

cyberwhy

SERVICE GUIDE

Penetration Testing

How an authorized attack simulation works, what you receive, and how to get the most out of it.

01 · OVERVIEW

Why it matters

A penetration test is an authorized, simulated attack. Specialists try to get into your web applications, APIs and networks the way a real attacker would, and then show you how far they got, why it worked and how to close the gap.

A vulnerability scan lists what might be wrong. A penetration test shows what can actually be exploited, and how several small weaknesses combine into a real business impact. That makes it the most direct way to see how well your defenses hold.

Typical triggers

- A new application, API or major release is about to go live
- Customers, partners or tenders ask for independent security testing
- An audit or compliance review is coming
- You added AI features such as chat interfaces, agents or model APIs
- Infrastructure changed: a migration, a merger or a new cloud environment
- After an incident, to confirm the door is really closed

Who benefits

Security lead

Independent evidence of what an attacker can reach.

CIO / IT manager

A ranked list of what to fix first, with effort in mind.

Product and engineering

Reproducible findings and concrete fixes.

Compliance lead

Documented testing for audits and customer questionnaires.

In this guide

01 Overview 2
02 What it is, and what it is not 3
03 How it works, step by step 4
04 What you receive 5
05 In practice 6
06 How to prepare 7
07 Questions, references and terms 8
08 How the services fit together 9

02 · SCOPE

What it is, and what it is not

What it is

+ Authorized and scoped: only agreed targets, times and rules
+ Largely manual and creative, going beyond automated scanners
+ Evidence based: every finding is demonstrated, not just claimed
+ A snapshot: it reflects the systems on the days they were tested

What it is not

– Not a vulnerability scan, which is broader and more automated
– Not a guarantee that no weakness exists
– Not an attack on live data: destructive actions are excluded by the rules of engagement
– Not a certification or an audit

What can be tested

Web applications

Login and sessions, access control, input handling, file uploads and business logic, tested against well-known weakness categories such as those in the OWASP Top 10.

APIs

Authentication, authorization per object and per function, excessive data exposure, rate limits and abuse of business flows.

Networks

External view: what the internet can see and reach. Internal view: what an attacker with a first foothold could reach next.

AI-enabled features

Where in scope: how prompts, tools and data connections can be abused, for example through prompt injection or over-broad permissions.

How much the testers know

Black box

No prior information, like an outsider. Realistic, but time goes into discovery.

Grey box

Limited access, such as a test account. A good balance of realism and depth.

White box

Documentation or architecture shared. The most depth for the time spent.

03 · PROCESS

How it works, step by step

A typical engagement, from the first conversation to clear next steps. Exact steps and timing are agreed with you when the work is scoped.

01 Discuss

Understand why you are testing and what worries you most.

- Talk through goals: release readiness, a customer request, compliance or a concern
- Map the systems involved and who owns them
- Learn about earlier incidents, known weak spots and business-critical processes

YOU PROVIDE
Your goals and concerns
An architecture overview, if you have one
A technical contact

YOU GET
A clear brief

02 Scope

Agree exactly what is tested, when and under which rules.

- Define targets: URLs, API endpoints, address ranges, environments
- Choose the knowledge level: black, grey or white box
- Set the rules of engagement: time windows, excluded actions, emergency contacts
- Agree how test accounts and data are handled

YOU PROVIDE
Targets and constraints
Written authorization from the system owner
Test accounts for each role, if grey box

YOU GET
An agreed scope and rules of engagement

03 Test

Find out what an attacker could really do.

- Reconnaissance: what is exposed and how it is built
- Search for weaknesses with tools and by hand
- Exploit carefully within the rules, chain weaknesses together and record evidence
- Critical issues can be flagged during the test, as agreed in the rules of engagement

YOU PROVIDE
Access to the agreed targets
Availability during the agreed time window

YOU GET
Verified findings with evidence

04 Report

Turn technical results into decisions and fixes.

- Verify findings and remove duplicates
- Rate each finding by risk, considering likelihood and impact in your context
- Write an executive summary for decision makers and technical detail for engineers
- Add a tailored remediation plan with expert recommendations

YOU PROVIDE
Nothing further

YOU GET
A detailed report and remediation plan

05 Review

Make sure the results lead to action.

- Walk through the results with your team
- Clarify questions and discuss practical fixes
- Agree next steps, for example how fixes will be checked

YOU PROVIDE
Your questions and priorities

YOU GET
Clear next steps

04 · DELIVERABLES

What you receive

The report is written for two audiences: leadership who must decide, and engineers who must fix. Each finding stands on its own, so it can be handed to the team that owns it.

01
Executive summary
Overall picture, the most important findings and what needs attention first, in plain language.

02
Scope and method
What was tested, when, with which knowledge level and under which rules.

03
Findings with evidence
What was found, where, and how it was demonstrated, with steps to reproduce.

04
Risk rating
Each finding ranked so the order of work is clear.

05
Remediation plan
Tailored, practical recommendations for each finding and for the root causes behind them.

06
Walk-through
A conversation to go through the results and agree next steps.

How findings are rated

Critical	9.0 to 10.0	Fix immediately. Exploitation is likely and the impact is severe.
High	7.0 to 8.9	Fix soon. Serious impact or easy to exploit.
Medium	4.0 to 6.9	Plan the fix. Needs specific conditions or has limited impact.
Low	0.1 to 3.9	Fix as part of regular work.
Info	None	Observation or hardening advice, no direct risk.

Scores follow the common CVSS scale. The rating also takes into account how easy the weakness is to exploit and what it would mean in your environment, so a lower score can still rank high.

05 · IN PRACTICE

Seeing it at work

After the report: from findings to fixed

1 Triage
Go through the findings with the owners of each system and confirm priorities.

2 Fix
Start with critical and high findings, and with quick wins that remove several issues at once.

3 Verify
Check that each fix works, so a finding is closed with evidence rather than assumption.

4 Learn
Feed root causes back into design, code review and testing, so the same flaw does not return.

EXAMPLE OF A FINDING (FORMAT ONLY)

Missing authorization check on an invoice API

High

AFFECTED	GET /api/invoices/{id}
DESCRIPTION	The API returns an invoice for any valid ID. It does not check whether the invoice belongs to the logged-in customer.
IMPACT	A customer can read other customers' invoices by changing the ID.
EVIDENCE	Request and response pairs for two test accounts, with steps to reproduce.
FIX	Check ownership on the server for every request, and use non-guessable identifiers.

FOR AI PROJECTS

Testing projects that use AI

AI features add new paths into your systems: chat inputs that steer a model, tools and plugins an agent may call, and data sources a model can read. A test looks at what a hostile input could make the feature reveal or do, and whether permissions are limited to what the feature needs.

06 · PREPARATION

How to prepare

A little preparation makes the work faster, safer and more useful. Here is what helps most.

Before we start

- Name an owner who can authorize testing, and a technical contact
- List systems and environments, and anything that is off limits
- Decide whether to test staging or production, and when
- Prepare test accounts for each user role
- Tell hosting providers and internal teams that testing is coming
- Avoid risky changes during the test window

Typical ground rules

- + Testing starts only after written authorization
- + Only the agreed targets are touched
- + Data is handled confidentially and kept to the minimum needed
- + Destructive actions and denial of service are excluded unless explicitly agreed

Common misconceptions

~~We have a firewall, so we are covered.~~	**In practice:** Most attacks on web applications and APIs arrive through allowed traffic. A firewall does not understand your application logic.
~~A clean scan means a clean test.~~	**In practice:** Scanners find known patterns. Testers find logic flaws and chains of small weaknesses.
~~We tested last year.~~	**In practice:** Every release changes the attack surface. Testing is most useful after major changes.

07 · REFERENCE

Questions, references and terms

Frequently asked questions

How long does a test take?
It depends on the size and depth of the scope. The time is estimated and agreed while scoping.

Can testing disturb our systems?
Tests are designed to avoid disruption. Risky actions are discussed beforehand, and a staging environment is often the safer place for the most intrusive checks.

How often should we test?
After major changes and on a regular cycle that fits your risk. Many standards and customers expect regular independent testing.

Do you need our source code?
Not necessarily. Black and grey box tests work without it. Sharing code or architecture documents (white box) gives more depth for the time.

Can you test production?
Yes, if agreed, with care about timing, accounts and excluded actions.

What happens after the report?
You fix the findings in priority order. We walk through the results with you and agree how fixes will be checked.

Reference frameworks and sources

OWASP Web Security Testing Guide	A widely used method for testing web applications.
OWASP Top 10 and API Security Top 10	Common weakness categories for web applications and APIs.
PTES and NIST SP 800-115	Reference descriptions of how a technical security test is planned and run.
MITRE ATT&CK	A catalog of real attacker techniques.
CVSS (FIRST)	The common scale for scoring severity.

Terms in plain language

Attack surface. Everything an attacker can reach: pages, APIs, ports, accounts.

Exploit. Using a weakness to gain access or an effect that was not intended.

Rules of engagement. The written agreement on what may be done, when and by whom.

Black, grey, white box. How much the testers know in advance.

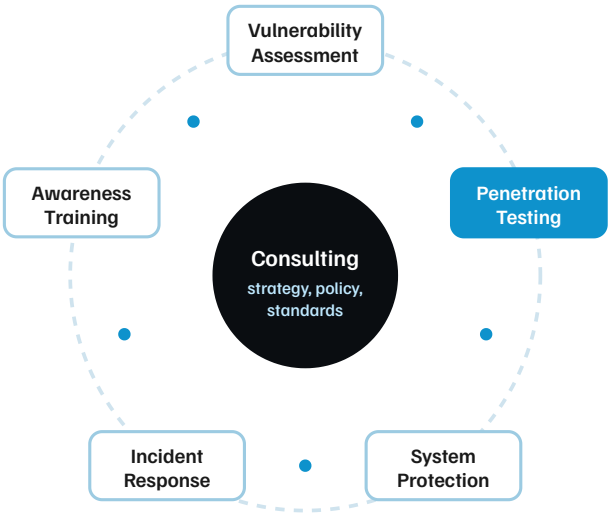
Remediation. Fixing a weakness or reducing its risk.

Foothold. A first point of access from which an attacker moves on.

08 · NEXT STEPS

How the services fit together

Security works best as a cycle: find weaknesses, test them, protect, prepare to respond and keep people alert, all guided by a strategy. You can start with any one of them.



Vulnerability Assessment
Finds known weaknesses across your systems and ranks them.

Penetration Testing (this guide)
Shows what an attacker can actually exploit.

System Protection
Closes gaps with layered, maintained defenses.

Incident Response and Forensics
Acts when something gets through, and learns from it.

Security Awareness Training
Prepares the people who meet threats first.

Cyber Security Consulting
Sets strategy, policy and standards that tie it together.

Talk to a specialist about penetration testing.

Tell us what you are working on and what you need to protect.

EMAIL
info@cyberwhy.com

WEB
cyberwhy.com/services/penetration-testing/

This guide describes how work of this kind typically runs. Scope, methods, timing and deliverables are agreed with you for your situation, and nothing in it is a guarantee. References to standards and frameworks are for orientation and do not mean certification. © CyberWhy. All rights reserved.