

AL-KERNEL Perl CVE Triage Pipeline

Architecture, Neural-Network Baseline, LLM Review Stages, Prompt Semantics, and Final Reconciliation (generated on 19 Sept 2026 by LLM based on source-code analyses and LLM prompts analyses, and mark “[AL READ/AGREED]” below means that Alexander Larkin read this part of text carefully and agreed with it and sometimes added some phrase from himself manually to this document too)

The idea is first to have documentation for historical solution by Alexander Larkin (for automatic Kernel CVEs classification regarding impact level) and next step going to be one more document about new similar solution for Aegis

Implementation documented: `alkernel_curver_daemon(1).pl`

Document status: implementation-derived technical documentation. This document describes the behavior visible in the supplied Perl source. Where rationale is inferred rather than explicitly encoded, it is labeled as interpretation. *[AL READ/AGREED]*

Scope: the legacy/Perl AL-KERNEL implementation, including its daemon interface, CVE and patch processing, 49-feature neural-network classifier, CVSS/triage-hints LLM call, ActionableScore LLM call, KERNEL_PANIC validation, post-promotion HIGH review, deterministic reconciliation, persistence, and output. *[AL READ/AGREED]*

1. Executive Summary

AL-KERNEL is a hybrid Linux kernel CVE triage system. The supplied Perl implementation combines conventional program logic, hand-engineered patch/CVE features, a small feed-forward neural network, CVSS-oriented LLM analysis, a separate ActionableScore LLM analysis, and two specialized LLM precision-review stages. The final severity is therefore not the direct output of any single model. *[AL READ/AGREED]*

The legacy neural network produces one of four internal classes: HIGH, MODERATE, LOW, or NONE. The current operational vocabulary used by later experiments is not identical to these internal names. In particular, the Perl implementation uses a KPANIC flag on MODERATE cases to represent a special review-routing state comparable to what later work calls MODERATE_KPANIC or MODERATE7. HIGH is the Perl implementation's high-severity class and is broadly comparable to the later IMPORTANT operational class, but this document preserves the source-code terminology whenever describing actual control flow. *[AL READ/AGREED]*

The key architectural property is layered reconciliation. The neural network establishes a baseline. The first LLM call provides a paranoid CVSS vector, CWE information, an English description, and a manual-check recommendation. The second LLM call independently computes an ActionableScore and semantic primitive classification. Deterministic Perl rules then promote or demote the baseline. If KPANIC is present, a dedicated KERNEL_PANIC review can remove a false-positive panic/review signal or decide that non-crash evidence still supports high severity. Cases promoted to HIGH are additionally subjected to a post-promotion precision review. *[AL READ/AGREED]*

This design intentionally separates several questions that are often conflated: Is the bug technically memory corruption? Is kernel panic supported? Is there a strong non-crash exploitation primitive? Should the CVE receive manual review? Is automatic LOW closure safe? The implementation is historically evolved and contains explicit workarounds and threshold rules, but the overall direction is a safety-biased cascade followed by targeted false-positive filtering. *[AL READ/AGREED]*

1.1 High-level execution model

```
CVE request / patch override
|
v
CVE, patch, applicability and source-data acquisition
|
v
49-feature extraction from description / patch / metadata
|
v
Legacy neural network: 49 -> 123 -> 4
|
+--> HIGH
+--> MODERATE
+--> LOW
+--> NONE
|
v
Initial deterministic compatibility/workaround adjustments
|
v
LLM Stage 1: general triage / paranoid CVSS / CWE / manual-check hints
|
v
```

```

LLM Stage 2: ActionableScore + primitive / exploitability analysis
|
v
Deterministic CVSS + ActionableScore reconciliation
|
+--> severity promotion/demotion
+--> KPANIC review-routing state
|
+--> if KPANIC: LLM Stage 3, KERNEL_PANIC false-positive review
|
+--> if pushed to HIGH: LLM Stage 4, post-promotion HIGH review
|
v
Final Perl severity, flags, report text, cache/output [AL READ/AGREED]

```

1.2 Source-of-truth rule used in this document

The Perl source is treated as the source of truth. Prompt text is documented as instructions supplied to an LLM, not as guaranteed behavior. Deterministic Perl conditions are documented separately from prompt intent. Comments that describe a reason for a workaround are treated as implementation notes, not as proof that every runtime case follows that rationale.

2. Runtime Modes and Service Interface

The program supports both a single-CVE invocation and a network-listening service mode. The command-line usage accepts a CVE identifier, a skip mode, and optionally a patch identifier or patch file. The special CVE argument NETWORK switches the program into listening mode. *[AL READ/AGREED]*

In network mode, the program creates a TCP socket bound to 10.0.109.87:8080 with a listen backlog of 50. Each accepted request is expected to contain a CVE identifier, a numeric skip flag, and an optional patch identifier/path-like token. The daemon processes one request through the same classification pipeline and sends a tab-separated result back to the client. *[AL READ/AGREED]*

The source contains daemonization support through `Daemon::Daemonize`, but the actual `daemonize()` call is commented out for the current 'simple service' deployment. PID-file handling and HUP/USR1 handlers remain present. The reason that program actually works in daemon mode, but in simplified mode where daemonization not required. *[AL READ/AGREED]*

Before new analysis, the program checks `/list/llm/<CVE>`. If a prior cached result already contains the same CVE and a LOW, MODERATE, or HIGH result, it returns that existing result rather than re-running the expensive analysis. Specialized LLM traces (results of different type prompts to LLM for specific CVE) are also persisted under `/list/llm` with `AS_`, `KP_`, and related prefixes. *[AL READ/AGREED]*

3. Core Severity Model

Internal value	Perl name	Meaning in this implementation	Approximate later operational analogue
0	HIGH	Highest normal	IMPORTANT

		NN/final severity class	
1	MODERATE	Moderate severity; may carry KPANIC for mandatory-review- style routing	MODERATE or MODERATE_KPANIC
2	LOW	Low severity; candidate for low- touch/automatic handling	LOW
3	NONE	Not a security issue / not included; not supported by some downstream integration	No direct current four-class analogue (and in resulted output just being used as Low)

The constants are represented as one-hot neural-network targets. HIGH=[1,0,0,0], MODERATE=[0,1,0,0], LOW=[0,0,1,0], and NONE=[0,0,0,1]. The winner() result is an integer index 0-3. *[AL READ/AGREED]*

The program contains a compatibility conversion from NONE to LOW when the NN result is consumed by OSIM-oriented behavior because NONE is not supported there. It also contains post-NN workarounds that can lower HIGH to MODERATE when a build/applicability condition says the component is not built (the idea that no affected products, so can lower particular this CVE impact level from Important to Moderate-7.0), or raise LOW to MODERATE when a pre-existing KPANIC-related condition requires it. *[AL READ/AGREED]*

3.1 MODERATE plus KPANIC

KPANIC is not a fifth neural-network class. It is a flag carried in the per-CVE flags string. A MODERATE result with KPANIC is treated specially by later thresholds and LLM review. In prompt language this state is frequently called MODERATE7: a Moderate issue that remains visible and is intended to receive manual security review. *[AL READ/AGREED]*

This distinction is essential. Removing KPANIC can change workflow semantics without changing the numeric severity from MODERATE. Conversely, adding KPANIC can make a Moderate case eligible for escalation to HIGH under ActionableScore rules. *[AL READ/AGREED]*

4. Legacy Neural-Network Baseline

The baseline classifier uses AI::NeuralNet::Simple with 49 input units, 123 hidden units, and 4 output units. The model is trained at process startup from an embedded corpus of CVE labels in %answers and feature vectors generated by the same extraction machinery. *[AL READ/AGREED]*

```
my $net = AI::NeuralNet::Simple->new(49, 123, 4);
```

Training labels are stored directly in the Perl source as CVE-to-class assignments. The corpus includes historical HIGH, MODERATE, LOW, and NONE examples. The program constructs feature vectors for those CVEs, trains the network, performs embedded self-tests, and then calls winner() on the current CVE's 49-element vector. *[AL READ/AGREED]*

AL-KERNEL Perl CVE Triage Pipeline - implementation documentation

4.1 The 49 feature inputs

#	Feature
1	KASAN evidence
2	lock-related change
3	write-related signal
4	read-related signal
5	printf/fix-printing signal
6	remote signal
7	danger signal
8	BPF
9	NULL pointer
10	disk/storage
11	initialization
12	warning-only
13	out-of-bounds
14	deadlock
15	trace
16	USB
17	no-memory-check
18	virtualization
19	simple fix
20	netfilter
21	NVMe
22	race
23	leak
24	error path
25	UAF
26	networking
27	not-included component
28	skb
29	DMA
30	logger
31	improvement-only
32	kernel-panic-plus-UAF
33	timer

34	Spectre
35	hardware
36	server-to-client failure
37	TIPC
38	debugfs
39	Linus-related signal
40	outdated CVE
41	boot
42	test
43	qdisc
44	IPv6
45	io_uring
46	syzbot
47	kernel panic
48	memory
49	packet

Most of these inputs are heuristic indicators derived from patch text, commit metadata, subsystem names, CVE descriptions, or surrounding processing. The network therefore does not consume raw source text. It consumes a compact feature representation produced by Perl regexes and procedural extraction. *[AL READ/AGREED]*

Because the same extraction code is used to construct training vectors and the current vector, changes to feature extraction can alter both training behavior and inference behavior. This is one reason the source contains many regression/self-test calls after training. *[AL READ/AGREED]*

4.2 Embedded training and self-tests

The source contains a large %answers mapping of CVEs to one-hot classes. For CVEs with a non-empty generated trainset, the network is trained against the corresponding expected answer. A large collection of TestRes() calls checks that known vectors still produce expected class indices. *[AL READ/AGREED]*

These tests are implementation-level anti-regression checks for the legacy NN. They do not validate the later LLM reconciliation layers. *[AL READ/AGREED]*

5. Input Acquisition and Pre-LLM Processing

Before the LLM stages, the Perl program performs substantial CVE and patch processing. It can operate from the normal kernel CVE flow or from an explicitly supplied patch identifier/file. The code retrieves and parses patch/CVE material, checks applicability and previously fixed states for RHEL-related trees, extracts changed paths and text, normalizes HTML, and derives the 49 NN features. *[AL READ/AGREED]*

The implementation also records flags and auxiliary values used later by reconciliation, including KPANIC and subsystem/bug-class indicators. The resulting HTML/text representation of input patch data (that is input CVE) is stored in \$htmltext and becomes the primary natural-language input to the LLM stages. The HTML tags of course removed before being used as input to LLM prompts. *[AL READ/AGREED]*

The exact patch-acquisition logic is historically heterogeneous and contains fallbacks for multiple page/patch representations. Documentation consumers should treat this layer as data preparation rather than a security decision engine: its main security effect is through the features and textual evidence it makes available downstream. *[AL READ]*

6. LLM Transport and Conversation State

All four LLM stages use the OpenAI Responses API through HTTP::Tiny. The API key is read from /API_KEY. Requests use a long timeout and truncation='auto'. The implementation currently names model gpt-5.5 in the supplied source. *[AL READ/AGREED]*

The code stores previous_response_id and a local history array. If previous_response_id is present, it is supplied on the next LLM request, allowing multi-turn state. Responses are read from output_text when available, with a fallback that walks output message/content objects and concatenates output_text fragments. For each CVE the analyses starts from scratch, so older history of previous LLM prompt of previous CVE cleaned. *[AL READ/AGREED]*

Each stage writes diagnostic state/output files. At the end of the CVE pipeline the code reloads /state.json, effectively refreshing the shared state object for subsequent work. Because state handling is shared historical infrastructure, documentation should not assume that every stage is logically independent merely because each has a separate prompt. *[AL READ]*

7. LLM Stage 1 - General Triage, CVSS, CWE, and Manual-Check Hints

request_llm_hints() is the oldest and least formally structured of the LLM stages. It prepends a large accumulated instruction block to the CVE/patch text. The instruction block mixes Russian operational guidance with English security-analysis rules and examples. *[AL READ/AGREED]*

7.1 Purpose

- Produce one or more CVSS v3.1 vectors, including a higher/paranoid interpretation when defensible.
- Identify relevant CWE identifiers.
- Produce a concise English report description suitable for later reporting.
- Choose one of the fixed manual-check phrases: YES REQUIRES MANUAL CHECK or NO MANUAL CHECK REQUIRED COULD BE AUTOCLOSED.
- Bias against dangerous false negatives while still prohibiting unsupported conversion of crash-only/resource bugs into memory-corruption or privilege-escalation classes. *[AL READ/AGREED]*

7.2 Important prompt principles

The prompt explicitly asks for patch/source analysis rather than blind trust in a short commit description. It emphasizes a 'highest still-defensible' paranoid CVSS, but repeatedly states that paranoid escalation must remain within the vulnerability class supported by evidence. *[AL READ/AGREED]*

Privilege is treated carefully. Local code execution or a local process context normally implies PR:L rather than PR:N. Host-admin assumptions should not be introduced merely because a capability is mentioned. Network-adjacent storage such as NVMe is specially calibrated so that deployment reachability is not automatically treated as public-network AV:N. Means that this LLM prompts specifically asks to calculate privileges part of CVSS as good as possible. *[AL READ/AGREED]*

The prompt contains special rules for races, resource leaks, KVM/MMU reverse mappings, stale mappings, UAF candidates, and manual-review sensitivity. It also instructs the model to use fixed output phrases that Perl can parse. *[AL READ/AGREED]*

7.3 Perl-side parsing

The Perl parser extracts every CWE-N token and every CVSS:3.1 vector matching the expected metric layout. It computes the base score of each vector with the CVSS Perl module and chooses the highest-scoring vector as the 'best (paranoid) CVSS'. The “paranoid” CVSS value (higher numerical value from few possible CVSS vectors) later being used for decisions regarding impact level together with other information, but not being used as is (so weights only part in overall later decision). *[AL READ/AGREED]*

A preferred-CWE map decorates selected CWE values with one or two asterisks in the generated summary. The longest quoted report-like fragment matching the permitted character set is retained as DESCR (and this Descr goes to private CMT of the flaw in OSIM). The manual-check fixed phrase is also copied into the summary. *[AL READ/AGREED]*

```
The best (paranoid) CVSS is '<vector>'
*CWE-...
Other CVSS '<vector>'
BEST CVSS score: '<numeric score>'
DESCR '<report text>'
YES REQUIRES MANUAL CHECK
or
NO MANUAL CHECK
```

The returned Stage-1 summary is later flattened to semicolon-separated text and becomes both a direct input to deterministic CVSS rules and contextual input to ActionableScore (that is next LLM prompt in cascade of rules and prompts). *[AL READ/AGREED]*

8. LLM Stage 2 - ActionableScore

`request_llm_calculate_awareness()` is the largest and most formal prompt in the Perl implementation. It is a second-opinion security triage system rather than a CVSS replacement. It is explicitly designed to distinguish ordinary Low/Moderate issues, actionable Moderate cases that should remain in review, Important-class vulnerabilities, and high-end Important/possible Critical candidates. Originally author noted that CVSS doesn't work good at all and this value doesn't help so much to analyst regarding decisions about CVE impact level, so actually human analyst does

more and different type of analyses apart from CVSS, so it make an idea to design a new type of value that is ActionableScore and better for decision about Kernel CVE impact level than CVSS value is. The ActionableScore designed with the usage of OpenAI LLM (ChatGPT), so if older Perl code and first LLM Prompt designed mostly by author himself, then next LLM prompts designed primarily with the more powerful LLM model to make it work as good as possible in previous LLM model version. However such way of designing prompts similar like if for programming happened in small steps with running tests after each step (like improve LLM prompt for the new type of CVE and then check that it still works good for other already used before types of CVEs), so you cannot think that only LLM did the work and actually it was controlled by author. *[AL READ/AGREED]*

8.1 Machine-readable contract

```
ActionableScore=NUM
ActionableScoreLower=NUM          # when lower/conservative differs

PrimitiveClass=NONE|CLASS_A|CLASS_B|CLASS_C
RealUAF=YES|NO
HostAdminRequired=YES|NO|UNKNOWN
NamespaceScopedPrivilege=YES|NO|UNKNOWN
ImportantCandidate=YES|NO
ManualReview=NO|RECOMMENDED|REQUIRED
AutoDowngradeAllowed=YES|NO

Final recommended bucket: **...**::
```

CLASS_A denotes a directly established strong Important-preserving primitive. CLASS_B denotes a real potentially exploitable corruption/lifetime primitive that requires leverage analysis. CLASS_C denotes primarily crash/DoS/invalid-state behavior without an independently useful corruption primitive. NONE means no meaningful corruption, lifetime, ownership, authorization, or similar security primitive is established. *[AL READ/AGREED]*

8.2 Evidence philosophy

The prompt makes the patch/commit authoritative for the low-level root cause but allows supplied authoritative advisory or exploit evidence to establish reachability, victim objects, privilege, confidentiality/integrity impact, boundary bypass, LPE, or an exploit chain that the upstream commit does not describe. *[AL READ]*

A recurring principle is that absence of proof of a strong primitive is not automatically proof of safety, but a generic dangerous bug class is also not enough to invent exploitation. The prompt repeatedly separates demonstrated primitive, theoretical consequence, and demonstrated high-impact consequence. *[AL READ]*

Negative scoring signals require concrete evidence. Uncertain words such as 'typically', 'probably', or 'may require' should normally be recorded as uncertainty rather than converted into a negative point. *[AL READ]*

8.3 Major ActionableScore rule families

Rule family	What it contributes
Reachability and exploitability	Remote/network trigger, firmware-mediated influence, local unprivileged trigger, high-control data-plane API, practical exploit chain.

Active-object lifetime influence	Whether the same attacker can repeatedly create/replace/update objects and drive the lifetime transition relevant to stale use.
Privilege-escalation leverage	Controlled reclaim, directed writes, object replacement, callback influence, credential/refcount takeover, semantic reinterpretation.
Memory corruption taxonomy	Strong corruption, constrained buffer overwrite, weak/indirect corruption, read-only OOB handling.
UAF/RCU/lifetime	Real UAF base signal, active structured object vs fixed stale storage, repeated release/refcount mutation, structured stale-object leverage.
KVM calibration	Separates malicious guest, host VMM process, and host administrator; prevents combining different principals into a fictional stronger attacker.
Race/timing	Treats genuine timing difficulty as a constraint but avoids automatic downgrades solely because a bug is a race.
Privilege and namespaces	Separates host-admin-equivalent privilege, reduced/delegated capability, namespaced capability, ordinary user, and unknown.
Availability/resource exhaustion	Separates immediate strong DoS from bounded leak/exhaustion requiring sustained abuse.
Filesystem/COW/ownership	Recognizes shared-page, page-cache, reflink/clone, and ownership/COW boundary violations as potentially strong security primitives.
Authorization/security boundary	Recognizes missing mandatory authorization, ptrace/pidfd-like access, sensitive-resource theft, and security-state lifetime mismatch.
Semantic reinterpretation	Treats validation-to-use changes in semantic meaning as strong when attacker-controlled bytes become active operations/resources.
Subsystem-specific calibration	Networking, qdisc/TAPRIO, BPF, device/DMA, KVM, filesystem, and configuration exposure rules.
Call-site/return/error-path analysis	Requires examination of actual helper semantics, rollback paths, return-value

	consequences, and whether cleanup wording hides a real lifetime mutation.
--	---

[AL READ]

8.4 UAF handling

The prompt devotes extensive space to preventing both UAF inflation and UAF dismissal. A concrete UAF or equivalent lifetime violation receives a base signal, but CLASS_A requires stronger exploitation evidence. Fixed stale storage such as a DMA/URB buffer remains different from an active structured object. A cleanup/destructor stale dereference may remain constrained CLASS_B if the demonstrated result is only a fault. [AL READ]

A special repeated-release/refcount rule recognizes that a second lifetime-mutating operation through the same potentially dangling reference is stronger than a passive stale read. It can justify manual review and a higher floor without automatically becoming arbitrary write or CLASS_A. [AL READ]

The prompt also has explicit floors for real UAF cases, exceptions for BPF/privileged control planes, RCU lifetime mismatch rules, teardown/per-CPU calibration, and a distinction between crash corruption and non-crash exploitation. [AL READ]

8.5 KVM attacker-role separation

For virtualization bugs the prompt requires separation of malicious guest code, the host userspace/VMM process with /dev/kvm access, and host administrator/root. A complete trigger that requires actions from more than one principal is a compound trigger unless evidence shows that one attacker can induce all stages. This prevents accidental guest-to-host severity inflation by combining guest-controlled page tables with VMM-controlled memslot or MMU lifecycle operations. [AL READ]

8.6 Authorization and COW/ownership primitives

The ActionableScore prompt contains explicit high-value treatment for authorization bypasses and ownership/COW violations. It can preserve Important-grade review even when there is no classic memory-corruption exploit. Examples include mandatory authorization checks that disappear during task/object teardown, privileged FD/resource access, shared-page/page-cache corruption, or writes that cross an ownership/COW boundary. [AL READ]

This is a major design evolution relative to a pure bug-class classifier: security-boundary semantics can independently justify manual review or high severity. [AL READ]

8.7 Score interpretation

Score	Prompt interpretation
0-2	Ordinary Moderate / Low-like
3-4	Borderline; manual review recommended
5-6	Actionable Moderate; should not be automatically closed
7-8	Strong Important candidate

9+	High-end Important / possible Critical candidate
----	--

The prompt additionally distinguishes a conservative/lower score from a paranoid score. The higher value is emitted as ActionableScore and the conservative value as ActionableScoreLower. This idea is similar as in CVSS calculation: should not calculate only one severity-level value, but instead would be good to calculate two values in parallel where one is paranoid kind one and other more conservative one. *[AL READ/AGREED]*

8.8 Critical implementation detail: the Perl score is an average

The deterministic pipeline does not directly use the emitted paranoid score. Perl parses ActionableScore and ActionableScoreLower, limits an excessive lower/upper gap to at most three points, and then returns $\text{ceil}((\text{upper} + \text{lower}) / 2)$. If ActionableScoreLower is absent, it is set equal to ActionableScore.

```
if ($sc != $lsc && $lsc < $sc && $sc - $lsc > 3) {
    $lsc = $sc - 3;
}
return (ceil(($sc+$lsc)/2), ...);
```

This averaged value is the \$alimpactscore used by subsequent deterministic thresholds. Consequently, prompt documentation and runtime behavior must not equate 'ActionableScore=7' in raw LLM text with an effective Perl score of 7 unless the lower score is also 7 or absent. *[AL READ]*

9. Deterministic Reconciliation: CVSS Layer

After the NN result and both initial LLM analyses are available, Perl applies a sequence of threshold-based corrections. These are procedural rules, not learned behavior. *[AL READ/AGREED]*

The first group uses the Stage-1 best CVSS score, selected CVSS metrics, CWE/description text, manual-check phrase, and existing flags. Representative behaviors include:

- Lower HIGH to MODERATE/KPANIC when the guessed CVSS is below the configured high-severity threshold.
- Raise MODERATE to HIGH for sufficiently high CVSS, with additional CIA-high conditions for some ranges and exclusions for BPF, btrfs, and NVMe cases.
- Lower selected local, low-C/I Moderate cases toward regular MODERATE or LOW when manual review is not requested.
- Lower some HIGH NULL-pointer/deadlock/resource-leak-like cases when the CVSS and attack-vector evidence do not support high severity.
- Raise LOW to MODERATE/KPANIC when Stage 1 explicitly requires manual review or when a sufficiently high CVSS conflicts with an otherwise low result.

The exact ordering matters because the source uses a \$lowered flag to prevent some later CVSS rules from repeatedly lowering the same case. This is an example of historical procedural state that should be preserved when reimplementing behavior. The original idea by author was that LLM gives mistakes often and cannot rely only on results from LLM request, so should use some simple rules (kind of IF/ELSE logic) that has more weight than answer from LLM, so bad answer from LLM

can decrease or increase impact level by one step, but should not crack it in catastrophic way. Later author observed that LLM can return different answers between runs (for same input data) and started to think what can do about this, so to get same Impact level results for same input data from run to run with same CVE (patch) input. *[AL READ/AGREED]*

10. Deterministic Reconciliation: ActionableScore Layer

After CVSS corrections, the effective averaged ActionableScore is applied. The main transitions visible in the source are:

Condition	Result / effect
ActionableScore >= 5, severity MODERATE/LOW, no KPANIC	Set severity MODERATE and add KPANIC/MODERATE7 routing.
ActionableScore > 5, severity not HIGH, KPANIC present	Promote to HIGH; mark pushed_to_high.
ActionableScore >= 6, severity not HIGH	Promote to HIGH; mark pushed_to_high.
ActionableScore < 6, severity HIGH, best CVSS <= 7	Lower to MODERATE/KPANIC.
ActionableScore >= 3, severity LOW	Raise to MODERATE/KPANIC.
ActionableScore < 4, severity HIGH/MODERATE with KPANIC	Set regular MODERATE and remove KPANIC.
ActionableScore < 3, severity MODERATE, no Stage-1 manual-check requirement, downgrade-count guard passes	Lower to LOW and remove KPANIC.

A local \$dec_cnt counter limits chained downward decisions in this block. The flags string records the reason for each transition, which is valuable for post-hoc debugging and should be considered part of the implementation's observability contract. *[AL READ/AGREED]*

11. LLM Stage 3 - KERNEL_PANIC Validation

request_llm_kpanic() is invoked when KPANIC remains present after the initial reconciliation. Its primary role is explicitly described as a false-positive filter for KERNEL_PANIC. Its secondary role is independent: when panic is unsupported, decide whether non-crash evidence still requires preserving HIGH/IMPORTANT or whether HIGH-to-MODERATE7 is safe. Overall idea by author was that original KPANIC value being calculated without NN usage on RegExps only, so very often KPANIC being selected aggressively (for cases where not required really), so need to use separate LLM prompt to decrease (remove) KPANIC when strong evidence by LLM prompt work that should remove it. *[AL READ/AGREED]*

11.1 Why panic and severity are separated

The prompt states that panic validity and severity downgrade are independent. A bug can lack credible kernel-panic evidence yet still contain a strong non-crash exploitation primitive. Conversely, a real crash/UAF does not automatically justify HIGH. *[AL READ/AGREED]*

The prompt's operational model defines MODERATE7 as a review bucket rather than a low-severity sink. This allows the reviewer to downgrade uncertain HIGH cases without losing human review. Overall you can think about it that no reason to keep everything that looks as potentially dangerous as Important, but less dangerous could be decreased to Moderate 7.0 (that was original architect kind of idea by author about a year ago and separate support of Moderates 7.0 in OSIM and by design of manual review became available as ask from author). *[AL READ/AGREED]*

11.2 Evidence precedence and calibration

The KPANIC prompt reuses technical findings from the prior ActionableScore and CVSS analyses, but it is instructed not to preserve HIGH merely because earlier text called the issue Important and not to use a previous numeric ActionableScore threshold as a prerequisite. It must independently evaluate the underlying primitive. *[AL READ/AGREED]*

The prompt contains specific preservation profiles and downgrade calibration for remote bugs, UAF, double release, active-object UAF, physical/peripheral corruption, limited authorization bypass, file-backed/page-cache corruption, parser confusion/OOB read, and reachability constraints. *[AL READ/AGREED]*

A notable calibration is that remote reachability plus a generic UAF/OOB/parser issue is not sufficient by itself for HIGH. However, a directly remote, repeatable active-object UAF with a demonstrated post-free write/state mutation can veto automatic downgrade even without controlled reclaim. *[AL READ/AGREED]*

11.3 Perl-side effect

The function returns a compact result consumed by two downgrade rules. If the review rejects KPANIC and the case is MODERATE with bounded CVSS, non-network AV, and ActionableScore below 5, Perl removes KPANIC and keeps regular MODERATE. If the review rejects KPANIC for a HIGH case under configured CVSS/ActionableScore bounds, Perl lowers HIGH to MODERATE7. *[AL READ/AGREED]*

Therefore Stage 3 is not simply a Boolean 'panic or no panic' checker. Its output participates in preserving or relaxing workflow severity. *[AL READ/AGREED]*

12. LLM Stage 4 - Post-Promotion HIGH Review

`request_llm_afterpushedtohigh()` is invoked only when `$pushed_to_high` is true and the current severity is HIGH. This makes it a targeted precision filter for cases that reached HIGH through the current pipeline rather than a universal HIGH classifier. *[AL READ/AGREED]*

The prompt receives the original CVE/patch text, the full previous ActionableScore analysis, the previous triage/CVSS description, and an explicit consistency instruction. In the supplied call site, HIGH should not be preserved merely because previous text says Important. The call highlights concrete non-crash shared-page/page-cache/COW/ownership primitives as examples of evidence that can justify preservation. *[AL READ/AGREED]*

The prompt itself is a detailed downgrade/preservation classifier. It evaluates evidence precedence, strong primitives, memory-corruption categories, reachability constraints, high-impact fields, default-bias behavior, and whether the promoted HIGH is better represented as MODERATE7. One

of ideas to have this new prompt was that original Perl rules should be quite aggressive regarding increasing impact level (better to choose Moderate 7.0 instead of Moderate regular when something between and similar cases), so too many CVEs can reach Important level as result and need additional check that really there is reason to keep it as Important as last step of analyses. *[AL READ/AGREED]*

If the function returns its downgrade outcome and the current case still has KPANIC, is HIGH, and satisfies the configured CVSS/ActionableScore ceilings, Perl lowers it to MODERATE7 and records DECREASED_TO_MODERATE7_BASED_ON_FALSEPOSCHECKOFDH. *[AL READ/AGREED]*

13. End-to-End Decision Flow

1. Receive CVE request or explicit patch override.
2. Check for an already cached completed result.
3. Acquire CVE/patch/applicability information and normalize source text.
4. Extract the 49 legacy features and build flags.
5. Train/load the embedded neural-network behavior and run regression/self-tests (the train happens only once when program starts and then it uses already trained neural net for next decisions about next input CVEs for analyses).
6. Run NN winner() for the target CVE.
7. Apply immediate compatibility/workaround adjustments to the NN result.
8. If the result is not NONE and patch text is available, call Stage 1 general triage/CVSS LLM prompt N1.
9. Parse the highest CVSS, CWE information, description, and manual-check phrase.
10. Call Stage 2 ActionableScore LLM prompt N2 with the patch plus Stage-1 hints.
11. Convert raw upper/lower ActionableScore to the effective averaged score.
12. Apply deterministic CVSS-based promotion/demotion rules.
13. Apply deterministic ActionableScore promotion/demotion and KPANIC-routing rules.
14. If KPANIC remains, call Stage 3 KERNEL_PANIC prompt LLM N3 validation and apply eligible downgrades/removal.
15. If the case was pushed to HIGH and remains HIGH, call Stage 4 LLM prompt N4 post-promotion review and apply eligible downgrade.
16. Refresh LLM state, compose final resolution text, write cache/output, and return the tab-separated result.

[AL READ/AGREED]

14. Operational Safety Semantics

The code predates the current four-class evaluation terminology, but several safety goals are visible in both prompts and deterministic rules. *[AL READ/AGREED]*

14.1 LOW is special

LOW is the state most closely associated with low-touch or automatic handling. The LLM prompts therefore include explicit manual-review recommendations and auto-downgrade controls, while deterministic rules can raise LOW when CVSS, manual-check guidance, or ActionableScore

indicates unresolved risk. In other words the Low decision was designed carefully to happen rarely and only in cases when obvious that is is kind of Warning impact level bug or memory leak bug only, but not Kernel Crash or other severe one, because Lows could be closed automatically without any manual review at all. *[AL READ/AGREED]*

14.2 MODERATE7/KPANIC is a safety buffer

MODERATE plus KPANIC is used as a review-preserving middle state. It allows the system to avoid claiming HIGH when exploitation evidence is weak while still preventing an uncertain CVE from disappearing into ordinary Moderate/Low handling. The original request to have something like Moderate 7.0 came from engineers from Engineering, because they were saying that cannot skip all Moderates without analyses and fixes at all when could be some Kernel Crash or other high impact cases that doesn't have high level of security impact. *[AL READ/AGREED]*

14.3 HIGH precision is improved through secondary review

The system has multiple promotion paths but also two later filters: KPANIC validation and post-promotion HIGH review. This architecture reflects a deliberate preference for high recall early in the pipeline followed by evidence-focused precision checks. *[AL READ/AGREED]*

15. Prompt-by-Prompt Reference

15.1 General triage/CVSS prompt (LLM prompt N1)

Aspect	Behavior
Function	request_llm_hints
Primary question	What is the technically defensible CVSS/CWE/impact interpretation and does the case require manual review?
Input	CVE/patch text appended to a large accumulated instruction string.
Output style	Human-readable analysis with parseable CVSS vectors, CWE tokens, quoted report text, and fixed manual-check phrase.
Perl parsing	Select maximum CVSS score; collect CWE; select longest report-like quoted description; copy manual-check marker.
Downstream consumers	CVSS threshold rules, ActionableScore context, KPANIC review context, final report.

[AL READ/AGREED]

15.2 ActionableScore prompt (LLM prompt N2)

Aspect	Behavior
Function	request_llm_calculate_awareness

Primary question	How actionable/security-significant is the bug beyond conventional CVSS, and what primitive/review state is supported?
Input	CVE/patch text plus Stage-1 hints.
Output style	Strict machine-readable fields followed by concise technical analysis.
Core semantic outputs	PrimitiveClass, RealUAF, privilege/namespace state, ImportantCandidate, ManualReview, AutoDowngradeAllowed.
Numeric behavior	Upper/paranoid and optional lower/conservative scores.
Perl parsing	Gap clamp then ceil(mean upper, lower).
Downstream consumers	Promotion/demotion thresholds; full trace is also passed to Stage 3/4.

[AL READ/AGREED]

15.3 KERNEL_PANIC review prompt (LLM prompt N3)

Aspect	Behavior
Function	request_llm_kpanic
Primary question	Is KPANIC actually supported, and independently, does non-crash evidence preserve HIGH?
Input	Patch/CVE text + complete prior ActionableScore analysis + prior triage/CVSS summary + consistency instructions.
Bias	False-positive filter for panic; MODERATE7 is considered a safe manual-review destination.
Key distinction	Panic validity != Important/HIGH validity.
Downstream effect	May remove KPANIC from MODERATE or lower HIGH to MODERATE7 under deterministic gates.

[AL READ/AGREED]

15.4 Post-promotion HIGH review prompt (LLM prompt N4)

Aspect	Behavior
Function	request_llm_afterpushedtohigh
Primary question	Was a HIGH promotion justified by a strong non-crash primitive, or is MODERATE7 more appropriate?
Invocation	Only when pushed_to_high is true and current

	severity remains HIGH.
Input	Patch/CVE + prior ActionableScore + prior CVSS/triage description.
Bias	Precision review after an intentionally escalation-sensitive pipeline.
Downstream effect	Can lower HIGH to MODERATE ⁷ under final Perl thresholds.

[AL READ/AGREED]

16. ActionableScore (N2) Prompt: Detailed Semantic Inventory

Prompt section	Role
Machine-readable output	Defines mandatory numeric and categorical fields so later code can parse a stable contract.
Core evidence rules	Allows patch, context, call sites, kernel architecture knowledge, subsystem semantics, lifetime/RCU semantics, deployment defaults; forbids invention of exploit primitives.
Advisory/exploit override	Allows trusted supplied evidence to establish stronger reachability, authorization, disclosure, LPE, or exploit facts absent from the commit.
Conservative semantics	Conservative means conservative about unproven consequences, not choosing the most restrictive hypothetical deployment.
Negative-evidence standard	Negative points require positive evidence; unknown privilege/race/hardware is neutral rather than automatically restrictive.
Scoring principles	Avoids double counting labels that describe the same underlying mechanism.
Reachability	Separates direct remote, firmware-mediated, local unprivileged, namespace-scoped, and high-control API reachability.
Lifetime manipulation	Rewards concrete attacker influence over creation/replacement/lifetime only when relevant to stale use.
Exploitation leverage	Separates strong leverage, weak concern, and zero-value weak corruption.
Memory corruption	Strong corruption, constrained overwrite, weak/indirect candidate.

Read-only OOB	Requires a disclosure/control path before assigning confidentiality.
UAF/RCU	Defines real lifetime violation and base signal.
Security-state lifetime	Recognizes authorization state disappearing before resource accessibility ends.
Repeated release/refcount	Scores a concrete second lifetime mutation through a potentially dangling reference.
Fixed stale vs active object	Prevents DMA/URB fixed stale storage from being treated like structured object reuse.
Structured stale leverage	Requires attacker influence plus meaningful later field consumption.
KVM	Separates boundary relevance from proof of guest-to-host exploitation.
Constrained UAF	Allows crash-oriented cleanup/async cases to remain constrained.
RCU mismatch	Recognizes concrete referenced-object lifetime failures.
Race/timing	Only materially difficult timing is a reducing factor.
Privilege	Separates unknown, host-admin, reduced capability, namespace capability, ordinary user.
Prior setup	Accounts for separate privileged configuration prerequisites.
Availability/resource	Separates crash, leak, and sustained exhaustion.
Filesystem/COW	Recognizes cross-ownership/shared-page corruption.
Authorization	Recognizes mandatory-check bypass and protected resource access.
Process exit/ptrace/pidfd	Special lifecycle/authorization calibration.
Memory-layout invariant	Scores fixes restoring interdependent allocation/layout variables.
Semantic reinterpretation	Recognizes validation-to-use semantic changes as potentially strong primitives.
Networking/qdisc/device/BPF	Subsystem-specific reachability and privilege calibration.
Call-site and error path	Requires actual helper/use semantics rather than naming heuristics.

Synchronization/translation	Covers mandatory retirement/synchronization and translation-domain reinterpretation.
Important preservation	Defines CLASS_A/B/C semantics and when review must be retained.
Score sanity and lower-score semantics	Constrains numeric results to semantic classification.
Manual-review floor	Prevents automatic closure where concrete unresolved security primitives remain.
Auto-downgrade	Defines when AutoDowngradeAllowed must be NO.
Final self-check	Forces consistency checks before emitting the machine-readable result.

17. Deterministic Rule Ordering and State

The final result depends not only on conditions but on their order. Several mutable variables carry history through the reconciliation block:

Variable	Role
\$severity	Current class index; mutated repeatedly.
\$severityfromnn	Original NN result retained for reporting.
\$flags{\$cve}	String-based provenance and operational state, including KPANIC and reason tags.
\$lowered	Prevents selected CVSS downgrade rules from applying repeatedly.
\$pushed_to_high	Controls whether the post-promotion HIGH reviewer is invoked.
\$dec_cnt	Counts selected ActionableScore/reviewer downgrades and guards further lowering.
\$sc	Best/paranoid CVSS numeric score from Stage 1.
\$alimpactscore	Effective averaged ActionableScore used by Perl thresholds.
\$rs	Stage-1 summary augmented with ActionableScore recommendation.

Any future reimplementation that turns the rules into a declarative policy must preserve these ordering dependencies or intentionally redefine them. A logically equivalent set of individual predicates can still produce different results if executed in another order.

18. Persistence, Diagnostics, and Output

DumpRes() appends results to cve_analysed_daemonver.txt, copies that file to /list/, logs the result, and in network mode sends it back over the client socket. The result includes CVE, severity text, decision, resolution, build/applicability state, accumulated flags, RHEL fixed-state flags, severity type, and patch ID.

The final resolution embeds 'Hints by AL-KERNEL' followed by the Stage-1/ActionableScore textual summary. The decision field also records the original NN impact for traceability.

LLM-specific artifacts are persisted separately. For example, ActionableScore full output is written to /list/llm/AS_<CVE>_ActionableScore, and the KPANIC stage writes /list/llm/KP_<CVE>_KernelPanic. The final cached CVE result is written to /list/llm/<CVE>.

[AL READ/AGREED]

19. Failure Handling and Defaults

The LLM transport dies on non-success HTTP responses, so an API failure is not silently converted into a low-severity answer. Text extraction has a fallback path and substitutes a '[No text output found in response JSON]' marker if necessary.

ActionableScore parsing has explicit numeric defaults. If ActionableScore cannot be parsed, the code uses 7.0. If ActionableScoreLower is missing, it uses the upper score. This is safety-biased but can produce escalation when formatting fails.

Stage-1 CVSS parsing also supplies a default local/high-impact vector if no best vector is found. These defaults are important operational behavior and should be treated as fail-safe escalation rather than normal model output.

The source contains many regex-based parsers. Therefore prompt formatting is part of the API contract. Changing labels, punctuation, or fixed phrases can change deterministic behavior even if the natural-language meaning is unchanged. *[AL READ/AGREED]*

20. Historical Architecture and Compatibility Behavior

The source clearly evolved over time. The original architecture was a feature-based NN with procedural workarounds. LLM hints were added as an experimental/test layer, then ActionableScore became a second independent analysis, followed by specialized false-positive filters. *[AL READ/AGREED]*

Evidence of this evolution includes comments describing 'workaround' transitions, old prompt versions retained in comments, OSIM compatibility handling for NONE, accumulated Russian/English instructions in the first LLM prompt, and later prompts with much stricter machine-readable schemas. *[AL READ/AGREED]*

This history explains why the implementation contains both numeric thresholds and semantic LLM classifiers. The system should not be interpreted as a clean single-model architecture; it is a layered production experiment designed to preserve earlier behavior while adding security-sensitive review. *[AL READ/AGREED]*

21. Maintenance Guidance

21.1 If changing feature extraction

- Re-run NN self-tests.
- Check whether training vectors and target vector change together.
- Inspect HIGH/MODERATE/LOW/NONE distribution before evaluating LLM effects.

21.2 If changing the Stage-1 prompt

- Preserve parseable CVSS:3.1 vectors.
- Preserve CWE-N tokens.
- Preserve the fixed manual-check phrases.
- Verify that the quoted report description still matches the Perl regex.

21.3 If changing ActionableScore

- Preserve exact machine-readable field names.
- Remember that Perl uses ceil(mean upper, lower), not simply the upper score.
- Test threshold boundaries 3, 4, 5, and 6.
- Check KPANIC creation/removal and pushed_to_high behavior.
- Review both numeric score and semantic class consistency.

21.4 If changing KPANIC review

- Keep panic validity separate from non-crash HIGH preservation.
- Test MODERATE+KPANIC -> regular MODERATE behavior.
- Test HIGH+KPANIC -> MODERATE7 behavior.
- Ensure strong non-crash primitives are not lost merely because panic is rejected.

21.5 If changing post-promotion review

- Test only cases that actually set pushed_to_high.
- Include COW/page-cache/ownership and authorization controls.
- Check that review does not become circular by trusting previous 'Important' wording.

[AL READ]

22. Recommended Regression Dimensions

Dimension	Cases to include
LOW safety	True LOW, Moderate near threshold, dangerous corruption that must never auto-close.
Manual-review routing	Regular Moderate vs MODERATE7/KPANIC; cases where panic is false but review is still needed.
HIGH precision	Strong COW/ownership, authorization bypass, controlled corruption, versus crash-only UAF.

UAF calibration	Fixed stale buffer, cleanup UAF, repeated refcount mutation, structured stale object, controlled reclaim.
KVM	Guest-only trigger, VMM-only trigger, compound trigger, demonstrated guest-to-host primitive.
Privilege	Ordinary user, namespace capability, reduced capability, host-admin, unknown.
Reachability	Direct remote, internal network, firmware/device mediated, local-only.
Parsing robustness	Missing lower score, malformed fields, extra prose, missing CVSS, missing manual-check phrase.
Ordering	Cases that can be lowered by multiple rules; verify \$lowered and \$dec_cnt effects.
Nondeterminism	Repeat LLM cases to quantify variation around decision thresholds.

[AL READ]

23. Worked Conceptual Flows

23.1 Safe LOW-like case

A case begins as LOW from the NN. Stage 1 produces a low CVSS and no manual-check requirement. ActionableScore is below 3 and does not establish a meaningful primitive. No KPANIC review is needed. The result remains LOW. This is the desired automation path. [AL READ]

23.2 LOW with unresolved security evidence

A case begins LOW but Stage 1 requests manual review or ActionableScore reaches the Moderate-review threshold. Perl raises the case to MODERATE and attaches KPANIC/MODERATE7 routing. The system therefore converts uncertainty into review rather than automatic closure. [AL READ]

23.3 MODERATE promoted to HIGH

A MODERATE case receives a sufficiently high effective ActionableScore or satisfies a high-CVSS promotion rule. Perl sets HIGH and pushed_to_high. If KPANIC is present, Stage 3 independently checks panic/non-crash evidence. Because pushed_to_high remains true, Stage 4 can also review whether the HIGH promotion is supported by a strong primitive. A failed precision review can return the case to MODERATE7 rather than LOW. [AL READ]

23.4 HIGH with weak crash-only evidence

A case begins or becomes HIGH, but the best CVSS and ActionableScore do not support strong non-crash exploitation. KPANIC review rejects the panic or HIGH-preserving primitive. Under the

deterministic gates, the case is lowered to MODERATE7, preserving review while reducing false Important/HIGH classification. *[AL READ]*

24. Design Interpretation: What the Pipeline Is Optimizing

The following is an interpretation of the implemented structure rather than a single explicit source-code statement. The pipeline appears designed to optimize asymmetric security risk: false LOW/automatic closure is treated as more dangerous than an unnecessary review, while false HIGH is undesirable but recoverable through later precision filters. *[AL READ]*

The architecture therefore uses an escalation-sensitive middle layer and a review-preserving MODERATE7 state. Strong semantic primitives such as COW/ownership violations, authorization bypasses, controlled corruption, and meaningful lifetime manipulation can preserve high severity even when classic crash evidence is absent. Conversely, generic UAF, remote adjacency, or a complex kernel object does not automatically prove Important-grade exploitation. *[AL READ]*

This interpretation is consistent with the prompt language and deterministic routing, but future maintainers should validate intended policy against current operational requirements rather than treating historical thresholds as immutable design principles. *[AL READ]*

25. Known Implementation Risks and Technical Debt

- Prompt parsing is regex-dependent and therefore sensitive to output formatting changes.
- The first prompt (N1) is a long accumulated natural-language instruction string and is harder to reason about than the later schema-driven prompts.
- Conversation-state reuse can create hidden coupling if `previous_response_id` is not managed as expected (means that for one CVE the conversation being used with GPT-5.5 model, so when answering next LLM prompts it sees results from previous and only for one particular this CVE).
- String flags serve both provenance and control-flow roles; accidental text changes can affect logic.
- Threshold logic is order-dependent and uses mutable guards such as `$lowered` and `$dec_cnt`.
- ActionableScore semantic fields are richer than the subset directly consumed by the legacy Perl reconciliation.
- Fail-safe defaults can escalate malformed LLM responses, which is safer than auto-closing but can increase review load.
- The embedded training corpus and code are tightly coupled, making model maintenance less modular.
- Internal HIGH/MODERATE/KPANIC terminology differs from newer four-class evaluation terminology.
- Large prompt size and repeated full-context calls increase latency and can contribute to run failures or truncation.

[AL READ]

26. Terminology Crosswalk

Perl/source term	Meaning	Current documentation interpretation
HIGH	Internal class index 0	High-severity / broadly IMPORTANT-like
MODERATE	Internal class index 1	Normal Moderate unless KPANIC is present
LOW	Internal class index 2	Low / automation candidate
NONE	Internal class index 3	Not security/not included; sometimes mapped to LOW for integration
KPANIC	Flag, not NN class	Panic/review-routing marker; MODERATE+KPANIC behaves like MODERATE7
MODERATE7	Prompt term	Review-preserving Moderate state
ActionableScore	Second LLM numeric triage score	Effective Perl value is averaged upper/lower
pushed_to_high	Per-CVE procedural flag	Triggers post-promotion HIGH precision review
Manual check	Stage-1/ActionableScore review concept	Human review routing, not identical to numeric severity

[AL READ]

27. Reimplementation Checklist

17. Preserve the four legacy NN outputs and the 49-feature ordering if reproducing the old NN.
18. Preserve distinction between severity and KPANIC operational flag.
19. Implement Stage-1 parsing exactly or replace it with an explicit schema and update all consumers.
20. Implement ActionableScore upper/lower parsing, gap clamp, and ceil-average behavior.
21. Preserve deterministic rule ordering or document intentional changes.
22. Carry provenance for every promotion/demotion.
23. Keep panic validation independent from non-crash HIGH preservation.
24. Invoke post-promotion HIGH review only under the intended pushed_to_high semantics.
25. Preserve review routing when downgrading uncertain HIGH cases.
26. Regression-test false LOWs, manual-review misses, unnecessary review load, and Important/HIGH precision separately.

28. Appendix A - Key Source Locations

Area	Approximate source lines
Imports, API state, request_llm_hints	1-222
ActionableScore prompt/function	226-4728
KPANIC prompt/function	4730-5734
Post-promotion HIGH prompt/function	5736-6932
Runtime setup / daemon / severity constants / NN construction	6934-7060
Embedded answer corpus and training setup	7060-8920
NN self-tests	8922-9360
Target NN inference and compatibility workarounds	9360-9395
Stage-1 and Stage-2 calls	9397-9418
CVSS deterministic reconciliation	~9418-9545
ActionableScore deterministic reconciliation	~9545-9550
KPANIC review and effects	9551-9598
Post-promotion HIGH review and effects	9600-9635
Final state refresh and DumpRes/cache	9635-9663

29. Appendix B - Compact Decision Table

Stage	Input state	Evidence/threshold	Output state
NN	49 features	winner()	HIGH/MODERATE/ LOW/NONE
Compatibility	HIGH	not-built workaround	MODERATE
Compatibility	LOW	KPANIC workaround	MODERATE
Compatibility	NONE	OSIM compatibility	LOW
CVSS	HIGH	best CVSS below configured high threshold	MODERATE/KPANIC
CVSS	MODERATE	high CVSS plus supporting conditions	HIGH
Stage 1	LOW	YES REQUIRES MANUAL CHECK	MODERATE/KPANIC
Actionable	LOW	effective score ≥ 3	MODERATE/KPANIC
Actionable	MOD/LOW	effective score ≥ 6	HIGH
Actionable	MOD/LOW + KPANIC	effective score > 5	HIGH

Actionable	HIGH	effective score <6 and CVSS <=7	MODERATE/KPANIC
Actionable	HIGH/MOD + KPANIC	effective score <4	regular MODERATE
Actionable	MOD	effective score <3 + no manual-check + guard	LOW
KP review	MOD+KPANIC	panic rejected + deterministic gates	regular MODERATE
KP review	HIGH+KPANIC	panic/high preservation rejected + gates	MODERATE7
HIGH review	promoted HIGH	review recommends downgrade + gates	MODERATE7

[AL READ]

30. Appendix C - Documentation Boundaries

This document intentionally does not claim that an LLM always follows its prompt. Prompt sections describe requested behavior; only Perl parsing and deterministic conditions are guaranteed by the source. [AL READ]

This document also does not independently validate whether every CVSS or security heuristic is correct under external standards. It documents the supplied implementation. Likewise, it does not infer undocumented deployment topology, model availability, or external service behavior beyond what the source explicitly configures. [AL READ]

Finally, this first edition focuses on architecture and decision semantics. A future edition can add exact per-feature regex definitions, a complete inventory of embedded training CVEs, and line-by-line worked examples from captured production/regression logs. [AL READ]

31. Reader Guide: Where to Look for a Specific Question

Question	Documentation location
How does a CVE move through the system?	Sections 1, 13, and Appendix B.
What does the legacy neural network do?	Sections 3-5.
What are the 49 NN features?	Section 4.1.
How is CVSS obtained and parsed?	Section 7.
What is ActionableScore and why is it different from CVSS?	Section 8.
Why can raw ActionableScore differ from the score used by Perl?	Section 8.8.
What creates or removes the KPANIC/MODERATE7 routing state?	Sections 9-11.
When is a HIGH result reviewed again?	Section 12.
Which parts are deterministic and which parts are LLM judgments?	Sections 7-12 and 17.
What must be preserved during a rewrite?	Sections 17, 21, and 27.
Where are the relevant parts of the 9.6k-line source?	Appendix A.

32. Architectural Takeaways

- The Perl implementation should be understood as a cascade, not as a single classifier. The NN, CVSS analysis, ActionableScore, KPANIC validation, and post-promotion review each answer different questions. *[AL READ]*
- The effective ActionableScore consumed by Perl is a transformed value. Documentation, debugging, and regression reports should distinguish the raw paranoid score, lower/conservative score, and effective averaged score. *[AL READ]*
- KPANIC is simultaneously historical terminology and operational state. It must not be documented as if it were simply proof that a panic is factually demonstrated. *[AL READ]*
- MODERATE7 is an important safety valve: the system can reduce false HIGH classifications without losing manual review. *[AL READ]*
- The specialized review prompts are precision filters placed after deliberately recall-sensitive earlier stages. Removing them changes the error trade-off of the whole pipeline. *[AL READ]*
- The string-based flags are not merely logging. Several later predicates inspect those strings, so provenance and control state are partly coupled. *[AL READ]*
- Prompt schemas are APIs. Fixed labels and phrases are consumed by regexes, and prompt edits therefore have software-interface consequences. *[AL READ]*
- The current implementation is best maintained with end-to-end regression tests that measure both classification accuracy and operational outcomes such as false LOWs and manual-review misses. *[AL READ]*