Rings of Risk: Consequence and Reachability in Application-Code Vulnerability Chains

John Pellew john@kolega.ai
Faizan Raza faizan@kolega.ai
Kolega Labs
October 22, 2026

DRAFT v6.1 · Working research draft · Subject to revision

Abstract

Even when an application's outer perimeter is secured, weaknesses deeper in its code may combine to reach critical outcomes. A file, session or identifier accepted by one feature can cross into another that grants greater authority. Vulnerability counts and individual severity scores do not show whether such routes exist. The Common Vulnerability Scoring System (CVSS) also distinguishes severity from risk. **Rings of Risk** (RoR) traces these routes and profiles each protected outcome by consequence, entry conditions, distinct supported routes and the controls that could interrupt them. Each finding is represented once in the route map and may contribute to multiple chains that reach different outcomes.

We examine 15 public open-source applications, selectively hardening exposed direct routes in six local forks. Source analysis follows code flaws together with artifacts, victim actions and ordinary processing steps. The remaining source contains potential paths to serious outcomes, including multi-flaw chains. Securing an entry point therefore cannot substitute for independent controls deeper in the application code.

1. Introduction

Securing an application's outer perimeter closes its most visible direct routes to critical harm. Yet lower-trust input may still cross several code boundaries and reach a privileged operation. One flaw can supply an identifier, session or file that another flaw wrongly trusts. The application therefore needs independent controls at each handoff: that is the code-level defence-in-depth problem this paper examines.

By **outer perimeter** we mean exposed application entry points and their prominent direct routes to critical outcomes. Addressing those routes is a specified defensive step, not a claim that every external flaw is gone. Our selective fixes create a specified before-and-after version of that scenario in open-source software (OSS) code pinned to fixed revisions. We then examine routes through the controls that remain.

Consider an application whose vulnerability backlog falls from 1,000 to ten. That is progress, but the ten remaining findings could be confined to minor effects or combine into administrator control, customer-data exposure or server execution. The decisive question is which **protected outcomes remain reachable**.

Defence in depth here means independent controls **within application code**: authentication at entry, authorisation at the sensitive operation, tenant separation, safe file processing and checks before privileged execution. A profile-picture upload becomes consequential when attacker-controlled content reaches a vulnerable parser or worker. The upload, handoff and later operation must form a traceable path; accepting a file alone does not establish one.

Security attention may concentrate on conspicuous functions such as login, payments and customer data, while an ordinary feature supplies the first step of a serious chain. This is a motivation to trace cross-feature paths, not an observed pattern of developer or penetration-test behaviour in our sample.

Agentic tooling makes cross-feature search increasingly practical. Work on exploitation of published vulnerabilities and multi-stage activity reinforces the need to examine paths across application features. A one-finding-at-a-time report can obscure the consequential route those features form together.

This paper contributes an outcome- and capability-based risk model; a method for tracing code flaws together with ordinary or external prerequisites; and a 15-application OSS study that contrasts isolated scores with route inventories and examines routes remaining after selective hardening, including paths that the fixes break. It also visualizes how one finding can feed more than one route and how each route ends at a distinct outcome.

2. Background and Related Work

2.1 What existing measures actually measure

Security teams need different answers from a vulnerability assessment: how severe a flaw is, whether attackers exploit it, whether the affected system matters, and what an attacker can reach through it. These are related but different questions. A single finding can disclose a token that enables a second flaw; conversely, a high-severity finding may need an identity or configuration absent from the application being assessed. A score attached to either finding does not, by itself, describe that route.

CVSS v4 classifies and rates individual vulnerabilities. Its Base score expresses technical severity, while Threat and Environmental metrics let a consumer add threat and deployment context. FIRST, which maintains CVSS, explicitly warns against using Base alone as risk. The same user guide **does** explain how an analyst may score a *known* vulnerability chain by combining exploitability and impact metrics. RoR does not claim that CVSS ignores chains. The difference is that a chained CVSS score starts with an identified chain; it does not discover the application's routes, enumerate alternative routes to distinct outcomes or determine which code fixes interrupt them. Adding individual CVSS numbers is not the CVSS method and does not produce an application-risk score. A CVSS Base score is an ordinal severity ranking, so differences between scores are not measured differences in harm or exploitation probability.

The Common Vulnerabilities and Exposures (CVE) Program gives published vulnerabilities common identifiers; the National Vulnerability Database (NVD) enriches CVE records with CVSS, weakness and affected-product information. These are essential reference data for disclosed vulnerabilities, not a score for unpublished first-party findings or a map of an application's code handoffs.

Exploit Prediction Scoring System (EPSS) estimates the probability that exploitation of a published CVE will be observed in the wild over the next 30 days. FIRST says EPSS is not a complete risk score: it does not include the impact or the particular deployment. It updates daily for published vulnerabilities with CVE identifiers. FIRST also cautions against multiplying EPSS by a CVSS Base score: a calibrated probability times an ordinal severity ranking has no clear interpretation. The US Cybersecurity and Infrastructure Security Agency (CISA) maintains the Known Exploited Vulnerabilities (KEV) catalogue of CVEs with evidence of exploitation. It is an important remediation signal, not a probability. Neither can directly score an unpublished flaw in first-party code, and neither establishes whether two local findings compose. They can still inform a route's urgency when its steps have CVE identifiers.

Combining these signals can make a *published-CVE remediation queue* more manageable. In a study of 28,377 vulnerabilities, Shimizu and Hashimoto report roughly 95% less urgent remediation work while retaining 85.6% coverage of their exploitation reference set. That is a result for their filtering policy and data, not an estimate of how much application risk was removed. It does not reveal routes through unpublished findings in first-party code.

CISA's Stakeholder-Specific Vulnerability Categorization (SSVC) uses a stakeholder-specific decision tree to prioritise vulnerability response, incorporating exploitation and organisational impact. It makes a different decision from CVSS or EPSS: what action to take on a known vulnerability. A source-level route and the protected outcome it reaches could supply evidence to such a decision; RoR is not a replacement for that response policy.

2.2 Application and business context

The Open Worldwide Application Security Project's OWASP Risk Rating Methodology combines estimated likelihood and technical or business impact for a particular risk scenario. It can assess an application-specific chain when an analyst has identified one. RoR contributes a repeatable way to describe *which* code steps and ancillary handoffs reach each outcome before that scenario is rated. OWASP's Top 10 is an awareness list of risk categories, not a score for one application; Application Security Verification Standard (ASVS) specifies verifiable security controls. Both help identify controls to inspect at a chain handoff, but neither is a route inventory. MITRE's Common Weakness Enumeration (CWE) provides a vocabulary for weakness types, while ATT&CK organises observed adversary behaviour. They help name a flaw or step without assigning this application's protected outcome or calculating a route.

The National Institute of Standards and Technology (NIST) describes risk in SP 800-30 through a contextual assessment of threats, vulnerabilities, likelihood and consequences for an organisation. Factor Analysis of Information Risk (FAIR) models the probable frequency and magnitude of loss. These broader frameworks are appropriate when there is evidence to estimate organisational impact and event frequency. This source-only study lacks deployment and loss data. Its reachability profile should therefore be an input to a risk assessment, not a numerical probability or expected-loss claim.

The NIST Cybersecurity Framework (CSF) 2.0 and OWASP Software Assurance Maturity Model (SAMM) address how organisations manage security work and improve application security practices. RoR can supply evidence about specific code paths to those programmes; this study does not assess programme maturity or whether an organisation implements their controls.

2.3 Operational prioritisation and attack paths

Commercial prioritisation goes beyond CVSS Base. Tenable's Vulnerability Priority Rating (VPR) combines technical impact and threat intelligence; Rapid7 Active Risk adds exploit and threat signals to CVSS; Qualys TruRisk aggregates findings with asset criticality and exposure. Developer-facing Snyk prioritises by severity, exploit maturity, fixability and project importance. Its reachable-vulnerability analysis can build a call graph for dependencies. These products can use deployed asset data, threat signals or dependency reachability to prioritise a finding. Our study has no deployed asset inventory or live threat telemetry; its contribution is tracing how findings and ordinary code steps connect to protected application outcomes.

Dependency call reachability can show that an application invokes a vulnerable library. RoR asks a different question: whether first-party code flaws and ordinary steps connect a stated starting actor to a protected outcome. A vendor's priority score and that route profile can be used together.

Attack paths are also established practice. MulVAL derives attacker capabilities from facts and prerequisite rules in a network model. Topological Vulnerability Analysis represents dependencies among exploits and the paths to a network attack goal without listing every path separately. The k-zero-day safety literature asks how many unknown flaws a network path would require; its general network formulation is NP-hard. These works establish why combinations and alternative routes matter. They take host, network and configuration facts as inputs rather than deriving application-code handoffs from source. That complexity result does not describe the cost of RoR's bounded review of known code findings.

Microsoft Security Exposure Management and Tenable Attack Path show routes among assets, identities and exposures and identify choke points for remediation. Wiz's Security Graph similarly connects cloud and AI assets to surface consequential combinations. RoR uses the same prerequisite-to-capability idea to examine handoffs within first-party application code and its ordinary consumers: sessions, files, workflows, parsers, logs and jobs. It does not infer a network or cloud path that the source and stated deployment do not establish.

Industry analyses also use paths to narrow remediation. In a reported customer analysis, Praetorian reduced roughly 500,000 vulnerability findings to 14 endpoints exposed to complete browser exploit and sandbox-escape combinations. This is a vendor case, not a prevalence estimate for application-code chains. Cycode describes lower-severity application findings that can contribute to serious outcomes through intermediate steps. RoR takes the complete route as the unit to review, including the code handoff, starting conditions and protected outcome. Vendor examples cannot validate a proposed route in our corpus; that requires review of the pinned application source.

2.4 Program analysis and agent benchmarks

Code property graphs combine syntax, control flow and data dependencies so queries can locate vulnerable code patterns. IRIS combines a large language model with static analysis to infer taint specifications and detect vulnerabilities across Java repositories. Both help discover flaws; RoR starts with findings and follows the capabilities they grant through other code and ordinary processing. VEX-Bench tests whether agents can assess the exploitability of known vulnerable dependencies in downstream projects. That is a different reachability question from composition among first-party application findings.

Agent exploitation benchmarks show why capability and outcome should be kept distinct. CVE-Bench evaluates agents against known web-application CVEs and reports that its strongest framework resolved up to 13% in that test. ExploitBench grades 16 successive capabilities on 41 bugs in the V8 JavaScript engine, from reaching vulnerable code to arbitrary code execution. AgentCyberRange tests web footholds and subsequent movement in ranges with 156 internal hosts; its best reported system solved 16.1% of web-exploitation tasks and 31.7% of post-exploitation tasks under that evaluation. These figures describe particular agents, tasks and budgets, not the likelihood of any RoR route. RoR characterises source-supported routes from a stated start, including unpublished flaws and non-code handoffs; it does not infer actual agent success from those routes. An exploit benchmark tests whether an agent achieves a specified task.

2.5 Why aggregation needs routes

One reachable critical outcome is already critical. A second route to the same outcome does not double the *harm*, but it can increase the opportunity to reach that harm and require another fix. Route count alone is not a probability: routes may share a step or prerequisite. Administrator control and mass customer-data disclosure are also distinct outcomes despite both being critical. RoR therefore keeps consequence, entry difficulty, independent supported routes and the fixes needed to interrupt them for **each protected**

outcome. This is the gap an isolated severity score, a threat-intelligence signal or a finding total cannot fill by simple addition.

If routes had known, independent success probabilities, we could subtract the product of their failure probabilities from one: $1 - \prod(1 - p_i)$ for route probabilities p_i. But application routes often share a credential, vulnerable operation, victim action or deployment condition, so independence rarely holds. This study has no measured p_i values and does not turn route counts into a probability. It instead reports which outcomes each route reaches, what its entry conditions demand, and which fixes interrupt shared or separate paths. Possible route combinations can multiply while the harm of one compromised outcome remains bounded.

Prior work	Unit of analysis	Relationship to RoR
CVSS, EPSS, KEV and operational priority scores	A flaw, published CVE, or affected asset	Useful severity, threat and context signals; RoR adds source-traced routes to specific outcomes.
MulVAL, Topological Vulnerability Analysis and exposure graphs	Network or asset paths	Supply prerequisite-and-capability reasoning; RoR examines handoffs within application code.
Code property graphs, IRIS and VEX-Bench	Vulnerability discovery or dependency exploitability	Help identify or assess individual flaws; RoR follows interactions among known findings and ordinary steps.
Agent exploitation benchmarks	Agent success on specified tasks	Measure what an agent achieved in a test; RoR profiles what routes the assessed source can support.

3. The Rings of Risk model

3.1 Outcome, capability and transition

For each application, a **context card** identifies important functions and assets before findings are graded. Each protected outcome names the resource, operation, affected population and consequence: for example, reading customer records across tenants or executing code in the application-server process. The execution environment matters; the label "remote code execution" alone does not identify what was compromised.

A **capability** is something an actor can do or know, such as reading a file, supplying a workflow property, holding a role or knowing an object identifier. A **transition** uses stated prerequisites and a mechanism to produce another capability. The mechanism may be a code flaw or ordinary application behaviour; a victim action may also be required. Code claims cite source, while external conditions remain explicit assumptions. This prerequisite-to-capability representation draws on attack-graph reasoning such as MulVAL.

A large language model (LLM) proposes capabilities and transitions; source review tests whether each handoff works under the stated identity, permissions and configuration. A route is **source-supported** when those reviewed transitions connect the stated starting actor to a protected outcome. Only reviewed transitions enter the graph. Deterministic code then derives the outcomes reachable from a declared starting position. Citation existence is checked mechanically, while the source review establishes what the cited code actually does.

3.2 Consequence bands and aggregation

Band	Meaning	Illustrative outcome
B1 Critical	Loss of an essential function or authority	Server-process execution, platform-administrator control or mass credential disclosure
B2 Major	Substantial data or operational harm	Cross-tenant disclosure of customer records

Band	Meaning	Illustrative outcome
B3 Moderate	Meaningful but bounded and recoverable harm	Alteration of one team's non-authoritative records
B4 Limited	Local, readily reversible harm	Disclosure of a version string

B1 Critical is the highest consequence, followed by B2 Major, B3 Moderate, and B4 Limited.

Each band grades a specified outcome, not an application. RoR keeps distinct critical outcomes separate and does not add their bands or count additional routes as additional harm. A highest-band headline can point to the full profile, but cannot replace it.

3.3 Chains and prerequisites

An **attack route** connects a starting actor to a protected outcome. A **chain** has at least two distinct security-relevant transitions: one step supplies an artifact or capability that a separate step uses to cross another trust or authority boundary. At least one step involves a code flaw. A *multi-vulnerability chain* contains two or more distinct code flaws. A *mixed chain* includes a separate, capability-changing handoff through a credential, victim action, document, browser, log, workflow or other ordinary behaviour. We record the full route and the number of code flaws separately. A prerequisite does not automatically become a chain link: SQL injection used to dump a database is one flaw's direct route, even if it takes multiple queries. Uploading a file to trigger a vulnerable parser is likewise not a chain merely because storage and processing are separate operations. Phishing can bridge steps, but must be stated as a condition unless its role and outcome are independently supported.

A consequential chain matters even when a shorter route also exists, and it need not reproduce the outcome of a flaw chosen for repair. After hardening, we ask whether its own steps still connect to its own outcome; a route that requires patched code no longer survives. Each handoff must preserve the actor's capability across environments: browser execution, server execution, continuous-integration (CI) execution and PDF creation are different outcomes unless code connects them.

Code-adjacent handoffs provide the search map for those connections. A file, message, credential or URL may carry attacker influence without itself being a vulnerability. The important event is its later consumption by a different principal or more privileged component. We search for the producer, carrier, consumer, trigger and resulting capability, then test that handoff against source and stated deployment conditions.

Bridge to examine	Evidence needed to admit the handoff
Human and identity: phishing, approval, login, token or invitation	Who acts; how a valid credential or role is obtained; its scope, expiry and receiving operation.
Browser and client: rendered content, localhost request, extension or native app	Viewer action, origin, session and permissions; an extension or client must actually be installed and reached.
Artifact and document: upload, model, archive, backup, image or PDF	Attacker control of bytes; storage and later parser, importer, viewer or worker; the exact processing trigger.
Log, message, cache, search result or agent input	Writer and reader; how stored data becomes a displayed value, leaked secret, tool instruction or privileged action.
Webhook, callback, queue or scheduled job	Producer, service identity, access control, message or URL validation, and the consumer's operation.
Build, plugin and deployment component: CI runner, package, cloud identity or guest	Installation or job trigger; runner or service permissions; any further boundary to server, cloud asset or host.

This is a catalogue of *possible* bridges, not a list of observed attacks. File-upload processing, log consumption, CI script evaluation and agent use of untrusted text illustrate distinct consumer boundaries. Browser content does not inherit a browser extension's permissions; PDF scripting depends on the viewer's controls. Crossing from a browser, viewer, runner or guest into the application server or host requires another supported transition.

3.4 Access ring, reach ring and effort

The **Access Ring (AR)** places an attacker, or a protected asset, by the privilege tier it holds or requires. Rings are numbered from the core, as the consequence bands are, so AR1 is the innermost.

Ring	Name	Position	Example
AR5	Outside	No account or foothold: anonymous, or holding only a lure, a leaked link or an identifier	Anonymous request; enumerated object ID
AR4	Standard user	A low-privilege account, including a self-registered or tenant-member account	Ordinary logged-in session
AR3	Delegated authority	Another principal's authority: an administrator or owner role, or a victim's authenticated session	Administrator session taken through stored script injection
AR2	Control plane	Platform, realm, tenant or organisation administration; secret and key stores; cross-tenant reach	Realm administration; stored credentials
AR1	Crown jewels	Host or server-process code execution; control of the whole datastore or of the identity root	Server-process execution; arbitrary SQL

A vulnerability finding is an arrow. Its tail is the ring the attacker occupies when exploiting it, and its head is the ring the attacker holds afterwards, so its length is the number of rings it crosses: what the finding gives. A finding that discloses an identifier but confers no authority leaves the attacker in AR5 and enables a later arrow instead of moving the attacker inward. Every arrow ends in a circle. A final outcome is a circle inside the ring whose privilege the outcome puts under attacker control. An intermediate outcome is the position or capability a finding gives on the way, such as an administrator session or a leaked secret, and the next finding starts from that circle, which makes the chain and its hand-offs visible. A finding used by several routes is drawn once.

The ring and the consequence band are different measures. The band grades the harm of an outcome; the ring says where the attacker stands. A B1 Critical account takeover can end in AR3, whereas server-process execution ends in AR1. Rings are assigned from the reviewed capability and outcome of each step under a stated rule, not from CVSS: an outcome takes the ring of the privilege needed to perform that function legitimately, and a capability takes the ring of the authority it confers. Judgement calls are marked in the classification table. An enabling step that puts an account, credential or a victim's session within the attacker's reach, such as open self-registration, a phished password or a lure that makes a privileged user open attacker content, is drawn as a dashed arrow into the ring it reaches (AR4 for an ordinary account, AR3 for a privileged victim), and the first code arrow starts from that ring. A configuration, file or missing control gives the attacker no position of its own, so such a route starts in AR5. A first code flaw whose starting position the ledger does not record is drawn entering from outside the rings, and no starting ring is assumed.

The **Reach Ring (RR)** classifies a finding by what it can reach from a declared starting position. The consequence band grades an outcome; RR grades a finding's role in reaching it. In ring terms, an RR1 or RR2 finding is a single arrow from the declared start to a B1 Critical or B2 Major outcome, whereas an RR3 finding is an arrow that ends short of the outcome and contributes only as an earlier segment of a longer path. RR4 Bounded reach describes a finding; B1 Critical describes an outcome.

Level	Name	Rule in the declared scenario
RR1	Critical reach	The finding can reach B1 Critical without another code flaw
RR2	Major reach	It can reach B2 Major without another code flaw, but not B1 Critical
RR3	Chain-dependent	It contributes to B2 Major or B1 Critical only with another code flaw
RR4	Bounded reach	Its reachable outcomes are B4 Limited or B3 Moderate only
RR5	No established reach	No outcome is established from the declared start

Victim actions, credentials and configuration are recorded alongside each RR assignment. A route involving one code flaw and a victim action can still be RR1 Critical reach or RR2 Major reach; the action affects entry difficulty, not the code-flaw count. A finding at either level can also carry a chain flag. RR5 No established reach means that reach was not established from this starting position.

Figure 1 introduces the visual notation. The rings are drawn uncurled, as consecutive columns from AR5 at the left to AR1 at the right, so that the distance an arrow travels reads as the number of columns it crosses. Vulnerability findings are arrows that cross rings, protected outcomes are circles, and a shared finding is drawn once with later arrows starting where it ends. Dashed links mark supporting conditions such as credentials, victim actions or configuration. The example is schematic; it does not depict a corpus application.

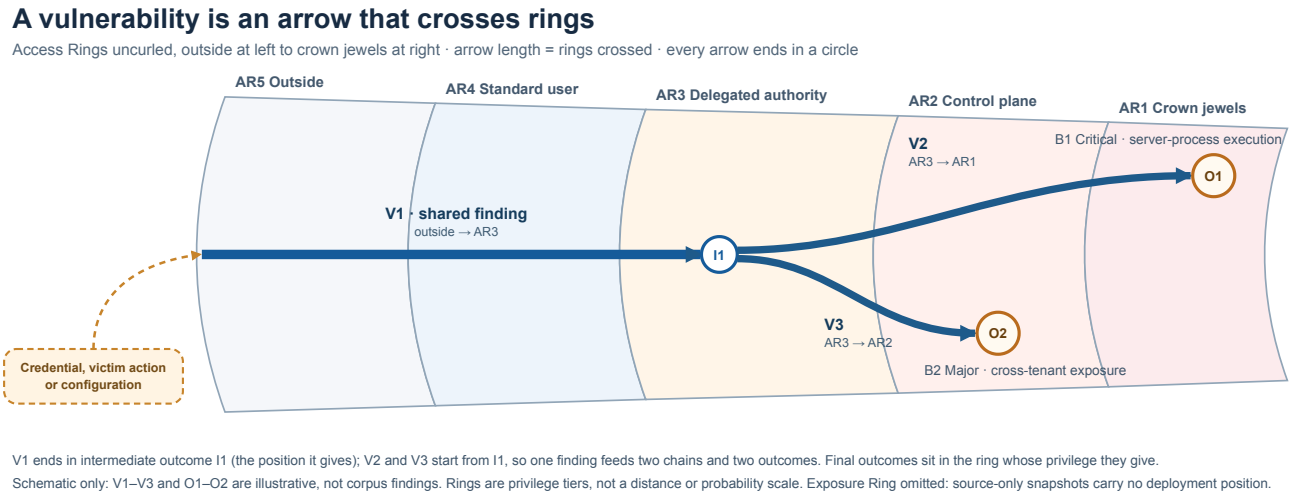


Figure 1. The rings are drawn uncurled, from the outermost (AR5, left) to the crown jewels (AR1, right). Vulnerabilities are arrows that cross Access Rings and end in circles. V1 is a shared finding that takes the attacker from AR5 to AR3 and ends in intermediate outcome I1; V2 and V3 start from I1 and reach different final outcomes. The example is schematic; V1–V3 are not corpus findings.

For each outcome, RoR records the demands of every supported route: starting access, victim action, configuration, code-flaw count and evidence state. It then identifies the least demanding routes. When one needs a victim click and another needs an extra code flaw, neither is automatically easier; both remain visible.

An **Exposure Ring** can separately describe where a component sits in a deployment. This source-only study has no consistent entry-point or network policy data, so it records application entry conditions without assigning deployment exposure levels.

3.5 Outcome profiles and known-route closure

RoR reports a **structured profile for each protected outcome**:

Field	Question answered
Consequence	What exact resource, operation and population are affected, and at which band from B1 Critical to B4 Limited?
Entry difficulty	What access and conditions does each supported route require, and which routes are least demanding?
Route count	How many distinct source-supported routes reach this outcome? Candidate records are displayed separately.
Known-route closure	What is the smallest feasible set of code fixes or controls that interrupts every *known supported route* to this outcome?

For each outcome, the profile contains its consequence band, the entry requirements and least demanding routes, the number of distinct supported routes, and the minimum number of fixes needed to interrupt known routes. The profile keeps different kinds of evidence visible instead of adding them into an opaque score. Outcomes at the same band can have different entry conditions and remediation work. Route count and entry difficulty indicate attack opportunity; they are not a calibrated probability of exploitation.

Closure depends on shared mechanisms. Twenty variants through one vulnerable operation that consumes attacker input may require one fix; three independent routes may require three. The field covers **known supported routes**. A compact profile might read: "critical customer-data disclosure; ordinary-account entry; three supported routes; two fixes to close the known routes." The underlying paths and fixes remain available for review.

The current OSS ledger is an input to this profile, not a completed outcome profile. Its records include overlapping variants and unreviewed handoffs. Distinct supported-route and closure counts require those records to be adjudicated against specified outcomes.

3.6 Found and residual risk

Here, **ground truth** means the study's reference inventory of findings used to evaluate scanner alerts. **Found risk** describes outcomes reachable through findings reported by a scan and can guide remediation without ground truth. **Scanner residual risk** is a benchmark quantity: after treating matched ground-truth findings as fixed, it asks which outcomes remain reachable through missed findings. **Post-hardening residual reach**, used in this study, instead inspects routes remaining after documented fixes to pinned source. A live system's unknown misses cannot be measured directly, so these quantities answer different questions.

4. Study design

4.1 Corpus and route inventory

Provenance of the findings. Every finding in the ground-truth inventory was already public. Each of the 879 finding rows is a published GitHub Security Advisory for one of the 15 applications, and 736 of them also carry a CVE identifier. We matched each advisory to the pinned snapshot and reviewed it only to confirm that the described flaw and its location are present in that source; 486 rows also record an upstream fix commit. This study discloses no new vulnerability. It did not run the applications or exploit any finding. What it adds is the route analysis: how these known findings, together with artifacts, victim actions and ordinary processing steps, could combine. The route records below are therefore source-supported candidates assembled from public advisories, not newly discovered flaws.

We compare the available CVSS scores for ground-truth findings with candidate route records across **all 15** pinned applications. Their simple score sums are shown only to test what an additive finding-level summary conveys; they are not CVSS risk scores. The six locally changed forks provide the narrower before/after comparison. The nine unchanged repositories supply baseline source data, not observed results of a fix budget.

The chain ledgers contain 801 proposed pairs of steps for all 15 applications. A pair is a route proposal, not necessarily a complete chain to an outcome. We retain non-rejected routes labelled B2 Major or B1 Critical, remove routes requiring a patched flaw in the relevant modified fork, and collapse identical ordered first-step/second-step/outcome records. We assess whether each handoff exists regardless of any shorter route. The result is **246 route records carrying source references**: 123 beginning with a code flaw, 123 with an ancillary step, and 178 labelled with a B1 Critical outcome. An ancillary step is a condition such as a credential, victim action or ordinary application operation, rather than another code flaw. Nineteen records have a stale or imprecise source reference. These counts describe the review inventory, not 246 distinct established chains.

We inspect source, fixes and configuration without running applications. A route is **source-supported under conditions** when the code establishes its steps and handoffs, and any victim or deployment conditions are stated. A record with citations but an unverified capability transfer remains a *candidate*; contradicted or patched paths are removed. This source standard does not claim that an exploit was executed.

To identify longer paths, we also join a route's resulting finding to another route that begins with that finding, then check whether the first step supplies the second step's required capability, role or token. This exact-ID pass produced **54 joined path records** across Budibase, Ghost, Keycloak and Open WebUI. They are overlapping graph paths, not 54 distinct chains: source review rejects the two Open WebUI empty-password LDAP joins, and some apparent Keycloak starts do not provide the DCR token required by the next step. The retained longer paths and their conditions are described in §5.2; the composed-path register records every generated path.

4.2 Selective interventions

In six local forks, we fix selected direct routes from exposed application entry points to serious outcomes, then examine the routes still present in the code. Each intervention names the flaw and fix, making the before/after comparison reproducible. This models outer-perimeter hardening at the named routes without claiming that every exposed flaw was closed.

Fork	Local intervention	CVSS sum: pinned → minus targeted findings	Retained B2 Major/B1 Critical records	Using two code flaws	Records labelled B1 Critical
Ghost	Upstream fix for `CVE-2026-26980`	318.6 → 309.2	21	18	18
Budibase	Upstream fix for `CVE-2026-35216`	415.3 → 406.3	11	9	5
Gradio	Upstream fixes for `CVE-2023-51449`, `CVE-2024-1561`, `CVE-2024-34511`	150.3 → 127.7	5	0	1
Parse Server	Upstream fixes for `CVE-2026-30965`, `-30966`, `-32248`	562.4 → 533.2	15	3	7

Fork	Local intervention	CVSS sum: pinned → minus targeted findings	Retained B2 Major/B1 Critical records	Using two code flaws	Records labelled B1 Critical
pyLoad	Locally required an operator-supplied first/reset admin password	152.4 → 152.4	8	3	7
Langflow	Upstream fixes for `CVE-2026-48519`, `-55450`	197.2 → 178.3	7	6	2
Total	**Ten finding IDs plus pyLoad credential hardening**	**1,796.2 → 1,707.1**	**67**	**39**	**40**

The six modified forks retain 67 route records, 39 involving two code flaws. Several share steps or outcomes, so the count is not an aggregate harm score. The fixes also eliminate some proposed chains: Gradio's file-read patch breaks seven recorded pairs that depended on that code, including four proposals labelled B1 Critical. The intervention therefore reveals both surviving routes and routes a shared fix closes.

4.3 Route assessment

For each route, we identify the lower-trust producer and carrier, the exact consumer, the trigger that brings them together, and the capability gained at that boundary. We then record the starting actor, cited code steps, credential or configuration conditions, protected outcome and execution environment. The source review checks that the produced artifact, identity or value can actually reach the named consumer with the claimed authority. We distinguish a separate capability-changing handoff from the ordinary delivery, trigger or consequence of one flaw. We also check whether a selected fix removes a required step. The question is whether **this route's steps compose** to reach **this outcome**, regardless of another available route.

5. Findings

5.1 The full 15-application view

All 15 applications enter the cross-sectional comparison below. Six local forks have documented changes, permitting a before/after route check; the other nine remain pinned baselines. Every application has route records labelled B2 Major or B1 Critical, but those records are not distinct harms or a completed RoR score. The CVSS column deliberately shows what simple addition yields; it is **not** a CVSS-sanctioned measure of application risk. Its sums describe the pinned ground-truth inventories, while the route columns use modified forks in six cases and pinned baselines in nine. The figures are descriptive inputs, not a ranking of these applications.

Application	Ground-truth findings	Sum of available CVSS scores*	Local state	B2 Major/B1 Critical route records	Records labelled B1 Critical
Airflow	32	203.0	Pinned	5	1
Budibase	56	415.3	Selectively hardened	11	5
Flowise	92	719.3	Pinned	9	6
Ghost	52	318.6	Selectively hardened	21	18
Gradio	22	150.3	Selectively hardened	5	1
Jenkins	30	170.6	Pinned	8	6
Keycloak	66	381.5	Pinned	49	42
Langflow	25	197.2	Selectively hardened	7	2

Application	Ground-truth findings	Sum of available CVSS scores*	Local state	B2 Major/B1 Critical route records	Records labelled B1 Critical
MLflow	30	249.8	Pinned	11	11
n8n	125	917.5	Pinned	13	5
Open WebUI	60	386.9	Pinned	8	5
Parse Server	81	562.4	Selectively hardened	15	7
pyLoad	22	152.4	Credential hardened	8	7
Weblate	33	171.4	Pinned	13	8
XWiki	153	1,276.6	Pinned	63	54
Total	**879**	**6,272.8**	**Six changed, nine unchanged**	**246**	**178**

*Three ground-truth findings lack a numeric CVSS score (two in Budibase, one in MLflow). The sum uses the other 876 scores. Route records vary in review depth and may share steps or outcomes, so their count is not an alternative risk score. Only the six changed forks support an observed before/after comparison; placing hypothetical fixes in the nine baselines would require a separate intervention and route review.

MLflow's ledger contains 13 conditional victim- or untrusted-artifact routes. Eleven are provisionally labelled B1 Critical and enter the table; two browser-card routes are B3 Moderate and do not. Nine routes labelled B1 Critical have source-supported processing handoffs under stated artifact or victim conditions. The MLproject-parameter and card-template routes remain candidates because attacker control of the later input is unconfirmed. No MLflow route in this ledger links two code flaws, and its pinned fork has not been selectively hardened. The ledger is a scoped source review of victim/artifact handoffs, not an exhaustive enumeration of possible pairs.

Route records cannot be read as independent risks. XWiki's 63 include victim/configuration variants and repeated sinks; Keycloak's 49 include specific identities and integrations. Their eventual outcome profiles must group shared routes, retain distinct outcomes and state each route's entry conditions. A shorter route changes immediate fix priority without erasing a deeper path.

5.2 Source-supported chain patterns

Artifact to processing. MLflow's pinned fork provides a clear reason to include code-adjacent steps: an untrusted model or recipe artifact must be selected and later loaded or rendered by a user or worker. The ledger records eleven artifact- or parameter-to-processing routes labelled B1 Critical, nine with source-supported handoffs under that condition. Two more need confirmation that attacker input reaches the later command or template. These paths end in the loading or rendering process; they do not show that the public tracking server automatically executes an uploaded file.

Lower-trust content to privileged processing. In pyLoad's credential- hardened fork, an account with the ADD permission can place content in a Downloads view. When an administrator opens it, the cited rendering flaw can run script in that administrator's browser. A separate session-store/upload weakness is proposed as a later step toward server execution; that browser-to-server handoff still needs source confirmation. The password change leaves the cited code steps in place. The route illustrates why the browser and server capabilities must be traced separately.

Identity and permission handoff. Weblate, an unchanged baseline, shows how a low-rated permission defect could open a deeper operation. In the pinned source, ordinary permission collection skips a team that enforces two-factor authentication (`weblate/auth/models.py:791–799`), while global permission collection

uses that team's roles (`:874–880`). Project creation checks `project.add` (`weblate/trans/views/create.py:112–115`). A separate backup-restore flaw can write version-control content (`weblate/trans/backups.py:918–935`); later processing of that content is the proposed server-execution step. This conditional route requires a privileged password, the specified team policy and that later processing trigger. It is not an anonymous-entry route.

Privileged browser to server. Ghost's local fix closes the selected anonymous Content API route. Remaining records propose a different path: attacker-influenced Portal content reaches an administrator's browser, then a browser/session capability is used toward a theme-helper path to server execution. The relevant code-step files remain after the fix. An obsolete fixture citation and an incomplete permission handoff prevent those records from establishing the full server outcome yet; they identify a concrete cross-boundary path for adjudication.

Routes a fix actually breaks. Four proposed Gradio file-read chains labelled B1 Critical used `CVE-2023-51449` as their second step. The local fix removes that step, so those chains do not survive in the modified fork. Other Gradio records involve a victim browser reaching localhost or `share=True` tunnel behaviour. The browser, tunnel and CI runner remain distinct environments; the source does not connect them into one server-execution chain.

Other proposed routes use an identifier disclosed before a gated Flowise or Langflow operation, an Airflow log or import error as an information carrier, an n8n OpenID Connect flow requiring a real identity-provider account, or Open WebUI content rendered in a browser before PDF creation. The catalogue directs attention to the next consumer in each case. Reading a log does not execute it; exporting a PDF does not run embedded code. None of the reviewed source routes establishes a malicious browser extension or virtual-machine escape.

Longer compositions and shared findings. Joining source-reviewed route segments shows why a finding list alone hides route structure. In Budibase, `CVE-2026-73406` exposes identity information that can feed the SSO account takeover in `CVE-2026-73302`; the resulting global account session can reach a separate license-control flaw (`GHSA-4wr8-5c3p-rjcr`). This is a conditional three-flaw path to account and license control. In Ghost, a staff password hash exposure can feed a device-verification bypass; the resulting staff session can then reach separate administrator or staff operations. Four recorded hash-source findings and three downstream outcomes produce 12 ledger combinations, but they share mechanisms and are not 12 independent chain types. In Keycloak, a valid initial-access token can open a DCR client management flaw, a victim administrator's visit can create an admin session, and separate realm-permission and component/LDAP flaws can extend that session to server execution. The route depends on the token, victim action, role and component configuration.

Long end-to-end paths also include non-code stages. In Weblate, a compromised privileged password, a team 2FA/global-permission flaw, project creation and backup/VCS processing form a conditional path involving two code flaws and a later worker operation. In pyLoad, low-privilege content reaches an administrator's browser, then a separate server-side flaw; the victim's view is the bridge. These paths have more attack stages than code flaws. A finding shared by several paths is counted once in the map and appears in every route that uses it.

The 54 joined records are a mechanical search result, not the study's chain count. Exact finding-ID joins find only links represented in the ledgers; capability-based joins can also connect a credential, role, artifact or ordinary operation to another vulnerable feature. The composed-path register lists all 54 generated paths, including rejected joins, and the full route register lists all 246 underlying route records. The present review

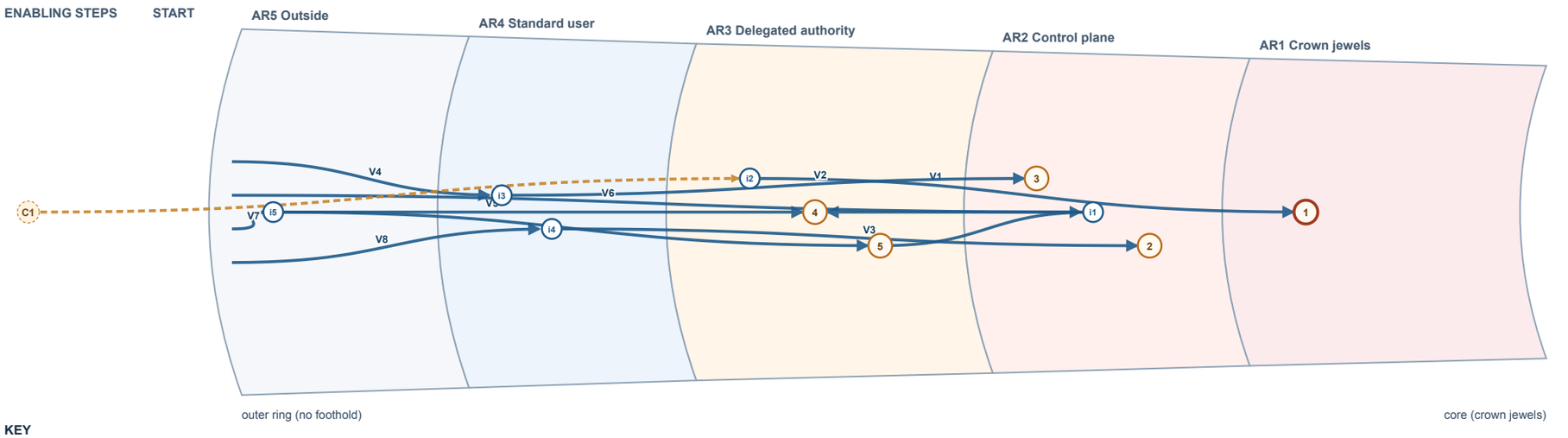
supports longer candidate families; it does not claim to have enumerated every possible capability-based composition in all 15 repositories.

5.3 Repository route maps

Figures 2–16 show one Access Ring map for each pinned application, with the rings drawn uncurled from AR5 (left) to AR1 (right). Vulnerabilities are labelled V1, V2, … and each map's legend gives the audit identifier for every code. Each map includes every source-anchored route record in the current audit. A vulnerability is one labelled arrow from the ring the attacker holds to the ring it reaches, so its length shows what it gives. Every arrow ends in a circle: a final outcome in the ring whose privilege it puts under attacker control, or an intermediate outcome (the position or capability a finding gives), and the next arrow starts from that circle. A finding used by several routes is drawn once. Dashed amber steps are supporting conditions (credentials, victim actions, configuration, files) and are not counted as vulnerabilities.

Rings are assigned from the reviewed capability and outcome of each step under the rule in §3.4; judgement calls are marked in the classification table. Where the ledger does not record a starting position for a first code flaw, its arrow enters from outside the rings and no starting ring is assumed. Exposure Rings are omitted because the snapshots carry no deployment position. Section 6.4 reads what the maps show. Full source gates and conditional assumptions remain in the route register and the route audit. All supporting files are in the project repository.

Figure 2. Airflow 3.1.0: all 5 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- i1 — Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- 1 — Final outcome, B1 Critical (#).
- 2 — Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- C1 — Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2025-54550
V2 · CVE-2025-65995
V3 · CVE-2025-68675
V4 · CVE-2026-24098
V5 · CVE-2026-30911
V6 · CVE-2026-31987
V7 · CVE-2026-38743
V8 · CVE-2026-45426

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · exec:worker via example_xcom (UI users, even admin, are not meant to run code)
2 · AR2 · B2 · proxy credentials/client secret from unauthorised task logs
3 · AR2 · B2 · secrets from other teams' DAG kwargs
4 · AR3 · B2 · DAG author of one DAG approves/rejects HITL of another team's task
5 · AR3 · B2 · non-DAG-author with worker-log access forges HITL decisions for other tasks

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

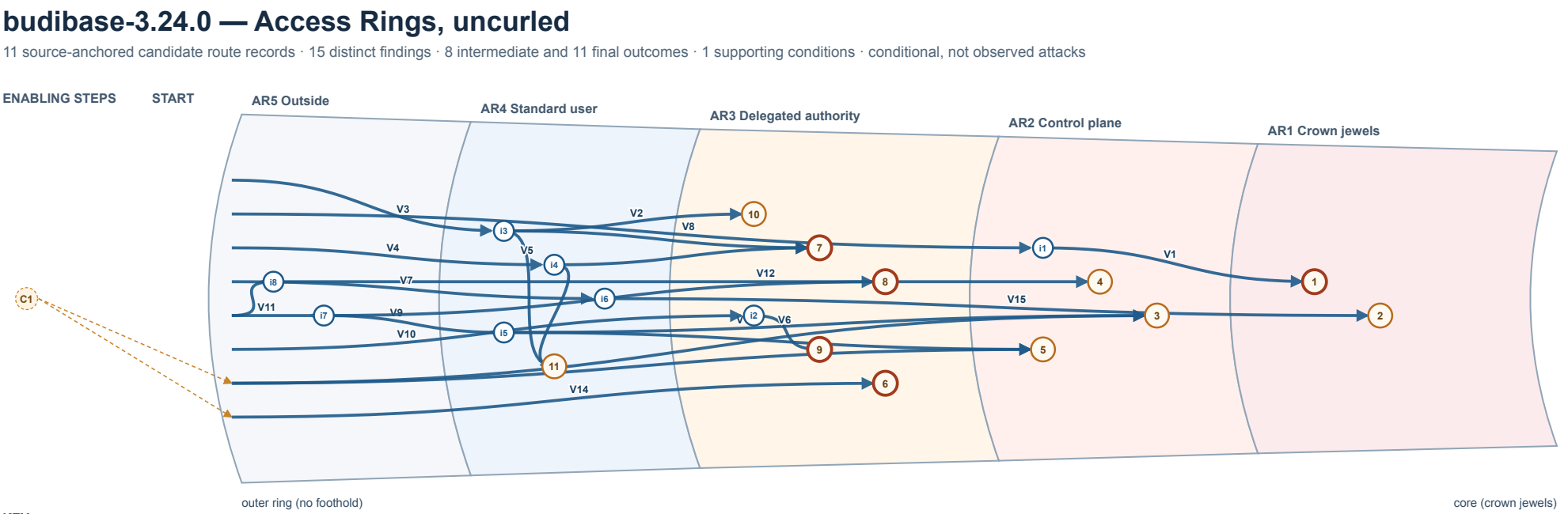
i1 · AR2 · secret:task execution-API JWT
i2 · AR3 · privileged credential held
i3 · AR4 · read:import errors of DAGs the user cannot read
i4 · AR4 · read:other tasks' log files on worker log server
i5 · AR5 · leak:task_instance_id (uuid)

Enabling steps (not vulnerabilities)

C1 · credential: admin password generated at first start (16 random chars, airflow-core/src/airflow/api_fastapi/auth/managers/simple/simple_auth_manager.py:128-130,373-374), …

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 3. Budibase 3.24.0: all 11 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- (i1) Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- (1) Final outcome, B1 Critical (#).
- (2) Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- (C1) Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2026-27702
V2 · CVE-2026-35218
V3 · CVE-2026-42239
V4 · CVE-2026-48147
V5 · CVE-2026-48149
V6 · CVE-2026-48150
V7 · CVE-2026-48151
V8 · CVE-2026-54352
V9 · CVE-2026-73302
V10 · CVE-2026-73305
V11 · CVE-2026-73406
V12 · GHSA-4946-qf2m-wrh5
V13 · GHSA-4wr8-5c3p-rjcr
V14 · GHSA-j82g-67x3-xcwh
V15 · GHSA-x7h8-ww3q-xv7c

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · builder reads env then abuses eval
2 · AR1 · B2 · shape webhook body schema then unauth SQLi via webhook
3 · AR2 · B2 · any SSO-autocreated user deletes/activates license
4 · AR2 · B2 · enumerate then read another tenant's app package
5 · AR2 · B2 · hijacked account deletes license
6 · AR3 · B1 · SSO-autocreated global user rewrites admin email -> account recovery -> admin
7 · AR3 · B1 · stored XSS in Text component steals admin auth JWT -> full ATO
8 · AR3 · B1 · targeted OIDC account-takeover of a named victim (incl admin)
9 · AR3 · B1 · workspace builder escalates via public roles API
10 · AR3 · B2 · builder Command Palette XSS steals builder JWT
11 · AR4 · B2 · CSRF-bypass lets a victim-driven request write attacker-controlled bound data (stored XSS seed)

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

i1 · AR2 · read:server-file (env secrets, e.g. cloud eval reachability)
i2 · AR3 · write:role-assignment (grant self app-builder)
i3 · AR4 · cookie readable by JS (httpOnly:false)
i4 · AR4 · csrf-bypass on protected endpoints
i5 · AR4 · session (logged-in global user)
i6 · AR4 · write:automation-trigger-outputs (unauth schema mutation)
i7 · AR5 · knows:victim-email/ssoId
i8 · AR5 · leak:appId/tenant identifiers

Enabling steps (not vulnerabilities)

C1 · config:sso-autocreate

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 4. Flowise 3.0.7: all 9 source-anchored route records.

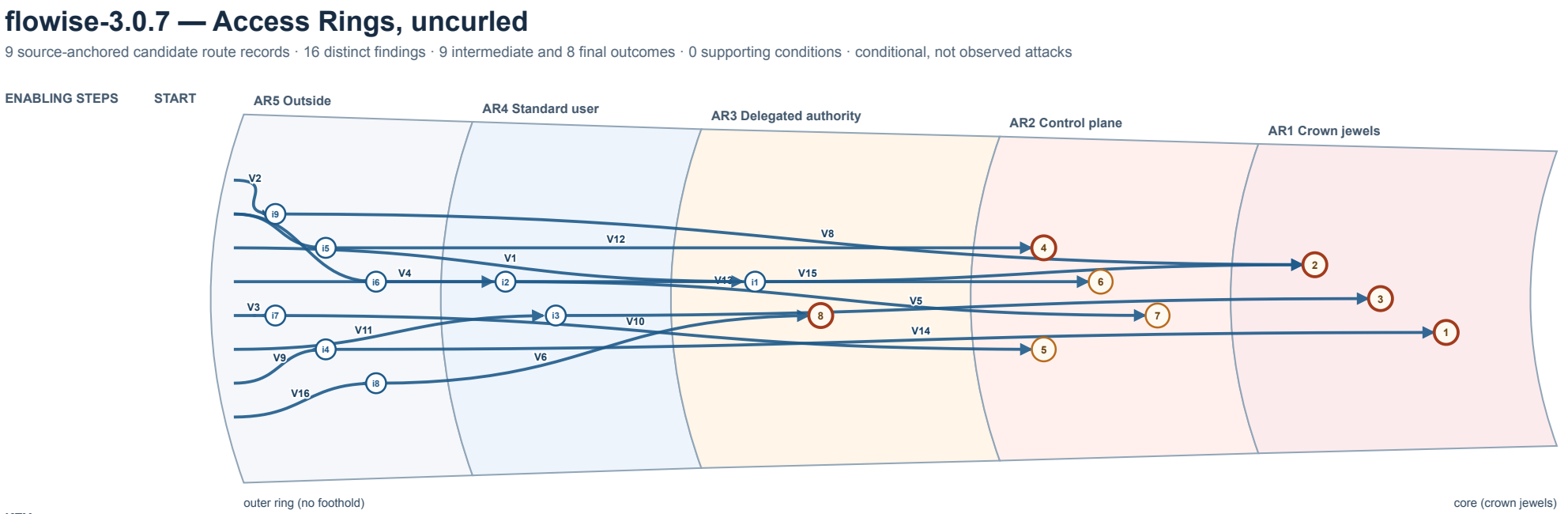
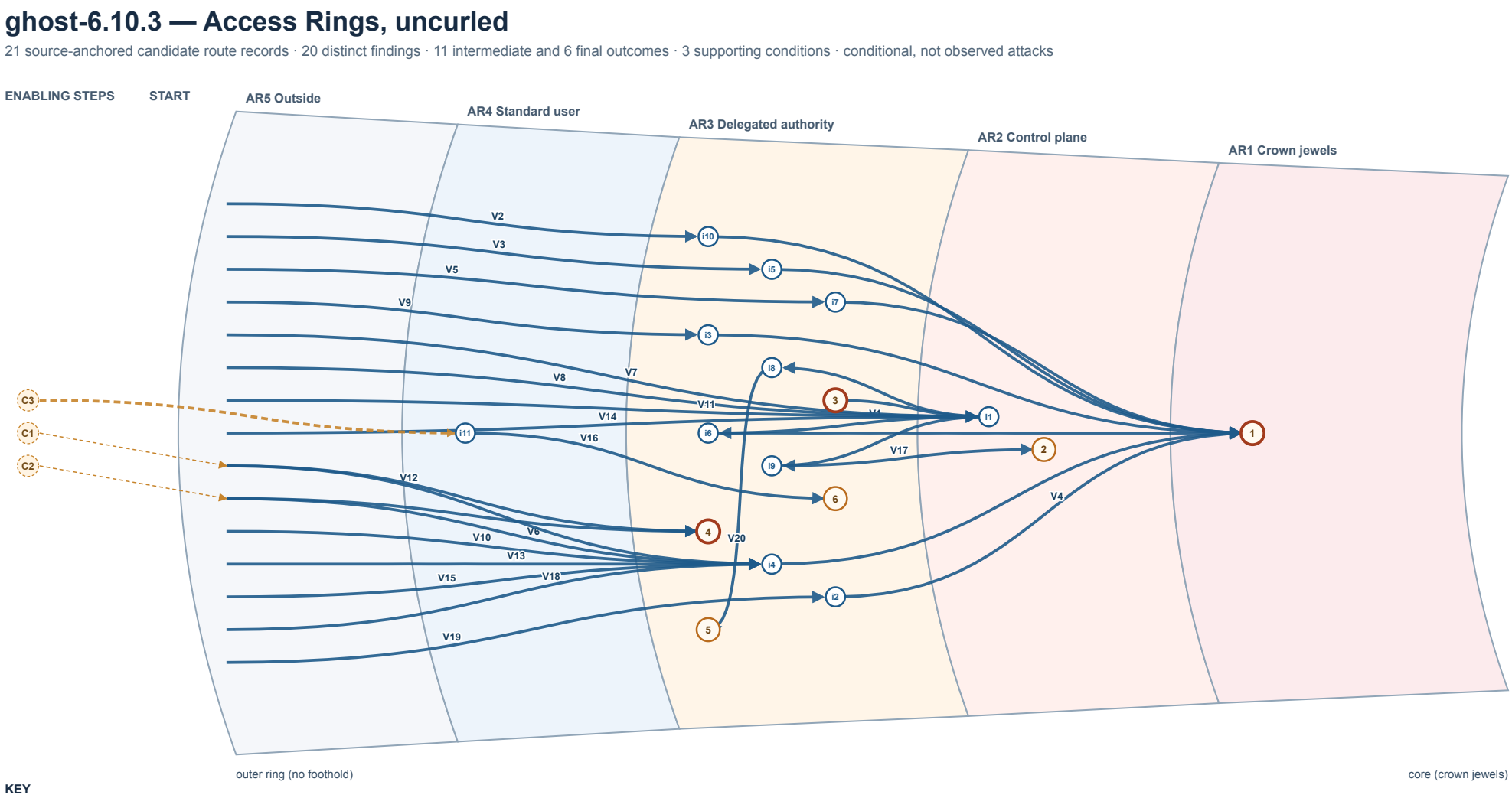


Figure 5. Ghost 6.10.3: all 21 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- i1 — Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- 1 — Final outcome, B1 Critical (#).
- 2 — Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- C1 — Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2026-22594
V2 · CVE-2026-22596
V3 · CVE-2026-24778
V4 · CVE-2026-29053
V5 · CVE-2026-29784
V6 · CVE-2026-53943
V7 · CVE-2026-53949
V8 · CVE-2026-70590
V9 · CVE-2026-70596
V10 · GHSA-347q-26qq-h2p6
V11 · GHSA-53vv-xm9f-mm82
V12 · GHSA-5pxf-whwv-g54g
V13 · GHSA-8575-cr6v-7jh4
V14 · GHSA-ghq6-q78f-2cpg
V15 · GHSA-mp9q-4q6m-44f5
V16 · GHSA-q734-xjgc-vpj9
V17 · GHSA-q98w-jm4q-wch7
V18 · GHSA-v3xr-g6p2-fvgv
V19 · GHSA-v6q3-xqxm-6f5v
V20 · GHSA-x3mg-q38v-m562

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · exec:server
2 · AR2 · B2 · secret:third-party API keys
3 · AR3 · B1 · session:Owner/Administrator past 2FA
4 · AR3 · B1 · staff session via victim
5 · AR3 · B2 · delete any post
6 · AR3 · B2 · reactivated staff session

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

i1 · AR2 · secret:staff password (hash from oracle + offline cracking)
i2 · AR3 · role:Administrator via pending Admin invite token
i3 · AR3 · session:Administrator (victim opens post history)
i4 · AR3 · session:Administrator (victim)
i5 · AR3 · session:Administrator (victim, same origin)
i6 · AR3 · session:Administrator/Owner past 2FA
i7 · AR3 · session:Administrator/Owner past device verification
i8 · AR3 · session:Author (with Author password)
i9 · AR3 · session:Contributor+ past 2FA (with that staff password)
i10 · AR3 · session:Owner/Administrator via DB read
i11 · AR4 · standard-user account or credential held

Enabling steps (not vulnerabilities)

C1 · config:comments_enabled on
C2 · config:shared cache in front of Ghost
C3 · credential:former staff (suspended) mailbox

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 6. Gradio 4.8.0: all 5 source-anchored route records.

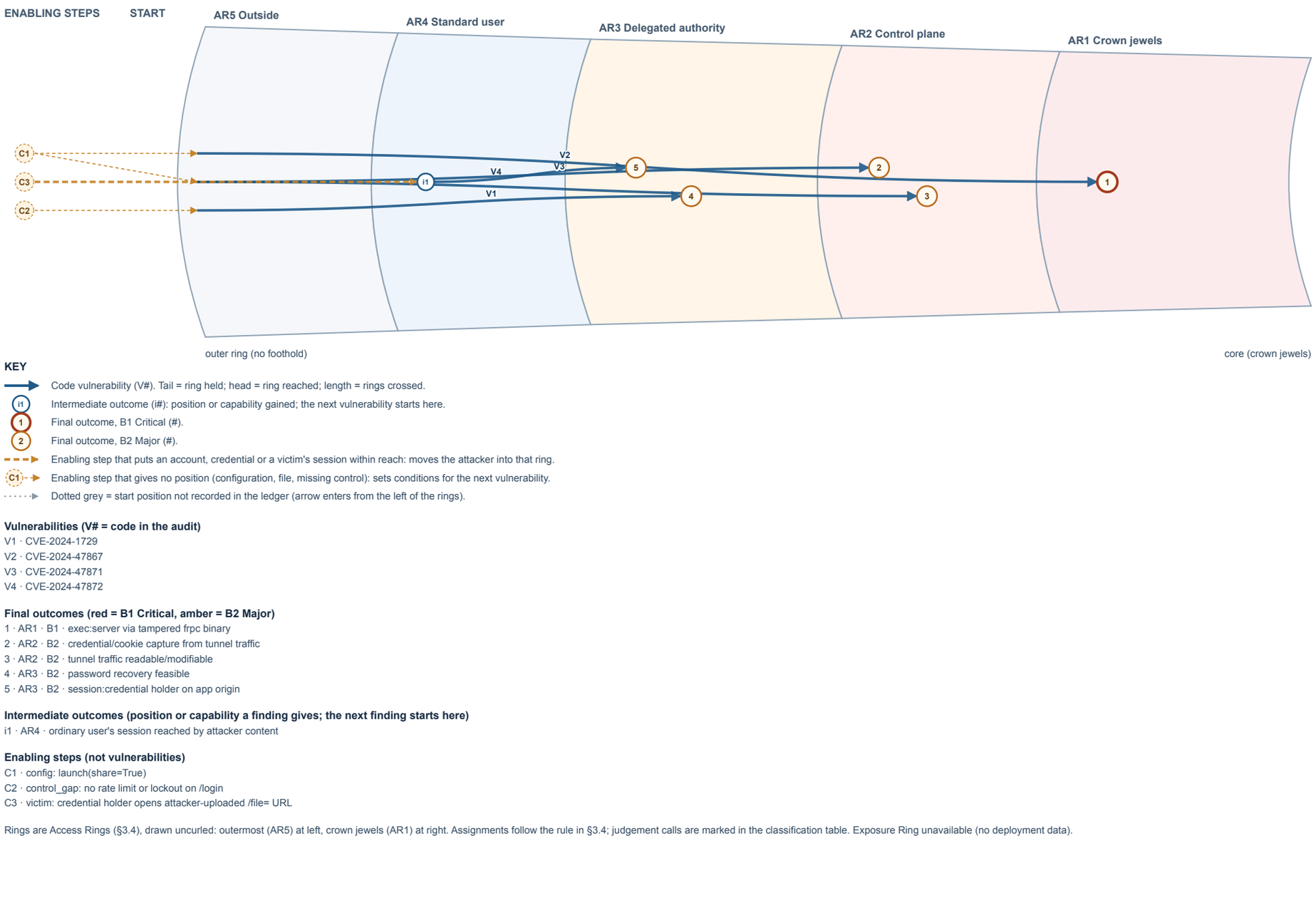
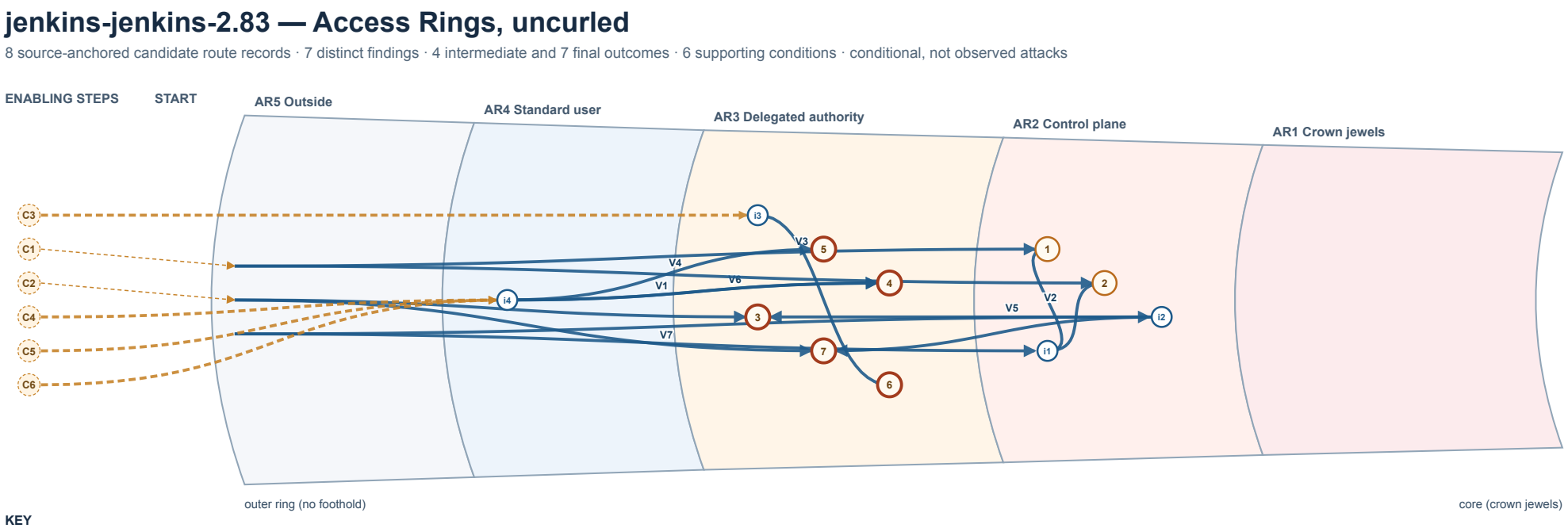


Figure 7. Jenkins 2.83: all 8 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- (i1) Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- (1) Final outcome, B1 Critical (#).
- (2) Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- (C1) Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2017-1000392
V2 · CVE-2017-1000401
V3 · CVE-2017-1000504
V4 · CVE-2018-1999007
V5 · CVE-2018-1999045
V6 · CVE-2023-43495
V7 · CVE-2024-23897

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR2 · B2 · secret:form passwords in logs
2 · AR2 · B2 · secret:logged form passwords
3 · AR3 · B1 · cookie still authenticates as admin
4 · AR3 · B1 · session:admin
5 · AR3 · B1 · session:admin (any account = admin)
6 · AR3 · B1 · session:admin actions
7 · AR3 · B1 · session:admin despite remember-me disabled

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

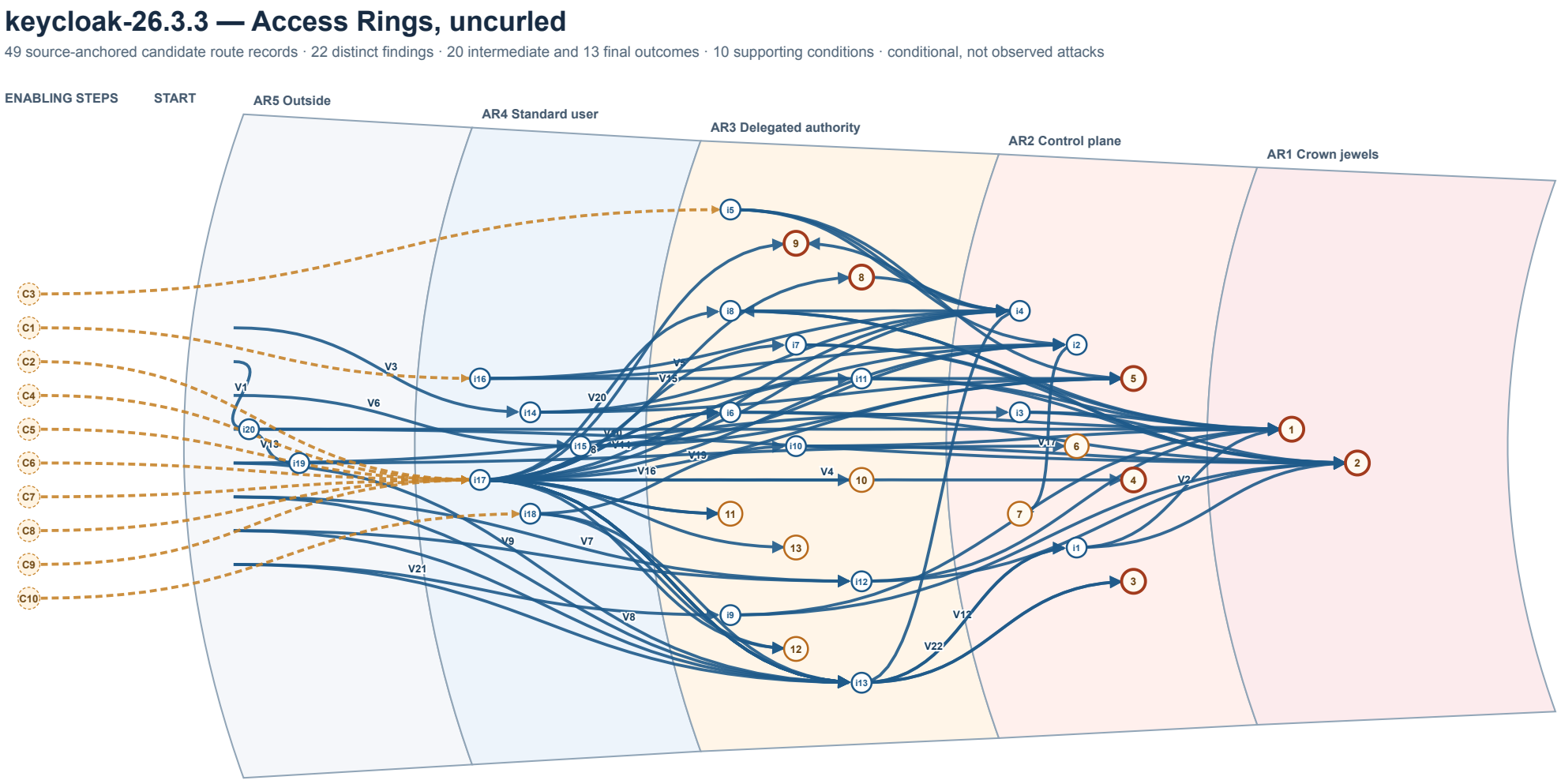
i1 · AR2 · read:server files (access log on disk)
i2 · AR2 · secret:remember-me signing inputs + user record
i3 · AR3 · victim's privileged session reached by attacker content
i4 · AR4 · ordinary user's session reached by attacker content

Enabling steps (not vulnerabilities)

C1 · config: HTTP access logging enabled
C2 · config: disableRememberMe=true (admin mitigation)
C3 · victim: admin browses attacker page during startup window
C4 · victim: logged-in user opens crafted 404 URL (needs non-default Stapler trace)
C5 · victim: logged-in user uses auto-complete with attacker-fed suggestions
C6 · victim: logged-in user views console with plugin caption

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 8. Keycloak 26.3.3: all 49 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- i1 — Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- 1 — Final outcome, B1 Critical (#).
- 2 — Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- C1 — Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2025-10939
V2 · CVE-2025-13467
V3 · CVE-2026-1529
V4 · CVE-2026-15572
V5 · CVE-2026-16102
V6 · CVE-2026-16442
V7 · CVE-2026-16443
V8 · CVE-2026-18963
V9 · CVE-2026-2092
V10 · CVE-2026-2603
V11 · CVE-2026-3009
V12 · CVE-2026-3121
V13 · CVE-2026-3190
V14 · CVE-2026-3429
V15 · CVE-2026-37982
V16 · CVE-2026-4325
V17 · CVE-2026-4630
V18 · CVE-2026-4636
V19 · CVE-2026-7507
V20 · CVE-2026-9086
V21 · CVE-2026-9087
V22 · CVE-2026-9099

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · exec:server beyond realm admin
2 · AR1 · B1 · exec:server with externally held admin credential
3 · AR2 · B1 · admin:realm
4 · AR2 · B1 · disallowed mapper -> forged claims
5 · AR2 · B1 · manage-clients on DCR, all client secrets
6 · AR2 · B2 · grant self on other users resources
7 · AR2 · B2 · write resources of other resource servers
8 · AR3 · B1 · session:admin victim
9 · AR3 · B1 · session:admin victim who follows client link
10 · AR3 · B2 · replay action token -> session:victim
11 · AR3 · B2 · session:linked users
12 · AR3 · B2 · session:victim
13 · AR3 · B2 · session:victim full, OTP removed

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

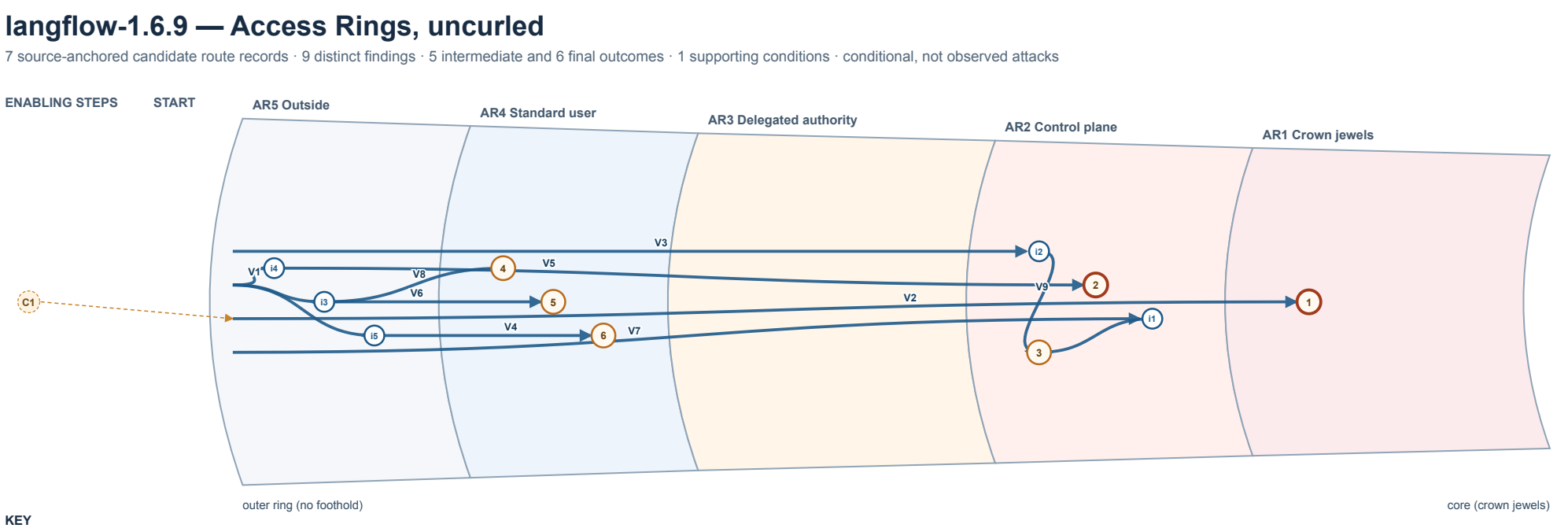
i1 · AR2 · admin:realm incl. manage-realm
i2 · AR2 · secret:resource-server client secret (uma_protection)
i3 · AR2 · session:realm admin via reset-flow bypass (needs resetPasswordAllowed)
i4 · AR2 · write:any client URL via DCR manage-clients
i5 · AR3 · privileged credential held
i6 · AR3 · session:admin linked to disabled IdP
i7 · AR3 · session:admin via replayed execute-actions
i8 · AR3 · session:admin via stored XSS in client URL
i9 · AR3 · session:admin victim via IdP link
i10 · AR3 · session:admin victim via session fixation
i11 · AR3 · session:admin with MFA deleted
i12 · AR3 · session:brokered admin
i13 · AR3 · session:delegated admin victim
i14 · AR4 · account via forged org invitation
i15 · AR4 · account via link-only SAML IdP
i16 · AR4 · standard-user account available (open registration or granted permission)
i17 · AR4 · standard-user account or credential held
i18 · AR4 · ordinary user's session reached by attacker content
i19 · AR5 · leak:victim resource uuids
i20 · AR5 · reach:/admin behind path-filtering proxy

Enabling steps (not vulnerabilities)

C1 · config:self-registration enabled (registrationAllowed default false, model/jpa/src/main/java/org/keycloak/models/jpa/entities/RealmEntity.java:70)
C2 · credential:control of or login at a disabled IdP
C3 · credential:initial access token (admin-issued; anonymous DCR blocked by default Trusted Hosts, services/src/main/java/org/keycloak/services/clientregistration/policy/Def…
C4 · credential:initial access token (as above)
C5 · credential:initial/registration access token
C6 · credential:issued account (phished/reused password of any realm user)
C7 · credential:leaked execute-actions link
C8 · credential:phished password of MFA-protected user
C9 · credential:used action token from victim mailbox
C10 · victim:user follows crafted login link

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 9. Langflow 1.6.9: all 7 source-anchored route records.



Vulnerabilities (V# = code in the audit)
V1 · CVE-2026-21445
V2 · CVE-2026-27966
V3 · CVE-2026-33497
V4 · CVE-2026-33760
V5 · CVE-2026-48520
V6 · CVE-2026-55255
V7 · CVE-2026-55447
V8 · GHSA-4hmc-cfm3-w43c
V9 · GHSA-jxw3-mjmx-3pqm

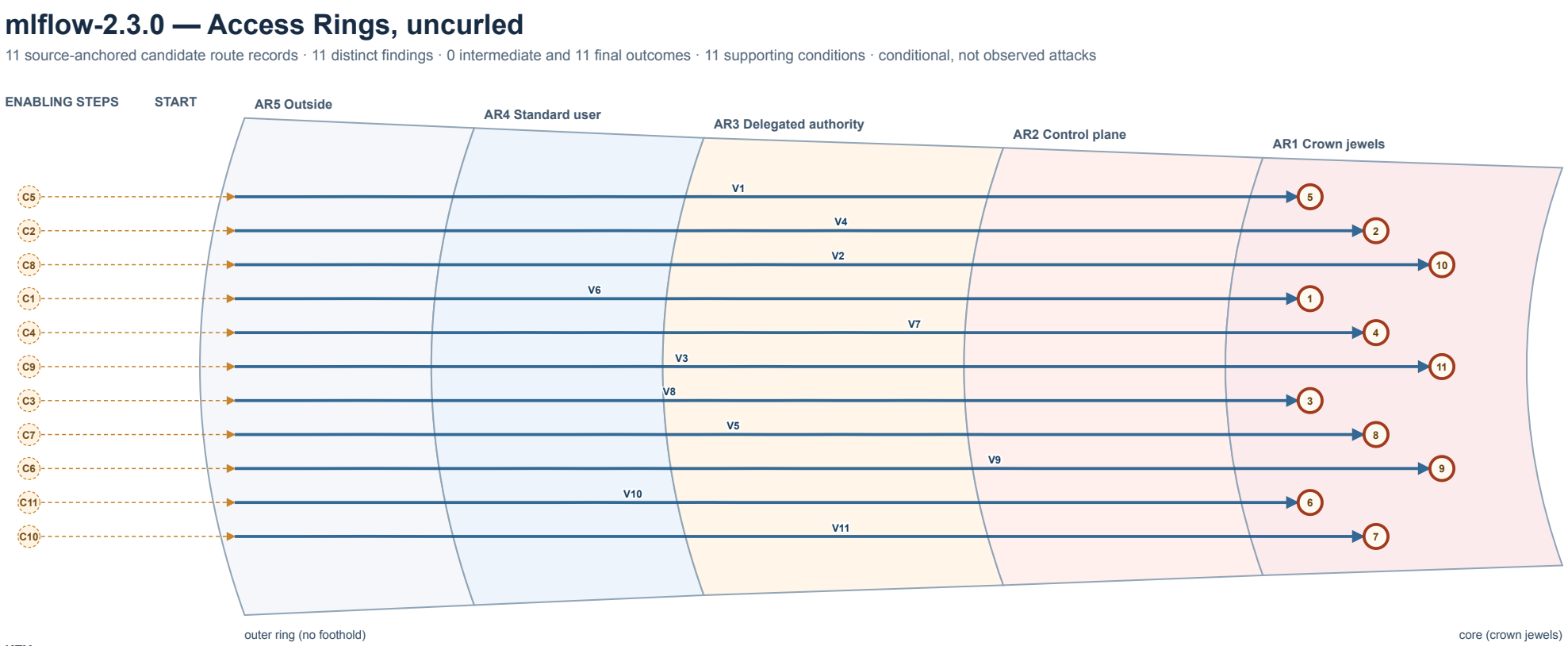
Final outcomes (red = B1 Critical, amber = B2 Major)
1 · AR1 · B1 · exec:server
2 · AR2 · B1 · read:server files
3 · AR2 · B2 · secret:decrypted stored credentials
4 · AR4 · B2 · read:other users' flow files
5 · AR4 · B2 · run another user's flow; read its outputs
6 · AR4 · B2 · write:modify other users' messages

Intermediate outcomes (position or capability a finding gives; the next finding starts here)
i1 · AR2 · read:encrypted credentials
i2 · AR2 · read:encrypted credentials in config_dir sqlite (env-set short key => no key file)
i3 · AR5 · leak:flow_id
i4 · AR5 · leak:flow_id (candidate public flow ids)
i5 · AR5 · leak:flow_id/session_id/message_id

Enabling steps (not vulnerabilities)
C1 · config: public flow containing CSV Agent

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 10. MLflow 2.3.0: all 11 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- i1 — Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- 1 — Final outcome, B1 Critical (#).
- 2 — Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- C1 — Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2023-6709
V2 · CVE-2023-6940
V3 · CVE-2024-37052
V4 · CVE-2024-37054
V5 · CVE-2024-37055
V6 · CVE-2024-37056
V7 · CVE-2024-37057
V8 · CVE-2024-37059
V9 · CVE-2024-37060
V10 · CVE-2024-37061
V11 · GHSA-gqvg-gmmx-x4hm

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · execution in the LightGBM model-loading process
2 · AR1 · B1 · execution in the PyFunc model-loading process
3 · AR1 · B1 · execution in the PyTorch model-loading process
4 · AR1 · B1 · execution in the TensorFlow model-loading process
5 · AR1 · B1 · execution in the card-rendering process
6 · AR1 · B1 · execution in the local project-runner process
7 · AR1 · B1 · execution in the model-loading process
8 · AR1 · B1 · execution in the pmdarima model-loading process
9 · AR1 · B1 · execution in the recipe-inspection process
10 · AR1 · B1 · execution in the recipe-rendering process
11 · AR1 · B1 · execution in the sklearn model-loading process

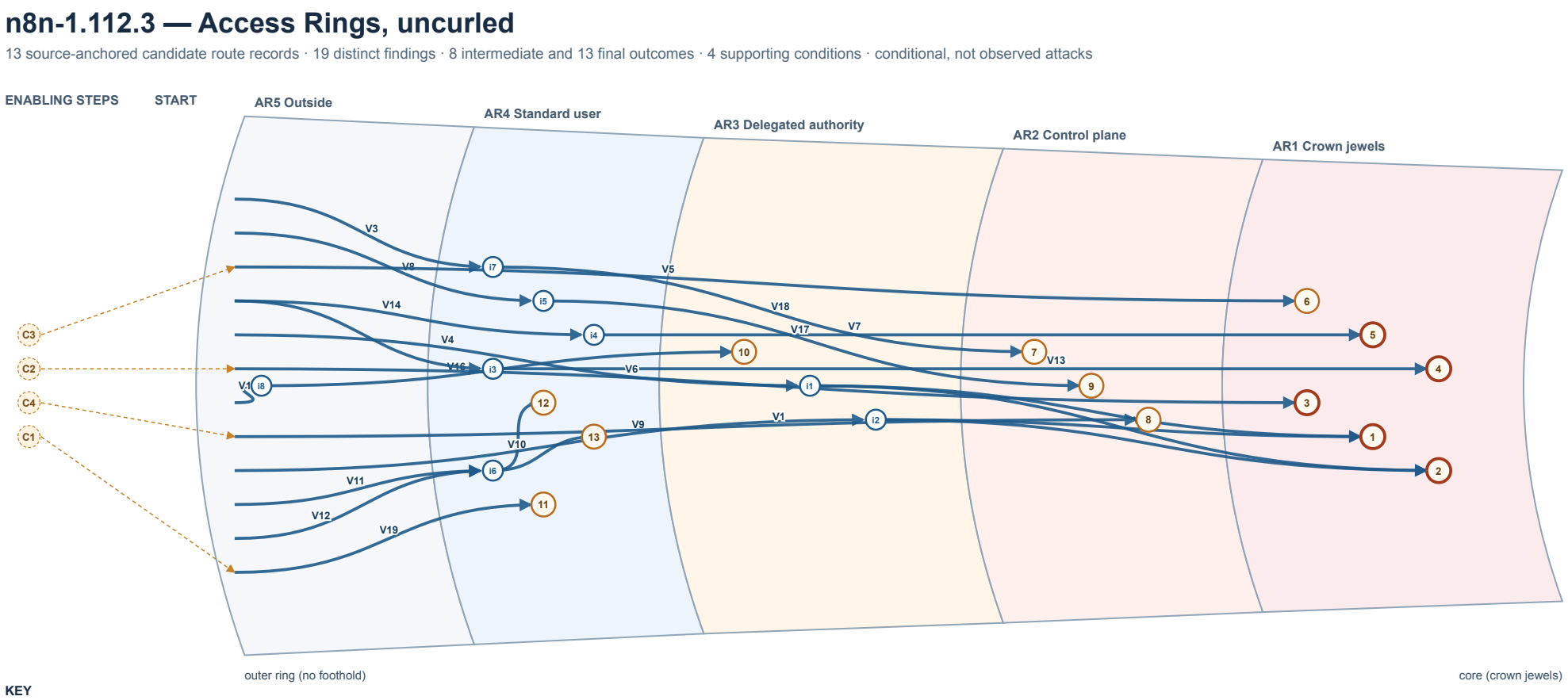
Intermediate outcomes (position or capability a finding gives; the next finding starts here)

Enabling steps (not vulnerabilities)

C1 · artifact: untrusted LightGBM sklearn-flavor model selected for loading
C2 · artifact: untrusted PyFunc python-model file selected for loading
C3 · artifact: untrusted PyTorch model selected for loading
C4 · artifact: untrusted TensorFlow custom-objects file selected for loading
C5 · artifact: untrusted card-tab template chosen for rendering
C6 · artifact: untrusted card.pkl placed in recipe output inspected by a user
C7 · artifact: untrusted pmdarima model selected for loading
C8 · artifact: untrusted recipe or profile YAML chosen for rendering
C9 · artifact: untrusted sklearn model selected for loading
C10 · artifact: untrusted statsmodels model selected for loading
C11 · input: untrusted extra parameter key reaches an MLproject run

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 11. n8n 1.112.3: all 13 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- i1 — Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- 1 — Final outcome, B1 Critical (#).
- 2 — Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- C1 — Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2026-21858
V2 · CVE-2026-21893
V3 · CVE-2026-21894
V4 · CVE-2026-25054
V5 · CVE-2026-25055
V6 · CVE-2026-27493
V7 · CVE-2026-33660
V8 · CVE-2026-33663
V9 · CVE-2026-33665
V10 · CVE-2026-44790
V11 · CVE-2026-54306
V12 · CVE-2026-59206
V13 · CVE-2026-86076
V14 · CVE-2026-86084
V15 · CVE-2026-86994
V16 · GHSA-597w-c3jh-g8fg
V17 · GHSA-64xh-79j6-r5v8
V18 · GHSA-953p-jm2c-8h5j
V19 · GHSA-w24g-6454-7w7f

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · admin community-package command injection
2 · AR1 · B1 · admin session -> community-package install command injection
3 · AR1 · B1 · anon expression evaluation -> exec:server with no account
4 · AR1 · B1 · anon obtains a valid member session via disabled-OIDC endpoints, then reaches member-default code execution
5 · AR1 · B1 · anon obtains member session, then authors a Merge/AlaSQL workflow -> exec:server
6 · AR1 · B2 · anon arbitrary file write on a remote host
7 · AR2 · B2 · anon NoSQL injection via a signature-verifying trigger
8 · AR2 · B2 · anon arbitrary server file access
9 · AR2 · B2 · name-based credential theft places a domain-restricted foreign credential in the attacker workflow; the allowed-domains bypass then exfiltrates its p…
10 · AR3 · B2 · cross-project approval forgery
11 · AR4 · B2 · anon cross-session chat-history disclosure
12 · AR4 · B2 · anon public-webhook prototype pollution supplies the process-wide PP primitive that the Git node arbitrary-file-read requires
13 · AR4 · B2 · authenticated PP primitive feeds Git file read

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

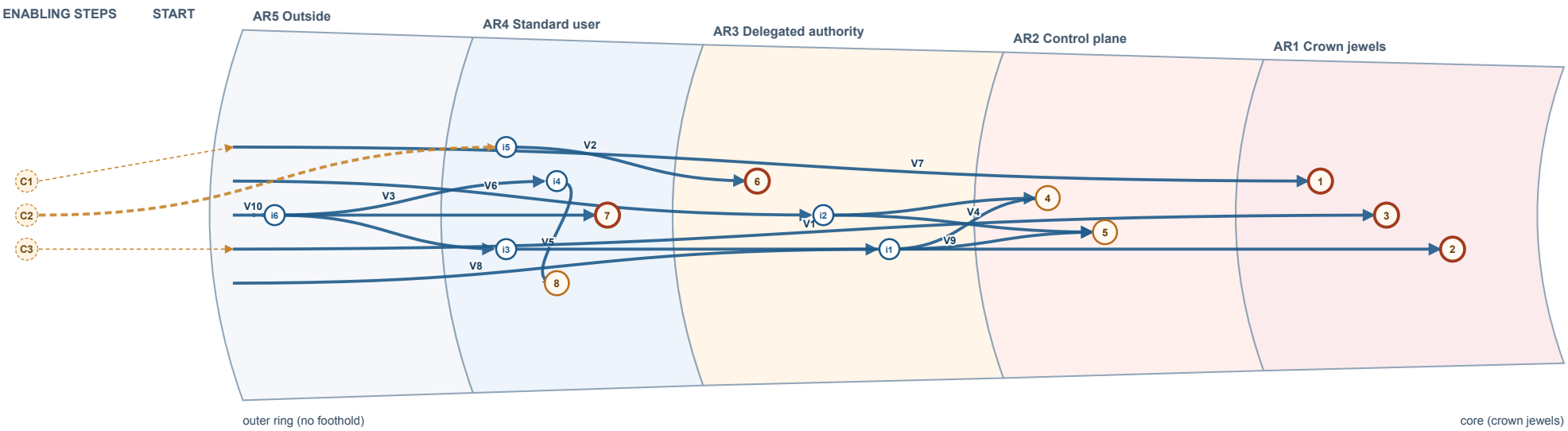
i1 · AR3 · session:admin (stored markdown XSS viewed by an owner/admin victim) -> admin
i2 · AR3 · session:owner/admin (LDAP e-mail linking hijacks a local account whose email the attacker sets in LDAP) -> supplies 'admin'
i3 · AR4 · session:member
i4 · AR4 · session:member (disabled-OIDC endpoints still issue a valid cookie) -> supplies 'account' an outsider cannot self-register
i5 · AR4 · use-of-cross-user-credential
i6 · AR4 · write:Object.prototype
i7 · AR4 · write:inject-workflow-data (forged Stripe event, no signature check) delivers attacker JSON into a MongoDB node
i8 · AR5 · leak:active-workflow-ids narrows the guessable node-id space for the signed-resume forgery

Enabling steps (not vulnerabilities)

C1 · config:public-ChatTrigger
C2 · config:public-Form-node
C3 · config:public-upload-webhook
C4 · config:public-webhook

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 12. Open WebUI 0.6.30: all 8 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- (i1) Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- (1) Final outcome, B1 Critical (#).
- (2) Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- (C1) Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2025-64495
V2 · CVE-2026-44551
V3 · CVE-2026-44552
V4 · CVE-2026-44560
V5 · CVE-2026-45315
V6 · CVE-2026-45672
V7 · CVE-2026-45675
V8 · CVE-2026-59216
V9 · CVE-2026-59218

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · code execution despite ENABLE_CODE_EXECUTION=false
2 · AR1 · B1 · cross-user tool/code execution as another user
3 · AR1 · B1 · stored DOM XSS -> ATO/RCE
4 · AR2 · B2 · admin-gated Redis cache poisoning reached without native admin
5 · AR2 · B2 · admin-gated Redis tool_servers cache poisoning across instances
6 · AR3 · B1 · code execution in victim browser (new Function)
7 · AR4 · B1 · session:user via empty-password LDAP bind
8 · AR4 · B2 · read cross-user files/KB with a bootstrapped account

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

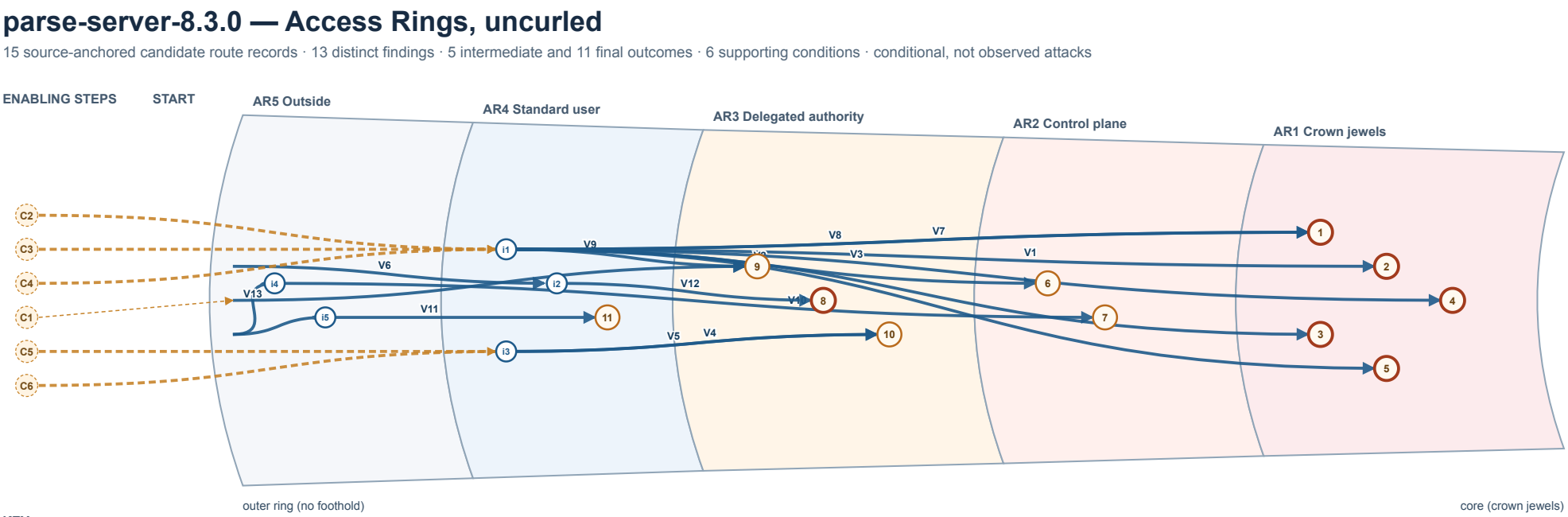
i1 · AR3 · admin session (first-user race mints extra admin)
i2 · AR3 · session:admin (stored XSS lands in an admin browser -> account takeover)
i3 · AR4 · session:user
i4 · AR4 · session:user / account (account is not free: DEFAULT_USER_ROLE=pending)
i5 · AR4 · ordinary user's session reached by attacker content
i6 · AR5 · leak:username -> knows:username needed to bind

Enabling steps (not vulnerabilities)

C1 · config:CODE_EXECUTION_ENGINE=jupyter
C2 · victim-adds-attacker-model-server
C3 · victim-enables-insertPromptAsRichText

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 13. Parse Server 8.3.0: all 15 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- (i1) Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- (1) Final outcome, B1 Critical (#).
- (2) Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- (C1) Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2026-29182
V2 · CVE-2026-30228
V3 · CVE-2026-30229
V4 · CVE-2026-30863
V5 · CVE-2026-30949
V6 · CVE-2026-31872
V7 · CVE-2026-32234
V8 · CVE-2026-33539
V9 · CVE-2026-34215
V10 · CVE-2026-34784
V11 · CVE-2026-53726
V12 · GHSA-863r-39r9-vfcf
V13 · GHSA-97pr-9hgg-3p8r

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · SQL write
2 · AR1 · B1 · cloud hooks -> all hooked reads/writes
3 · AR1 · B1 · full write as any user
4 · AR1 · B1 · full write as any user via /loginAs
5 · AR1 · B1 · hooks take over triggers/functions
6 · AR2 · B2 · create/delete any file
7 · AR2 · B2 · protected file read
8 · AR3 · B1 · LDAP account takeover
9 · AR3 · B2 · MFA secret -> login despite MFA
10 · AR3 · B2 · session:victim
11 · AR4 · B2 · protected relation read on ACL-hidden owner

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

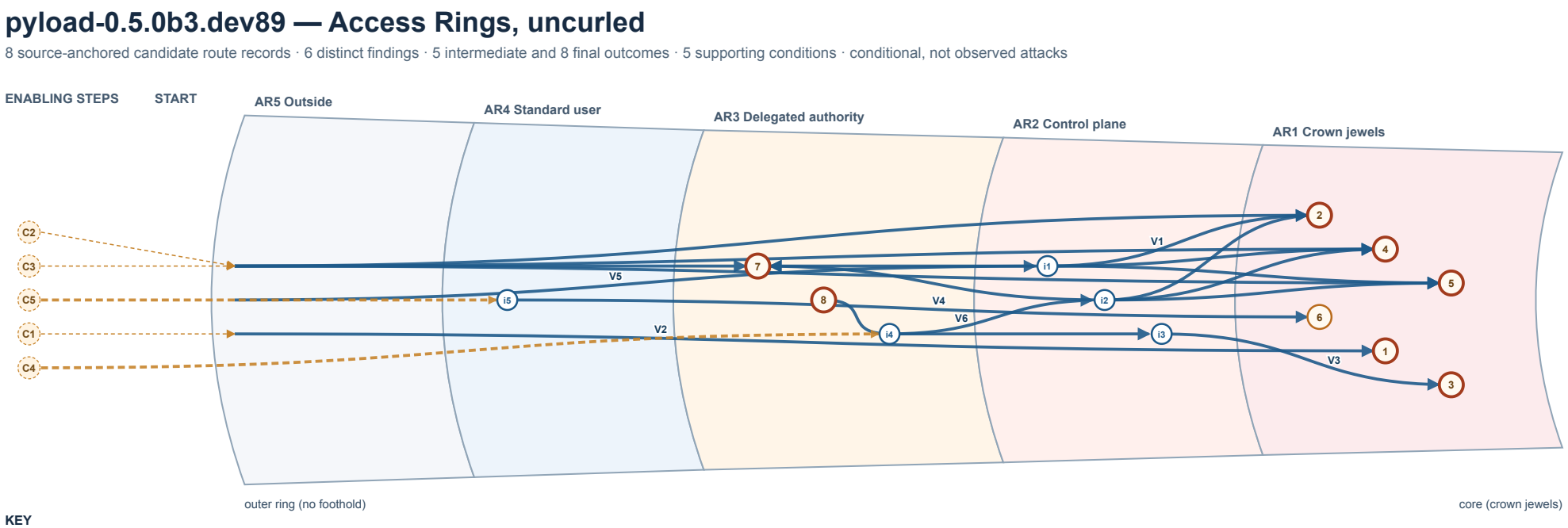
i1 · AR4 · standard-user account or credential held
i2 · AR4 · leak of user emails/usernames
i3 · AR4 · ordinary user's session reached by attacker content
i4 · AR5 · file names in leaked object bodies
i5 · AR5 · object ids from leaked bodies

Enabling steps (not vulnerabilities)

C1 · control_gap: no login lockout or rate limit by default (accountLockout, rateLimit unset; src/Config.js:333)
C2 · credential: readOnlyMasterKey held by lower-trust party
C3 · credential: readOnlyMasterKey held by lower-trust party (analytics/dashboard); note src/middlewares.js:265-277 applies no masterKeyIps check to it
C4 · credential: victim password (reuse/stuffing)
C5 · victim: victim signs into an attacker-controlled Google/Apple OAuth client
C6 · victim: victim uses another client in the same Keycloak realm that the attacker controls

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 14. pyLoad 0.5.0b3: all 8 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- (i1) Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- (1) Final outcome, B1 Critical (#).
- (2) Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- (C1) Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2026-33509
V2 · CVE-2026-35463
V3 · CVE-2026-35464
V4 · CVE-2026-35592
V5 · CVE-2026-45306
V6 · CVE-2026-45348

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · SETTINGS user reaches OS command exec, but only if the addon is enabled
2 · AR1 · B1 · SETTINGS user reaches RCE, but only if reconnect is enabled
3 · AR1 · B1 · an ADD-only account, via XSS on a viewing admin, reaches SETTINGS and achieves server code execution
4 · AR1 · B1 · brute-force admin password then RCE; cheaper credential route exists (default creds)
5 · AR1 · B1 · same admin-session -> RCE, but 33509 needs non-default reconnect enabled
6 · AR1 · B2 · arbitrary file write on the server without an account (distinct from account-gated file-write rows)
7 · AR3 · B1 · SETTINGS user reads admin session file to become admin, but SETTINGS already reaches RCE directly
8 · AR3 · B1 · stored XSS executes in the admin's session (session:admin)

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

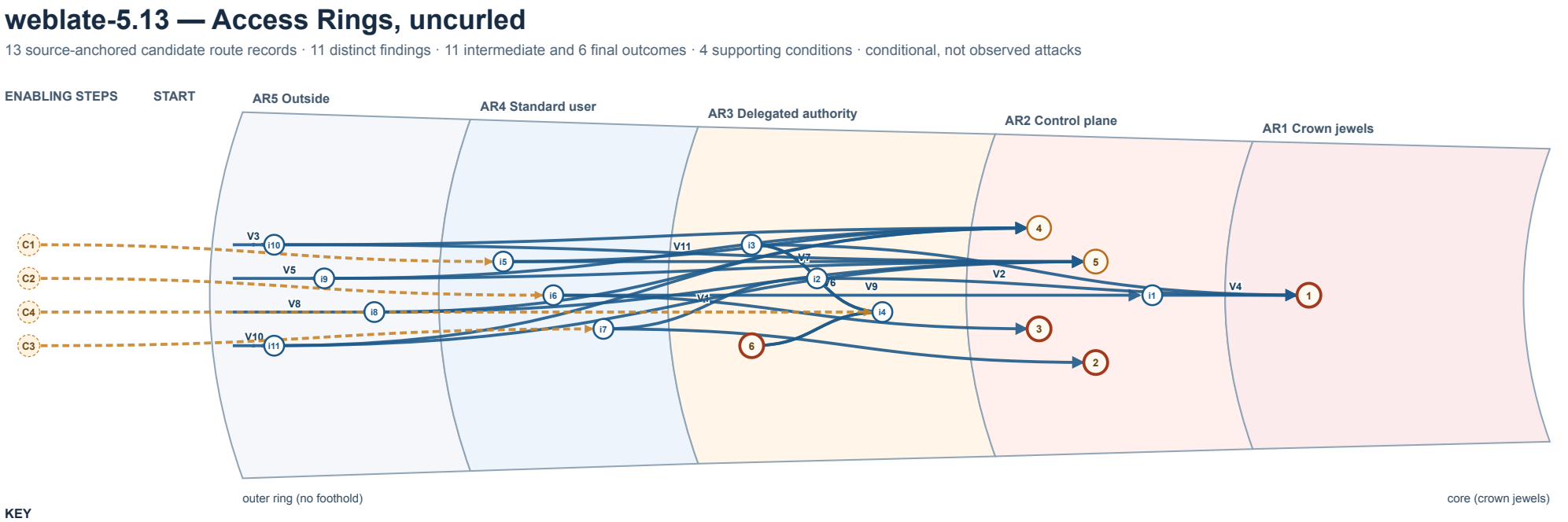
i1 · AR2 · read Flask session store -> session:admin -> role:SETTINGS
i2 · AR2 · session:admin -> role:SETTINGS
i3 · AR2 · session:admin -> role:SETTINGS+ADD
i4 · AR3 · victim's privileged session reached by attacker content
i5 · AR4 · ordinary user's session reached by attacker content

Enabling steps (not vulnerabilities)

C1 · config: AntiVirus addon enabled (non-default)
C2 · config: reconnect enabled (non-default)
C3 · control_gap: WebUI login has no rate limit or lockout
C4 · victim: an admin (or higher-priv user) opens the Downloads view that renders an attacker-planted link
C5 · victim: the app downloads and extracts an attacker-supplied archive

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 15. Weblate 5.13: all 13 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring held; head = ring reached; length = rings crossed.
- i1 — Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- 1 — Final outcome, B1 Critical (#).
- 2 — Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- C1 — Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2025-64725
V2 · CVE-2025-68398
V3 · CVE-2026-27457
V4 · CVE-2026-33435
V5 · CVE-2026-44263
V6 · CVE-2026-44264
V7 · CVE-2026-45106
V8 · CVE-2026-55227
V9 · CVE-2026-61790
V10 · CVE-2026-77507
V11 · CVE-2026-86042

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · exec:server from a starting point of account/credential/victim; no single vuln reaches B4 from anon/account
2 · AR2 · B1 · attacker's own registered account joins invited team, or becomes superuser
3 · AR2 · B1 · global permissions despite team-enforced 2FA
4 · AR2 · B2 · attacker component linked to another project's private repository (read+write); dominated by CVE-2026-33435 exec from the same project.add preconditi…
5 · AR2 · B2 · enables backup import
6 · AR3 · B1 · session:<victim role>

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

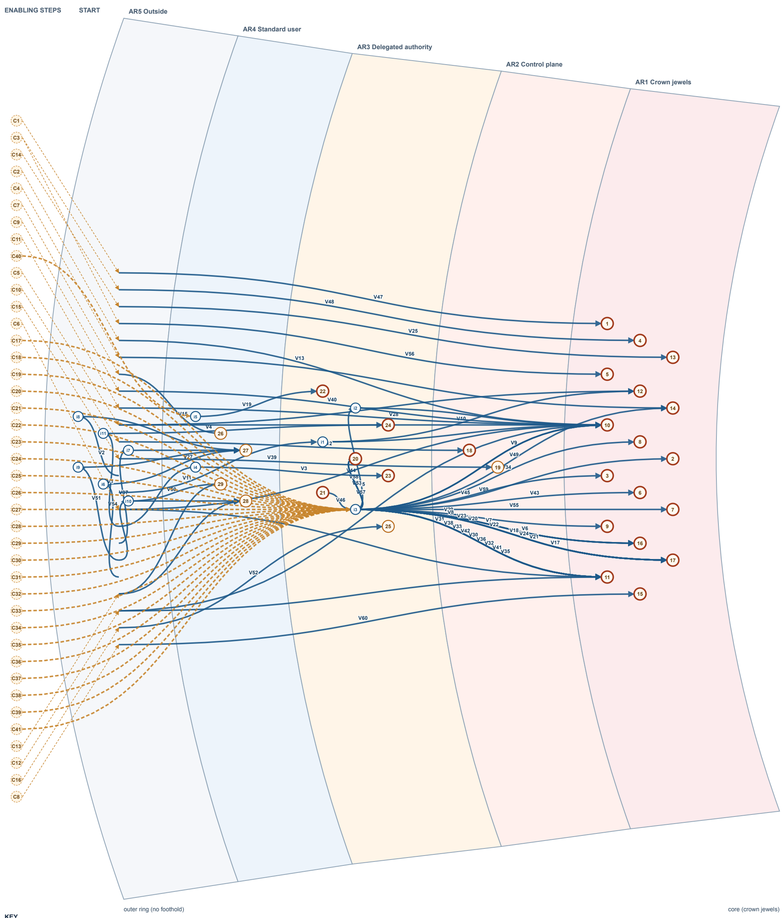
i1 · AR2 · role:project.add as global permission with 2FA bypassed
i2 · AR3 · role:component-admin via invitation into project Administration team (or superuser)
i3 · AR3 · session:project-admin of victim (requires relaxed CSP)
i4 · AR3 · victim's privileged session reached by attacker content
i5 · AR4 · standard-user account available (open registration or granted permission)
i6 · AR4 · standard-user account or credential held
i7 · AR4 · ordinary user's session reached by attacker content
i8 · AR5 · leak:confirms guessed private object exists (account)
i9 · AR5 · leak:confirms guessed private translation exists (account)
i10 · AR5 · leak:private component slugs from add-on API (anon)
i11 · AR5 · leak:private project/component slugs from language feed (anon)

Enabling steps (not vulnerabilities)

C1 · config: project.add granted (hosted billing trial / Project creators)
C2 · credential: phished/reused password of privileged user lacking 2FA (lockout at accounts/models.py:517 rules out brute force)
C3 · victim: invitation link forwarded/exposed by invitee
C4 · victim: project admin / superuser views attacker content

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figure 16. XWiki 14.5: all 63 source-anchored route records.



KEY

- Code vulnerability (V#). Tail = ring.held; head = ring reached; length = rings crossed.
- Intermediate outcome (i#): position or capability gained; the next vulnerability starts here.
- Final outcome, B1 Critical (#).
- Final outcome, B2 Major (#).
- Enabling step that puts an account, credential or a victim's session within reach: moves the attacker into that ring.
- Enabling step that gives no position (configuration, file, missing control): sets conditions for the next vulnerability.
- Dotted grey = start position not recorded in the ledger (arrow enters from the left of the rings).

Vulnerabilities (V# = code in the audit)

V1 · CVE-2022-41933
V2 · CVE-2022-41935
V3 · CVE-2023-29473
V4 · CVE-2023-26476
V5 · CVE-2023-26480
V6 · CVE-2023-29202
V7 · CVE-2023-29206
V8 · CVE-2023-29207
V9 · CVE-2023-29213
V10 · CVE-2023-29214
V11 · CVE-2023-29515
V12 · CVE-2023-29517
V13 · CVE-2023-29521
V14 · CVE-2023-34464
V15 · CVE-2023-34467
V16 · CVE-2023-35153
V17 · CVE-2023-35155
V18 · CVE-2023-35156
V19 · CVE-2023-35157
V20 · CVE-2023-35158
V21 · CVE-2023-35159
V22 · CVE-2023-35160
V23 · CVE-2023-35161
V24 · CVE-2023-35162
V25 · CVE-2023-36468
V26 · CVE-2023-37277
V27 · CVE-2023-37910
V28 · CVE-2023-37913
V29 · CVE-2023-40572
V30 · CVE-2023-45134
V31 · CVE-2023-45135
V32 · CVE-2023-45136
V33 · CVE-2023-45137
V34 · CVE-2023-46242
V35 · CVE-2023-46732
V36 · CVE-2023-48240
V37 · CVE-2023-48241
V38 · CVE-2023-50722
V39 · CVE-2024-31464
V40 · CVE-2024-31983
V41 · CVE-2024-31986
V42 · CVE-2024-31988
V43 · CVE-2024-37899
V44 · CVE-2024-37900
V45 · CVE-2024-41947
V46 · CVE-2024-43400
V47 · CVE-2024-55662
V48 · CVE-2024-56158
V49 · CVE-2025-23025
V50 · CVE-2025-29924
V51 · CVE-2025-29925
V52 · CVE-2025-29926
V53 · CVE-2025-32430
V54 · CVE-2025-46554
V55 · CVE-2025-49580
V56 · CVE-2025-54385
V57 · CVE-2025-66472
V58 · CVE-2026-24128
V59 · CVE-2026-40105
V60 · CVE-2026-48047

Final outcomes (red = B1 Critical, amber = B2 Major)

1 · AR1 · B1 · RCE from account
2 · AR1 · B1 · admin session -> programming-right actions
3 · AR1 · B1 · admin/programming session
4 · AR1 · B1 · arbitrary SQL as anon
5 · AR1 · B1 · arbitrary SQL from script right
6 · AR1 · B1 · attacker profile content runs with admin/programming right
7 · AR1 · B1 · attacker widgets run with programming right
8 · AR1 · B1 · content rendered with victim's programming right
9 · AR1 · B1 · doc created/rendered as victim with programming right
10 · AR1 · B1 · exec:server
11 · AR1 · B1 · exec:server / session:admin
12 · AR1 · B1 · exec:server with panel author PR
13 · AR1 · B1 · old privileged code executed via rev
14 · AR1 · B1 · programming-right execution
15 · AR1 · B1 · server file write on HTML export
16 · AR1 · B1 · session:admin -> grant programming
17 · AR1 · B1 · session:admin -> right:programming
18 · AR2 · B1 · read:server-files -> secret:DB credentials
19 · AR2 · B2 · password hash in history diff
20 · AR3 · B1 · admin session
21 · AR3 · B1 · script in admin session
22 · AR3 · B1 · script in victim (admin) session
23 · AR3 · B1 · session:<victim, incl. admin>
24 · AR3 · B1 · session:<victim, incl. admin> without cracking
25 · AR3 · B2 · create subwiki as owner/admin
26 · AR4 · B2 · leak:emails(all users)
27 · AR4 · B2 · move attachments off non-viewable pages (read + remove)
28 · AR4 · B2 · read closed-subwiki pages
29 · AR4 · B2 · read closed-subwiki pages as guest

Intermediate outcomes (position or capability a finding gives; the next finding starts here)

i1 · AR3 · right:space-admin (from AWM app creation) -> set own space panel columns
i2 · AR3 · session:programming-user via stored XSS on victim view (representative of all stored-XSS A; R10)
i3 · AR3 · victim's privileged session reached by attacker content
i4 · AR4 · plaintext password at rest (after victim reset) -> readable via arbitrary select
i5 · AR4 · ordinary user's session reached by attacker content
i6 · AR5 · leak:attachment names/metadata on non-viewable pages -> knows:source page+attachment
i7 · AR5 · leak:document references/content via Solr suggest -> knows:source page+attachment
i8 · AR5 · leak:names of restricted documents via livetable -> knows:source page+attachment
i9 · AR5 · leak:page references of private pages -> knows:source page+attachment
i10 · AR5 · leak:subwiki page names (anon REST listing)
i11 · AR5 · plaintext password at rest (after victim reset) -> leaked as credential, not hash

Enabling steps (not vulnerabilities)

C1 · config: Extension Repository Application installed
C2 · config: Macros.VFSTreeMacro installed
C3 · config: Oracle database backend
C4 · config: Realtime WYSIWYG editor installed and enabled (not default at 14.5)
C5 · config: admin adds Panels.IncludedDocuments to a panel column
C6 · config: entity name validation enabled
C7 · config: mail obfuscation enabled
C8 · config: malicious WebJar extension available in configured repository
C9 · config: multilingual wiki
C10 · config: office server connected
C11 · config: office server enabled
C12 · config: realtime WYSIWYG editor installed
C13 · config: subwiki with 'prevent unregistered users to view' enabled
C14 · config: vulnerable privileged revision retained in history (post-upgrade)
C15 · config: xobject holding a password deleted from a viewable page
C16 · config: xwiki-platform-wiki-rest-default extension installed
C17 · victim: admin (programming right, xwiki-platform-distribution/xwiki-platform-distribution-ui/xwiki-platform-distribution-ui-base/src/main/resources/XWiki/XWikiPreference…
C18 · victim: admin disables/enables attacker's account in user admin
C19 · victim: admin opens crafted ConfigurableClass link
C20 · victim: admin opens crafted create link
C21 · victim: admin opens crafted link
C22 · victim: admin opens crafted revision link
C23 · victim: admin opens create form for existing page
C24 · victim: admin opens create form on attacker template provider
C25 · victim: admin uploads a file with an attacker-suggested name
C26 · victim: admin views Scheduler.WebHome
C27 · victim: admin views attacker page with ClassEditSheet
C28 · victim: admin views attacker-edited page
C29 · victim: admin views rendered diff of attacker-edited page
C30 · victim: logged-in admin visits attacker page (REST CSRF)
C31 · victim: privileged user opens edit form of attacker's class property
C32 · victim: privileged user resolves an edit conflict on a page the attacker edited
C33 · victim: programming-right user follows create link and saves
C34 · victim: programming-right user opens attacker GET link (no CSRF token)
C35 · victim: programming-right user opens attacker link to create action
C36 · victim: programming-right user opens attacker link to edit action (CSRF, no token)
C37 · victim: programming-right user opens crafted GET link
C38 · victim: programming-right user renames a page the attacker's page links to
C39 · victim: script/programming-right user co-edits page in realtime and refreshes/saves
C40 · victim: user opens crafted delattachment URL and accepts CSRF warning
C41 · victim: user with programming/admin right views attacker-edited page

Rings are Access Rings (§3.4), drawn uncurled: outermost (AR5) at left, crown jewels (AR1) at right. Assignments follow the rule in §3.4; judgement calls are marked in the classification table. Exposure Ring unavailable (no deployment data).

Figures 2–16. Shared vulnerability arrows appear once per repository graph. Routes remain conditional candidates as described in §4.3.

6. Discussion

6.1 What selective hardening changes, and what it does not

Securing an exposed direct route changes what an attacker can reach most easily, but leaves the rest of the application's code to assess. The six changed forks retain route records through other weaknesses and ordinary conditions, some labelled B1 Critical. Gradio also shows that one fix can remove several proposed chains sharing a vulnerable step. The relevant result is therefore the path left after each named fix: which handoffs still connect to which protected outcome, and under what conditions?

The 1,000-to-ten example illustrates why a small backlog does not establish low risk; it is not a reduction measured here. Our intervention targeted ten finding IDs across five forks and an administrative credential condition in a sixth. It tests how named fixes affect recorded routes, rather than how often such chains occur across production software.

6.2 Implications for remediation and scanners

Remediation should follow outcomes and routes. A flaw with a bounded effect alone may become urgent if it supplies the identifier or authority needed by a critical path. A proposal labelled B1 Critical with an impossible handoff should be rejected. Teams can then see whether one fix interrupts several routes or whether independent routes to the same outcome need separate controls.

Two scanners with equal finding recall can leave different protected outcomes reachable through their misses. RoR's benchmark residual-risk measure makes that difference visible. The six modified OSS forks address a related question: which routes remain after documented fixes? Both views move evaluation from finding totals toward consequences.

6.3 Agent-assisted search

Agent-assisted search can explore cross-file and cross-feature paths that single-finding triage misses. Its value here is to propose consequential handoffs for source review, not to assume every proposed path works. Defence in depth means tracing and breaking those paths in code so that one failed control does not grant the next protected capability.

6.4 From a list of findings to a map of chokepoints

A list of vulnerabilities ranks each finding alone. It cannot show which findings hand an attacker the same position, or which single finding stands between that position and the crown jewels. The Access Ring maps add both. Each circle is a position that several findings can reach, and each arrow is a finding that moves the attacker from one position to the next. Counting the routes that share a circle or a finding turns the same inventory into a statement about where a fix removes the most paths. Three patterns appear across the fifteen maps.

Dispersed chains. Airflow and Budibase are ordinary applications whose routes do not share a centre. No circle carries more than two route records (Airflow: five records over eight findings, each circle on one record; Budibase: eleven records over fifteen findings, the busiest circle on two). Reaching AR1 takes a

separate chain each time: one route in Airflow and two, through different findings, in Budibase. Here the map confirms that no single fix removes the exposure, so findings are best treated one by one, and the ring view shows which of them completes a route into AR1.

A convergent chokepoint. Ghost has twenty-one route records, and twelve of them end in AR1 through one finding, CVE-2026-29053 (server-process execution). Twelve different findings lead into that finding through seven circles at AR3, all of which are administrator or owner positions. The busiest, i4 (an administrator session reached through a victim), is entered by six findings alone. A list presents these thirteen findings as separate items. The map shows that one of them guards the route to AR1 for the other twelve, so correcting the step between the AR3 circles and AR1 removes every AR1 route in the ledger, while correcting any one entry finding removes only its own route. A second, smaller chokepoint sits one ring out: CVE-2026-22594 carries the four routes that start from a cracked staff password.

Interlocking chains. Keycloak has forty-nine route records over twenty-two findings and twenty intermediate circles, and its chains cross. Three structures stand out. First, one finding, CVE-2025-13467, ends all thirteen AR1 routes. Second, the circle for a delegated administrator session (i13) carries twenty of the forty-nine records, entered by ten findings and left through two. Third, the standard-user circle (i17) is entered only through supporting steps (seven credentials, registration tokens and issued accounts), never through a code flaw, and eight findings leave it towards seven different outcomes. Because the routes leaving i17 end at AR2 or AR3, reaching AR1 from there depends on joining them with the realm-administrator routes, which the composed-path register treats as candidate joins still subject to token and role review. The practical reading is that the controls behind those seven steps (how initial access tokens are issued, who may register, how credentials are handled) deserve review to the same standard as the code, since a failure in any one of them opens several routes at once.

The maps therefore support three working rules: prioritise the finding that every route to a ring shares; review the supporting steps that feed a shared circle as carefully as the code behind it; and, where chains do not converge, fall back to ordering findings individually. All counts are of source-supported candidate routes in the ledger, and ring assignments follow the rule in Section 3.4. They locate where a defect would matter most; they do not show that any route has been exploited.

7. Limitations and next evidence step

This selected corpus cannot estimate chain prevalence across production software or establish how organisations prioritise review. Advisory/version matches and merged finding rows may be imperfect. The ledgers contain route variants, free-text outcomes and conditional assumptions; their B2 Major or B1 Critical labels need confirmation against exact resources and affected populations. We reviewed selected code handoffs and 54 exact-ID path joins, not every possible capability-based composition, and did not run the applications. A source citation locates evidence; it does not by itself establish a capability transfer.

The next evidence step is an adjudicated case set from the six changed forks. Each case should show the pinned revision, named fix, before/after code, remaining path, external conditions, execution environment and source-review judgement. Include both supported chains and failures such as Gradio's patched shared sink or a missing victim-session handoff. This resolves the current inventory into defensible outcome profiles without requiring live exploits.

8. Conclusion

Securing an application's outer entry points does not settle the risk deeper in its code. Weaknesses can compose across ordinary features, identity checks and privileged operations; independent controls at those handoffs are the code-level defence in depth that matters. Rings of Risk expresses this risk per protected outcome, showing consequence, entry conditions, distinct routes and the fixes needed to interrupt known paths. Across 15 OSS applications, selective hardening in six local forks leaves a substantial route inventory while also breaking chains that share patched code. The remaining source paths need outcome-level adjudication, but they already show why counting fixed findings alone cannot answer what remains reachable.

Internal evidence record (remove or replace for submission)

- Fork manifest: pinned revisions, selective fixes and resulting local heads.
- Static route audit and count notes: 15-repository table, 246-route inventory and source-reference gaps.
- Attack-vector chain review, full route register and 54-path exact-ID register: longer path families, shared finding reuse and rejected joins.
- Full chain map and non-code surface catalogue: conditional handoffs and environment distinctions.
- Ground-truth audit: 879 finding rows.
- All-15 CVSS comparison and six-fork fix-budget pilot: descriptive score inputs and the limits of current route-priority claims.

References

1. FIRST. CVSS v4.0 User Guide.
2. FIRST. EPSS FAQ.
3. CISA. Known Exploited Vulnerabilities Catalog and SSVC Guide.
4. OWASP. Risk Rating Methodology, Top 10 and ASVS.
5. NIST. SP 800-30 Rev. 1: Guide for Conducting Risk Assessments.
6. FAIR Institute. FAIR Standard v3.0.
7. Tenable. Vulnerability Priority Rating and Attack Path.
8. Rapid7. Active Risk strategy.
9. Qualys. Understanding TruRisk Score.
10. Microsoft. Security Exposure Management attack paths.
11. Ou et al. MulVAL: A Logic-based Network Security Analyzer. USENIX Security, 2005.
12. IRIS: LLM-Assisted Static Analysis for Detecting Security Vulnerabilities. 2024.
13. CVE-Bench: A Benchmark for AI Agents' Ability to Exploit Real-World Web Application Vulnerabilities. 2025.
14. AgentCyberRange: Benchmarking Frontier AI Systems in Realistic Cyber Ranges. 2026.
15. MITRE. Common Weakness Enumeration and ATT&CK.
16. CVE Program. Overview; NIST. NVD CVE FAQ.
17. Snyk. Create a fix strategy and reachable-vulnerability analysis.
18. Wiz. Cloud and AI Security Platform.
19. NIST. Cybersecurity Framework; OWASP. SAMM.
20. Shimizu and Hashimoto. Vulnerability Management Chaining. *IEEE Access*, 2026.
21. Jajodia, Noel and O'Berry. Topological Analysis of Network Attack Vulnerability. 2005.
22. Wang et al. k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities. *IEEE TDSC*, 2014.

23. Yamaguchi et al. Modeling and Discovering Vulnerabilities with Code Property Graphs. *IEEE S&P*, 2014.
24. Lee and Brumley. ExploitBench: A Capability Ladder Benchmark for LLM Cybersecurity Agents. 2026; Shi et al. VEX-Bench. 2026.
25. Praetorian. Exploit Chain Analysis. 2026; Cycode. Attack Chaining. 2026. Both are vendor accounts.