Securify

THE COAST MODEL

Cybersecurity Operations Autonomy Scale for Tooling

THE INDUSTRY FRAMEWORK

Abstract

The cybersecurity industry lacks a shared vocabulary for describing and discussing the degree of autonomy exhibited by security tooling. As artificial intelligence and automation continue to transform how security work is performed across all disciplines, practitioners, vendors, and clients increasingly need a common framework to reason about what tools can do independently, where human oversight is required, and what operational constraints are appropriate in each context. This holds true regardless of the discipline, from offensive testing to incident response, threat hunting, and malware analysis.

This paper introduces COAST, the Cybersecurity Operations Autonomy Scale for Tooling. A framework consisting of three complementary classification components: six COAST Levels (CL0–CL5) defining the spectrum of autonomy in cybersecurity tooling from fully manual operation to full autonomous execution, supplementary Audit Levels (AL0–AL3) addressing logging and reconstructability requirements for autonomous systems, and supplementary Data Handling Levels (DH0–DH2) addressing data processing boundaries. Together, these classifications provide a shared language for tool classification, rules of engagement definition, procurement decisions, and regulatory discussion across all cybersecurity disciplines.

Securify

Content

Introduction

01

1.1 The problem

When a security operations team deploys an AI-driven alert triage system, how much of the decision-making is the system doing independently? When a vendor markets a product as "autonomous threat detection" or "AI-powered security testing," what does that actually mean in practice? When rules of engagement specify that "automated tools are permitted," what degree of autonomous action does that actually authorise? When a regulator asks whether a security operation was conducted with "appropriate human oversight," what standard is being applied?

These questions arise across every cybersecurity discipline, offensive testing, incident response, threat hunting, malware analysis, vulnerability management, and security operations. Today there is no agreed framework to answer them. The result is real and compounding: clients cannot make informed decisions about what they are authorising. Practitioners cannot precisely communicate their methodology. Vendors cannot objectively position their products. Regulators cannot write enforceable requirements. And as security tooling becomes increasingly autonomous, from simple scanners to AI-driven agents capable of independent multi-phase operation, the consequences of this imprecision grow more serious.

1.2 The Opportunity

Other industries have solved analogous problems. The Society of Automotive Engineers (SAE) defined a six-level autonomy scale for vehicles[1] that has become a universal reference for manufacturers, regulators, insurers, and consumers alike. The SAE model works because it focuses on a single, well-defined axis, the degree of human involvement in the driving task. It defines clear, observable boundaries between levels.

The cybersecurity industry is requires an equivalent model. The underlying axis, the degree of human involvement and control in security operations, is just as well-defined. What has been missing is the model itself.

[1] SAE International. SAE J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Warrendale, PA: SAE International.

1.3 Why Cybersecurity is Different

The SAE analogy is instructive, but imperfect. Vehicles always perform the same fundamental task, transportation. Cybersecurity professionals do not perform the same task over and over. A Red Team operator pursuing lateral movement, a malware analyst reverse engineering a binary, a threat hunter investigating an anomaly, a SOC analyst triaging alerts, and an OSINT analyst profiling a subject are performing fundamentally different types of work with different notions of scope, risk, and goal.

COAST addresses this by focusing on the control axis, who decides what to do next and what triggers human involvement, rather than the specific task being performed. This makes the model applicable across cybersecurity disciplines while remaining precise enough to be operationally useful.

1.4 About This Framework

COAST consists of three complementary classification components designed to address different but related dimensions of autonomous tool operation.

COAST Levels (CL0–CL5) form the core of the framework. They define six levels of autonomy based on observable differences in who decides what to do next and what triggers human involvement. Every tool, system, or engagement can be assigned a COAST level. The levels span from fully manual operation at CL0, where the human initiates every action, to full autonomous execution at CL5, where the system pursues a defined goal without any human approval during execution.

Audit Levels (AL0–AL3) are a supplementary classification for systems operating at CL3 and above. At these levels, autonomous reasoning should be generated by the system independently of any human decision or pre-authored logic. Without explicit logging, this reasoning is simply lost. Audit levels define the degree to which that reasoning and the actions it produced should be logged and reconstructable after the fact.

Data Handling Levels (DH0–DH2) are a supplementary classification applicable at any COAST level where the sensitivity of engagement data is a concern. They define the boundaries within which data may be processed during tool operation, independently of the tool's autonomy level. A CL1 tool can send data to an external service just as a CL5 system might. Data handling requirements must be defined separately from autonomy level and are relevant at any COAST level.

Where data sensitivity is a concern at CL0–CL2, the classification is expressed as CL[x], DH[x], for example, CL1, DH1. For systems operating at CL3 and above where data sensitivity is also relevant, the complete classification is expressed as CL[x], AL[x], DH[x], for example, CL4, AL2, DH1.

COAST is designed to complement, not replace, existing cybersecurity frameworks and methodologies. Frameworks across the industry address what techniques are used, how work is structured, what vulnerabilities to look for, or how incidents should be handled. COAST addresses a distinct and orthogonal dimension: the degree of autonomy at which tools and systems operate. It can be applied alongside any methodology, standard, or discipline to characterise and communicate the level of autonomous operation involved in a given tool or engagement.

1.5 What COAST Does Not Measure

COAST measures a single, well-defined property: the degree of human involvement in tool operation. It deliberately does not measure result quality, coverage, or accuracy; the inherent risk or danger of an operation; operator or practitioner skill; or legal and regulatory compliance. A higher COAST level does not imply a better, safer, or more competently executed outcome. It only describes who was in control of what happened next. Section 7 discusses this in full, alongside the trust-based nature of COAST classification.

1.6 Versioning

COAST uses a versioned classification model. The current version is COAST 1.0.

When citing classifications in engagement documentation, reports, or rules of engagement, the COAST version should be included to ensure precision over time. The recommended notation is:
COAST [version]: CL[x], AL[x], DH[x]
For example: COAST 1.0: CL4, AL2, DH1
At CL0–CL2 where no AL classification applies: COAST 1.0: CL2, DH1

When new versions of COAST are published, existing classifications remain valid under their stated version. Classifications made under an earlier version are not automatically invalidated, but practitioners should review whether updated definitions affect the accuracy of prior classifications when applying them to new engagements. Changes to level definitions that would materially affect existing classifications will be clearly documented in the version history accompanying each new release.

1.7 What COAST Does Not Measure

Section 1 introduces the problem, explains the three-part classification system, and briefly notes what COAST does and does not measure.
Section 2 provides the glossary of terms used consistently throughout the framework.
Section 3 explains the distinction between capability level and operational level. This is essential for procurement, rules of engagement, and compliance conversations.
Section 4 defines all six COAST levels in full detail.
Sections 5–6 cover the supplementary Audit Level and Data Handling Level classifications.
Section 7 expands on what COAST does not measure and addresses the trust-based nature of COAST classification and its current limitations.
Section 8 provides the classification process and important classification notes.
Section 9 covers operational best practices for deployment, guardrails, rules of engagement, and data handling.
Appendix A provides quick reference tables for all three classification systems.

Sections 2–4 should be considered essential reading for all audiences. They define the terms, the capability/operational level distinction, and the core classification levels that all subsequent sections build upon. After reading those:

Practitioners classifying a tool: proceed to Section 8.

Clients defining rules of engagement: focus additionally on Sections 5, 6, 7, and 9.

Vendors classifying products: focus additionally on Sections 5, 6, and 7.

Glossary

02

The following terms are used consistently throughout the COAST framework. Shared definitions ensure that practitioners, vendors, and clients apply the model consistently across different contexts and disciplines.

Atomic Operation: A single, indivisible operation that completes without making decisions based on intermediate results. An atomic operation takes input, performs one predetermined action, returns output, and stops. It has no awareness of prior or subsequent actions.

Audit Level (AL): A supplementary classification that describes the logging and reconstructability of autonomous system behaviour. Audit levels apply only to tools and systems operating at CL3 and above, where the system generates autonomous reasoning that exists independently of any human decision or pre-authored logic. See Section 6 for full definitions.

Capability Level: The COAST level reflecting the maximum autonomous capability of a tool when fully utilised. The capability level is an intrinsic property of the tool and does not change based on how it is deployed in a specific engagement. Vendors and tool authors classify products at their capability level.

Data Handling Level (DH): A supplementary classification that describes the boundaries within which engagement data may be processed during tool operation. Data handling levels apply to all COAST levels where data sensitivity is a concern and must be defined independently of the operational level. See Section 7 for full definitions.

Deployment: The specific configuration and operational use of a tool or system within an engagement. A deployment is characterised by its operational level, audit level, and data handling level, which may differ from the tool's maximum capability level.

Engagement: A formally scoped and authorised security activity conducted by a practitioner or team on behalf of a client or organisation. An engagement defines the scope, objectives, rules of engagement, permitted tooling, and duration of the work. Engagements may span one or more operational phases and may involve multiple tools deployed at different operational levels.

Goal: The defined outcome that an operator or autonomous system is working toward. A goal may be outcome-oriented, achieving a specific result, or investigative, characterising or understanding a specific condition. The goal determines what constitutes meaningful progress and what constitutes an unrecoverable failure state requiring operator involvement.

Guardrails: Instructions or constraints provided to an autonomous system at CL3 or above that define what constitutes a high-impact action requiring human approval before execution. Guardrails are configured by the human operator prior to execution and govern the system's approval-seeking behaviour during operation. The responsibility for defining appropriate guardrails lies with the operator.

High-Impact Action: An action whose execution could reasonably result in a severe or unintended adverse effect on the confidentiality, integrity, or availability of systems, data, or services within or beyond the defined scope. High-impact actions include but are not limited to actions that are irreversible or difficult to reverse, involve access to highly sensitive or privileged systems or data, carry risk of consequence beyond the immediate subject, or have significant legal or operational implications in the current context.

Knowledge Base: A shared data store that accumulates results across tool operations and phases during execution, enabling subsequent operations to build upon prior findings. The knowledge base is the primary mechanism through which tools and systems at CL2 and above maintain state and pass information between operational phases.

Operational Level: The COAST level at which a tool is deployed in a specific context. The operational level may be equal to or lower than the capability level, reflecting deliberate decisions about rules of engagement, client risk appetite, or regulatory requirements. Operational level can be defined per engagement, per operational phase, or per subject scope and it is not required to be uniform across an entire engagement.

Operational Phase: A distinct stage of work in which the nature of the operation being performed remains consistent. A phase is defined by what type of action is being taken, not by which specific tools implement it. The definition of phases varies by discipline. For example, in offensive security phases may include discovery, enumeration, exploitation, and post-exploitation; in DFIR they may include collection, processing, analysis, and reporting; in malware analysis they may include static analysis, dynamic analysis, and behavioural characterisation.

Operator: The person or team actively running and overseeing a tool or system during a deployment. The operator is responsible for configuring the system within its operational level, defining guardrails, responding to approval requests at CL3 and CL4, and ensuring compliance with rules of engagement. The operator is distinct from the vendor who developed the tool and the client who commissioned the engagement.

Phase Boundary: The point at which automation transitions from one operational phase to another, triggered by the results of the preceding phase. Crossing a phase boundary means the nature of the operation changes qualitatively. The ability to autonomously cross phase boundaries is the distinguishing characteristic between CL1 and CL2.

Scope: The defined boundary within which autonomous action is authorised. Scope defines the subjects, systems, data sources, or assets that a tool or operator is permitted to interact with during an engagement. In the COAST framework, scope is always defined by a human operator. At CL3 and above, scope enforcement must be implemented as a hard architectural constraint within the system, autonomous reasoning alone should never be relied upon to determine or respect scope boundaries.

Capability Level vs. Operational Level

03

A fundamental distinction in the COAST framework is between a tool's **capability level** and its **operational level**.

A tool's capability level is fixed. It reflects the maximum degree of autonomy the tool can exercise when fully utilised. This is an intrinsic property of the tool that does not change based on context or configuration.

A tool's operational level reflects how it is deployed in a specific context. A CL5-capable tool may be deliberately operated at CL4, with human approval gates enabled for high-impact actions, because a client's rules of engagement require it, because regulation demands it, or because the operator's own risk assessment calls for additional oversight. The tool's capability has not changed; only the operational constraints applied to it.

This distinction has direct practical consequences:

- **Vendors** classify their products at capability level. This reflects what the product can do and enables informed procurement decisions.
- **Practitioners** classify engagements at operational level. This reflects how the tool is being used and supports accurate methodology documentation.
- **Clients** set a permitted operational level ceiling. This defines what degree of autonomous action they are authorising within the rules of engagement.

All three classifications are valid and serve different purposes. A complete engagement record should note both the capability level of tools used and the operational level at which they were deployed.

3.1 Operational Level is Not Uniform Across an Engagement

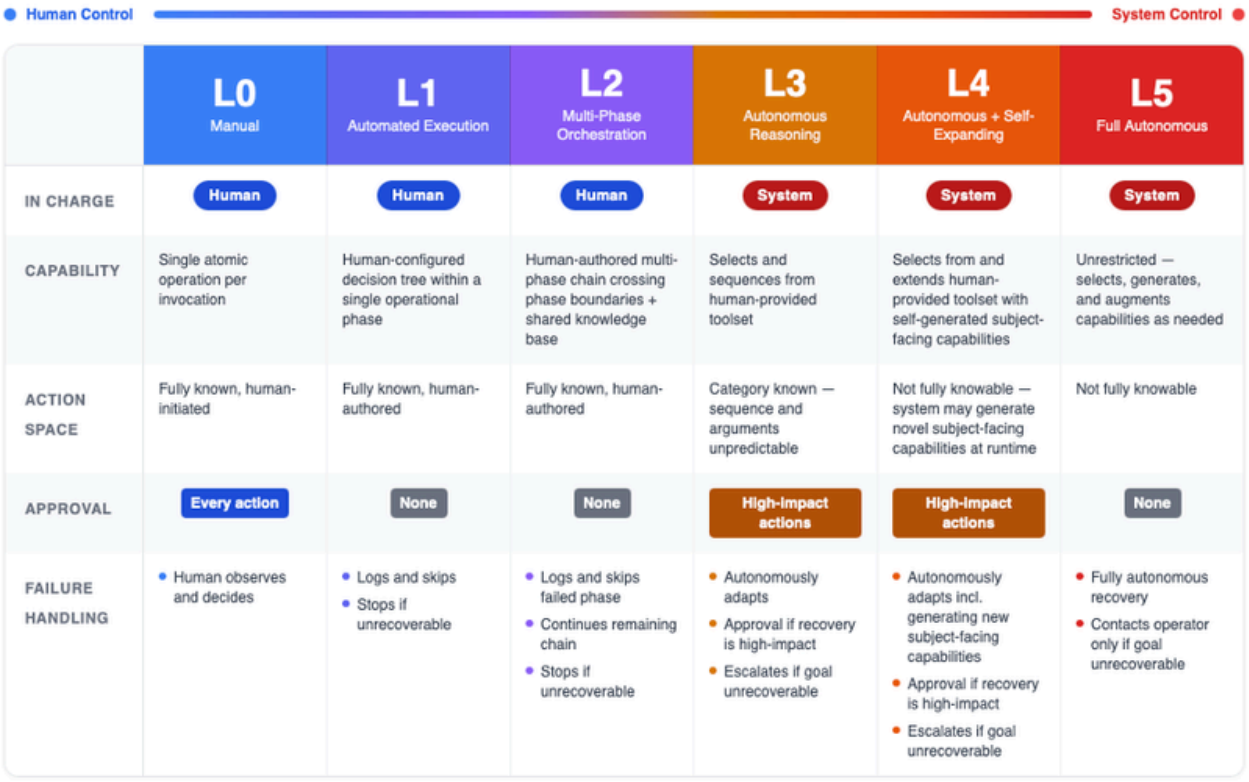
Operational level does not need to be a single setting applied uniformly across an entire engagement. Clients and practitioners should consider defining permitted operational levels per phase or per subject scope, as different phases carry meaningfully different risk profiles.

Higher autonomous operation may be appropriate during lower-risk phases such as passive information collection, while strict human control may be required during higher-risk phases such as active system interaction, privileged access operations, or sensitive data handling. This per-phase approach provides more precise control over risk and oversight than a single engagement-wide operational level, and is the recommended approach for engagements spanning multiple distinct operational phases.

COAST Levels

04

The six COAST levels are defined below. Each level is described across four dimensions: who holds decision-making authority and why, what the human operator is responsible for, how the system or tool behaves at runtime, and how failures are handled. The definitions are technology-agnostic, they describe observable behaviour, not implementation choices. Examples at the end of each level are illustrative rather than exhaustive; they are drawn from multiple disciplines to demonstrate applicability. For guidance on applying these definitions to classify a specific tool, see Section 8. A condensed overview is provided with the following image:



4.1 Level 0 - Manual Operation

Who is in charge and why?

The human entirely. No tool takes any action beyond its single invoked operation. The human is the sole connection between all actions.

Human responsibility

The human explicitly initiates every action and manually interprets all output. Information is carried between tools by the human. There is no automatic passing of results from one action to the next. Every subsequent action is a conscious, deliberate decision.

Runtime behaviour

Executing a single, atomic, predetermined operation per invocation and returning output. The tool makes no decisions based on intermediate results and performs no follow-up actions.

Failure handling

The tool surfaces the error in its output and stops. No recovery is attempted. The human observes the failure and decides whether to retry, adjust the approach, or abandon the action.

Examples

- Manually submitting input to a subject system interface and directly observing the response (e.g. entering a test input into a form field, sending a crafted request with curl, manually querying a log source).
- Running a single targeted query against a subject to identify one specific characteristic (e.g. a single port scan against one host, a single manual memory analysis command).
- Using a basic utility to process a discrete, already-collected piece of data (e.g. grepping through a log file, reading a captured response, manually examining a binary with a hex editor).
- Directly interacting with a discovered service or resource using a standard client (e.g. navigating to an address in a browser, connecting to an open port, manually querying a discovered API endpoint).

4.2 Level 1 - Automated Execution within Fixed Scope

Who is in charge and why?

The human. Although the tool operates automatically during execution, every action it takes was explicitly pre-configured by the human. The tool cannot deviate from or expand upon its configured instructions.

Human responsibility

The human defines the scope and configures all tool behaviour before execution begins. This includes any branching logic the tool will follow. The human is effectively authoring the decision tree the tool will execute. Once the tool is running, the human monitors output but is not required to intervene. The human remains the sole decision maker between separate tools and actions.

Runtime behaviour

Executing a human-authored decision tree deterministically within a single operational phase. When intermediate results reveal new in-scope items, the same configured operation is automatically applied to them. The tool automatically uses intermediate results to continue its configured operation, but makes no independent decisions about what to do with those results beyond what was explicitly configured. The nature of the operation does not change based on what is discovered.

Failure handling
The tool logs the failure, skips the failed action, and continues with its remaining configured operations where possible. If the failure is unrecoverable within its configuration, the tool stops and surfaces the error to the human. No autonomous recovery or adaptation is attempted.

Examples

- Automatically applying a fixed methodology or input set against all discovered items within a defined boundary, using intermediate results only to continue the same configured operation (e.g. recursive enumeration using a fixed wordlist, credential testing against discovered authentication endpoints, spidering within a defined domain).
- Running a pre-configured set of checks against a defined subject, with check selection automatically adapted to what is discovered but remaining within the pre-configured check categories (e.g. a scanner running service-appropriate checks based on discovered service types, a signature tool applying relevant rules to discovered file types).
- Automatically applying a fixed ruleset, signature set, or query set against a defined collection of artifacts or data, flagging results for human review (e.g. static analysis rules against a set of binaries, detection queries against a bounded log dataset, pattern-matching rules against a collected sample set).

4.3 Level 2 - Automated Multi-Phase Orchestration

Who is in charge and why?
The human. Despite the depth and complexity of automation, every orchestration decision, every tool chain, and every data flow was explicitly programmed by the human prior to execution. The tool cannot take any action its programmer did not explicitly account for.

Human responsibility
The human defines the scope and authors the full orchestration logic before execution begins. This includes which tools are chained, how results flow between them, which operational phases are automated, and what actions are taken on newly discovered in-scope subjects. During execution the human monitors progress but is not required to intervene.

Runtime behaviour
Executing a human-authored chain of distinct operations in sequence, where the findings from each step automatically feed into a different type of operation next. Results accumulate in a shared knowledge base accessible to all stages of the chain. Newly discovered in-scope subjects are automatically subjected to the full chain. The logic governing each transition was entirely pre-authored by the human.

Failure handling

The tool logs the failure, skips the failed action or subject, and continues executing remaining chain operations. If a failure affects a critical dependency in the chain, the affected downstream phases are skipped for that subject. No autonomous recovery or adaptation of the chain logic is attempted. Failures are surfaced to the human in output or logs for review.

Examples

- Automatically orchestrating discovery, characterisation, and known issue identification across all subjects within a defined scope, with results feeding a shared knowledge base (e.g. automated network scanning feeding into service fingerprinting feeding into vulnerability identification; automated alert triage feeding into enrichment feeding into case creation).
- Automatically applying discovered information across all in-scope subjects, with newly confirmed findings added to the knowledge base and applied against remaining subjects (e.g. discovered credentials feeding automated validation across in-scope systems; indicators of compromise feeding automated searches across defined log sources).
- Automatically mapping all reachable items within a defined boundary, applying a pre-configured set of checks to each, and feeding confirmed findings into a structured results store (e.g. web application crawling feeding into automated vulnerability checks; memory analysis feeding into automated indicator extraction).
- Automatically executing a defined response workflow across collected artifacts. Such as processing evidence, correlating findings with reference data, and producing structured output for subsequent analysis phases (e.g. an automated pipeline that extracts indicators from collected memory images, correlates against threat intelligence, and creates structured investigation cases).

4.4 Level 3 - Autonomous Reasoning within Provided Toolset

Who is in charge and why?

The system, for the majority of actions. The system autonomously reasons about how to pursue the goal and decides what to do with what it finds. The human retains control over high-impact actions through approval gates, and retains control over the tools and capabilities initially made available to the system.

Human responsibility

The human defines the goal and scope, and the set of tools and capabilities made available to the system. The human provides guardrails that define what constitutes a high-impact action requiring approval. During execution the human responds to approval requests, verifying the system's reasoning and findings before making a decision.

Runtime behaviour

Autonomously reasoning about how to pursue the defined goal using the available toolset. The system decides what tools to invoke, in what order, and with what parameters based on its findings. It can generate code or scripts that operate internally, processing data, correlating findings, or parameterising provided tools. It does not generate capabilities that reach outside the system to interact with subjects in the target environment. It operates across phase boundaries independently, without requiring pre-authored orchestration logic. Results continuously inform subsequent reasoning and action selection. Actions classified as high-impact are paused and submitted for human approval before execution, while other aspects of the goal continue in parallel.

Failure handling

The system autonomously attempts to recover from or work around recoverable failures, adapting its approach without human involvement. If a recovery action would itself be classified as high-impact, the system pauses that action and requests human approval while continuing to pursue other aspects of the goal that do not require approval. Unrecoverable failures that block all progress toward the goal are escalated to the human.

Examples

- Autonomously invoking available tools with generated parameters based on discovered information, and processing the output with internally generated logic to determine next steps (e.g. generating a tailored nmap command based on discovered service characteristics, processing its output with a correlation script to identify patterns, then selecting the appropriate next tool from the available toolset).
- Autonomously working through a multi-phase assessment, selecting and invoking available tools at each step based on what was discovered, pausing for human approval before executing a flagged high-impact action and continuing other work in parallel while waiting (e.g. an agent that discovers an authentication endpoint, selects a credential testing tool from its registry, analyses the response, flags the next step for approval, and continues mapping other endpoints in parallel).
- Autonomously generating internal queries and processing logic tailored to discovered data structures, where all generated code operates within the system boundary (e.g. constructing structured database queries from discovered schema information, generating API requests from enumerated endpoint parameters, writing parsing scripts for discovered data formats).
- Autonomously processing collected artifacts, generating internal correlation logic to identify relationships across multiple data sources, and selecting the appropriate next investigative tool based on emerging patterns (e.g. an investigation agent correlating process, network, and file system activity to identify anomalous behaviour, then selecting an appropriate forensic tool to examine the most significant finding).

4.5 Level 4 - Autonomous Reasoning with Self-Expanding Capability

Who is in charge and why?

The system, for the majority of actions. The system autonomously reasons about how to pursue the goal and can generate capabilities, such as tools, scripts, probes, or code, that actively reach outside the system and interact with subjects in the target environment when its provided toolset is insufficient. The human retains control over high-impact actions through approval gates.

Human responsibility

The human defines the goal and scope, and an initial set of tools and capabilities made available to the system. The human provides guardrails that define what constitutes a high-impact action requiring approval. During execution the human responds to approval requests, verifying the system's reasoning and findings before making a decision.

Runtime behaviour

Autonomously reasoning about how to pursue the defined goal, selecting from and building upon the provided toolset. When existing capabilities cannot adequately interact with a discovered condition in the target environment, the system generates new capabilities, tools, scripts, probes, or code, that reach outside the system and actively interact with subjects. The system decides what to run, in what order, with what arguments, and can create new outward-facing capabilities based on its evolving understanding of the subject. Results continuously inform subsequent reasoning, action selection, and capability development. Actions classified as high-impact are paused and submitted for human approval before execution, while other aspects of the goal continue in parallel.

Failure handling

The system autonomously attempts to recover from or work around recoverable failures, including generating new outward-facing capabilities to overcome obstacles its provided toolset cannot address. If a recovery action would be classified as high-impact, the system pauses that specific action and requests human approval while continuing to pursue other aspects of the goal. Unrecoverable failures that block all progress toward the goal are escalated to the human.

Examples

- Autonomously identifying that no provided tool can interact with a discovered condition and generating a new subject-facing capability to address it (e.g. discovering a service running an unrecognised proprietary protocol and generating a custom probe, executing it against the subject, and processing the response to determine next steps).

- Autonomously augmenting provided tools or techniques with generated extensions that create new outward-facing interaction capabilities (e.g. generating a custom plugin that enables an existing tool to interact with a previously unhandled protocol or service type).
- Autonomously identifying an undocumented API endpoint and generating a custom interaction script tailored to the specific parameter patterns, authentication mechanisms, and response characteristics observed, executing it against the subject to map the endpoint's behaviour (e.g. generating a protocol-specific interaction script for a non-standard interface discovered during assessment).
- Autonomously developing novel external interaction approaches when provided capabilities cannot adequately characterise a discovered condition (e.g. generating a custom network probe for a novel service type, or a targeted analysis script for a proprietary binary format).
- Autonomously generating and validating new subject-facing capabilities before deployment, pausing for human approval where they would constitute a high-impact action (e.g. testing a generated probe in a controlled manner before executing it against the subject).

4.6 Level 5 - Full Autonomous Operation

Who is in charge and why?
The system entirely. The system autonomously pursues the defined goal without requiring human approval for any action, including those that would be classified as high-impact at lower levels. The human retains no active role during execution.

Human responsibility
The human defines the goal and scope before execution begins. No further involvement is required or expected during execution.

Runtime behaviour
Autonomously reasoning about and pursuing the defined goal using all available and self-generated capabilities. The system decides what to run, in what order, with what arguments, and generates or augments capabilities, including those that reach outside the system and interact with subjects as needed. All decisions, including those with high-impact or irreversible consequences, are made autonomously.

Failure handling
The system autonomously handles all failures, including high-impact recovery actions, without human involvement. This includes generating new capabilities to overcome obstacles where necessary. The system only contacts the operator when the goal has become completely unrecoverable and forward progress is impossible regardless of approach.

Examples

- Autonomously pursuing a complete objective from initial discovery through all required phases to completion, generating novel capabilities as needed, making all decisions including high-impact ones without human approval, and delivering results only when the goal is achieved or all approaches are exhausted (e.g. given a defined objective and scope, a system that autonomously discovers, assesses, and documents findings across the full engagement lifecycle without any human interaction).
- Autonomously identifying, developing, and applying novel approaches when existing capabilities are insufficient, including generating new outward-facing tools or techniques mid-execution (e.g. a system that encounters an unknown condition and independently develops and validates a new capability to address it).
- Autonomously making and executing decisions about high-impact or irreversible actions when they are necessary to progress toward the goal (e.g. a system that determines a highly invasive or consequential action is necessary and proceeds without approval).
- Autonomously recovering from failures by pivoting strategy, regenerating capabilities, or finding alternative paths, only surfacing to the operator when all approaches are exhausted (e.g. a system that silently retries, adapts, and reroutes before reporting an unrecoverable state).

4.7 Why COAST Stops at Level 5

COAST defines six levels, ending at CL5, full autonomous operation within a defined scope. A natural question is whether a further level is warranted: a fully autonomous system operating without any predefined scope, pursuing only a high-level goal.
The decision not to include this level is deliberate. Removing scope from an autonomous system does not change the degree of autonomy, it changes the blast radius. A scopeless autonomous system is not more autonomous than a CL5 system; it is simply less constrained in where it can direct that autonomy. The distinction between CL5 and a hypothetical CL6 is therefore not an autonomy distinction, it is a liability, legal, and ethical distinction that falls outside the domain this model is designed to address.

This is not a purely hypothetical scenario. Some current agentic frameworks already support open-ended, goal-only operation without a defined scope. Such a deployment is not a higher COAST level. It is a CL5 system operated without one of the two inputs the human is required to define at every COAST level: scope. A goal without a scope is not evidence of a capability beyond CL5. The absence of scope changes what the system might affect, not how autonomously it decides what to do next.

Audit Levels

05

For tools and systems operating at CL3 and above, autonomous reasoning is generated by the system itself and exists independently of any human decision or pre-authored logic. Without explicit logging, this reasoning is simply lost. This makes post-engagement review, accountability, and incident reconstruction significantly more difficult or impossible. Audit levels apply exclusively at CL3 and above; below CL3, standard professional documentation constitutes a sufficient audit trail.

A system at CL3 or above should include an audit level in its full classification: **COAST [version]: CL[x], AL[x], DH[x].** Log format, retention period, and accessibility are governed by applicable security standards, regulatory requirements, and law, not defined by COAST. See Section 7 for the complementary Data Handling Level classification.

5.1 AL0 - No Logging

No record of system actions, decisions, or reasoning is captured during execution. Post-engagement reconstruction of what the system did and why is not possible. This audit level represents a significant operational risk at any autonomous COAST level and should be avoided in professional engagements.

5.2 AL1 - Execution Logging

A record of what actions the system took is captured, but not the reasoning that led to those actions. It is possible to reconstruct the sequence of operations and their outputs, but not why the system chose each action. Sufficient for basic accountability but insufficient for meaningful post-engagement review of autonomous decision-making.

5.3 AL2 - Reasoning Logging

Both what actions were taken and why the system decided to take them are captured. The decision-making process must be reconstructable from the available records. Regardless of implementation, sufficient logging must exist to understand what triggered each action and why alternatives were not selected. Full input and output data per operation may not be available.

5.4 AL3 - Full Audit Trail

A complete record of all system reasoning, decisions, actions, inputs, and outputs is captured. Full post-engagement reconstruction is possible, including the complete chain of reasoning behind every autonomous decision..

Recommended for CL5 deployments and for any engagement where legal, regulatory, or contractual accountability requirements apply. Note that for this level it is important to log actions and output in such a way that no eldess logging/reasoning loop exists

5.5 Audit Level and COAST Level Interaction

The higher the COAST level, the more operationally critical the audit level becomes. A CL5, AL0 deployment, full autonomous operation with no logging, represents the maximum possible gap between what the system did and what can be verified after the fact. This combination should be considered unacceptable in any professional engagement context.

COAST Level	Recommended Minimum Audit Level
CL3	AL2
CL4	AL2
CL5	AL3

Data Handling Levels

06

Data handling levels describe the organisational and infrastructure boundaries within which sensitive engagement data must remain independently of the tool's autonomy level. They apply at any COAST level where data sensitivity is a concern, and become particularly important at CL3 and above where systems may autonomously process sensitive data through external AI infrastructure.

Where data sensitivity is a concern, include a data handling level in the classification. At CL0–CL2: **CL[x], DH[x]**. At CL3 and above: **CL[x], AL[x], DH[x].** See Section 6 for the complementary Audit Level classification.

6.1 DH0 - No Restriction

Data may be processed by any system or service without constraint during the engagement. No requirements apply to where processing occurs or by whom.

6.2 DH1 - Defined Boundary

Data must remain within a set of explicitly agreed and documented systems and services defined before the engagement begins. The boundary must be agreed by all parties and documented in the rules of engagement before execution begins. Any system or service outside this defined boundary must not process engagement data. All sub-processors used by agreed services must be explicitly considered and included in the boundary definition. The defined boundary is only as precise as the sub-processor inventory supporting it.

6.3 DH2 - Operator Controlled

Data must remain exclusively on infrastructure physically owned and operated by the party conducting the engagement. Cloud-hosted infrastructure, regardless of exclusivity of access, contractual controls, or dedicated tenancy arrangements, does not qualify under DH2. Note that at CL3 and above, DH2 requires that all AI reasoning and inference occurs on infrastructure physically owned and operated by the party conducting the engagement. External AI services, cloud-hosted AI infrastructure, and third-party managed systems are not permitted under DH2.

6.4 Why Data Handling Levels Stop at DH2

A natural question is whether a fourth, stricter level is warranted: a level where data may not be processed at all, by any system, under any condition.

The decision not to include this level is deliberate, and mirrors the reasoning behind why COAST Levels stop at CL5. DH2 already describes the strictest realistic operational boundary available for processing data: infrastructure physically owned and operated by the party conducting the engagement, with no exception for cloud hosting regardless of exclusivity, contractual control, or dedicated tenancy. If data is too sensitive to be processed even under DH2 conditions, the appropriate response is not a stricter data handling level, it is to exclude that data, system, or subject from the engagement's scope entirely.

6.5 Data Handling Level and COAST Level Interaction

Data handling requirements are independent of autonomy level. A CL1 tool that submits engagement data to an external service is subject to the same data handling considerations as a CL5 system. Operational level restrictions must not be used as a proxy for data handling requirements. The appropriate data handling level is determined by data sensitivity and applicable legal, regulatory, and contractual requirements, not by the COAST level. DH level should be agreed and documented independently of the operational level ceiling.

Limitations and Trust Model

07

Section 1.5 introduced the properties COAST deliberately does not capture. Each is expanded on below.

7.1 What COAST does not measure

Result quality, coverage, or accuracy.
A higher COAST level does not imply superior outcomes. The autonomy of a tool and the quality of its results are independent properties. A skilled practitioner operating manually at CL1 may identify critical findings that a fully autonomous CL5 system misses entirely, because human expertise, contextual awareness, and professional judgement operate independently of the autonomy axis. The appropriate COAST level for any engagement is determined by context and risk appetite, not by an assumption that greater autonomy produces better results.

Inherent risk or danger of the operation.
COAST level does not correlate directly with operational risk. A human manually executing a destructive action is CL0 but potentially catastrophic in consequence. An autonomous agent operating at CL5 could be just passively gathering data and correlating results and thus have a low impact. The risk of any operation is determined by the nature of the actions taken and the context in which they are taken, not by the degree of human involvement in initiating them.

Operator or practitioner skill.
COAST describes the behaviour of the tool, not the competence of the human operating it. Two practitioners operating the same tool at the same COAST level may produce vastly different outcomes depending on their experience, domain knowledge, and judgement. COAST classification says nothing about the qualifications or skill of the operator.

Legal or regulatory compliance.
A tool can be operated at any COAST level legally or illegally depending entirely on context and authorisation. COAST describes autonomy, not authorisation. The same tool at the same level may be entirely legitimate in one engagement and a criminal act in another. Compliance with applicable law, regulation, and rules of engagement is the responsibility of the operator and is wholly separate from COAST classification.

7.2 Trust model and known limitations

COAST operates on a trust-based classification model. Classifications are made by vendors, practitioners, and clients based on available information and good faith. COAST does not currently define an independent certification, audit, or verification mechanism. This has several practical implications as detailed below.

Vendor classification accuracy
Capability level classifications provided by vendors are taken at face value. The distinction between CL3 and CL4, specifically whether a system generates capabilities that reach outside itself to interact with subjects in the target environment, may not be transparently disclosed by all vendors. Practitioners should ask vendors explicit questions about this capability when the distinction is relevant to the engagement.

Approval gate quality
The distinction between CL4 and CL5 depends on the presence of meaningful approval gates. The quality, completeness, and enforceability of those gates relies on vendor implementation and cannot be independently verified without technical inspection. Practitioners and clients should evaluate approval gate coverage as part of tool selection.

Data handling compliance
Compliance with DH level restrictions relies on vendor and operator disclosure regarding their infrastructure and sub-processor arrangements. Independent verification requires contractual mechanisms and technical controls that fall outside the scope of this framework. Independent classification auditing and tool certification is acknowledged as a meaningful gap and an area for future development.

Classifying Tools Using COAST

08

Applying COAST to a specific tool requires answering the following questions in sequence:

1. Does the tool perform only a single atomic operation per invocation, with all interpretation and follow-up performed by the human? → CL0
2. Does the tool or system execute within a single operational phase, without crossing phase boundaries, and without autonomously initiating subsequent actions based on its own intermediate findings? → CL1
3. Does the tool execute a human-authored orchestration chain that crosses operational phase boundaries, feeding results into a shared knowledge base? → CL2
4. Does the system generate capabilities — tools, scripts, probes, or code — that actively reach outside the system and interact with subjects in the target environment? → If no → CL3. If yes, continue to question 5
5. Does the system operate without any human approval gates during execution, including for high-impact actions? → If no → CL4. If yes → CL5

The following image serves as a visual guide for going through the classification questions.

8.1 Complete Classification

For tools at CL0–CL2 where data sensitivity is a concern, a DH classification should be included: **CL[x], DH[x]**, for example, COAST 1.0: CL1, DH1.
For tools and systems operating at CL3 or above, a full classification must also include an Audit Level (Section 6). Where data sensitivity is a concern, include a Data Handling Level (Section 7). The complete classification is expressed as: **COAST [version]: CL[x], AL[x], DH[x]**, for example, COAST 1.0: CL4, AL2, DH1.

All AL and DH classifications should be agreed with the client and documented in the rules of engagement before execution begins.

8.2 Engagement classification example

The following illustrates how COAST classifications can be applied across the different phases of a red team engagement with some example activities:

Activity	Classification	Notes
IN phase		
OSINT	COAST 1.0: CL3, AL2, DH1	No direct interaction with client systems or personnel. AI-driven investigation agents permitted within DH1 boundary. Discovered credentials must not be validated against live systems without written approval.
External Compromise Assessment	COAST 1.0: CL3, AL2, DH1	Autonomous reasoning over discovered services permitted. System may autonomously chain reconnaissance findings and propose exploitation paths. No active exploitation without explicit phase transition approval. All generated capability must be reviewed by operator before execution.
Phishing	COAST 1.0: CL4, AL2, DH1	AI system permitted to generate tailored phishing content based on OSINT findings. Campaign targeting list requires written client approval before delivery. Harvesting infrastructure must remain within DH1 boundary. Each delivery wave requires operator sign-off. Maximum [X] targets per wave unless otherwise approved

Activity	Classification	Notes
THROUGH phase		
Initial Exploration	COAST 1.0: CL3, AL2, DH2	All tooling and payloads must execute from DH2-compliant infrastructure from this phase onward. System may autonomously select and sequence exploitation techniques from provided toolset. Actions classified as high-impact, including service-affecting exploits and credential relay attacks, require operator approval before execution.
Lateral Movement	COAST 1.0: CL4, AL3, DH2	System permitted to generate subject-facing scripts and probes for discovered internal services not covered by provided toolset. Full audit trail required at AL3. Actions risking service disruption are classified as high-impact and require operator approval. Each new subnet accessed must be logged. No propagation to systems outside defined internal scope.
Privilege Escalation	COAST 1.0: CL4, AL3, DH2	System permitted to autonomously generate and execute escalation techniques tailored to discovered environment configurations. Actions modifying security configurations, group memberships, or authentication mechanisms are classified as high-impact and require written operator approval before execution. Escalation beyond agreed privilege tiers requires client contact sign-off.
OUT phase		
Action on Objectives	COAST 1.0: CL2, AL3, DH2	Operational level reduced to CL2 for objective phase, all actions follow pre-authored operator logic only. No autonomous decision-making during objective access. No actual data exfiltration, simulated exfiltration limited to agreed indicator files. Access to each objective system must be confirmed with the designated client contact before proceeding. Real-time operator oversight required throughout.

8.3 Important Classification Notes

Classify at maximum capability

A tool with both manual and automated modes should be classified at its highest capable level. How it is used in a specific engagement determines the operational level, not the capability level. This applies equally to platforms and frameworks, a platform is classified at the highest level of autonomy any of its configurations can achieve.

The level is determined by behaviour, not architecture

A single binary that internally orchestrates multiple operational phases is CL2 regardless of how it is packaged. A suite of separately invoked tools requiring human intervention between each is CL0 or CL1 depending on individual tool behaviour. The number of components involved is irrelevant to classification.

The level is determined by autonomous sequential decision-making, not by the presence of internal reasoning

The key question is not whether internal reasoning or AI processing occurs, it is whether the system autonomously initiates subsequent actions based on its own findings without human initiation of each step. A deterministic system selecting from pre-configured checks is CL1 regardless of sophistication. A system that autonomously decides what to do next based on intermediate findings is CL3 or above.

The CL3/CL4 boundary is defined by where generated capabilities operate, not by whether code generation occurs

CL3 systems can generate code, parameters, data processing scripts, correlation logic, where all generated code operates within the system boundary. The boundary is crossed when the system generates capabilities that reach outside itself and interact with subjects. For example, generating a custom nmap command is CL3, parameterising a provided tool. Writing a custom NSE script to probe a novel vulnerability is CL4, a new capability that reaches out and touches the subject.

AI and LLM involvement does not automatically determine the COAST level — the system's behaviour within the interaction does

The COAST level is determined by what the system does to produce its output. If the system processes only the content the human provided, this is CL0 and considered a single atomic operation. If the system autonomously fetches or processes additional information from external sources to produce its output, this is CL1. The level rises to CL3 or above only when the system autonomously initiates subsequent actions based on its own findings, without human initiation of each step.

Classify the resulting pipeline, not the tool used to build it

Orchestration frameworks, scripting languages, and automation platforms are not themselves classified by COAST. The pipeline or workflow they implement is what is classified. A script implementing a multi-phase orchestration chain is a CL2 system regardless of the language used to build it.

Classification maintenance

Tool capabilities evolve. A tool classified at CL3 may become CL4 following an update that introduces subject-facing capability generation. Vendors are responsible for maintaining current and accurate capability level classifications. Practitioners should verify the current classification of any tool before deployment or engagement, and should not rely on classifications established prior to a significant tool update.

Operational Best Practices

09

The following practices are recommended for practitioners, operators, and clients deploying tools across COAST levels. They address the most common operational decisions that arise when applying the framework in practice. These are not prescriptive requirements of the COAST model itself, but guidance grounded in the principles the model establishes. Organisations with specific regulatory, contractual, or risk management obligations should supplement this guidance with their own applicable requirements.

9.1 Scope Enforcement at CL3 and Above

At CL2 and below, scope is enforced by the pre-authored logic of the tool, the system is architecturally incapable of pursuing subjects it was not configured to address. At CL3 and above, the system reasons autonomously and could theoretically pursue subjects outside the defined scope if its reasoning leads it there.

Scope must be enforced as a hard architectural constraint at the infrastructure or system level, not relied upon as a behavioural property of the system's reasoning. Practitioners and vendors deploying systems at CL3 and above should ensure that scope boundaries are implemented independently of the system's own judgement.

9.2 Defining Guardrails at CL3 and CL4

The quality of guardrails at CL3 and CL4 directly determines the quality of human oversight. Poorly defined guardrails, for example, specifying only "dangerous actions" without further definition, result in inconsistent approval requests and may allow high-impact actions to proceed without appropriate oversight.

Effective guardrails should be defined in terms of the High-Impact Action definition in the glossary (Section 3), and should explicitly address: irreversibility, privilege level, potential for out-of-scope consequence, and legal or operational sensitivity in the specific engagement context.

9.3 Defining Operational Level Per Phase

Operational level do not have to default to a single uniform setting across an entire engagement. Different operational phases carry meaningfully different risk profiles, and rules of engagement should reflect this. Clients and practitioners are encouraged to define the maximum permitted operational level per phase rather than per engagement as a whole.

Higher autonomous operation may be appropriate during lower-risk phases such as passive information collection, while strict human control may be required during higher-risk phases such as active system interaction, privileged access operations, or sensitive data handling. See Section 8 for a complete rules of engagement example applying COAST level, audit level, and data handling level across engagement phases.

9.4 Capability vs. Operational Level in Rules of Engagement

Rules of engagement should specify the permitted operational level, the maximum level at which tools may be operated, defined per phase where appropriate, rather than attempting to enumerate permitted tools. This approach is more durable as tooling evolves and more precise in communicating the degree of human oversight the client requires.

9.5 Data Handling and Confidentiality

Data handling and operational level must be defined independently, restricting operational level provides no data handling guarantees. A client may permit CL4 autonomous operation while requiring DH2 data handling, meaning the system must operate on infrastructure physically owned and operated by the party conducting the engagement. Both should be documented separately in the rules of engagement.

9.6 Auditability

Auditability is not a property of the COAST level, any tool at any level may or may not implement logging. At CL3 and above, autonomous reasoning exists nowhere else without explicit logging. Agree on a minimum audit level before the engagement, document it in the rules of engagement, and ensure logging is implemented.

9.7 Human Override

The ability to halt execution exists at every COAST level, it is not a level-defining characteristic. However, the consequences of halting differ. At CL0 and CL1, halting is clean and unambiguous. At CL2 and above, halting mid-execution may leave partial results in the knowledge base or the subject environment in an indeterminate state. Halt procedures should be defined and understood before execution begins at any level above CL1.

Conclusion

10

COAST provides the cybersecurity industry with a shared, precise vocabulary for reasoning about tool autonomy. By defining six clear levels grounded in observable behaviour, and supplementary Audit Level and Data Handling Level classifications, COAST enables practitioners to communicate their methodology accurately, vendors to position their products objectively, clients to define their risk appetite and data handling requirements precisely, and regulators to write requirements that are enforceable in practice.

The model is intentionally designed to be technology-agnostic and discipline-agnostic at its definitional core. As the industry's tooling continues to evolve toward greater autonomy, COAST provides a stable reference framework that does not need to change as the technology does. Only the distribution of tools across the levels will shift.

Companion documents providing extended discipline-specific application guidance for DFIR, malware analysis, threat hunting, and OSINT, as well as future work on independent classification auditing and tool certification, are planned as continuations of this work.

Questions, feedback, and errata regarding this document are welcome and can be directed to info@securify.nl.

Appendix A: Quick Reference

11

The table below is a quick visual guide to help illustrate the levels of capability classification in the COAST model.

Human Control — System Control

	L0 Manual	L1 Automated Execution	L2 Multi-Phase Orchestration	L3 Autonomous Reasoning	L4 Autonomous + Self-Expanding	L5 Full Autonomous
IN CHARGE	Human	Human	Human	System	System	System
CAPABILITY	Single atomic operation per invocation	Human-configured decision tree within a single operational phase	Human-authored multi-phase chain crossing phase boundaries + shared knowledge base	Selects and sequences from human-provided toolset	Selects from and extends human-provided toolset with self-generated subject-facing capabilities	Unrestricted — selects, generates, and augments capabilities as needed
ACTION SPACE	Fully known, human-initiated	Fully known, human-authored	Fully known, human-authored	Category known — sequence and arguments unpredictable	Not fully knowable — system may generate novel subject-facing capabilities at runtime	Not fully knowable
APPROVAL	Every action	None	None	High-impact actions	High-impact actions	None
FAILURE HANDLING	• Human observes and decides	• Logs and skips • Stops if unrecoverable	• Logs and skips failed phase • Continues remaining chain • Stops if unrecoverable	• Autonomously adapts • Approval if recovery is high-impact • Escalates if goal unrecoverable	• Autonomously adapts incl. generating new subject-facing capabilities • Approval if recovery is high-impact • Escalates if goal unrecoverable	• Fully autonomous recovery • Contacts operator only if goal unrecoverable

Audit Levels (AL)

The table below is a quick visual guide to help illustrate the audit levels referenced in the COAST model.

Level	Name	What is captured	Recommended for
AL0	No Logging	Nothing, no reconstruction possible	Not recommended in professional engagements
AL1	Execution Logging	Actions taken and outputs, but not the reasoning behind them	Basic accountability only
AL2	Reasoning Logging	Actions taken and the decision rationale behind them; full I/O data may not be available	CL3 and CL4 deployments (minimum)
AL3	Full Audit Trail	Complete record of all reasoning, decisions, actions, inputs, and outputs	CL5 deployments; legal, regulatory, or contractual requirements

Data Handling Levels (DH)

The table below is a quick visual guide to help illustrate the data handling levels referenced in the COAST model.

Level	Name	What is captured	Recommended for
DH0	No Restriction	No restriction on where data is processed or by whom	Permitted
DH1	Defined Boundary	Actions taken and outputs, but not the reasoning behind them	Permitted where within the agreed boundary
DH2	Operator Controlled	Physical infrastructure owned and operated by the party conducting the engagement only	Not permitted, cloud-hosted infrastructure does not qualify regardless of access exclusivity

Securify helps organisations identify, resolve, and prevent technical security risks through penetration tests, red teaming engagements, and security code reviews. The focus is on providing effective protection against the most significant business risks. Based on security roadmaps, we ensure that code, (mobile) applications, infrastructure, and the organisation as a whole are optimally secured. For many years, companies ranging from start-ups to major banks have relied on Securify to safeguard their systems and (mobile) applications. With hundreds of penetration tests and security code reviews carried out each year, the Securify team helps protect the data of millions of people in the Netherlands.

Securify also brings an innovative vision to application security for (business) critical systems. Our unique approach enables development teams to deliver software continuously, demonstrably, faster, and more securely. This way security is no longer a blocker to progress. Securify is part of the Solvinity Group. For more information, visit securify.nl/en.

Securify

Securify B.V.
Naritaweg 132
1143 CA Amsterdam

T +31(0)20 820 4516
E info@securify.nl
www.securify.nl/en