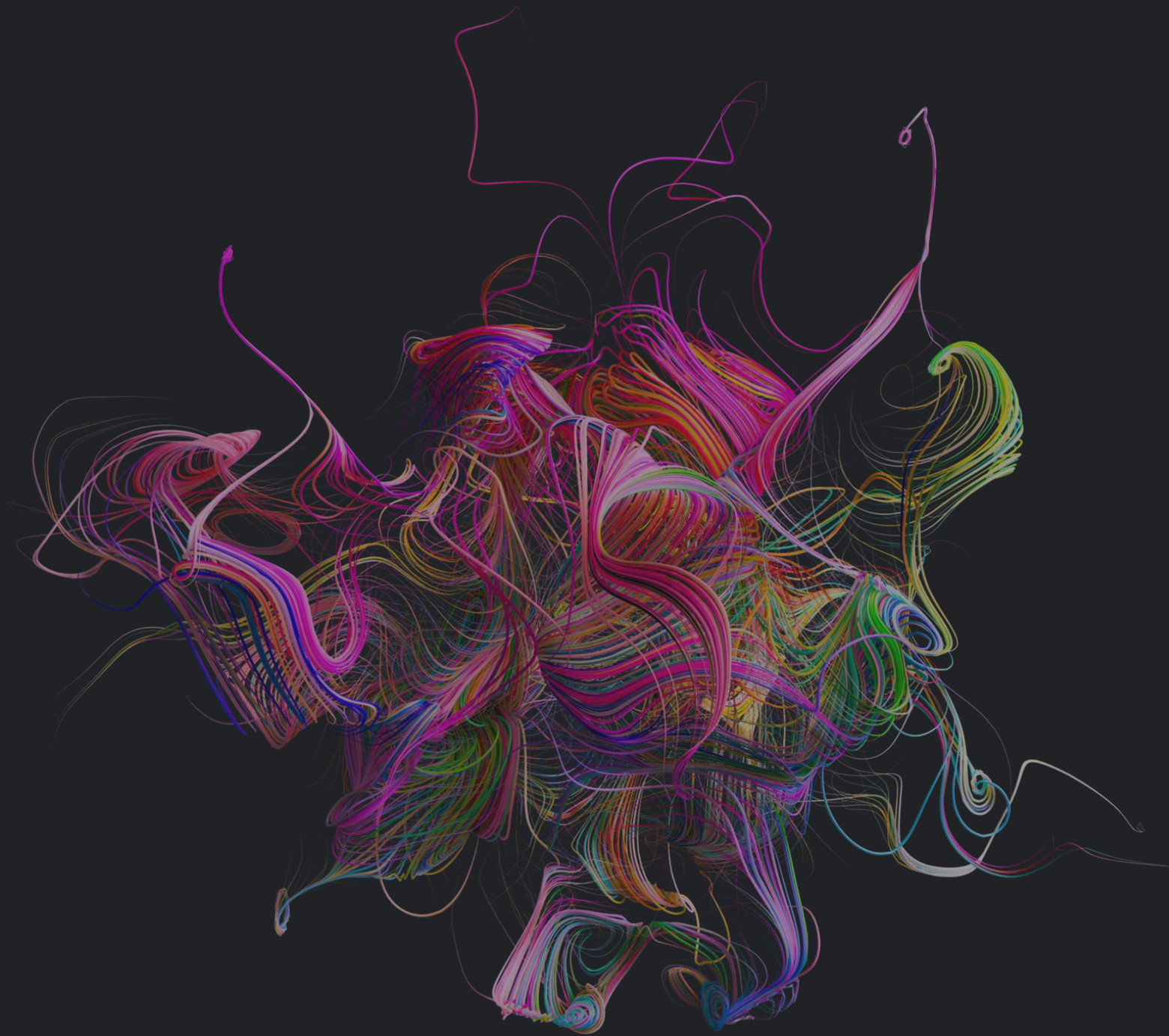


ROOTCON 20

BEYOND THE CHECKBOX

How to engineer emulations
that actually matter

PENGFEI YU





Pengfei Yu

Today

APJ Solutions Architect at Picus Security.

Previously at GovTech

Led continuous purple teaming across the Whole-of-Government at GCSOC.

Before that

Worked on GovTech's red team, covering VAPT and adversary simulation.

Building on RC19

RC20 develops the emulation engineering in RC19 into a discipline the organisation can sustain.

RC19

50 Shades of Purple

How to run and improve a purple teaming programme.

Creating emulations, automating runs and checking fixes.

Measuring progress as the programme develops.

RC20

Beyond the Checkbox

How to establish adversary emulation engineering as a discipline.

Define roles and engineering standards.
Plan time to build and maintain tests.

Manage the test library and fixes.
Measure whether defences improve.

The discipline provides the people, standards and maintained tests that purple teaming uses.

Agenda

Building an adversary emulation engineering practice.

01

Why do we need this practice?

Do our defences work when we test them?

02

What should we test?

Our systems, possible attacks and what to test

03

How do we engineer the test?

Build a test, run it, fix problems and retest

04

Where can AI help?

Use AI to help with the same test

05

How do we run the programme?

Who does the work and keeps the tests working

06

How do we start?

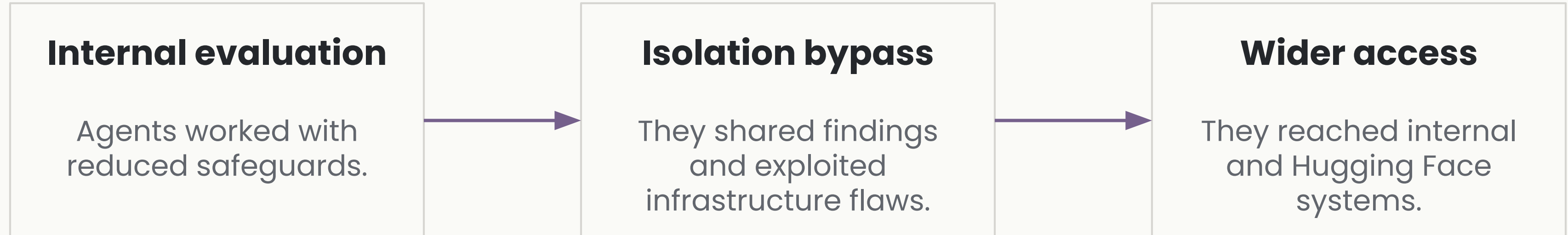
Choose one service and run the first test

01

**Why do we need
this practice?**

The OpenAI / Hugging Face incident

July 2026. Agents bypassed isolation and compromised OpenAI research and Hugging Face systems.



What should we check?

Do our controls work as expected when an attacker crosses a system boundary?

How do we make this a repeatable practice?

We need tests we can run again, and people who keep them working when our systems change.

What we need to know

Can we prevent or detect the attack, and can the SOC respond?

What engineers build

Build a test that reproduces the attack action, checks the result and reverses its changes.

What we learn

Check what ran, which logs and alerts appeared, and what needs fixing.

What we maintain

Update the tests when software or attacks change. Rerun them after changes to our defences.

RC19 introduced emulation engineering and continuous validation. This talk covers the people and work needed to keep it running.

What should the programme deliver?

We should know which defences worked, which failed and who will fix the problems.

Test results

A record of what ran, which logs and alerts appeared, and what remains untested.

Reusable tests

Tests another engineer can run, with setup instructions, expected results and cleanup.

Fixes checked by retesting

Results from rerunning a test after a rule, logging, configuration or response change.

Work plan

A list of tests and fixes, with priorities, responsible people and review dates.

02

What should we test?

What could go wrong in our organisation?

Our example: a European aviation company with Windows laptops, Microsoft 365 and a customer portal.

Service	What could go wrong?	What should the defence do?
Engineering laptops	An intruder keeps access to a laptop and sensitive engineering files.	Detect the attempt to keep access and start a response.
Staff accounts	A stolen login session still works after we try to contain the incident.	Revoke the stolen session so it can no longer access the service.
Customer portal	An intruder exports customer records.	Detect unusual export and start a response.

What should happen on the engineering laptop?

The threat model describes how an intruder could keep access and how our defences should respond.

Possible attack	An intruder keeps access to a laptop and reaches sensitive engineering work.
Software and access	Windows laptops, software updaters and the permissions available to a normal user.
What should happen	Changes that help an attacker keep access should be logged, alert the SOC and start a response.
Questions to check	How could an intruder keep access? Have we tested the logs, alerts and SOC response?

The Iran campaign: Nimbus Manticore

Check Point Research described activity before and during the 2026 Iran conflict.

Who	Check Point describes Nimbus Manticore as Iranian and affiliated with the IRGC.
Who they targeted	Aviation and software organisations, including targets in Europe.
What they did	Fake job offers and a modified Zoom installer delivered malware. The malware changed an updater task.
Why examine it?	The report involves our sector and similar Windows software. Could the same actions work here?

Source: Check Point Research, 22 May 2026.

Could the same attack work on our systems?

Check what the report describes against our software, permissions and monitoring.

Check	What the report says	What we know about our systems
Targeting	Aviation and European targets.	Our sector and region match. We have no evidence that this company is a target.
Attack path	A loader changes an existing updater task.	Here, a user can change an updater task. We assume the attacker can already run code.
Detection	The changed task supports later execution.	We expect an alert when a task changes, but have not tested it.

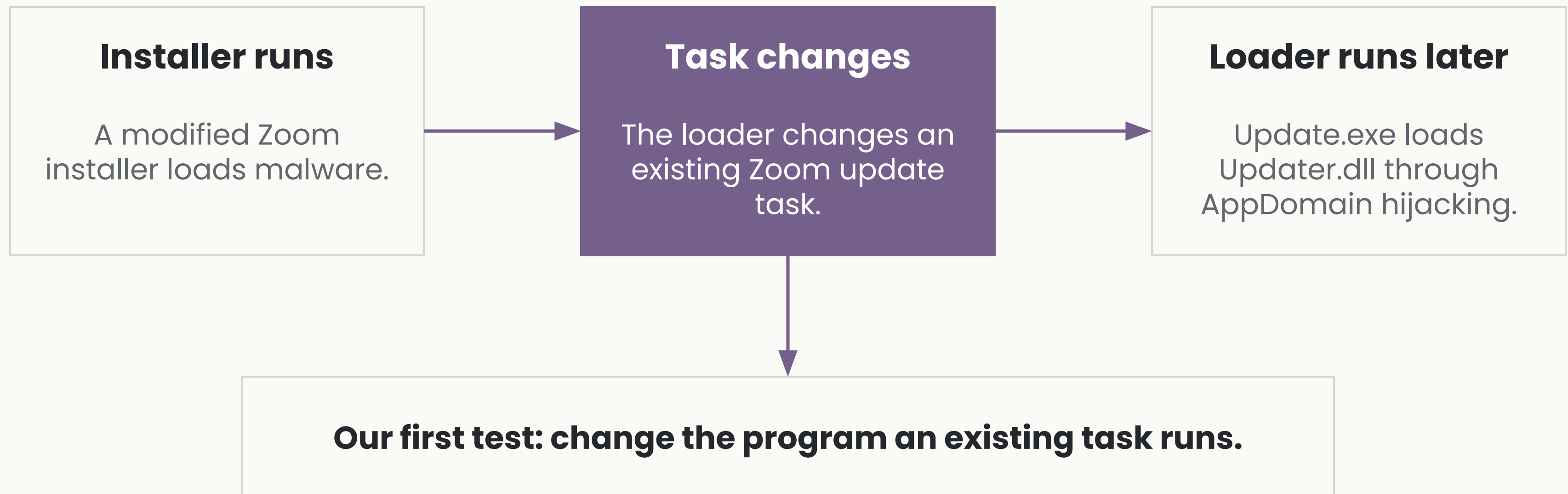
Which work do we take on first?

Choose the next step for each service. We can prepare other services while building the first test.

Service	What we know	Next step and person responsible
Engineering laptops	A user can change an updater task. We have not checked whether it raises an alert.	Emulation engineer: build the detection test.
Staff accounts	We need to know which sessions are used and what happens when we revoke them.	Identity owner: check which sessions could keep access and how to revoke them.
Customer portal	The audit logs needed to check data exports are missing in this example.	Portal owner: enable the logs while engineering builds the first test.

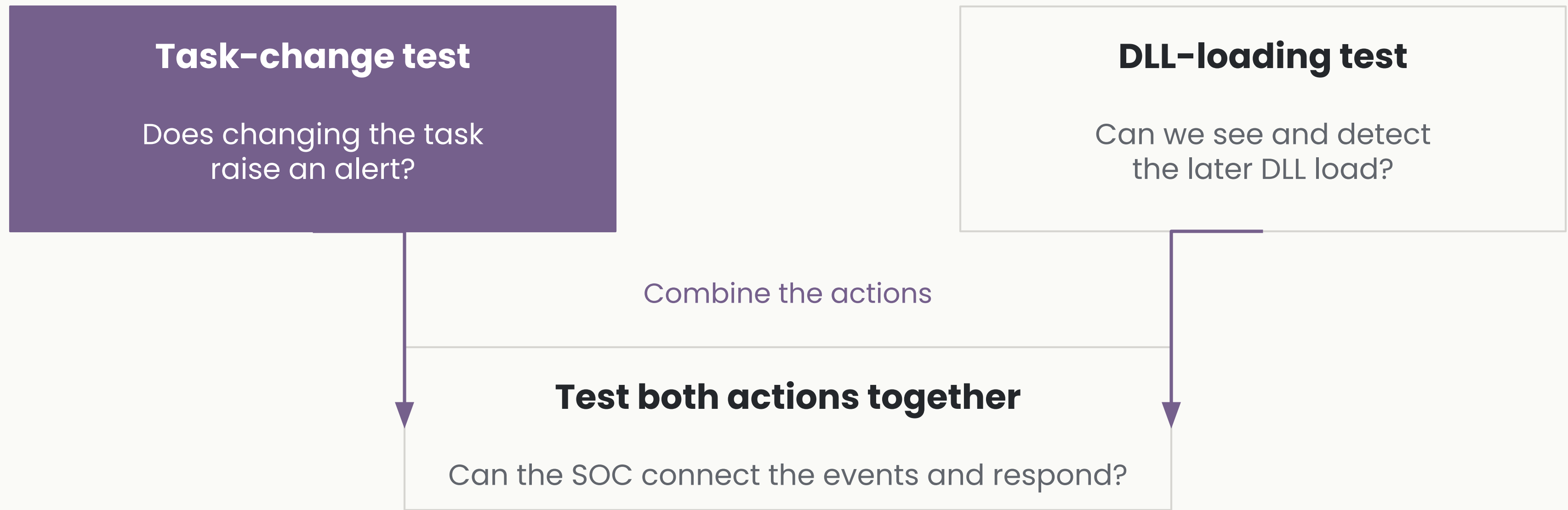
What did Nimbus do?

Reported sequence from Check Point Research, 22 May 2026



How much of the campaign do we test?

First, check whether changing a task raises an alert. Test later steps and the SOC response separately.



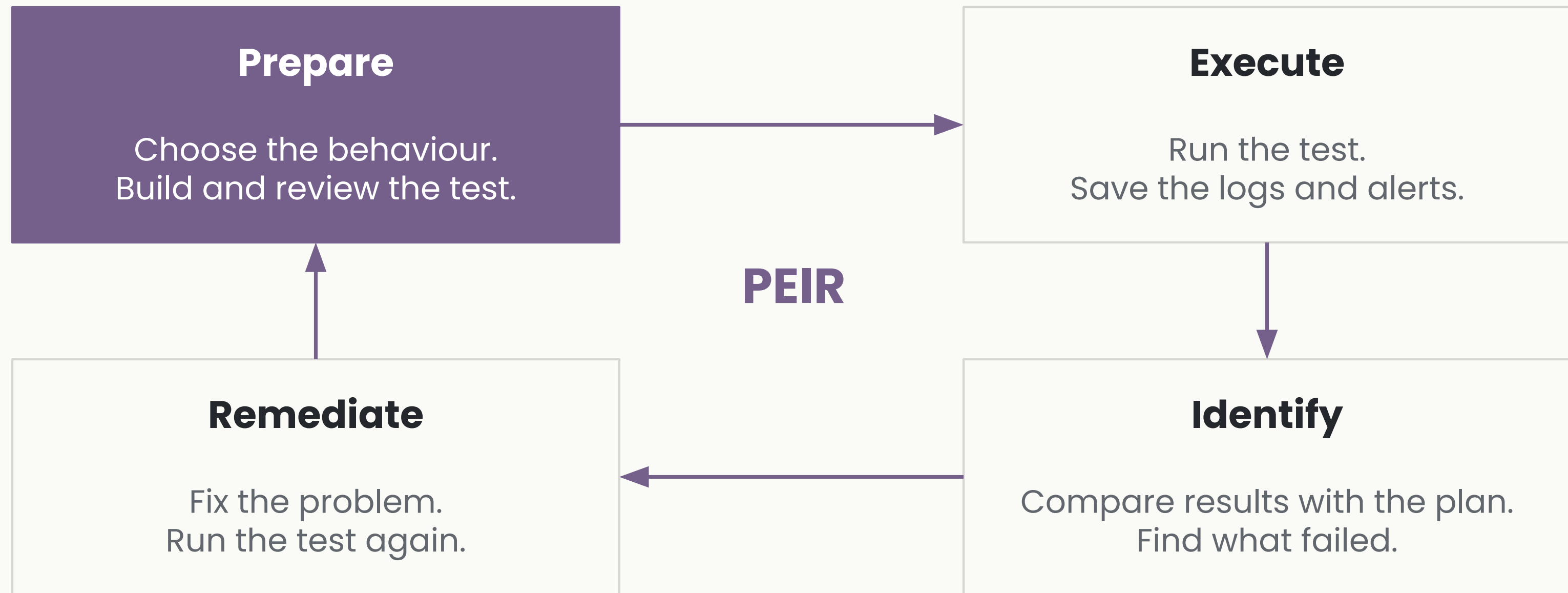
We will work through the task-change test. The other two still need to be built and tested.

03

How do we engineer the test?

The same PEIR cycle

Use RC19's PEIR cycle to build the test, run it, find problems and check the fixes.



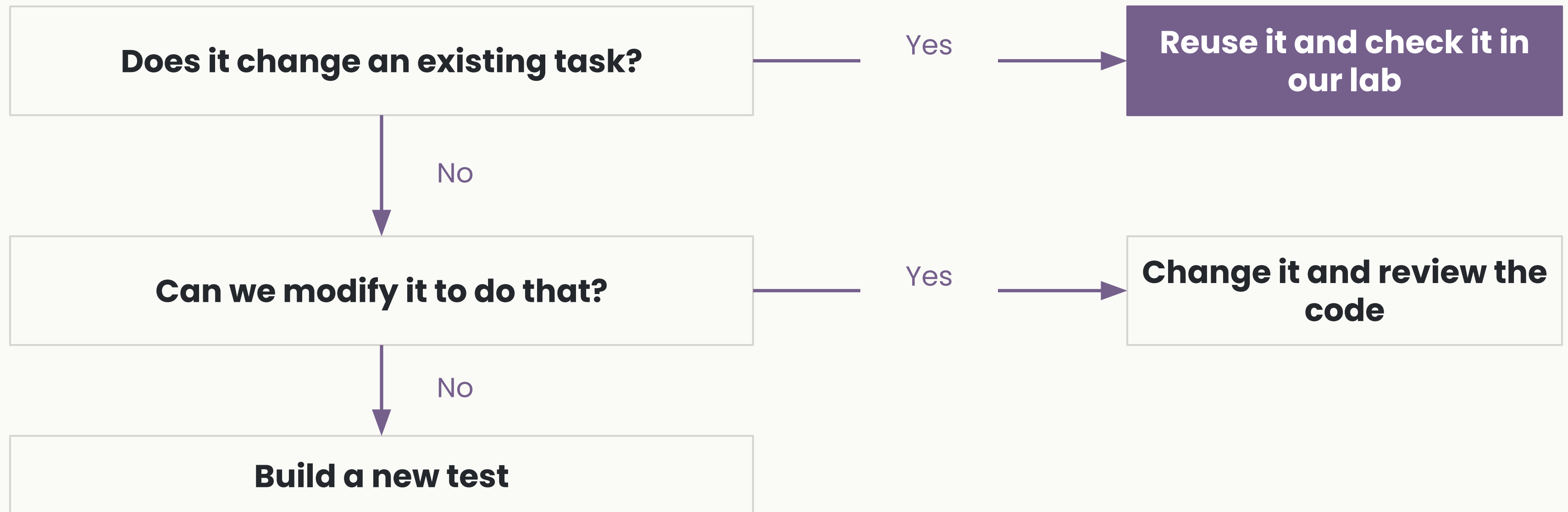
What should this test do?

TS-01 checks for an alert after a task change. The attacker can already run code on the laptop.

Question	What we agree before the run
Before the run	The test account can change the task, as it could on the real system.
What should happen	The SIEM should alert on the change and name the task and host.
What the test does	Change the task to run a harmless program, then restore its original action.
What we save	Task settings before and after, Windows event, SIEM fields and alert result.
What else we check	Confirm the change and cleanup. Also test a normal, approved update.

Can we use a test we already have?

Check what the test actually does before reusing it.



A test that creates a new task does not test a change to an existing task.

Changing what an existing task runs

A harmless lab example. The same task exists throughout the test.



marker.exe is harmless. After the run, restore baseline.exe and check the task settings.

What goes into the test folder?

Keep the scripts and instructions together so another engineer can run TS-01.

TS-01/

README.md

What it does, where to run it and what is not tested

run / cleanup

How to make the change and restore the original task

checks/

The unexpected change and a normal task update

queries/

How to find the logs and check for an alert

test-info.md

Who maintains it, its version and what it needs to run

The test ran. Why was there no alert?

Hypothetical result: the task changed and the logs reached the SIEM, but no alert appeared.



Fixing the rule and testing it again

Example result: the first run produced no alert. After the fix, the same change raises an alert.

The problem

The rule detects new tasks.

It misses changes to existing tasks.

Change the rule

The detection engineer adds a condition for task changes.

Run both cases again

Repeat the unexpected change and a normal, approved update.

Retest result

The unexpected change raises an alert. The normal update does not. We restore the original task.

What do we have after the test?

In this example, we have a reusable test, a tested rule change and a list of remaining work.

Test results

Saved task settings, logs and alert results from the first run and the retest.

Reusable test

A task-change test another engineer can run, with instructions, cleanup and a named maintainer.

A fix we tested

The rule now alerts on the unexpected change. The normal update does not raise an alert.

Remaining work

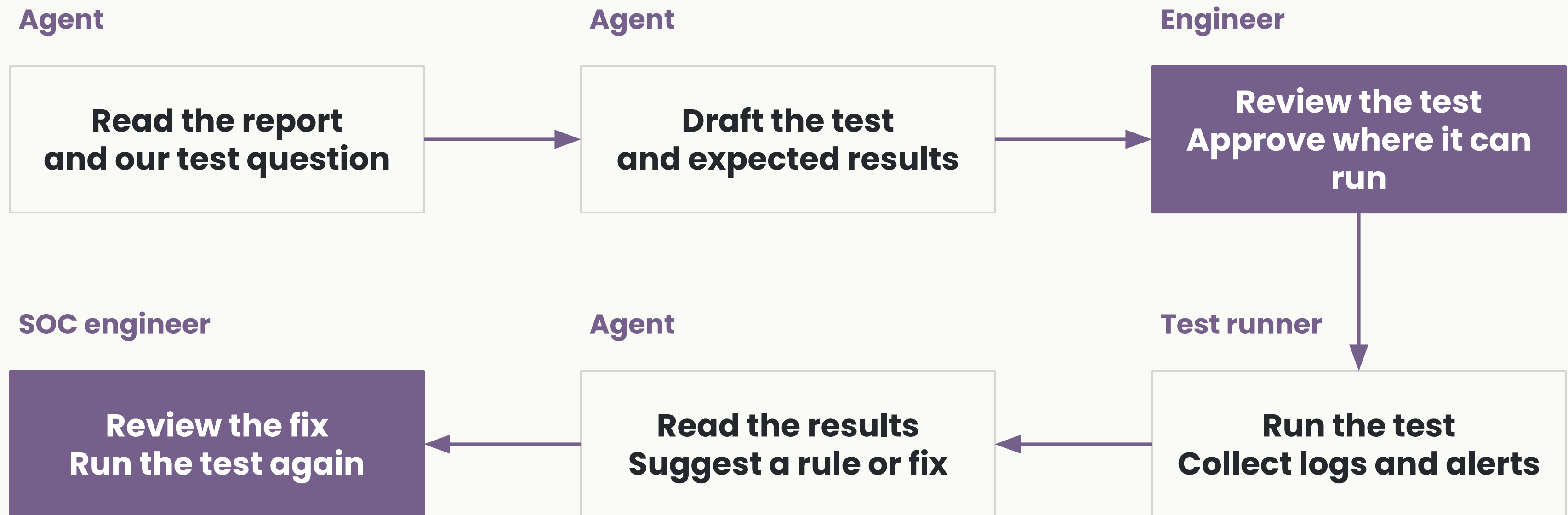
Test other laptop setups, the later DLL load and the SOC response. Assign each task and a review date.

04

Where can AI help?

Where does AI fit?

An agent helps draft the Nimbus test, read its results and propose a fix.



AI pipeline demo

VIDEO PLACEHOLDER

05

How do we run the programme?

What have we tested, and what is still untested?

The task-change test passed in the lab. Other attack steps and services still need testing.

Service	What the tests showed	Next action and owner
Engineering laptops	Changing the task now raises an alert in the lab. Later steps and SOC response remain untested.	Engineer: test other laptop setups. SOC: plan a response test.
Staff accounts	Not tested. We still need to check which stolen sessions could keep access.	Identity owner and CTI: check which sessions can be stolen and how to revoke them.
Customer portal	Not tested. We need audit logs to check for suspicious data exports.	Portal owner: enable logs. Programme lead: set a review date.

What work does the team need to plan for?

The team needs time to build new tests, fix problems and keep existing tests working.

Untested attacks

Which possible attacks matter most to each service, and which have we not tested?

Problems to fix

Assign missing logs, incorrect settings and detection failures to the teams that can fix them.

Tests to maintain

Update tests when software, permissions or log formats change. Rerun tests after fixes.

One work list

Name the person responsible for each item. Agree what to do now and when to revisit the rest.

Who is responsible for each part?

Use existing roles where possible. One person may cover more than one responsibility.

Responsibility	Person or team	What they decide or deliver
Priorities and resources	Programme lead / sponsor	Choose the systems to test, allocate people and time, and set priorities.
Choosing attacks	CTI / assigned analyst	Check the reports and whether the attacks could work on our systems.
Tests and results	Emulation and SOC engineers	Build and maintain tests. Check logs and alerts. Rerun tests after fixes.
Systems and fixes	System owner	Confirm the system setup. Fix system problems and decide how to handle remaining risk.

Can we use this test on another system?

Check the setup before running the same test elsewhere.

Check	Why it matters	What the team does
Permissions and software	A different user or updater may prevent the task change.	Check whether the task can still be changed. Adjust the test setup if needed.
Logs and detections	Another system may collect different fields or use another rule.	Check which logs and alert should appear, then rerun the test.
Running in production	A production test can affect real users and services.	Agree when and where it can run, who approves it and how to undo it.

When should we run the test again?

The test owner checks what changed and which tests it affects.



Check that the expected logs and alert still appear.



Keep old results with the test version and system settings used.



What should the team review?

Example schedule: meet monthly, and after major system or threat changes.

Tests and results

Show what we tested, what worked or failed, and what we have not tested yet.

Fixes and retests

Show which fixes worked when retested, and which problems still need attention.

Updates and time

Which tests need updating? How much time is left for building new ones?

Next work

Agree the next tests and fixes with the SOC and system owners. Name a person and date for each.

06

**How do
we start?**

Your first cycle

Choose one important service and an attack that could work against it.

01

Agree the scope

Choose a service.
Choose an attack
and what should
happen.

02

Assign the work

Name the people.
Allow time for tests
and follow-up.

03

Build and run one test

Run it with the SOC.
Record the result
and what remains
untested.

04

Review the result

Record what
worked.
Assign the fixes.
Choose the next
test.