

IN-PERSON WORKSHOP FOR TECHNICAL TEAMS

Secure development with AI

Code with Claude Code or Copilot without opening security holes in the codebase

One working day so development, architecture and security agree on three things: what the assistant may touch, when it has to ask, and which checks block an unsafe change before merge. We spend the last block on the company's own code.

5 to 8 h

on-site, depending on the team

5

hands-on labs

6

content blocks

1

block on the company's code

WHERE TEAMS START

AI is already part of how the team builds software, and the controls are still sitting outside

Development teams move faster with Claude Code or GitHub Copilot. Along with them came new things in the process: files like AGENTS.md or CLAUDE.md that change how the agent behaves, Skills downloaded from third parties, MCP servers with credentials, and modes that run commands without asking.

The risk starts before the code the assistant generates, with everything it reads, installs and can run. One line in an IDE settings file can switch off the confirmations, and a public issue can end up as an order for the agent running in the pipeline. And it keeps going after that, when a pull request looks fine because it compiles.

In this workshop we turn that into concrete decisions. By the end, the team knows what to review before enabling a capability, how to limit it, and what evidence to ask for before accepting the result, and has tried it on its own code.

The five doors the team learns to close

01 Context

Which instructions the assistant loads and who can change them.

02 Skills

Which behaviour gets installed before anyone reads a line.

03 MCP

Which systems, data and credentials the agent can reach.

04 Autonomy

What runs alone, what requires approval and what never runs.

05 Generated code

Which tests and gates decide whether a change reaches main.

Five surfaces, one control system the team tests on its own repository the same day.

WORKSHOP OUTCOME

Five decisions the team can make without improvising

Each one gets demonstrated during the day, first in the labs and then in the team's repository. Recognising the term on a slide does not count.

01

Separate the model from the orchestrator

Identify whether the risk comes from model output or from policy deciding which files, tools and permissions it receives.

Evidence: a trust map of the assistant the team uses.

02

Audit context, Skills and MCP

Find poisoned instructions, overbroad capabilities, tool descriptions trying to steer the agent, and remote servers that accept tokens issued for someone else.

Evidence: three unsafe artefacts reviewed and a documented decision.

03

Choose the right autonomy level

Separate reading, editing and execution. Apply ask-first, a deny list and a filesystem and network sandbox wherever credentials or system impact exist.

Evidence: the agent asks before env, installs and test changes.

04

Define when generated code is done

Turn risk into a Definition of Done covering unit, golden, smoke, end-to-end, SAST, secrets and dependencies, written into the project's instruction file.

Evidence: a reusable definition ready for the project.

05

Review without reading every line

Focus human review of a generated pull request on assertions, CI YAML, lockfiles, project instructions, assistant configuration and sensitive code paths.

Evidence: a generated change blocked by a test first seen red.

USE CASES

Situations any development team will recognise

These are decisions a team makes every day once an assistant works inside a repository, and most have a public incident behind them. Each risk carries its OWASP reference: LLM for model-based applications (2025) and ASI for agentic applications (2026).

CONTEXT POISONING • LLM01, ASI06

A repository file convinces the agent to write unsafe SQL

The instruction looks like another coding convention. The assistant treats it as authority and concatenates query parameters.

CONTROL PRACTISED

Review instructions as code, scan for hidden characters, and start a fresh chat after cleaning context.

EXCESSIVE AGENCY • LLM06

A read-only MCP reaches private repositories

The config shows a command and a token. It does not show the token scope or what the server does with it.

CONTROL PRACTISED

Dedicated agent identity, least privilege, allowlists, and server review before enablement.

UNEXPECTED CODE EXECUTION • ASI05

The agent edits its own configuration and stops asking

An instruction hidden in code the agent reads tells it to add a line to the IDE settings or to mcp.json. From then on it runs commands without confirmation.

CONTROL PRACTISED

Assistant configuration files kept outside the agent's write access and reviewed as code in every pull request.

GOAL HIJACK • ASI01, ASI02

A public issue gives orders to the pipeline agent

A workflow passes an issue title into the prompt of an agent with shell access and the job's secrets. Whoever writes the issue decides what runs.

CONTROL PRACTISED

Issue, PR and commit text treated as untrusted input, minimal tools, and the job that reads outside text kept apart from the job that publishes.

SUPPLY CHAIN • LLM03, ASI04

The assistant invents a dependency that happens to exist

The name sounds plausible. Someone registered it and the agent adds it to the lockfile as though it were established.

CONTROL PRACTISED

Every new package name triggers human review of its registry page, maintainer and lockfile diff.

TEST TAMPERING

Tests pass because the agent weakened them

The build turns green after an assertion is removed, a case is skipped or a snapshot is refreshed without approval.

CONTROL PRACTISED

Red first, no loosening tests, and human review of assertions, skips and goldens.

SENSITIVE INFORMATION DISCLOSURE • LLM02

A secret enters context without appearing in the diff

It can arrive through a pasted log, env output, an MCP reading .env or a value stored in the project file.

CONTROL PRACTISED

Paste hygiene, env and .env denial, prior redaction and immediate rotation after disclosure.

What is taught, block by block

The sequence follows a real request: context enters, the agent gains capabilities, acts on a laptop or in the pipeline, and proposes a change that must clear controls.

01 How a development assistant actually works

Risk changes when model and orchestrator are separated.

- Model, orchestrator, tools and tool results.
- Tokens, context windows and what disappears between sessions.
- Static context versus dynamic context.
- Trust boundaries inside the IDE and repository.
- Where each risk sits in the OWASP Top 10 for LLM Applications 2025 and the Top 10 for Agentic Applications 2026.

Real case: OWASP files EchoLeak under goal hijack (ASI01), the Amazon Q incident under tool misuse (ASI02) and Replit under rogue agents (ASI10).

02 Poisoned context and instruction chains

Text read by the agent can become policy.

- AGENTS.md, CLAUDE.md, and Copilot instructions as model-executed policy.
- Indirect prompt injection through repositories, issues, documents and tool output.
- Invisible characters, hidden instructions and persistence between chats.
- Provenance for internal documentation, RAG and vector stores.

Real case: Rules File Backdoor, Pillar Security, 2025: invisible Unicode in Cursor and Copilot rule files. EchoLeak, CVE-2025-32711: hidden instructions in an email that Microsoft 365 Copilot processes without the user clicking anything.

03 Skills and MCP without blind perimeter growth

Installing behaviour and connecting a service are different decisions.

- When to use a Skill, MCP or a plain CLI.
- Tool poisoning in names and descriptions.
- Scopes, tokens, human identity and agent identity.
- Authorising a remote MCP server per the specification: OAuth 2.1 with PKCE, tokens issued for that server (RFC 8707 and RFC 9728), and no passing tokens through to other services.
- Local STDIO servers take their credentials from the environment of whoever starts them.
- Pinning, third-party review and removing capability after use.

Real case: GitHub MCP, Invariant Labs, 2025: a public issue leads the agent to copy private repository data into a public PR. MCPoison, CVE-2025-54136: Cursor tied approval to the server name, not to what it ran.

04 Autonomy, sandboxing and secrets

The same prompt carries a different risk depending on what the agent can run and reach.

- Read-only, ask-first, auto-edit and full-auto.
- Allowlisting routine work and denying high-impact operations.
- A filesystem and network sandbox for the agent and its subprocesses; a dev container when the tool has none.
- Assistant configuration as attack surface: one change to settings.json or mcp.json switches off approvals.
- Production credentials, shell, cloud tools and destructive changes.
- What may be pasted, what gets redacted and what must be rotated.

Real case: CVE-2025-53773: an injection turns on Copilot auto-approval in VS Code and ends in code execution. Nx singularity, 2025: malicious packages launch Claude Code, Gemini CLI and Amazon Q with permission checks disabled. Replit, 2025: an agent deletes a production database during a code freeze.

05 Agents in the pipeline

An agent in CI inherits the job's permissions and reads text anyone can write.

- Issues, pull requests and commit messages as untrusted input, even when the workflow belongs to the team.
- Who can trigger the workflow, which token it receives and which tools the agent gets.
- Shared caches and publishing: keep the job that reads outside text apart from the job holding release secrets.
- CI tokens scoped to the task: the job that builds does not publish.

Real case: PromptPwnd, Aikido, 2025: issue text injected into agents in GitHub Actions; Google fixed Gemini CLI within four days. Cline, February 2026: an issue title ended in an unauthorised npm release that stayed up for about eight hours. Amazon Q, CVE-2025-8217: an overscoped token let code into a release.

06 DevSecOps and reviewing code nobody will read in full

Who writes the code changes, but the merge gates stay as they are.

- A persistent Definition of Done in AGENTS.md or the project file, not in the chat.
- Unit, golden, smoke and end-to-end tests, and what each proves.
- SAST, secret scanning, dependencies and slopsquatting.
- What a person reviews in a generated pull request: assertions, skips, CI, lockfiles, instructions, assistant configuration and sensitive paths.
- Git discipline when several agents share one checkout.

Real case: Spracklen et al., USENIX Security 2025: at least 5.2% of suggested packages invented by commercial models and 21.7% by open models, with 205,474 distinct names anyone could register.

Five labs, five controls each participant checks

Every lab works the same way: we see the unsafe behaviour once, apply the control and show that the result changes. The labs run in a workspace built to break without consequences. Then the same thing is repeated on the company's repository.

01

The project with bad instructions

Load a poisoned rules file, request product search, and find why the agent proposes vulnerable SQL.

Outcome: reviewed instructions, hidden characters found, and clean context in a new session.

02

The Skill that should never be installed

Review a Skill that demands environment dumping and removal of human confirmation.

Outcome: a refusal justified by a specific line, not by a vague impression.

03

Two MCP servers that look alike in config

Compare a scoped server with one able to read any file on the machine.

Outcome: only the allowlist is enabled and a request for secrets is blocked.

04

Generated code against the harness

AI implements a function under a Definition of Done, then the team breaks it to prove the injection test fails.

Outcome: red-first evidence and explicit acceptance criteria for the pull request.

05

The autonomy dial under pressure

A prompt tries to push the assistant into env, installs and test changes without approval.

Outcome: ask-first and the deny list proven in each participant's actual tool.

We do the last part on the company's own code

What we saw break in the labs is then looked for in one of the team's repositories. Each participant works on their own laptop while I go round the room answering questions and unblocking.

01

What the assistant reads

Find AGENTS.md, CLAUDE.md, and Copilot instructions, decide who may change them, and scan for invisible characters with the command used in the morning.

02

What it is connected to

List the MCP servers, Skills and assistant configuration files that affect the repository, with the real scope of each token. Anything nobody can justify is switched off that day.

03

What it can run

Set up ask-first, the deny list and the sandbox in each person's own tool, and check that the agent asks before reading the environment or touching a test.

04

When it is done

Write the Definition of Done into the project's instruction file and pick the first control that will go in by pull request, with an owner.

Everything happens on each participant's laptop, on their own repository. The code never leaves the company, I do not copy it or take it with me, and this is not an audit: there is no findings report. Whatever is written in that block stays in the company's repository.

Between 5 and 8 hours, depending on the team

The day alternates explanation, decisions and practice, and it ends on the company's code.

Block 1

Map the system the team already uses

Model, orchestrator, context and tools, with the OWASP risk map. Trust boundaries come before controls.

Block 2

Break trust in context

Poisoning, AGENTS.md, persistent instructions and lab 1. The team sees a repository rule alter generated code.

Block 3

Decide what connects and with which permissions

Skills, MCP and its authorisation, identity, autonomy, sandboxing and secrets. Labs 2, 3 and 5.

Block 4

Block an unsafe change before merge

Agents in the pipeline, Definition of Done, review of generated pull requests and lab 4.

Block 5

The company's code

Each participant opens their repository and their assistant. Inventory of instructions and capabilities, permissions in the real tool and a Definition of Done in the project. Everyone leaves with a first control chosen.

The workshop ends and the controls stay

The material is built for reuse in the repository and in conversations between architecture, security and development.

Definition of Done for AI changes

A template ready to adapt to AGENTS.md or the project's instruction file.

One-page control card

Context, Skills, MCP, autonomy, secrets, code and supply chain, each with its OWASP reference.

Context hygiene guide

What to redact before pasting and what to do after disclosure.

Checklist for agents in CI

Untrusted inputs, token permissions, agent tools, caches and a separate publishing job.

Reference pipeline

Azure DevOps sample with real pytest and the next gates identified.

Workspace with five labs

Write-ups, safe and unsafe fixtures, and tests.

Starter deny list

Git, shell, cloud and secret operations requiring a person.

Autonomy matrix

Equivalent modes, approvals and sandboxing in Copilot, Claude Code and Gemini CLI.

Multi-agent git discipline

Explicit paths, status before commit and file ownership.

WHAT COMES WITH THE COURSE

Four things that stay with the team, at no extra cost

They are what teams end up asking for after a course.

+1

Compliance evidence pack

A signed attendance sheet, an individual certificate with date, hours and contents, a syllabus mapped to the standards that apply to the company, and a one-page letter for the auditor.

+2

A 10-minute survey beforehand

Before the day, the team answers a short survey about its stack and where it gets stuck. I use it to adapt the labs to what the team really uses.

+3

Configurations on a branch of the company's repository

During the final block I leave the configurations we tested on a branch: assistant instructions and permissions, MCP rules and pipeline checks. It includes a technical template for an AI and MCP use policy; it is technical, not legal advice. The team reviews and merges them like any other change.

+4

Labs with solutions

The lab repository with its solutions and a 12-month internal-use licence, so new people on the team do not depend on repeating the day.

COMPLIANCE

What each standard asks for and what I hand over as evidence

The course does not make anyone compliant by itself. It serves as documented evidence for the training control the auditor already asks about, where the standard applies to the company.

STANDARD	WHAT IT ASKS FOR	WHAT I LEAVE AS EVIDENCE
PCI DSS 4.0 · 6.2.2	Training at least every 12 months for people who develop bespoke software: secure design, secure coding and testing tools, documented.	Syllabus mapped to those points, signed attendance and date.
ISO 27001:2022 · A.6.3	Security awareness, education and training appropriate to each person's role.	Individual certificate with date, hours and contents, and a per-person record.
ISO 27001:2022 · A.8.25 and A.8.28	Secure development life cycle and secure coding, including training for those who develop.	Mapped syllabus and the pipeline checks that stay on the company's branch.
SOC 2 · CC1.4 and CC2.2	A documented training programme, with completion records per person and depth matched to the role.	List with name, date, contents and role.
DORA · art. 13(6)	Digital operational resilience training for staff, proportionate to their role (banking and insurance).	Attendance record and course syllabus.
AI Act · art. 4	Measures to support AI literacy among staff, adapted to their role. It is an obligation of means and an internal record is enough.	Internal training record, with contents on AI risks and use for each role.

WHO IT IS FOR

For organisations that have moved from trying AI to depending on it

A strong fit for

- Security architects turning AI risks into controls development teams can apply.
- AppSec, DevSecOps and Platform Engineering owners of pull-request gates, pipelines and tool connections.
- Engineering managers and tech leads already seeing generated code in production repositories.
- Security champions and senior developers who will spread these practices through the team.
- Teams using Claude Code or GitHub Copilot and still deciding permissions case by case.

Not the right workshop if

- The team has never used an assistant on a real repository.
- The company wants an introduction to LLMs or prompt writing.
- What is needed is corporate policy, model governance or executive training.
- What is expected is an audit of the code with a vulnerability report. In the last block the team works on its repository, but the team does the review.

WHO TEACHES IT

Security and development taught by someone who has worked on both sides



I'm Daniel Alfocea, and I've spent twenty years in development and security. At Banco Santander I was Principal Engineer in the AI department while the area was being set up. At BBVA I was Cybersecurity Tech Lead. At 42Crunch, in the UK, I was responsible for one of its API security products.

I've presented research at RSA Conference and OWASP AppSec EU, and I'm a regular speaker at RootedCON. Spain's National Cybersecurity Institute (INCIBE) has asked me to judge its competitions several times. I founded Navaja Negra and the OWASP Madrid chapter. I'm the author of widely used security tools such as GoLismero, Plecost and Dockerscan, and of LAPIS, research published on arXiv on how to describe an API to a model with fewer tokens. With Alfonso Muñoz I wrote *Secure MCP: A practical guide for developers, software architects and tech leads*.

This workshop puts together two things that are almost always taught apart: how the assistant works inside a repository, and how to build controls that a development team can follow. And it finishes with those controls applied to the company's code.

GUARANTEES

Two commitments in writing, and one limit

If the configurations do not work, I fix them

If the configurations I leave in the company's repository do not work, I run a free remote session to fix them.

Stuck after 30 days, one free hour

If after 30 days the team has got stuck applying something from the course, I spend an hour with the team remotely, free.

This is not an audit or a promise of results: I do not guarantee percentage improvements or that there will never be an incident.

Questions worth resolving now

How much does it cost?

It depends on the duration, between five and eight hours, and on the size of the team. After the first conversation I send a fixed proposal.

Is it for security or development?

Both, as long as they share decisions about the SDLC. Architecture and AppSec set the limits, and development has to turn them into instructions, tests and gates it can follow every day. If only one side comes, the day gives less.

Do we need to install both Claude Code and Copilot?

No. Each participant needs one of the two, already signed in and able to work against a local folder. The ideas are the same, and only the file names and mode names change.

Does it adapt to our language or stack?

The labs use Python and the sample pipeline uses Azure DevOps. The controls apply across languages and CI systems, and in the last block each participant applies them to the team's real stack.

Can we use our repository during the workshop?

Usually yes, in the last block. The labs use a workspace designed to break safely, so the whole group sees the same failure. Afterwards each participant opens one of the team's repositories on their laptop and applies what they learned. If the company can't open its code that day, we work on a practice repository similar to its stack. The code never leaves the company.

What has to be sorted out before that block?

Each participant must be allowed, under the company's internal policy, to open the repository with their assistant. If a repository cannot go through that tool, we pick another one from the team that can. We confirm this in the first conversation.

Does it include a certificate?

It includes an individual attendance certificate with date, hours and contents, and no exam. It is evidence of attendance, not a qualification.

Can it be delivered remotely?

Yes, although in-person works better for configuration review and hands-on troubleshooting. Scope remains the same.

Does it guarantee we will have no AI incident?

No. I guarantee the five labs, the block on the company's code and the materials. Risk reduction depends on the organisation applying and maintaining the controls.

What if we need follow-up?

There is a 6-week programme on this same topic, with measurement before and after and review of the team's PRs. The details are on [the programme page](#).

FORMAT

One team, one room and one shared system for decisions

It covers preparation, between five and eight in-person hours depending on the team, the block of work on the company's repository and all working material. Small groups make it possible to review labs while they happen and for me to visit every desk in the last block; a crowded room turns practice into a demonstration.

- Participants from one team or collaborating functions.
- Delivered in English or Spanish.
- The company provides room, screen and wifi.
- Each participant brings their laptop, with their assistant and access to one of the team's repositories.
- Travel outside Madrid is agreed before the date is fixed.
- It includes the evidence pack, the configurations on the company's branch and the two commitments above.
- It is designed as a working day for a team that will apply what it learns. It is not an audit, a compliance certificate or a talk to sit through.

Write to me at

me@danielalfocea.com

In the first conversation we check whether the workshop is a fit, how many people will take part and which date works for the company.

Can be subsidised through FUNDAE (Spanish training credit).

EMAIL

me@danielalfocea.com

PROGRAMME ONLINE

danielalfocea.com/en/formacion/workshop-desarrollo-seguro-con-ia