AI Selects, a Simple Client Acts:

Human-Approved Playbooks for AI-Driven Security Response

Onur AKTAS

S4E.io, Ankara, Türkiye

Draft, 15 October 2026

DRAFT 15 October 2026 Not peer reviewed

Abstract

Large language model agents can now find and exploit known vulnerabilities with little human help, while most organizations still need hours to days to remediate a weakness after it is detected. Automating remediation with AI closes this gap, but an AI agent with privileged access to production systems becomes a new attack surface. We present an architecture in which people approve security playbooks, an AI system only selects among them, and a simple client executes the approved operations within the approved scope. The AI cannot change commands, playbook content or the approval policy. We describe the design as implemented in Opservant and its open source client Spark, and separate what is implemented today from what is proposed. We relate three forms of human oversight (pre-approval, human-in-the-loop and human-on-the-loop) to the workflow mechanisms of the OASIS CACAO 2.0 playbook standard. We then propose an optional CACAO extension, the AI Selection Record, which records which playbook an AI selected, for which target, on what evidence, and with what approval and execution outcome. Finally, we define five autonomy levels, an open database of AI-selected security actions, and an evaluation methodology for detection and remediation times. No measured results are reported in this version.

1 Introduction

The interval between detecting a security weakness and remediating it is one of the most exploitable windows in cyber defense. Detection has been automated extensively; remediation still depends on a person reading an alert, choosing a fix, obtaining approval and applying it host by host. That process takes hours or days and scales with staff rather than with the number of assets.

AI changes the attacker's side of this equation. Fang et al. showed that an LLM agent based on GPT-4 could exploit 87% of a set of 15 real one-day vulnerabilities when given their CVE descriptions [5]. Agents of this kind can scan and exploit continuously, so every hour between detection and remediation becomes time an attacker can use.

The natural response is to let AI remediate as well. The difficulty is that an AI with authority to change production systems is itself a target: its inputs, such as scan results, logs and tickets, can be manipulated. Work on AI control studies exactly this setting, asking how to keep a system safe when the model inside it may be wrong or adversarial [4]. We apply that perspective to security remediation.

Our approach separates selection from execution. People approve playbooks in advance; an AI system selects among the approved playbooks; and a simple client executes approved operations within the approved scope. Safety properties live in reviewed playbooks and in deterministic checks in the client, not in the model's judgment.

This paper makes the following contributions:

1. An architecture that separates AI selection from execution, together with a precise definition of what an approval covers: the playbook version, the oversight mode, the target scope and the parameter bounds (Section 3).
2. An analysis of how pre-approval, human-in-the-loop and human-on-the-loop oversight can be expressed with the workflow mechanisms of OASIS CACAO 2.0 [1] (Section 4).
3. Five autonomy levels that combine these oversight forms (Section 5).
4. The AI Selection Record, an optional CACAO extension that records each AI selection and its outcome (Section 6).
5. A design for an open database of AI-selected security actions and an evaluation methodology for detection and remediation times (Sections 7 and 8).

Throughout, we distinguish what the Opservant implementation does today from what we propose. Section 2 gives background, Section 9 analyses security, Section 10 discusses limitations and future work, and Section 11 concludes.

DRAFT 15 October 2026 Not peer reviewed

2 Background and Related Work

2.1 Detection and remediation times

We use two standard measures. Mean time to detect (MTTD) is how long a weakness or intrusion exists before the defender knows about it. Mean time to respond or remediate (MTTR) is how long it takes, after detection, to contain or fix it. Section 8 refines both into measurable intervals.

2.2 Two sides of AI security

AI affects security in two directions at once, summarized in Table 1. Most work on AI for security addresses the first column. The second column is where AI control applies: if a system lets AI decide what runs on production machines, the AI's inputs become part of the attack surface.

Table 1: Two sides of AI security.

	Security against AI	**Security of AI**
Threat	Malicious AI agents attack faster than humans respond	The defender's own AI is manipulated into doing harm
Examples	Automated scanning, exploit chaining, fast lateral movement	Prompt injection through logs or tickets, poisoned scan results, a hijacked decision
Failure without control	The remediation gap stays open	A defensive tool becomes a privileged attack path
Approach in this paper	Remediate in minutes, around the clock	AI only selects among human-approved playbooks; the client executes approved operations within the approved scope

2.3 Security playbooks and OASIS CACAO

A course-of-action playbook describes the steps that prevent, mitigate or remediate a threat. OASIS CACAO 2.0 standardizes how such playbooks are described, shared and executed [1, 2]. A CACAO playbook is a versioned object with authorship and validity metadata, a workflow of

steps (action, conditional, parallel and playbook-action steps), commands of defined types, agents and targets, optional data markings, optional digital signatures, and an extension mechanism. Example playbooks are published by the Technical Committee [3].

2.4 AI control

Greenblatt et al. introduced AI control as the problem of keeping a system safe even when the AI model inside it may intentionally subvert safety measures, and evaluated protocols such as trusted monitoring and editing of untrusted model outputs [4]. We borrow its central assumption: the model may be wrong or compromised, and the surrounding system must remain safe anyway. Our setting differs in that the model's output space is restricted to a choice among pre-approved artifacts rather than free-form code.

3 System Design

3.1 Architecture

In Opservant [6], people approve playbooks before any AI can use them. The AI selects among those approved playbooks; a simple client on the host, Spark, executes the approved operations within the approved scope [7]. The AI cannot change commands, playbook content or the approval policy. Figure 1 shows the components and the control boundary between selection and execution.

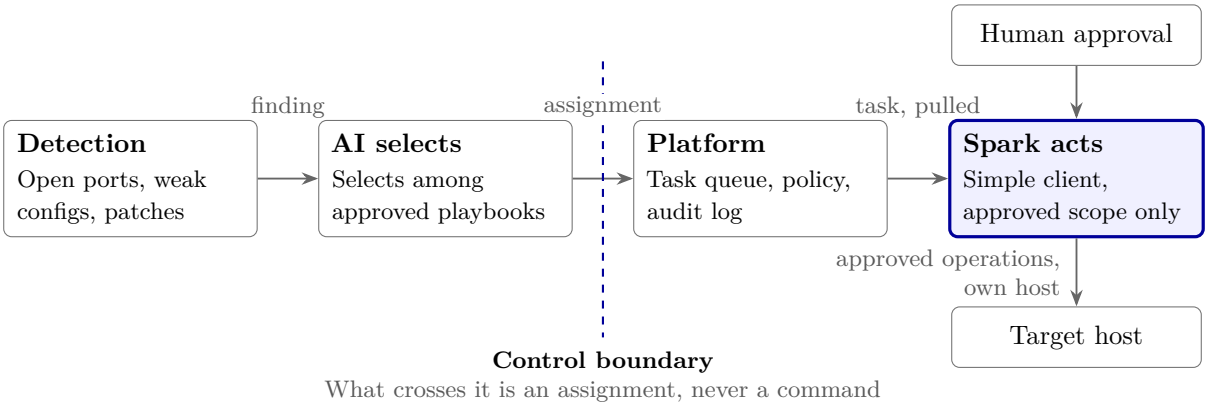


Figure 1: Opservant architecture. The AI's only output is an assignment of an approved playbook; Spark executes approved operations on its own host.

3.2 What an approval covers

Approving a playbook is more than approving its name. We define an approval as having four parts, each of which must be enforced where execution happens, not only where the selection is made:

1. **The playbook version:** the exact commands, rollback commands and timeouts that were reviewed.
2. **The oversight mode:** whether the action may run on its own or needs a person to decide at run time.
3. **The target scope:** which hosts or host groups the playbook may run on.
4. **The parameter bounds:** which variables, if any, may be set at run time, and which values they may take.

DRAFT 15 October 2026 Not peer reviewed

3.3 Client-side enforcement

Table 2 lists how the client protects each part of an approval, separating the current implementation from proposed extensions.

Table 2: Client-side protections in Spark.

Protection	How it works	Status
AI does not author content	The AI's output is an assignment of a playbook to a host; the playbook content comes from the reviewed library	Implemented
Target scope: own host only	Each Spark client polls only for assignments to its own host ID and runs commands only on the machine it is installed on; it has no remote execution	Implemented
No run-time parameters	The platform sends no run-time variables today, so commands run exactly as written in the playbook	Implemented
Run-time human decision	Actions marked `approval_required` pause for an interactive yes or no on the client; no answer means no	Implemented
Rollback and deadlines	Failed tasks revert completed actions in reverse order when `auto_revert_on_failure` and `revertable` are set; action, task and playbook deadlines apply	Implemented
Approved version integrity	The playbook format already carries hash fields, but the client does not verify them yet. Proposed: the client checks a signature or digest over the approved version and refuses anything else	Proposed
Target scope binding	The approval names the allowed hosts or groups, and the client refuses an assignment whose scope does not include its own host	Proposed
Parameter bounds	When run-time variables are introduced, each approved playbook declares them with a type and allowed values or ranges; the client rejects undeclared variables and out-of-bounds values	Proposed
Policy outside the AI's reach	The oversight mode and autonomy level are read from the approved playbook, never from the assignment	Proposed

Approval alone is not a guarantee: approvals can be rushed, and an approved version only protects the system if the client can tell it apart from a modified one. That is why the proposed integrity, scope and parameter checks sit in the client, the last component before a command runs. The model is assumed to be wrong some of the time; the safety properties live in reviewed playbooks and a simple, deterministic client, not in the model's judgment.

4 Human Oversight in CACAO

Spark's playbook model is inspired by CACAO 2.0 [1], but Spark playbooks are not yet CACAO documents. This section shows how CACAO's workflow model can express human oversight, and how Spark's fields map to CACAO concepts.

4.1 Three forms of human oversight

We distinguish three forms of oversight. CACAO 2.0 does not define fields with these names; Table 3 shows which of its mechanisms can express each one.

Spark also has a playbook-level `human_in_the_loop` field. Today the client enforces decisions per action through `approval_required`; the playbook-level field is checked by the validator, which warns when a critical playbook lacks it.

DRAFT 15 October 2026 Not peer reviewed

Table 3: Forms of human oversight and their expression in CACAO 2.0.

Oversight form	What it means	How it can be expressed in CACAO 2.0	Opservant today
Pre-approval of the playbook	People review and approve a playbook version before any automation may use it	The approval itself is an organizational process, but CACAO gives it something precise to approve: a versioned playbook with `created_by`, `modified`, `valid_from`, `valid_until`, `revoked`, and optional digital `signatures`	Playbooks come from a reviewed library; verifying the approved version on the client is proposed
Human-in-the-loop	A person must decide before a specific action runs	An `action` step whose command has type `manual`, which the standard defines as processed by a human or a system acting on behalf of one, assigned to an `individual` or `group` agent. The decision can be returned through `out_args` into a variable that an `if-condition` or `switch-condition` step evaluates before the automated step	Actions marked `approval_required` pause for an interactive yes or no on the client
Human-on-the-loop	Automation runs on its own; people monitor it and can intervene	No single construct. Monitoring happens outside the playbook; intervention paths can be modelled with `on_failure`, `workflow_exception` and revert steps	Execution logs and rollback on failure today; a remote stop or revert command is proposed

4.2 Mapping Spark playbooks to CACAO

Table 4 maps Spark's playbook fields to the closest CACAO 2.0 concepts.

CACAO describes playbooks: what they do, who authored them, and where people decide. It does not define a record of one particular use of a playbook: which system selected it, for which target, on what evidence, and what the approval and execution outcome was. For AI-driven response, that record is what accountability and measurement depend on, and it is what the extension in Section 6 adds.

A converter that exports Spark playbooks as valid CACAO 2.0 JSON, including human-in-the-loop steps, is planned and will be tested against the published examples [3].

5 Safe Autonomy Levels

We propose five autonomy levels (Table 5) that combine the three oversight forms of Section 4.1. Every level starts from pre-approval: the AI can only ever select a playbook that people approved. The level is part of that approval, together with the target scope and parameter bounds; the AI can never change it.

Four rules make the levels hold:

- **The level travels with the approval, not with the assignment.** The client reads it from the approved playbook, so a manipulated selection cannot raise it.
- **Scope and parameters are part of the approval.** A playbook approved for a group of hosts, with fixed or bounded parameters, is not approved for anything wider.
- **No rollback, no autonomy.** A playbook without a tested rollback cannot sit at A1, whatever its risk score.

DRAFT 15 October 2026 Not peer reviewed

Table 4: Spark playbook fields and the closest CACAO 2.0 concepts.

Spark playbook field	Closest CACAO 2.0 concept	Note
`slug`, `name`, `description`, `version`	Playbook `id`, `name`, `description`, `modified`	CACAO also requires `spec_version`, `created` and a STIX identity in `created_by`
`risk_level`, `risk_score`	Playbook `impact` (0 to 100)	Spark scores risk per action too
`tasks`, `actions`	Workflow of `action` steps linked by `on_completion`	Mechanical translation
`command`	Command data of type `bash` or `powershell`	Mechanical translation
`supported_os`	`agent_definitions` and `target_definitions`	Spark resolves the target at run time: the host it runs on
`timeout_seconds`	Step `timeout` (milliseconds)	Unit conversion
`approval_required`	A `manual` command step and a conditional step before the action	Human-in-the-loop pattern (Table 3)
`rollback_command`, `auto_revert_on_failure`	`on_failure` pointing to revert steps	Reversal as a workflow pattern
`hash`, `hash_algorithm`	Playbook `signatures`	Spark's fields exist but are not yet verified

- **Downgrade on doubt.** If the target or the evidence does not match the playbook's assumptions, the selection drops to the next stricter level.

Today, Spark enforces the pieces these levels are built from: per-action `approval_required` with an interactive decision on the client, `risk_level`, `revertable` with `rollback_command`, and automatic revert on failure. **Proposed** are the level names, the client reading the level from the approved version, scope and parameter binding, and the downgrade rule. The categories in the Opservant playbook library (information gathering, configuration, hardening, maintenance, reboot) map naturally onto A0 to A3 [8].

6 The AI Selection Record: A Proposed CACAO Extension

We propose an optional CACAO extension, working name *AI Selection Record*, that records which playbook an AI selected, for which target, on what evidence, and what the approval and execution outcome was. CACAO already lets authors express where people decide (Section 4.1); the extension does not replace or duplicate that. It adds a record of each individual use of a playbook by an AI system, which the standard does not define today.

It uses CACAO 2.0's existing extension mechanism (Sections 3.1 and 8 of [1]), so playbooks that reference it stay valid CACAO and consumers that ignore it can still run them.

6.1 Record format

One record is written per AI selection. It references the approved playbook version rather than copying it. Listing 1 shows an example.

The schema is not yet published. The `client_checks` block describes the proposed client-side checks, which Spark does not perform today.

DRAFT 15 October 2026 Not peer reviewed

Table 5: Proposed autonomy levels.

Level	Oversight	What happens after the AI selects	Typical playbooks
A0 Observe	Pre-approval, human-on-the-loop	Runs on its own; results are logged	Information gathering, read-only checks
A1 Fix, reversible	Pre-approval, human-on-the-loop	Runs on its own; people are notified and can revert	Low-risk configuration and hardening with a tested rollback
A2 Fix with a decision	Pre-approval, human-in-the-loop	Waits for a person to decide before it runs	Medium or high-risk hardening, service configuration
A3 Disruptive	Pre-approval, human-in-the-loop, scheduled	The AI may only propose; a person decides and schedules	Reboots, service restarts, anything that affects availability
A4 Not approved for AI	None	The AI cannot select it	Data deletion, credential changes, anything without a safe rollback

DRAFT 15 October 2026 Not peer reviewed

6.2 Extension definition

The extension definition follows Section 8 of the CACAO specification [1]: a `type` of `extension-definition`, a `name`, a `created_by` identity, a `schema` URL and a semantic `version`. Playbooks that may be selected by AI declare it in `extension_definitions`.

6.3 Rationale

The record serves four purposes:

- **Accountability.** Every action on a host traces back to an approved playbook version, a named selecting system, its evidence and, where required, a named person's decision.
- **Checkable against the approval.** Because the record names the approved version, the target and the parameters, anyone can check that what ran stayed inside what was approved.
- **Measurement.** The `timeline` block is exactly what MTTD, MTTR and time to contain need, recorded at the moment of action.
- **Sharing.** Records can carry CACAO's TLP data markings and digital signatures, so organizations can share them with a known handling policy and verify they were not altered.

We plan to publish the schema openly, implement the record in Opservant first, and bring it to the CACAO Technical Committee [2] for feedback through its public comment list.

7 Open Security Action Database

We propose a shared, public database of AI-selected security actions on real systems, built from the selection records of Section 6. Today, research on AI control and on AI in security runs mostly on sandboxes and simulations; almost no public data shows what AI decided on production infrastructure, what humans overruled, and what happened next.

What each record contains. The playbook and its autonomy level, the type of finding, the decision system, whether approval was required and given, the timeline, and the outcome

Listing 1: Example AI Selection Record. All identifiers, timestamps and the digest are illustrative.

```
{
  "type": "ai-selection-record",
  "id": "ai-selection-record--2b9e4c1d-6f7a-4e83-b5c2-9d0e1f2a3b4c",
  "extension_ref": "extension-definition--7f1c2a9e-0b4d-4c55-9a1e-3f6b8d2c1e01",
  "playbook_ref": "playbook--8c4e1a2b-3d5f-4a6b-9c7d-0e1f2a3b4c5d",
  "playbook_version": "2026-10-01T09:00:00Z",
  "playbook_digest": "sha256:<digest of the approved version>",
  "selected_by": { "system": "opservant-decision", "model": "<model id and version>" },
  "target": { "ref": "<salted hash of the asset id>", "in_approved_scope": true },
  "parameters": {},
  "evidence": [ { "type": "finding", "summary": "SSH root login enabled" } ],
  "rationale": "Finding matches the approved ssh-hardening playbook for this host group",
  "oversight": { "level": "A2", "mode": "human-in-the-loop" },
  "approval": { "decision": "approved", "by": "identity--a3c1f2d4-7e8b-4f90-8a1b-2c3d4e5f6071"
      , "at": "2026-10-14T08:05:02Z" },
  "timeline": {
    "detected_at": "2026-10-14T08:02:11Z",
    "selected_at": "2026-10-14T08:02:19Z",
    "started_at": "2026-10-14T08:05:09Z",
    "completed_at": "2026-10-14T08:05:15Z"
  },
  "execution": {
    "status": "success",
    "rolled_back": false,
    "client_checks": { "digest_match": true, "target_in_scope": true, "parameters_in_bounds":
        true }
  }
}
```

including any rollback. Free-text rationale is kept only after review for sensitive content.

What it never contains. Hostnames, IP addresses, usernames, customer names, raw command output or file contents. Targets appear only as salted hashes, and timestamps are coarsened before publication.

Table 6: Access tiers of the database.

Tier	Who	What they see
Public	Anyone	Aggregates: counts per playbook and level, approval and rollback rates, MTTR distributions
Research	Vetted researchers and CERTs, under a data use agreement	Individual anonymized records
Contributor	Organizations that share data	Their own full records, plus the research tier

Consent. Data enters only from organizations that opt in. Opt-out removes future records and, on request, past ones from non-aggregated tiers.

Dual use. A record of successful remediations is far less useful to an attacker than an attack dataset, but it can still reveal which weaknesses are common and how fast they are fixed. Records about active intrusions, rather than configuration weaknesses, stay in the research tier with a delay before any aggregate is published.

DRAFT 15 October 2026 Not peer reviewed

Open questions. Who governs the database long term (a neutral host such as a university, a CERT or a foundation is preferable to a vendor), and how contributors are credited.

8 Evaluation Methodology

We will measure speed and safety together, because a faster response that breaks production is not progress. All speed metrics come from the `timeline` and `approval` blocks of each selection record (Tables 7 and 8).

Table 7: Speed metrics.

Metric	Definition	Note
Time to detect	`detected_at` minus the time the weakness appeared	Often unknown; we report the scan interval as an upper bound
Selection latency	`selected_at` minus `detected_at`	Time the AI takes to choose a playbook
Decision latency	Approval `at` minus `selected_at`	Human-in-the-loop levels only; expected to dominate
Execution time	`completed_at` minus `started_at`	Includes rollback when it happens
Time to remediate (MTTR)	`completed_at` minus `detected_at`	The headline number
Time to contain	Time until the weakness is no longer exploitable	Equals MTTR for most configuration fixes; differs for intrusions

Table 8: Safety metrics.

Metric	Definition
Rollback rate	Share of runs that reverted, per playbook and level
Rejection rate	Share of A2 and A3 assignments a human declined
Wrong-assignment rate	Share of assignments reviewers later judge as the wrong playbook
Change-induced incidents	Outages or tickets traced to an AI-selected run

Method. We report medians and 90th percentiles, not means, because remediation times are heavily skewed. We compare three groups on the same playbooks: AI-selected at A0 and A1 (human-on-the-loop), AI-selected with a human decision at A2 and A3 (human-in-the-loop), and a manual baseline of the same fixes done by people before deployment. Splitting MTTR into its parts shows where the time actually goes; we expect human approval, not AI decision time, to be the largest part, which is itself a governance finding.

Status. No measured results yet. The first numbers will come from pilot deployments (Section 10.3).

9 Security Analysis

We assume the AI selection layer can be wrong or manipulated, and ask what an attacker can achieve through it. Human approval and the use of CACAO each help, but neither is a guarantee on its own; Table 9 shows which layer carries each risk and whether it exists today.

DRAFT 15 October 2026 Not peer reviewed

Table 9: Threats to the selection layer, mitigations and their status.

Threat	Example	Mitigation	Status	Residual risk
Prompt injection through scanned data	A service banner or log line tells the AI to pick a different playbook	The AI can only select among approved playbooks and cannot change commands, content or policy; human-in-the-loop at A2 and above	Selection from the library and per-action decisions implemented; levels proposed	A wrong but approved, reversible A0 or A1 playbook runs
Poisoned or spoofed findings	Fake findings trigger fixes that disable a needed service	A3 is propose-only; downgrade on doubt; scope limits per approval	Proposed	Availability impact from A1 fixes
Parameter manipulation	A crafted value is passed into a command at run time	Today the platform passes no run-time variables, so commands run as approved. Proposed: declared parameters with a type and allowed values or ranges, checked by the client	No run-time parameters today; bounds proposed	Bound definitions that are too loose
Modified playbook in transit or at rest	The approved playbook is altered between approval and execution	Proposed: the client verifies a signature or digest over the approved version and refuses anything else	Hash fields exist, not verified	**High until verification ships**
Compromised platform	An attacker pushes assignments to every client	Each client runs only on its own host; per-action decisions stay local. Proposed: version verification and scope binding on the client	Partly implemented	**High until verification ships**; the priority fix
Flawed approved playbook	A reviewed playbook has a hidden mistake	Review, versioning, tested rollback	Review and rollback implemented	A mistake every reviewer missed
Approval fatigue	People approve without reading	Track rejection rate; keep human-in-the-loop volume low; batch similar decisions	Proposed	Rubber-stamping at scale

The pattern is deliberate: each mitigation lives in a reviewed playbook or a simple, deterministic client check, not in the model's judgment. That is what lets the system tolerate a model that is wrong some of the time.

10 Limitations and Future Work

10.1 Limitations

This paper describes a design and a direction, not proven results. Its main limits:

- **No measured data yet.** Every speed and safety claim waits for pilot deployments.
- **Configuration weaknesses, not live intrusions.** Today's playbooks fix exposures such as open ports, weak settings and missing patches. Containing an active AI-driven intrusion needs runtime detection that Opservant does not yet have.
- **CACAO alignment, not conformance.** Spark playbooks are inspired by CACAO but are not yet CACAO documents; the converter is planned.
- **Client-side enforcement is incomplete.** Today the client limits execution to its own host, runs commands exactly as written and enforces per-action decisions. Verifying the

DRAFT 15 October 2026 Not peer reviewed

approved version, binding target scope and checking parameter bounds are proposed, not implemented. Until version verification ships, a compromised platform is the largest single risk.

10.2 Open questions

- Should autonomy levels be standardized across vendors, and if so, by CACAO or by a separate body?
- Which neutral organization should host the Open Security Action Database?
- What evidence would regulators and auditors need before accepting A1 autonomy for critical infrastructure?

10.3 Roadmap

Figure 2 shows the proposed roadmap. The one hard gate: no pilot starts until Spark verifies the approved version of every playbook it runs.

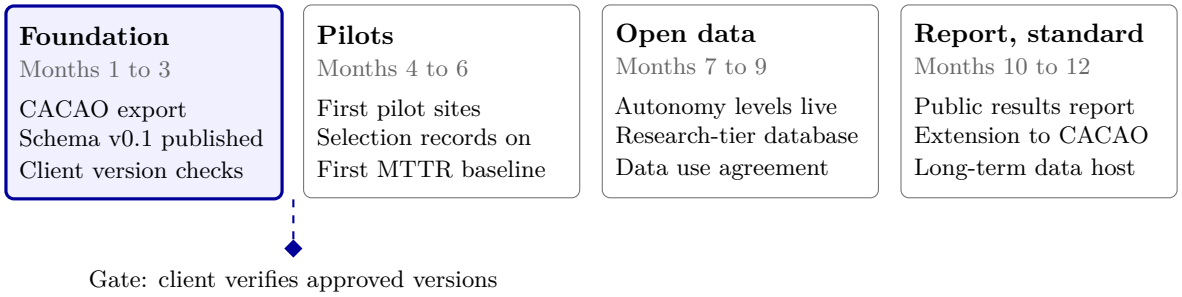


Figure 2: Proposed 12-month roadmap with one gate: the client verifies approved versions before any pilot starts.

11 Conclusion

AI-enabled attackers shorten the time defenders have to remediate, and AI-driven remediation is a natural response. We argued that it is safe only when selection is separated from execution: people approve playbooks, AI selects among them, and a simple client executes approved operations within the approved scope. We showed how pre-approval, human-in-the-loop and human-on-the-loop oversight map onto CACAO 2.0 workflow mechanisms, and proposed the AI Selection Record so that every AI selection can be traced, checked against its approval and measured. Neither human approval nor a standard format is sufficient on its own; the client-side checks of approved version, target scope and parameter bounds are the next step, followed by pilot deployments that produce the first measured results.

References

[1] B. Jordan and A. Thomson, editors. CACAO Security Playbooks Version 2.0. OASIS Committee Specification 01, 27 November 2023. https://docs.oasis-open.org/cacao/security-playbooks/v2.0/cs01/security-playbooks-v2.0-cs01.html

[2] OASIS. Collaborative Automated Course of Action Operations (CACAO) for Cyber Security Technical Committee. https://www.oasis-open.org/tc-cacao/

[3] OASIS CACAO TC. CACAO 2.0 examples. GitHub repository. https://github.com/oasis-tcs/cacao/tree/master/Examples

DRAFT 15 October 2026 Not peer reviewed

[4] R. Greenblatt, B. Shlegeris, K. Sachan, and F. Roger. AI Control: Improving Safety Despite Intentional Subversion. arXiv:2312.06942, 2023.

[5] R. Fang, R. Bindu, A. Gupta, and D. Kang. LLM Agents can Autonomously Exploit One-day Vulnerabilities. arXiv:2404.08144, 2024.

[6] Opservant. Project website. https://opservant.org

[7] Opservant Spark. GitHub repository. https://github.com/s4e-io/opservant-spark

[8] Opservant playbooks. GitHub repository. https://github.com/s4e-io/opservant-playbooks

DRAFT 15 October 2026 Not peer reviewed