONS

The false-positive surface is identical to the Suricata sibling: any TLS client sending a legitimately large ECH payload (128-512 bytes is common; padding-aggressive implementations may exceed 8 192 bytes in edge cases) will fire this rule if the extension_length field reads above 16 383. Baseline ECH extension sizes in production TLS traffic before deploying this rule. In environments where ECH traffic is rare, alert volume will be low regardless of threshold choice.

§3.1.6: TUNING NOTES

The `service:tls` keyword requires the Snort 3 TLS inspector to be enabled in `snort.lua` (`inspect { tls = ... }`). Deployments that have disabled the TLS inspector for CPU-overhead reasons cannot run this rule correctly; re-enable the TLS inspector for the patch-rollout window.

For rate suppression, add a standalone `threshold` entry in your Snort 3 threshold configuration: `threshold gen_id 1, sig_id 1000001, type limit, track by_src, count 1, seconds 60;`. `classtype:protocol-command-decode` maps to Snort 3's `classification.config` priority 3 by default; raise to priority 1 or 2 in environments where Wireshark-analyst workstations are high-value targets.

---

§3.2: CVE-2026-5403: SBC Codec Loop Accounting Overflow

§3.2.1: THE BUG

The SBC codec plugin's `codec_sbc_decode()` loop never decrements a remaining-capacity counter and never checks available buffer space before each decode call. An RTP stream supplying enough SBC frames to accumulate more than 8 192 bytes of decoded PCM output overflows the fixed heap-allocated output buffer. Full walkthrough: parent handout §2.

§3.2.2: WHAT WE WANT TO DETECT

Anomalous SBC-over-RTP traffic on the monitored network: either the presence of SBC sync bytes in UDP packets (low-noise detection in environments where Bluetooth audio over IP is uncommon) or a burst of such packets from a single source that approximates the frame-count volume needed to trigger the overflow.

§3.2.3: SNORT 3 RULE TEMPLATE

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (
    msg:"VCA-ADV CVE-2026-5403 anomalous Bluetooth-audio RTP stream SBC sync byte
burst loop-accounting-overflow shape wnpa-sec-2026-16";
    flow:to_server;
    service:rtp;
    content:"|9c|"; fast_pattern;
    detection_filter:track by_src, count 20, seconds 1;
    reference:cve,2026-5403;
    reference:url,www.wireshark.org/security/wnpa-sec-2026-16.html;
    classtype:protocol-command-decode;
    metadata:cve 2026-5403, service rtp;
    gid:1; sid:1000002; rev:1;
    # replace SID with site-allocated range per Snort community SID convention
)
```

§3.2.4: DETECTION RATIONALE

`alert udp` uses UDP as the transport because RTP runs over UDP in all standard Bluetooth A2DP and VoIP deployments.

`service:rtp` instructs Snort 3 to apply the RTP inspector to matching packets. The RTP inspector identifies RTP streams and provides structured access to RTP header fields; without this keyword, the rule matches raw UDP bytes containing the SBC sync byte regardless of whether they are actually RTP.

`content:"|9c|"; fast_pattern;` matches the SBC sync byte (0x9C). Every SBC audio frame begins with 0x9C; its presence in a UDP payload is a reliable indicator of SBC-encoded audio. The `fast_pattern` modifier anchors Snort 3's fast-path search to this one-byte signature.

`detection_filter:track by_src, count 20, seconds 1` is Snort 3's rate-based rule activation mechanism. Unlike Suricata's `threshold:` keyword, Snort 3 uses `detection_filter:` to suppress rule activation until the match rate from a given source exceeds the configured threshold. The rule fires only when 20 or more matching packets arrive from the same source IP within one second, approximating the frame-count volume needed to trigger the 8 192-byte output buffer overflow. This is the Snort 3 equivalent of the Suricata sibling's `threshold: type both, track by_src, count 20, seconds 1`.

`reference:cve,2026-5403` and the URL reference provide analyst triage anchors.

§3.2.5: FALSE-POSITIVE CONSIDERATIONS

Same as the Suricata sibling: enterprise environments with Bluetooth-audio-to-network bridges running legitimate A2DP traffic will fire this rule during normal operation. Baseline the detection_filter count against known-good A2DP bridge traffic before deploying; in high-density A2DP deployments, raise the count threshold to 200+ per second. In environments where Bluetooth audio over IP is entirely absent, lower the threshold to 1 (any SBC sync byte in a UDP packet triggers an alert).

§3.2.6: TUNING NOTES

The `service:rtp` keyword requires Snort 3's RTP inspector to be active. In minimal Snort 3 deployments (surveillance or lightweight sensor mode), the RTP inspector may be disabled; remove `service:rtp` and accept the broader UDP match in that case, with the understanding that the rule will fire on any UDP stream carrying 0x9C bytes at the threshold rate.

`detection_filter` in Snort 3 is a single-option suppressor: the rule does not fire at all until the threshold is crossed, then fires once per threshold period. Contrast with Suricata's `threshold: type both` which fires once at the crossing and once again on each subsequent crossing. The Snort 3 behavior is slightly more conservative; document the difference in SOC runbook notes when both engines are deployed in parallel.

---

§3.3: CVE-2026-5405: RDP ZGFX Uncompressed-Path Missing Bounds Check

§3.3.1: THE BUG

The RDP dissector's `rdp8_decompress_segment()` fast-path for ZGFX uncompressed segments performs a direct `tvb_memcpy()` with no bounds check against the 65 536-byte fixed output buffer. Any segment with the uncompressed flag set and a declared payload length above 65 535 writes past the output buffer. Full walkthrough: parent handout §3.

§3.3.2: WHAT WE WANT TO DETECT

RDP traffic to port 3389 containing a ZGFX segment with the uncompressed flag byte followed by a 4-byte length field whose value exceeds 65 535.

§3.3.3: SNORT 3 RULE TEMPLATE

```
 alert tcp $EXTERNAL_NET any -> $HOME_NET 3389 (
    msg:"VCA-ADV CVE-2026-5405 RDP ZGFX uncompressed-path oversized payload
missing-bounds-check shape wnpa-sec-2026-17";
    flow:to_server,established;
    content:"|03 00|"; depth:2; fast_pattern;
    content:"|00|"; within:128;
    byte_test:4,>,65535,1,relative,big-endian;
    reference:cve,2026-5405;
    reference:url,www.wireshark.org/security/wnpa-sec-2026-17.html;
    classtype:protocol-command-decode;
    metadata:cve 2026-5405, service rdp;
    gid:1; sid:1000003; rev:1;
    # replace SID with site-allocated range per Snort community SID convention
)
```

§3.3.4: DETECTION RATIONALE

`alert tcp ... -> $HOME_NET 3389` scopes to TCP traffic on port 3389, the IANA-assigned RDP port. Environments running RDP on non-standard ports must adjust the destination port accordingly.

`flow:to_server,established` matches client-to-server on an established TCP connection. A malicious ZGFX segment flows from the attacker toward the RDP server; in the analyst-exploitation scenario, the traffic was captured to a `.pcap` that the analyst opens in Wireshark.

Note that `service:rdp` is omitted from this rule. Snort 3 does not ship a dedicated RDP inspector in its default build; the `service:rdp` keyword is not universally available. The rule falls back to raw TCP byte matching against port 3389 rather than inspector-decoded matching.

`content:"|03 00|"; depth:2; fast_pattern;` matches the TPKT version byte (0x03) and reserved byte (0x00) at the start of every TPKT-framed RDP packet. This anchors the rule to TPKT-over-TCP traffic.

`content:"|00|"; within:128;` matches a null byte (0x00) within the first 128 bytes following the TPKT anchor. The ZGFX segment descriptor byte for the uncompressed path is 0x00. The `within:128` window is a simplified match; a production rule would compute the precise ZGFX-segment offset from the TPKT length field using `byte_jump`, then match the uncompressed flag at the exact location.

Precision note (same as Suricata sibling §3.4). The `content:"|00|"; within:128` match is intentionally simplified for teaching. Any zero byte in the first 128 bytes of an RDP packet will match, not just the ZGFX uncompressed flag. A production-grade Snort 3 rule uses `byte_jump` on the TPKT length field to land at the ZGFX header offset before testing the flag byte. ADV-101 capstone work teaches the refined variant.

`byte_test:4,>,65535,1,relative,big-endian` reads the 4-byte field at offset 1 from the null-byte match (the ZGFX payload-length field immediately following the uncompressed flag) and tests whether it exceeds 65 535. Any value above that threshold exceeds the fixed `outputSegment` buffer.

§3.3.5: FALSE-POSITIVE CONSIDERATIONS

Identical false-positive surface as the Suricata sibling: zero bytes are common in RDP protocol framing (keyboard-input PDUs, null-padded fields), making the first content match imprecise. The `byte_test` on the subsequent 4-byte field reduces false-positive rate substantially because most RDP header fields carry values well below 65 535. Monitor-mode deployment with 48-72 hours of baseline inspection before enabling any suppression rules is the recommended rollout.

§3.3.6: TUNING NOTES

During the patch-rollout window for CVE-2026-5405, enable this rule in monitor mode on all network segments carrying RDP traffic. Disable once 4.6.5 / 4.4.15 is confirmed deployed across all analyst workstations. For generalization to analogous ZGFX CVEs in other RDP implementations (FreeRDP CVE-2022-39316, CVE-2022-39320), the rule shape is identical; update the `msg:`, `reference:`, `metadata:cve`, and `sid:` fields per additional CVE.

---

§3.4: CVE-2026-5656: Wireshark Profile ZIP Path Traversal

§3.4.1: THE BUG

Wireshark's profile-import ZIP extraction does not validate that each entry's resolved path stays within the extraction directory. An entry named `../../../plugins/auto-evil.lua` escapes the profile directory and lands in the Lua auto-load plugins path. On the next Wireshark startup, the plugin executes. Full walkthrough: parent handout §4.

§3.4.2: WHAT WE WANT TO DETECT

A ZIP archive being transferred over HTTP whose local-file-header entry names contain `../` path-traversal sequences, indicating a potential zip-slip payload targeting any application that auto-executes extracted content.

§3.4.3: SNORT 3 RULE TEMPLATE

```
alert http $EXTERNAL_NET any -> $HOME_NET any (
    msg:"VCA-ADV CVE-2026-5656 Wireshark profile ZIP path-traversal entry name
zip-slip shape wnpa-sec-2026-21";
    flow:to_server,established;
    http_client_body;
    content:"PK|03 04|"; fast_pattern;
    content:"../"; within:512; nocase;
    reference:cve,2026-5656;
    reference:url,www.wireshark.org/security/wnpa-sec-2026-21.html;
    classtype:policy-violation;
    metadata:cve 2026-5656, service http;
    gid:1; sid:1000004; rev:1;
    # replace SID with site-allocated range per Snort community SID convention
)
```

§3.4.4: DETECTION RATIONALE

`alert http` uses the `http` protocol label, which activates Snort 3's HTTP inspector (the `http_inspect` preprocessor). Unlike the TCP-with-service-keyword approach for TLS and RDP, `http` as a protocol label is a first-class Snort 3 inspector activation.

`http_client_body` is Snort 3's sticky buffer for the HTTP request body. After this keyword, subsequent `content:` matches operate against the decoded HTTP client request body rather than the raw TCP stream. This is the Snort 3 equivalent of Suricata's `file.data` sticky buffer for HTTP uploads; the name differs but the function is the same.

`content:"PK|03 04|"; fast_pattern;` matches the ZIP local file header magic bytes (ASCII `PK` = 0x50 0x4B, followed by 0x03 0x04). Every conforming ZIP file begins with this four-byte signature; `fast_pattern` anchors Snort 3's Boyer-Moore search to this distinctive sequence.

`content:"../"; within:512; nocase;` matches the path-traversal sequence `../` within 512 bytes of the ZIP magic. In a ZIP local file header, the filename field begins at offset 30; `within:512` allows for the fixed header bytes plus a reasonable filename before the

traversal sequence appears. `nocase` covers capitalization variants. The Windows-path variant `..\\` requires a separate content match or a `pcre:` option; a production rule adds both.

`classtype:policy-violation` matches the Suricata sibling. This is a file-policy violation (an archive should not contain path-traversal entries), not a protocol parsing error, so `policy-violation` is more accurate than `protocol-command-decode`.

§3.4.5: FALSE-POSITIVE CONSIDERATIONS

Identical to the Suricata sibling: conforming ZIP archives rarely contain `../` in entry names; the PKWARE specification does not permit path-traversal sequences in conforming archives. The primary false-positive risk is developer-generated ZIP archives containing relative symlinks on platforms where those are permitted, or archives with legitimate `../` substrings in documentation filenames rather than entry names. The `within:512` constraint limits false matches to files where the traversal sequence appears near the start of the archive's entry list. A production rule iterates over multiple local file headers via `byte_jump` for more complete coverage.

§3.4.6: TUNING NOTES

`http_client_body` in Snort 3 requires HTTP request-body reassembly to be enabled in the `http_inspect` configuration. In high-throughput deployments where request-body inspection is disabled for performance, this rule will not fire; enable body inspection for the relevant sensor placements.

For SMTP coverage, replace `alert http` with `alert tcp ... -> $HOME_NET 25` (or 587/465) and replace `http_client_body` with `file_data` applied to MIME-encoded attachments. For SMB coverage, use `alert tcp ... -> $HOME_NET 445` with the SMB inspector's file-data equivalent. `classtype:policy-violation` maps to Snort 3 priority 2 by default in `classification.config`.

---

§4: Snort 3 vs Suricata 7: What Differs Across This Rule Set

See `cve-suricata-rules-reference-wireshark-quartet-2026-05.md` for the Suricata 7 siblings of all four rules in this handout.

§4.1: Keyword-level differences

Feature	Suricata 7	Snort 3	Notes
App-layer TLS scoping	`tls.handshake.type:1` sticky buffer	`service:tls;` + `content:` match	Suricata uses per-handshake-field keywords; Snort 3 uses a service label to activate the inspector
File body inspection	`file.data;` (dot notation)	`http_client_body;` or `file_data;` (underscore)	Functional equivalent; syntax differs; `file_data` in Snort 3 applies to multi-protocol file extraction
Rate-based activation	`threshold: type both, track by_src, count N, seconds N;`	`detection_filter:track by_src, count N, seconds N;`	Snort 3's `detection_filter` suppresses all alerts until the threshold is crossed (does not fire-then-suppress); Suricata's `threshold: type both` fires once at crossing
CVE reference metadata	`metadata: cve CVE-2026-XXXX, ref wnpa-sec-2026-XX;`	`reference:cve,2026-XXXX; reference:url,...;`	Snort 3 uses dedicated `reference:` options; Suricata embeds references in `metadata:`
Performance hint	Implicit (Suricata auto-selects fast-path content)	`fast_pattern;` modifier required for explicit declaration	Snort 3 rewards explicit `fast_pattern` placement; omitting it forces the engine to heuristically choose
Generator ID	Implicit (GID 1 assumed)	`gid:1;` explicit	Including `gid:1;` in Snort 3 is best practice for local rules
Protocol label for TLS	`alert tls ...` accepted	`alert tcp ... service:tls;` preferred	Snort 3 accepts `alert tls` but resolves it to the TCP+service pathway; the explicit form is clearer

§4.2: SID Range Convention

The Suricata sibling uses placeholder SIDs in the `1000XXXX` (8-digit) range (10001402, 10001403, 10001405, 10001656), patterned after the CVE number for mnemonic value. The Snort 3 templates use the 7-digit range (1000001 through 1000004), which is the conventional Snort 3 local-rule starting range. In mixed Suricata+Snort 3 deployments, maintain a single site-wide SID registry to avoid collision between the two engines' local rules.

§4.3: Deployment Footprint Differences

Suricata 7 is the more common choice in open-source NSM deployments (Security Onion, Zeek+Suricata stacks, commodity cloud-based sensors). Snort 3 is the canonical choice in Cisco-sponsored enterprise SOCs, federal-agency detection platforms, and environments where the TALOS Subscriber ruleset is a primary feed. Both engines share the same core rule syntax for 80-90% of rule options; the differences above matter primarily for rules that use app-layer inspector keywords, rate-based filtering, and metadata formatting.

For students in SEC-101 and ADV-101, both dialects appear in real-world SOC environments. Cross-engine fluency is the academic goal; the four rules in this handout paired with the four in the Suricata sibling give eight concrete data points to compare.

---

§5: Cross-Bug-Class Shape Comparison

CVE	Bug-class shape	Delivery vector	Snort 3 primitive	Detection difficulty	Genera
CVE-2026-5402 (TLS/ECH)	Integer truncation -> heap overflow	Live TLS capture or crafted `.pcapng`	`service:tls` + `content:` ECH type + `byte_test` on length field	Medium; ECH extension_length threshold requires baseline calibration	Any TLS extensio inconsist inner/ou length fi recurs a TLS-diss CVEs
CVE-2026-5403 (SBC codec)	Loop accounting failure -> heap overflow	RTP-over-Bluetooth capture	`service:rtp` + `content:` SBC sync byte + `detection_filter` volumetric	Low precision without A2DP baseline; high recall	Any fixe output-b codec di where accumul decode anomalo large
CVE-2026-5405 (RDP/ZGFX)	Missing bounds check on uncompressed fast path -> heap overflow	ZGFX segment in RDP capture or live traffic	TPKT `content:` anchor + `content:` uncompressed-flag + `byte_test` on length field	Medium; simplified uncompressed-flag match generates false positives from other zero bytes in RDP framing	Any pars fast-path path asymme validatio recurs a RDP impleme
CVE-2026-5656 (Profile/ZIP)	Path traversal + Lua auto-execute	HTTP/SMTP file transfer	`http_client_body` + `content:` ZIP magic + `content: ../` pattern	Low false-positive rate in enterprise environments; requires HTTP body reassembly enabled	Zip-slip i applicati combini archive extractio auto-exe of extrac content

Reading the four rules together, the detection primitives fall into two families. The three heap-overflow CVEs (5402, 5403, 5405) each produce a numeric-field invariant: a length, a count, or a capacity value that the attacker drives past a boundary. The appropriate Snort 3 primitives are `byte_test` and `detection_filter`, both of which test

numeric values against thresholds. The zip-slip CVE (5656) produces a string-pattern invariant: the path-traversal sequence `../` in a filename field. The appropriate primitive is a `content:` match against file-body bytes. This two-family structure mirrors the structural divide in the bug classes: numeric bounds violations vs. path-validation logic bugs.

---

§6: Cross-Track Linkage

Course	CVE-2026-5402 (TLS/ECH)	CVE-2026-5403 (SBC codec)	CVE-2026-5405 (RDP/ZGFX)	CV
SEC-101 Module 4	Snort 3 rule template introduced as reference; students read each line and match it to the integer-truncation bug-class shape; compare against Suricata sibling to begin cross-engine fluency	Anomaly-detection pattern introduced; `detection_filter` vs Suricata `threshold` comparison surfaces cross-engine behavioral difference (suppress-until-crossed vs. fire-and-suppress)	Simplified uncompressed-flag match introduced; students identify the simplified match as the motivation for production refinement; FreeRDP analogy named	`ht` Su co re eq sy
ADV-101 Belt-5	Capstone IDS track: students write Snort 3 rules from packet captures and compare against both this template and the Suricata sibling; evaluate which engine's rule is more ergonomic for this CVE class	Optional Zeek script capstone: students implement a per-session SBC decoded-byte counter and compare the precision of inspector-level tracking against the `detection_filter` approximation	Students produce a refined Snort 3 rule using `byte_jump` on the TPKT length field to compute the precise ZGFX header offset; compare against the simplified template's FP rate	St ru lo op va +
NET-101 Wk 8 + Wk 11	Week 8 (TLS module): Snort 3 `service:tls` introduced as the inspector-activation pattern for app-layer detection; contrast with raw-TCP matching	Week 11 (RTP/VoIP module): `service:rtp` + `detection_filter` pattern anchored as 2026-currency; students contrast with network-flow-level RTP anomaly detection	Not a primary NET-101 anchor; named in cross-CVE comparison as the exception (no Snort 3 RDP inspector available in base build)	No in co tra th mo

---

§7: Authorization and Safe-Use Reminder

All hands-on rule testing in this module runs against lab-owned, intentionally-vulnerable Wireshark 4.6.4 or 4.4.14 instances inside the academy `fwlab` container, or against pre-recorded `.pcap` / `.pcapng` files supplied by the instructor. Production analyst workstations are never the test target.

The `--authorized-by` discipline that ADV-101 enforces for the SB6141 capstone applies unchanged: a written authorization for the specific lab target, a controlled capture-file provenance chain, and explicit refusal to test against systems the academy does not own. SOC teams adapting these rule templates for production work own their own production-tuning, change-management approval, and regression-testing discipline; those processes are outside the scope of this teaching handout.
