

Vulnerability Assessment of Networked Systems

JOÃO PAULO BARRACA



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

1



universidade
de aveiro

Welcome to the vulnerability assessment part of AEV. This lecture is deliberately not about writing exploits; it is about the professional process that surrounds them: how you decide what to test, how you bound the test, how you record what you find and how you make it actionable. Most of your grade comes from practical assignments and CTF-style work, so treat every slide here as the discipline that turns a talented hacker into someone an organisation can trust.

Everything you have already touched in the lab - nmap, Wireshark, Ghidra, ZAP, OpenVAS, metasploit, the Linux VMs and Podman images - is a tool inside this process. The course keeps coming back to the same idea: document everything. That instruction is not bureaucracy; it is what separates a reproducible finding from an anecdote you cannot defend.

Keep in mind the people in the room with you. You will do a group CVE discussion assignment and a software-vulnerabilities project, and you will be graded partly on how well you communicate a technical finding. The skills on these slides - scoping, evidence, risk language - are exactly the skills that assignment is checking.

As you go through, ask one question of every slide: what would I actually do next Monday in a real engagement? The answer is rarely 'run a tool'; it is usually 'agree scope, record evidence, write it down, get authorisation in writing.'

Vulnerability Research

The process of finding and analyzing new vulnerabilities

- Through direct experimentation
- Through analysis of the architecture, code or system behavior

Important to many different stakeholders:

- Product owners: prioritize actions/budget on the product lifecycle
- Developers: understand what created the vuln, how it can be avoided
- Administrators: assess impact and deploy defense/recovery measures
- Vuln. Researchers: to pivot to new vulnerabilities

Takeaway *Research output is a decision - what to fix, what to defend, what to budget for.*



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

2



Vulnerability research is the upstream activity that feeds everything else: finding and analysing vulnerabilities that are not yet public, either by experimenting directly or by reasoning about architecture, code and behaviour. This is different from an assessment, which mostly looks for weaknesses that are already known. Research is where a previously unknown bug becomes a finding.

The slide lists who cares, and the list matters because it tells you the economics. Product owners care because a vulnerability discovered after a release forces a costly lifecycle change; developers care because understanding the root cause prevents the same class of bug recurring; administrators care because they must assess blast radius and deploy defences; and other researchers care because one finding often reveals the next. Your CVE discussion assignment is a small version of this: you will take a public vulnerability and walk through what it means for each of these stakeholders.

Think about how research actually happens with the tools you have. Direct experimentation is fuzzing an input parser, sending malformed requests to a service you control, or throwing unusual arguments at a binary in Ghidra or gdb. Analysis is reading source and reasoning about state - where does the developer trust an assumption that an attacker can break?

Vulnerability Assessment - Objective

Process to analyze, evaluate and review entities (software applications, devices, networks, systems)

Identify and categorise issues that may be exploited, or that constitute a risk to the normal operation of the entity

Takeaway *Assessment is analysis of your own entities against risk - neither compliance nor exploitation.*



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

3



universidade
de aveiro

An assessment is a process: analyse, evaluate and review entities - software applications, devices, networks, systems - and then identify and categorise issues that could be exploited or that threaten normal operation. Keep the word categorise: a good assessment does not just dump a list of problems; it classifies them so that the organisation can prioritise. That classification is where your report starts to be useful.

Note that the objective includes issues that constitute a risk to normal operation, not only issues that an attacker can exploit. A misconfiguration that causes an outage, a service with no redundancy, an undocumented dependency - these are assessment findings too, even though no one 'exploits' them in the offensive sense.

This framing also explains why an assessment is broader than a penetration test: it is not only about what an attacker could reach, but about the health and posture of the entity. In your assignments, when you assess one of the lab targets, do not stop at 'I could log in with the default password'. Also record what else was wrong - exposed services, missing hardening, information that leaked.

Keep this objective sentence as a checklist when you write your report. For every finding ask: did I identify the issue, did I categorise it, and did I show why it threatens operation or exploitation? If a finding fails any of those, it is not ready to hand to a client.

Definitions: the primitives

Asset: anything of value to protect - data, systems, people, reputation, availability

Vulnerability: a weakness of an asset or control that can be exploited (code flaw, misconfiguration, missing control)

Threat: a potential cause of harm. Threat agent: who might do it, with what motive, opportunity and capability

Attack: use of a technique to exploit a vulnerability.

Exploit: the code or procedure that performs it

Exposure: an unmitigated vulnerability that a threat can reach (asset reachable from that vantage point)



These definitions are the primitives of the whole discipline, and the single most common student mistake is to use them interchangeably. An asset is anything of value you protect - data, systems, people, reputation, availability. A vulnerability is a weakness of an asset or a control that can be exploited: a code flaw, a misconfiguration, a missing control. The distinction between asset and vulnerability is the first thing to get straight before you read any report.

Threat is the potential cause of harm, and a threat agent is the one who might do it - with a motive, an opportunity and a capability. That last part is practical: a threat is only real if an actor has all three. The classic example is a public Wi-Fi network: the motive (steal credentials) and capability (sniffing tools) may be present, but if your laptop only talks to the VPN, the opportunity is largely removed. Assessments are about closing the opportunity side.

An attack is the use of a technique to exploit a vulnerability; the exploit is the code or procedure that performs it. This is not a semantic distinction. In your midterm challenges you will write attacks; the exploit is the deliverable. Exposure is the point that makes everything concrete: an unmitigated vulnerability that a threat can reach - your asset must be reachable from that vantage point for exposure to exist.

Use these words precisely in your reports and in your CVE discussion. A precise vocabulary lets two analysts disagree productively ('is this an exposure or a vulnerability?') instead of talking past each other. Precision here is not pedantry; it is the difference between a finding that gets fixed and one that gets ignored.

Definitions: the primitives

Control / countermeasure: mechanism that prevents, detects or recovers; compensating control: an alternative that limits exposure

Severity: intrinsic property of the vulnerability.

Likelihood: probability of exploitation.

Impact: consequence

Risk = $f(\text{threat, likelihood, asset value, impact})$

Vulnerability severity is NOT risk: two systems with the same CVE can carry opposite risk

- Controls: Filter packets, connections

Takeaway *If asset, threat, exposure and impact are not separate ideas, risk collapses into a CVSS score.*



A control or countermeasure is a mechanism that prevents, detects or recovers; a compensating control is an alternative that limits exposure when the ideal control is impossible. The compensating idea is important in real engagements because clients rarely have the budget to fix everything. If you cannot patch, you can isolate; if you cannot isolate, you can monitor. Each is a control, just weaker.

Severity is an intrinsic property of the vulnerability - how bad it is in isolation. Likelihood is the probability of exploitation in your environment, and impact is the consequence. Risk is a function of threat, likelihood, asset value and impact. The slide's formula is deliberately informal: risk is a judgement that combines several inputs, not a number you can compute with one arithmetic operator.

The sentence that will be on the exam is the last one: vulnerability severity is NOT risk. Two systems with the same CVE can carry opposite risk because one is internet-facing with customer data behind it and the other sits behind a firewall in an isolated VLAN. Severity tells you how bad the bug is; risk tells you how bad it is for you.

The example 'controls: filter packets, connections' is there to make it concrete: the same remote-code-execution bug has almost no risk on a system that is unreachable, because the control removes the exposure even though the vulnerability remains. When you write a report, always separate the score (severity) from the risk statement, and justify the risk with exposure and asset value.

Assessment vs Audit

Audit: determines compliance with a standard

- Scope: A given standard and its control points

Assessment: determines how good/bad something is

- Scope: may be broad. Driven by risk, compliance, contractual requirements
- Aims to help improve systems
- Done before the audit, to identify any loopholes
- Done after the audit, to measure how effective the audit was

SANS Institute, "Scoping Security Assessments: A Project Management Approach"
(white paper 33673), sans.org/white-papers/33673

Takeaway *Audit asks 'were the rules followed?'. Assessment asks 'are we actually exposed?'. You need both.*



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

6



universidade
de aveiro

Assessment and audit are easily confused, and the slide gives you the clean distinction: an audit determines compliance with a standard, while an assessment determines how good or bad something is. An audit has a fixed yardstick - the standard and its control points - and the result is pass/fail per control. An assessment has no fixed yardstick; its scope can be broad and is driven by risk, compliance or contractual requirements.

The deeper point is that an assessment is aimed at improvement, not at judgement. It is done before an audit to find loopholes that would fail the audit, and after an audit to measure how effective the audit was. That makes assessment the more flexible and the more forward-looking activity, which is why this course is called assessment rather than audit.

For your group project, think about which of the two you are actually doing. If you are checking the lab target against a checklist (CIS benchmarks, OWASP top ten) you are auditing; if you are trying to find what is wrong and why, you are assessing. Most real work is a mix, and being honest about which hat you are wearing changes how you write the conclusions.

The slide cites SANS white paper 33673 on scoping security assessments. Scoping - deciding what is in and what is out - is the most neglected and most consequential part of the process, and you will see it again in two slides. Read the white paper if you want the project-management framing; the lecture gives you the technical framing.

Assessment vs Penetration Test

A penetration test focuses on infrastructure and systems, from an outside-in and inside-out perspective

- Outside: outside the domain (another domain or the Internet)
- Inside: in the domain

Tests the capability of entering a domain and its impact

- How an attacker entered (which flaws or bugs were used)
- How/if an attacker moved laterally
- What other systems may it have reached
- What data/systems were impacted
- Was data exfiltrated?

Takeaway *A pentest proves one path exists; an assessment counts how many paths you have - and their cost.*



A penetration test is a specific kind of exercise, focused on infrastructure and systems, and viewed from outside-in and inside-out. Outside means from outside the domain - another domain or the Internet; inside means from within the domain. The test is really a rehearsal of an attacker's journey: how did they enter, which flaws or bugs were used, did they move laterally, what other systems did they reach, and what data or systems were impacted.

The list of questions on the slide is essentially a list of what a defender most wants to know after a breach. A penetration test answers those questions proactively, before the real attacker does. That is its value and its purpose: it converts 'we have no idea how they got in' into a rehearsed, documented answer.

Contrast this with an assessment: an assessment identifies and categorises vulnerabilities across a broad scope, while a penetration test goes deep on a specific path and demonstrates impact. The assessment finds that a service is exposed; the pen test proves what an attacker could actually do with that exposure. Both are needed, and they answer different questions.

When you see the phrase 'penetration test' in a job description or a client request, notice the scope it implies: infrastructure, systems, entry and lateral movement, data impact, exfiltration. If the client actually wants a broad posture review, that is an assessment, not a pen test, and the contract should say so - the words determine the legal authorisation you have.

Why assess? (1) The organisational landscape

An essential process in current organizations, products and systems

- Two distinct views: Internal and External

Current organizational landscape is complex

- Heterogeneous computing environment
 - Servers, desktops, laptops, BYOD...
- Multiple applications
 - From multiple vendors
 - Developed over time, using different tools, languages and stacks
- Rely on communication networks
 - Not all confined (e.g. Wi-Fi)
- Rely on external services and actors

Important to understand what the risks are, what to address, and what processes should be in place

Takeaway *You cannot assess what you have not enumerated: asset inventory is the first assessment task.*



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

8



The first reason to assess is simply the reality of the organisation you work in. No modern organisation is homogeneous: heterogeneous computing environments with servers, desktops, laptops and BYOD; applications from many vendors, built over time with different tools, languages and stacks; communication networks that are not all confined (Wi-Fi reaches beyond the building); and reliance on external services and actors. Every one of those is a surface an attacker can reach.

The two distinct views - internal and external - matter because they see different things. An external view sees the public perimeter: DNS, web servers, mail, VPN. An internal view sees what is behind the firewall: admin panels, databases, printers, internal APIs. An assessment that only takes one view is incomplete, because real attacks usually start outside and continue inside.

Complexity is the enemy of security, and this slide is the motivation for the whole course. When you cannot enumerate what you have, you cannot protect it. That is why later slides push an inventory-first mindset (ASM/EASM) and why standards like CPE exist: you cannot assess what you have not named.

When an organisation asks 'why should we spend money on assessments?', the answer is the landscape you just described: the risk is not a single bug, it is the interaction of a heterogeneous environment you cannot fully predict. Assessments exist to make the unpredictable visible.

Why assess? (2) Defences alone are not enough

Standard defensive measures are not enough

- They help create and operate software with greater security
- They are also limited to the mindset of the developers/ops

Defensive technologies are limited in capabilities

- **Firewall:** Filter packets, connections
 - Mostly deployed as perimeter control devices; visibility inside the network depends on deployment (micro-segmentation)
 - Inspects packets in the clear, or publicly visible metadata (ports, IP addresses, protocol), but struggles with encrypted traffic (TLS)
- **WAF:** Filter HTTP requests
 - Matches profiles of known attacks (deny list) or of allowed requests (allow list), but can be circumvented
- **IDS: Network- and Host-based Intrusion Detection Systems monitor network traffic or OS changes**
 - Matches profiles of known attacks, but can be circumvented
 - An IDS detects and alerts; it does not stop what already happened (blocking requires an inline IPS)

Takeaway *Firewalls, WAF and IDS answer the questions you expected; assessment asks what they will miss.*



The second reason to assess is that defensive measures alone are not enough. Firewalls, WAFs and IDS are limited by their design and by the mindset of the developers and operators who configure them. This slide is the honest engineering case for offence-informed defence: you can only defend against what you can imagine, and assessment broadens the imagination.

Firewalls filter packets and connections. They are mostly perimeter devices, so visibility inside the network depends on deployment - micro-segmentation gives you internal visibility, a flat network gives you none. They inspect packets in the clear and public metadata like ports, IPs and protocols, but they struggle with encrypted TLS traffic, because they simply cannot see inside it. That is not a failure; it is a physics limit.

A WAF filters HTTP requests by matching profiles of known attacks (deny list) or of allowed requests (allow list), but it can be circumvented - encodings, logic bugs, and requests that simply do not look like the profile. An IDS, network- or host-based, monitors traffic or OS changes and matches known-attack profiles, but it can be circumvented too, and critically it detects and alerts - it does not stop what already happened. Blocking requires an inline IPS.

The lesson to carry into the labs: when you test with ZAP or a fuzzer, you are often doing nothing more than finding requests that pass the WAF. When you assess a host with OpenVAS or manual checks, you are finding what the IDS does not flag. Defenders build profiles; attackers look for what is outside the profile. Assessment is how you find your own blind spots before they become someone else's entry point.

Scope

The definition of what systems/software/endpoints/approaches are considered

The most important component of setting up a successful security assessment

Too broad: Mimics a powerful attacker

- Too expensive
- May lead to a never-ending assessment
- May lead to lack of depth (missing vulns)

Too narrow: mimics a focused attack

- Cheap, fast, repeatable
- May miss easily found issues
 - Like hardening the bulletproof front door while leaving a glass window in the wall

Takeaway *Scope is a risk decision, not a technical one: too wide never ends, too narrow misleads.*



Scope is the definition of what systems, software, endpoints and approaches are considered, and the slide calls it the most important component of a successful assessment. Everything else - methods, tools, findings, legal authorisation - depends on a clear scope, because you cannot authorise what you have not defined.

Too broad a scope mimics a powerful attacker but is expensive, may become a never-ending assessment, and paradoxically may lead to lack of depth - you spread thin and miss vulnerabilities that a focused test would find. Too narrow a scope mimics a focused attack: cheap, fast and repeatable, but it may miss easily found issues. The slide's image is the memorable one: hardening the bulletproof front door while leaving a glass window in the wall.

The tension is real and there is no perfect answer, only a negotiated one. In your group assignment, scope is the single biggest lever on your effort and your grade: scope too broad and you cannot finish; scope too narrow and you miss the interesting findings. Practise writing a scope sentence that names the systems, the endpoints, the approaches and what is explicitly excluded.

Professional scope is written, not implied. It names what is in, what is out, and why. It also names the time window, the credentials provided, and the techniques allowed - which is the Rules of Engagement you will see later. A scope you cannot defend in writing is a scope that will come back to haunt you when something breaks.

Limitations

Assessment is only valid at a given point in time

- Other vulnerabilities may exist before or after the assessment

The researcher must be aware of the latest vulnerabilities

- Risk of false negatives

Limited to the scope, location and methods used

- Different domains may have different firewall access rules or security policies

Tests specific entities, not the overall security controls

- A vulnerability may exist, but the security controls may limit/block its exploitation

Takeaway *Read the small print first: as of today, within this scope, by these methods.*



Limitations are the honesty of the discipline, and the first one is temporal: an assessment is only valid at a given point in time. The network, the software, the configuration and the attacker all change, so vulnerabilities may exist before or after your assessment window. A report is a snapshot, not a promise.

The second limitation is that the researcher must be aware of the latest vulnerabilities, or risk false negatives - missing something real because you did not know about it or did not think to look. This is why the course pushes you toward current sources (NVD, vendor advisories, KEV) and why your CVE discussion assignment keeps you in touch with recent disclosures.

Scope, location and methods bound everything. Different domains may have different firewall rules or security policies, so a finding in one domain does not transfer to another. And you test specific entities, not the overall security controls: a vulnerability may exist, but a compensating control may limit or block its exploitation. The reverse is also true - a control that works today may be removed tomorrow.

Read these limitations as the structure of your report's 'assurance' section. A professional report states explicitly what was and was not covered: which assets, which time window, which methods, which domains. When you read another team's report, look for that section first; a report without stated limitations is a report you should not fully trust.

Before you probe: legal and ethical constraints (1/2)

Written authorisation before any testing: who, what systems, what techniques, what time window

Rules of Engagement (RoE): depth of testing, credentials provided, out-of-scope systems, escalation contact

Scope boundaries are physical and legal: a wireless signal leaves the building; cloud/SaaS/CDN may forbid testing

- Third-party services often require prior approval (pen-test gates, provider terms)

Data handling: minimise collection, no copying of personal data, secure storage and destruction (GDPR)



This slide is about the paperwork that makes testing legal, and it is not optional. Before any testing you need written authorisation that states who, what systems, what techniques and what time window. That single document converts your activity from a criminal act into a contracted service, which is the difference that matters under Portuguese and EU law.

Rules of Engagement (RoE) refine the authorisation: depth of testing (are you allowed to exploit, or only to prove existence?), credentials provided, out-of-scope systems, and the escalation contact. The escalation contact is the person you call the moment something goes wrong - a service crashes, you find production data, you realise you are on the wrong system. You should know their number before you start, not after.

Scope boundaries are physical and legal at the same time. A wireless signal leaves the building, so a Wi-Fi test may touch neighbours' devices. Cloud, SaaS and CDN services often forbid testing in their terms, and third parties frequently require prior approval or a pen-test gate. Even a 'public' service may be legally protected, so you cannot assume consent from visibility.

Data handling is a separate obligation, largely GDPR-driven: minimise collection, never copy personal data, store findings securely and destroy them when done. In the lab this seems theoretical, but in a real engagement the data you collect - credentials, logs, personal records - is the most sensitive thing you touch. Handle it as if it were the production data of your own employer.

Before you probe: legal and ethical constraints (2/2)

Portuguese / EU context: unauthorised access is a criminal offence; act on contract, not on good intentions

When testing disrupts production: stop, notify immediately, agree a rollback path

Disclosure: how and when findings become known

- co-ordinate with the vendor;
- no public naming during remediation

If you find data or credentials accidentally: stop reading, report at once, document the incident

Takeaway *Authorisation and Rules of Engagement precede the first probe - an illegal finding is an incident.*



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

13



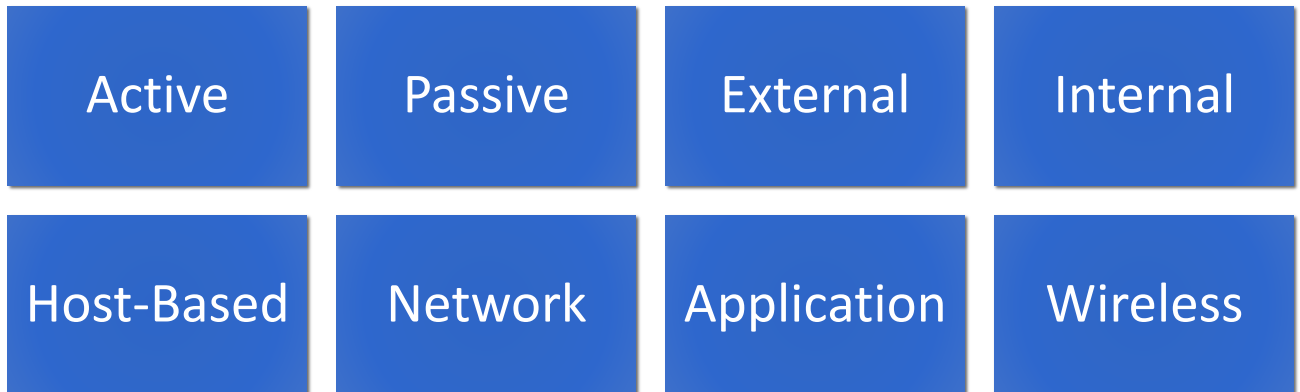
The Portuguese and EU legal framing is worth stating plainly: unauthorised access to a computer system is a criminal offence, so you act on a contract, never on good intentions. Good intentions are not a defence. Every lab exercise in this course is lawful because the systems are yours; a real engagement is lawful only because of the signed authorisation from the previous slide.

When testing disrupts production - a service crashes, a WAF blocks a legitimate user, a scan saturates a link - the rule is to stop, notify immediately, and agree a rollback path. Do not try to fix it silently; that is how small incidents become large ones. The escalation contact from the RoE is the person you call first.

Disclosure is about how and when findings become known. The professional default is coordinated disclosure: report to the vendor, agree a remediation window, and do not name the issue publicly while it is being fixed. Public naming during remediation helps attackers and hurts everyone. In your CVE discussion assignment you will see exactly this tension play out in real disclosures.

The last rule covers the accidental find: if you stumble on data or credentials you were not authorised to touch - someone's password in a log, a database you were not scoped to test - you stop reading, report it at once, and document the incident. Reading further converts an accident into an intentional breach. This is the behaviour a professional is judged on.

Types of assessment (company scale)



Takeaway *These types are not alternatives: a working programme runs several of them continuously.*



This is a picture of the eight assessment types at company scale: active, passive, external, internal, host-based, network, application, wireless. Before you read the individual type slides, notice that these eight are not eight separate species - they are eight combinations of a few underlying dimensions. That is the key to the whole taxonomy, and it is the point of the next slide.

Think of the eight as a menu you order from, not a classification you must fit into. A real engagement rarely is purely one type. An internal network assessment is internal plus network; an external application test is external plus application plus active. The useful question is not 'which type is this?' but 'which axes does it use?'

For your own work, map the eight types onto the lab. nmap and OpenVAS are active network scans; Wireshark captures are passive; testing the public-facing web app with ZAP is external application; testing the internal services from the VM is internal. You have already been doing most of these types without naming them - the taxonomy just gives you the vocabulary to describe and scope them.

Keep the picture in mind as a completeness check. When you plan an assessment, run through the eight and ask which you are including and which you are deliberately excluding, and record why. That is the difference between a thorough plan and a lucky scan.

Four independent axes (why the 'types' overlap)

The eight types mix three different dimensions.

Describe an engagement as a coordinate instead:

- **Method:** active (probe the target) vs passive (observe what it says)
- **Vantage point:** external (no access) vs internal (some access, privileged or not)
- **Target scope:** network, host, wireless, application
- **Knowledge:** black box, gray box, white box (what the tester is given)
 - Example: "active, external, network-based, gray-box, credentialed for 3 host classes"

Each axis trades coverage, realism, disruption risk and credentials required

Benefit: no more arguing whether a test is 'host-based or internal' - it is both

Takeaway *Name the coordinate, not the type: active, external, network, gray-box. The argument ends there.*



Here is the conceptual core of the taxonomy: the eight types are not eight categories but combinations of four independent axes. The first is method - active (you probe the target) versus passive (you observe what it says). The second is vantage point - external (no access) versus internal (some access, privileged or not). The third is target scope - network, host, wireless, application. The fourth is knowledge - black box, gray box, white box (what the tester is given).

The slide gives a model description: 'active, external, network-based, gray-box, credentialed for 3 host classes.' That one sentence is a complete engagement in five words of axes. This is the notation you should use when you write your scope, because it removes the ambiguity of 'is this host-based or internal?' - the answer is that it is both, along different axes.

Each axis is a trade-off between coverage, realism, disruption risk and the credentials required. Active gives you real evidence but risks disruption; passive is safe but blind to encrypted payloads and silent hosts. External is realistic for outside attackers but misses what only insiders see. White box gives coverage but over-states what an attacker can do; black box is realistic but low-coverage.

The benefit the slide claims is real: with axes, you stop arguing about taxonomy and start describing engagements precisely. When you present your group project, describe it in the axes notation rather than with a single label. It will make your scope defensible and your findings reproducible.

Type: Active

Runs software to discover network hosts

- Send probes
- Checks information repositories

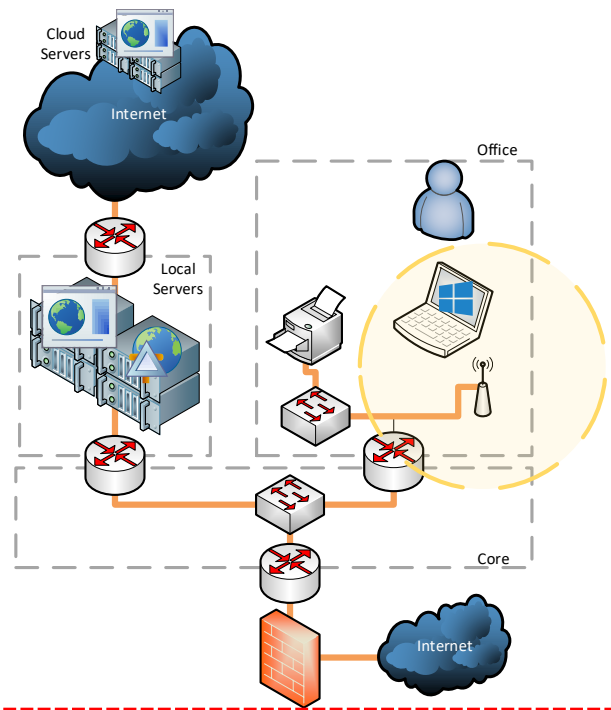
Runs tools to actively test software/systems

- Sends crafted arguments, payloads, packets
- Simulates attack techniques
- MiTM, DoS, etc...

May disrupt systems! Run in agreed windows, with rate limits, exclusions and a rollback plan

- Detection of vulnerability may have impact

Takeaway *Active probing trades certainty for production risk - schedule it like a change, with rollback.*



An active assessment is the kind you are most familiar with from the lab because it is what nmap and OpenVAS do. It runs software to discover network hosts - sending probes and checking information repositories - and then runs tools to actively test software and systems, sending crafted arguments, payloads and packets, simulating attack techniques, including man-in-the-middle and denial of service.

Because you are actively sending traffic, an active assessment can disrupt systems. The slide is blunt: run it in agreed windows, with rate limits, exclusions and a rollback plan. Even a benign scan can trip an IDS, lock an account after too many attempts, or saturate a link. Detection of a vulnerability may itself have impact - an active probe can crash the very service you are testing.

In practice, active testing is where you must be most careful about your own tooling. A rate-limit that looks polite to you may look like an attack to the target's monitoring. The discipline is to record what you sent, when, and what you observed - that log is both your evidence and your defence if the target operator asks what you were doing.

The distinction from passive is the trade-off you will see in your assignments: active gives you direct evidence that a service is present and a version is vulnerable, but it only tells you about what answers your probes. Hosts that are down, filtered, or silent to your scan are invisible to you - which is exactly the limitation the next slide explains.

Type: Passive

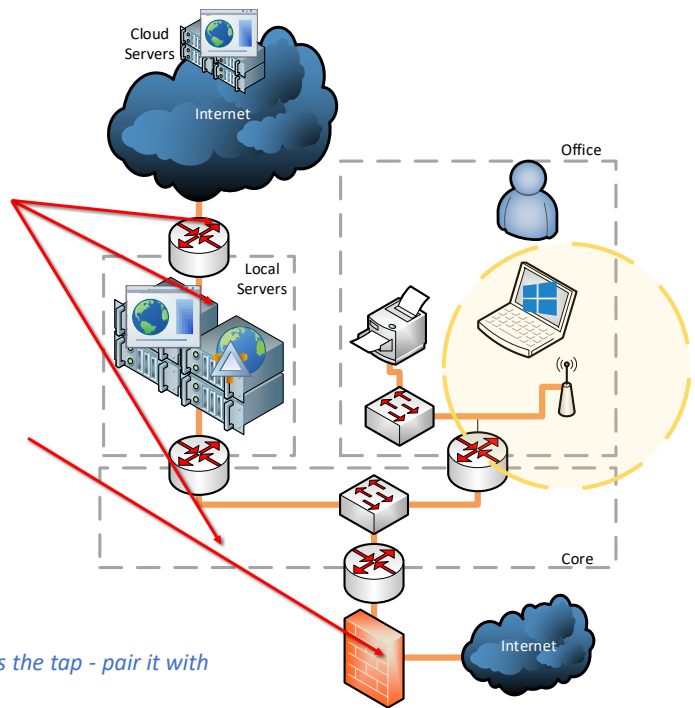
Runs software to eavesdrop on traffic

Observes logs and dumps

- Network logs
- Service/application logs
- Host logs
- May be run for a long time in production

Minimal impact - but blind to what does not cross the tap (silent/patched hosts, encrypted payloads)

Takeaway *Passive is safe but silent about what never crosses the tap - pair it with credentialed checks.*



Passive assessment flips the active model: instead of sending traffic, you eavesdrop on traffic and observe logs and dumps - network logs, service and application logs, host logs. Because you send nothing, you can run it for a long time in production without disrupting anything, and you get a picture of what actually happens, not what a probe makes happen.

The big advantage is zero impact and the ability to run continuously, which is why passive techniques are the backbone of monitoring and why later slides make monitoring part of the life cycle. In your lab, Wireshark captures and service log analysis are the passive side: you learn what the system really does when no one is deliberately probing it.

The limitation is the counterpart: passive is blind to what does not cross the tap. Silent or patched hosts that generate no traffic are invisible; encrypted payloads hide their contents; and a service that is never contacted never shows a vulnerability. You can only observe what is actually transmitted, which means the absence of an observation is not evidence of absence of a problem.

The practical lesson is to combine the two. Passive observation tells you what to prioritise - which services are really used, what credentials actually appear, what legacy protocol is still on the wire - and active testing then verifies hypotheses on that shortlist. That ordering is how a professional spends their effort.

Type: External

Focus on the public exposition

- External attackers

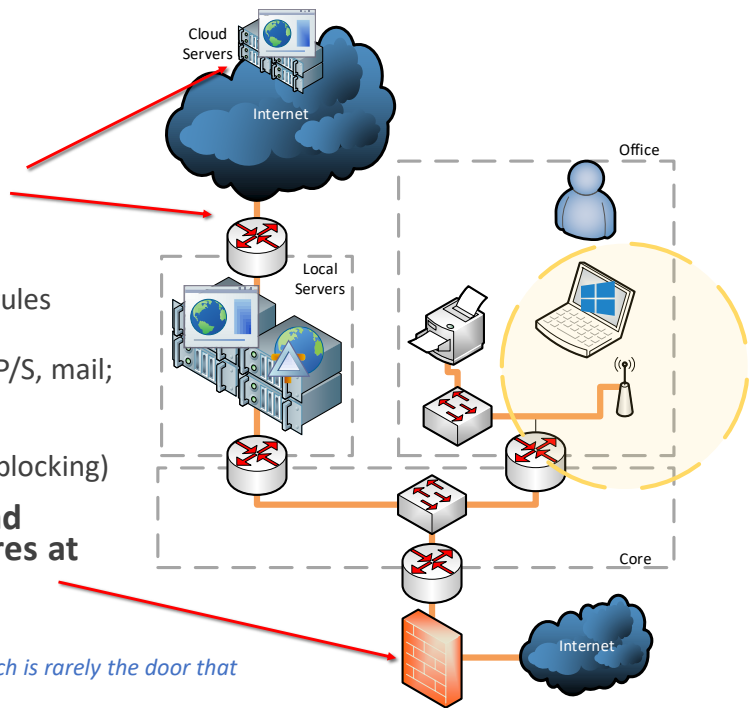
Targets:

- Publicly exposed routers and firewall rules
- Publicly available IP Ports
- Public services and records (DNS, HTTP/S, mail; zone transfers, SPF/DMARC)
- Information exposed to the public
- Security mechanisms (throttling, TLS, blocking)

It allows finding vulnerabilities and enables deploying countermeasures at the firewalls

- For assessment and exploitation

Takeaway *External shows your worst public door, which is rarely the door that gets you breached.*



An external assessment focuses on the public exposition - what an outside attacker sees with no access to the organisation. It targets publicly exposed routers and firewall rules, publicly available IPs and ports, public services and records (DNS, HTTP/S, mail; zone transfers, SPF and DMARC records), information exposed to the public, and the security mechanisms themselves (throttling, TLS, blocking).

The mention of DNS records and email authentication is a good reminder that external assessment is not just port scanning. Zone transfers, SPF and DMARC tell you about the organisation's email posture and which servers are public. Even DNS itself is a reconnaissance goldmine: every public record is a potential entry point. Your enumeration work in the lab is the same activity.

Because you only see the perimeter, external assessment finds what a real outside attacker can reach - which is precisely the threat model of an internet-facing business. It also lets you deploy countermeasures at the firewalls, both for assessment and for exploitation: the finding tells you which rule to add or change.

For your project, external is the natural first pass on any public-facing target: enumerate DNS and ports, fingerprint services, and check the security mechanisms for misconfiguration. Remember the limitation though - external only sees the outside. The internal view, which is where most real incidents end up, is a separate vantage point you will cover next.

Type: Internal

Run from inside the perimeter, with normal user credentials (or none)

Covers what the outside never sees

- admin panels, databases, printers, internal APIs, dev hosts

Reveals flat networks and lateral-movement paths

- SMB, RPC, shares, weak internal ACLs

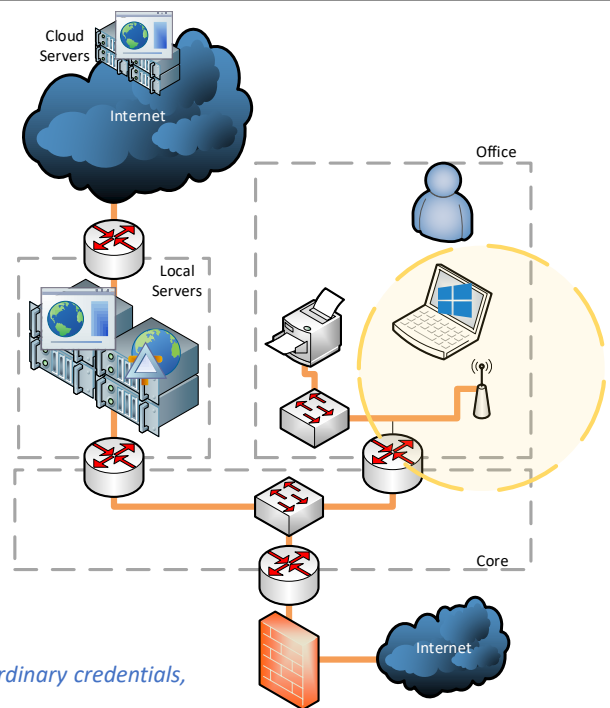
Most real incidents end here

- initial access outside, then expansion inside

Needs strict scope control

- internal systems usually hold production data

Takeaway *Most real incidents end here: inside the perimeter, with ordinary credentials, moving sideways.*



An internal assessment runs from inside the perimeter, with normal user credentials or none at all. It covers what the outside never sees: admin panels, databases, printers, internal APIs, dev hosts. This is the vantage point that most real incidents end up at - the attacker gets initial access outside and then expands inside, and the internal view is what shows you what that expansion would reach.

The slide calls out flat networks and lateral-movement paths - SMB, RPC, shares, weak internal ACLs. These are exactly the findings you will look for when you assess an internal lab: can a user with low privileges reach a file share, an SMB service, or a database they should not? A flat network means every host can talk to every other host, which is what makes lateral movement trivial.

Internal assessment needs strict scope control because internal systems usually hold production data. The moment you are inside the perimeter with credentials, the blast radius of your own mistakes is larger - you may read data you were not authorised to see, or change a production configuration. The rules about accidental findings from earlier slides apply with full force here.

In the lab, the internal view is where your VM-based assessments live: you are already inside, with some level of access, and you are checking what an insider could do. Practice writing the internal findings with the same discipline as external ones - evidence, impact, and what the attacker would actually gain.

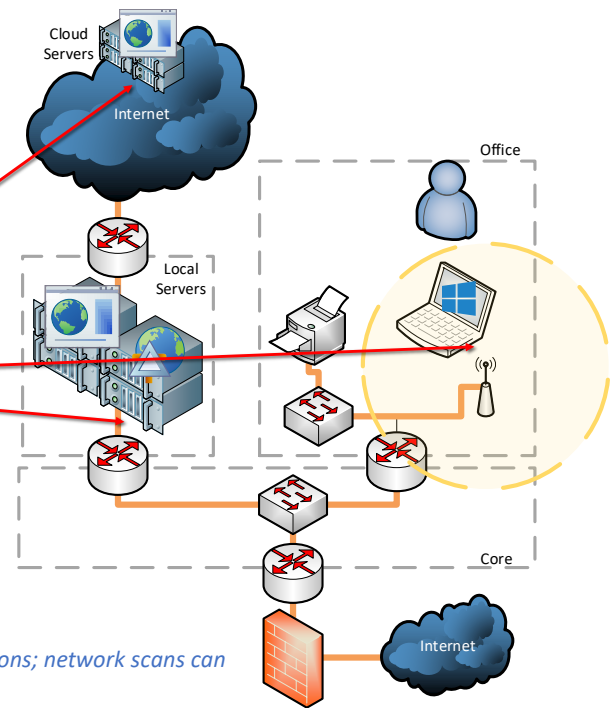
Type: Host-Based

Focus on misconfigurations, permissions, existing software, updates

Targets:

- Servers
- VMs
- Workstations and Laptops

Allows finding vulnerabilities that could be exploited by insiders or by an attacker who gained access to the systems



Takeaway *Host checks read the truth - patches, configuration, versions; network scans can only infer it.*



Host-based assessment focuses on the individual system: misconfigurations, permissions, existing software and updates. Its targets are servers, VMs, workstations and laptops. The question it answers is not 'what is exposed on the network?' but 'what is wrong with this host itself?' - which is a different and complementary question.

Think about the classes of findings this reveals. A service running as root; world-writable files; an unpatched package with a known CVE; default credentials left in a config file; a user with more permissions than their role needs; a kernel that has not been updated in years. None of these are network findings, but each is exploitable by an insider or by an attacker who has already gained access to the system.

The slide makes the threat model explicit: host-based findings can be exploited by insiders or by an attacker who has gained access to the systems. That is why host-based assessment matters even when the perimeter looks solid - once someone is on a host, the misconfigurations you left there become their path to privilege escalation or to other systems.

In your lab, host-based assessment is the OpenVAS-style and manual hardening checks you run against the VMs: review users, permissions, running services, package versions, and compare against a baseline (CIS benchmarks, STIGs). Write the findings with the attacker's viewpoint: what would a user who just got a shell on this host be able to do?

Type: Network

Focus on the communications of the network infrastructure

- Rules, misconfigurations, updates
- Individual services (FTP, Telnet, SMTP, LDAP, SMBv1, SNMP v1/v2c) - cleartext legacy services are themselves a finding

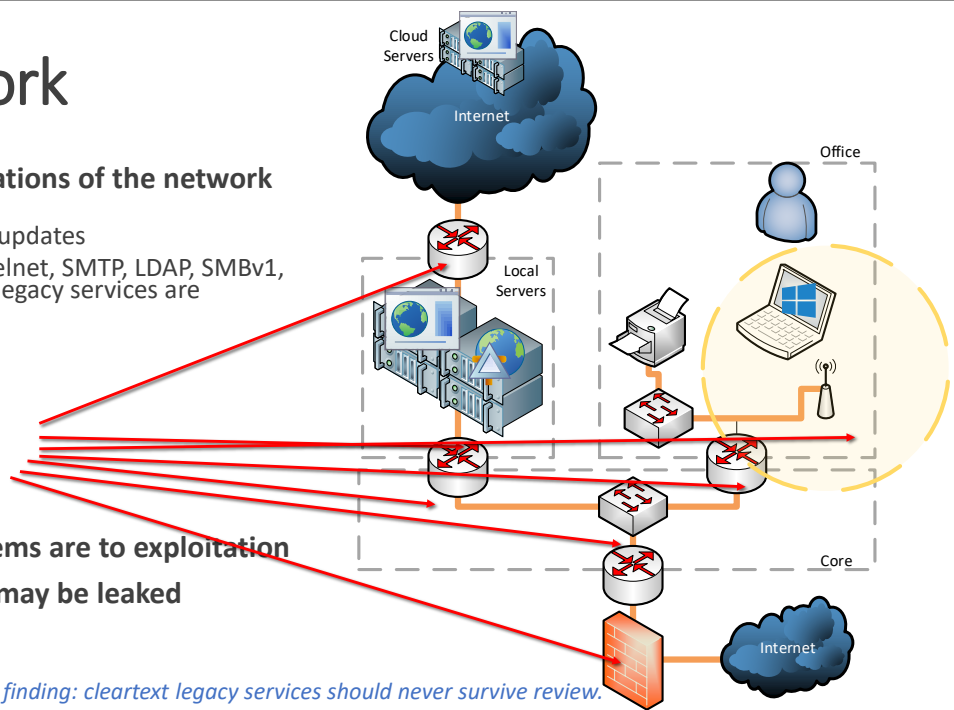
Targets:

- Communication links
- Networking Gear

Finds how exposed systems are to exploitation

Finds what information may be leaked

Takeaway *Reachability is itself a finding: cleartext legacy services should never survive review.*



Network assessment focuses on the communications of the network infrastructure: rules, misconfigurations, updates, and the individual services themselves. Its targets are communication links and networking gear. It finds how exposed systems are to exploitation and what information may be leaked.

The slide lists the classic cleartext legacy services - FTP, Telnet, SMTP, LDAP, SMBv1, SNMP v1/v2c - and makes a sharp point: a cleartext legacy service is itself a finding. Credentials and data crossing the wire unencrypted is a vulnerability regardless of anything else, because anyone on the path can read it. That is why the course keeps returning to TLS and why you will learn to spot plaintext protocols.

Network assessment is where your nmap and Wireshark skills come together. Nmap tells you which services respond; Wireshark tells you what crosses the wire and in what form. The combination reveals not just what is reachable but what leaks - unencrypted logins, SNMP community strings in the clear, legacy protocols still enabled because nobody removed them.

The distinction from host-based is the focus: host-based is about the host's own state, network is about the links and the services on the wire. When you assess the lab, separate the two in your report - a network finding (SMBv1 exposed) and a host finding (weak local account) are different findings with different remediations.

Type: Wireless

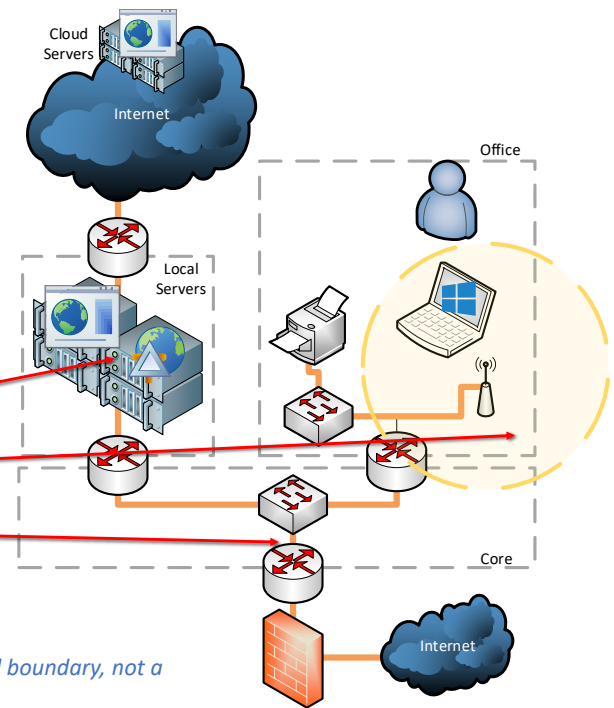
Focus on the wireless communications of the network infrastructure and support services

- Rules, misconfigurations, updates
- Authentication, confidentiality, access control
- Guest access

Targets:

- Wireless Networking Gear
- Authentication servers
- Networking Gear (VLANs)

Similar to network, but with specific tools due to range and authn/authz



Takeaway *The signal leaves the building, so wireless scope is a legal boundary, not a technical one.*



Wireless assessment focuses on the wireless communications of the network infrastructure and its support services: rules, misconfigurations, updates, authentication, confidentiality, access control and guest access. Its targets are wireless gear, authentication servers, and the networking gear that carries the traffic (VLANs).

It is similar to a network assessment but with specific tools and a different threat model, because of range and authentication. A wireless signal is not confined to your building - it reaches the street, the neighbouring office, the car park - so anyone in range is a potential attacker. The physical boundary from the scope slide is literally real for Wi-Fi.

The authentication and access-control angle is the wireless-specific part: is WPA2/3 configured correctly, are weak passphrases in use, is guest access separated onto its own VLAN so guests cannot reach internal hosts, are rogue access points detectable? These are the questions that do not arise on a wired network.

For your lab, wireless assessment is where you apply the sniffing skills you have: capture the beacon and the authentication handshake, check the security configuration, and verify the separation of guest from internal traffic. Remember the legal boundary though - testing a network that is not yours, even one that is 'open', requires authorisation, and a signal that leaves the building can reach devices you were never scoped to touch.

Type: Application

Focus on a single application

- Inputs and outputs
- Logic errors
- Authentication and authorization processes
- Operational assumptions
- Related services (databases, firewalls)

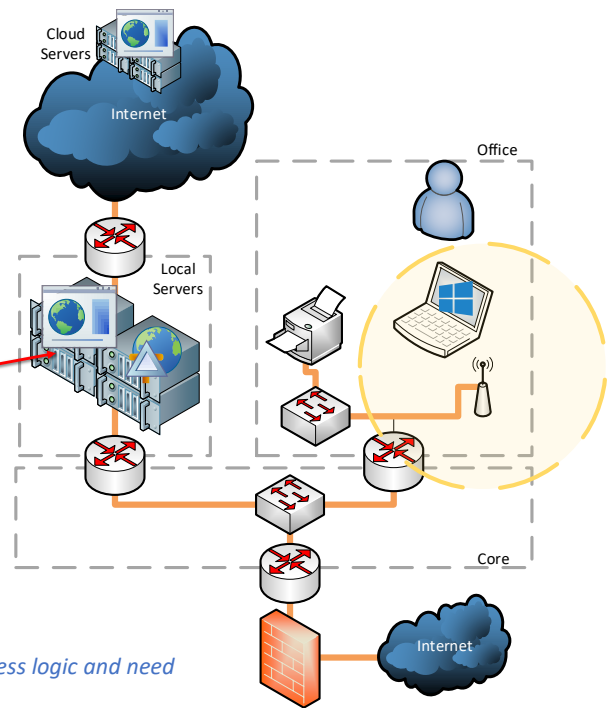
Targets:

- Application
- Service

Finds software vulnerabilities in the targeted application

- Bugs or flaws

Takeaway *Scanners find the shell; application flaws live in the business logic and need human reasoning.*



Application assessment focuses on a single application: its inputs and outputs, logic errors, authentication and authorisation processes, operational assumptions, and the related services it depends on (databases, firewalls). The target is the application or service, and the goal is to find software vulnerabilities - bugs or flaws - in that specific piece of software.

Unlike network or host-based assessment, which look at systems broadly, application assessment goes deep on one thing. That depth is what makes it the natural home of your injection, XSS and authentication work: the questions are about how the application handles input, how it decides who can do what, and what it assumes about its environment.

The list of focus areas maps directly to the OWASP-style categories you will study: inputs and outputs (injection, XSS), logic errors (broken access control, business-logic flaws), authentication and authorisation (the broken-authentication lecture), operational assumptions (what the developer assumed that an attacker will not violate). Related services matter because an application is not an island - its database and firewall are part of its attack surface.

Your ZAP and manual web-testing work is application assessment. The discipline is to test each input, each authorisation decision, and each assumption, and to record the request and response as evidence. That evidence is what lets the developer reproduce the bug - a finding without a reproducing request is a rumour, not a finding.

What the classic taxonomy misses (1/2)

Cloud: misconfiguration and posture (CSPM), identity and permissions, exposed storage, image/AMI hygiene

Containers and Kubernetes: vulnerable images, cluster RBAC, secrets in env/files, runtime policy

CI/CD and build chain: build-agent compromise, dependency confusion, unsigned artefacts, provenance

APIs: broken object-level authorisation, mass assignment, shadow APIs

- OWASP API Top 10



The classic taxonomy you just saw - active/passive, external/internal, host/network/wireless/application - was built for a simpler world. This slide and the next list what it misses, and the first gap is the cloud. Cloud assessment is about misconfiguration and posture (CSPM), identity and permissions, exposed storage, and image/AMI hygiene. In the cloud, the misconfiguration is the vulnerability: a public S3 bucket or an over-permissive IAM role is a finding of the highest order.

Containers and Kubernetes add a second world: vulnerable images, cluster RBAC, secrets in environment variables or files, and runtime policy. A container image you pull from a registry may ship with known vulnerabilities; a cluster with weak RBAC lets a compromised pod move sideways. In your Podman-based labs you are already living in this world, so these findings are directly relevant to the targets you will assess.

CI/CD and the build chain are the third gap: build-agent compromise, dependency confusion, unsigned artefacts and weak provenance. An attacker who poisons a dependency or a build agent owns every artefact it produces. This is the supply-chain angle you will see in the CVE discussions - the vulnerability is not in the product but in how the product was built.

APIs are the fourth: broken object-level authorisation, mass assignment, shadow APIs - and the slide points to the OWASP API Top 10 as the reference. APIs are the new perimeter because every mobile app and every integration is an API call. When you assess, treat the API as a first-class target with its own authorisation model, not as a minor part of the web app.

What the classic taxonomy misses (2/2)

Mobile and MDM: app hardening, enrolment and certificate distribution, jailbreak/root detection

IoT and OT/ICS: long-lived, unpatchable systems - active testing often prohibited, so passive + lab replication

Continuous attack-surface management (ASM/CAASM/EASM): inventory first, point-in-time scanning second

Same life cycle applies to all of them - only the tools and constraints change

Takeaway New surfaces change the tools, not the life cycle: inventory, assess, remediate, verify, monitor.



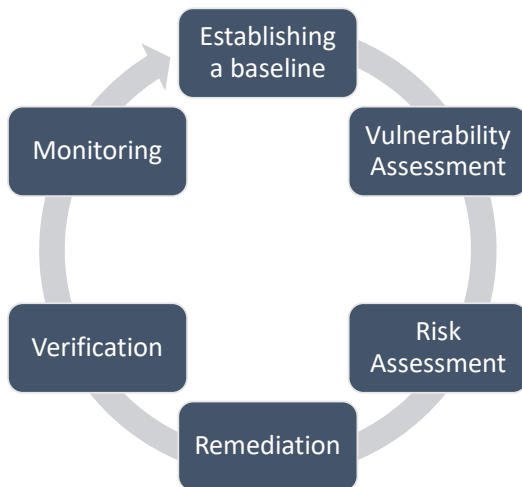
The second half of what the taxonomy misses starts with mobile and MDM: app hardening, enrolment and certificate distribution, and jailbreak/root detection. A mobile app is an application plus a device plus a trust chain - the device must be enrolled, certificates distributed, and the app protected from being tampered with on a rooted device. Each of those is an assessment surface.

IoT and OT/ICS are a world apart: long-lived, often unpatchable systems where active testing is frequently prohibited and the answer is passive observation plus lab replication. An industrial controller that cannot be patched, rebooted or even scanned without risking a production line changes the entire assessment method - you test a replica, not the live system. The slide names that constraint explicitly.

Continuous attack-surface management (ASM, and its CAASM/EASM variants) is the modern answer to the point-in-time problem from the limitations slide: inventory first, point-in-time scanning second. Instead of a periodic assessment, you maintain a continuous inventory of what is exposed and scan it continuously. This is why the life cycle you will see next ends in monitoring.

The last line is the unifying one: the same life cycle applies to all of these - baseline, assess, remediate, verify, monitor - only the tools and constraints change. Cloud, containers, mobile and OT each change the tooling and the rules, but they do not change the discipline. Keep that in mind: learn the life cycle once, and adapt it per domain.

Vulnerability Management Life Cycle



Takeaway *Assessment is stage 2 of six. Skip the other five and findings become paperwork.*



This slide is the picture of the vulnerability management life cycle: baseline, assessment, risk assessment, remediation, verification and monitoring, with the current stage highlighted. It is the process that turns assessment from a one-off event into a continuous practice, and the six stages are the spine of the rest of the lecture.

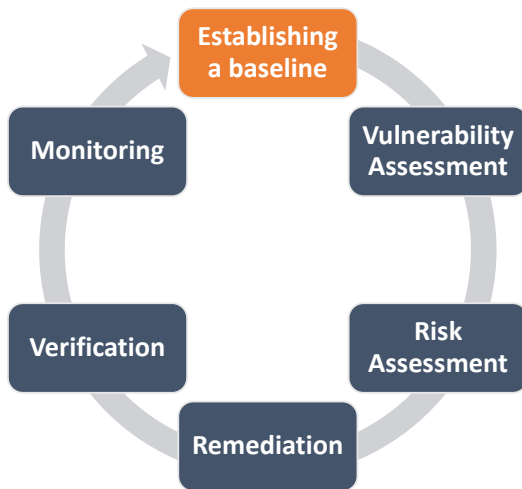
Read it as a loop, not a line. After monitoring you go back to the baseline, because the inventory and the risk picture change over time. The point of the loop is that security is not a project with an end date; it is a process that must keep running as the organisation, its software and the threat landscape change.

Each stage has a specific deliverable: the baseline produces an agreed inventory with priorities; the assessment produces a report of findings; risk assessment produces risk levels per finding; remediation produces fixes and workarounds; verification proves the fixes hold; monitoring produces evidence that nothing regressed. If you can name the deliverable of each stage, you understand the loop.

In your group project, you will walk this loop for a small target. Practise producing the deliverable of each stage, even briefly, rather than jumping from 'found a bug' straight to 'fixed it'. The loop is what a professional organisation actually runs, and the slide is the map.

Vulnerability Management Life Cycle - 1/6

Establishing a baseline



Establish a Baseline

Select the assets to be assessed and define their priorities

- Some assets may be excluded due to potential impact or cost

Characterize the systems/software state

Determine what is known and what must be assessed

- Known vulnerabilities may be ignored from the assessment

Takeaway *Decide what you will not scan, and why - and put an owner on every asset you do scan.*



The first stage of the life cycle is establishing a baseline, and it is the most boring and most important slide in the lecture. You select the assets to be assessed and define their priorities; some assets may be excluded because of potential impact or cost. Then you characterise the state of the systems and software, and determine what is known and what must be assessed.

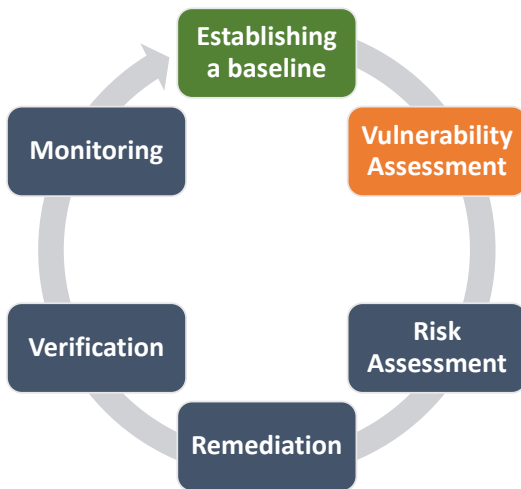
The key decision is what is in and what is out - exactly the scope discipline from earlier. Excluding an asset is a decision, not an accident; it must be recorded and justified, because the assessment can only be judged against the baseline you actually agreed. If a high-value asset was excluded for cost, that is a risk the organisation accepted, and the baseline should say so.

Characterising the state means knowing the software, versions, and configuration of each asset - which is where CPE comes in later: you cannot assess what you have not named. And 'determine what is known' means you can deliberately ignore already-known vulnerabilities to focus effort where it is needed, as long as that choice is explicit.

The exit criteria are the practical test: the baseline is done when the inventory is agreed, every asset has an owner, risk tiers are assigned, and scan windows and credentials are agreed. If those four are not true, your baseline is not done, and everything downstream will be built on sand.

Vulnerability Management Life Cycle - 2/6

Vulnerability Assessment



Vulnerability Assessment

Assess the entities for vulnerabilities

- Takes in consideration priorities
- Takes in consideration scope

Constructs a detailed report with:

- What vulnerability was found
- What are the affected entities
- What are the recommendations to handle it

An assessment usually does not exploit the vulnerability or build an exploit chain

- It's not a penetration test

Takeaway Report finding, affected entities and the fix; resist the temptation to exploit it yourself.



The second stage is the assessment itself: you assess the entities for vulnerabilities, taking into account the priorities and the scope you set in the baseline. The output is a detailed report with three things for every finding: what vulnerability was found, what entities are affected, and what the recommendations are to handle it.

The slide makes an important boundary statement: an assessment usually does not exploit the vulnerability or build an exploit chain - it is not a penetration test. The distinction matters for legal authorisation and for safety: proving a vulnerability exists is different from demonstrating full impact. You will learn both in the course, but know which one your current assignment is asking for.

The priorities from the baseline shape the assessment - you spend your effort where the risk tier is highest, not on the easiest target. And the scope bounds it: you assess what you agreed to assess, nothing more. Both constraints protect you from the twin failure modes of too broad and too narrow that you saw on the scope slide.

For your reports, the discipline is the three deliverables per finding: what, which entities, and what to do. A finding that lists only the vulnerability without the affected entities or the recommendation is incomplete, and a client will rightly send it back.

Assessment Methods

Close to software testing, but with a focus on security-related impact

- Extensively studied in the Robust Software course

Highly dependent on the scope of the assessment

- Application: SAST, DAST, IAST, SCA (software composition analysis), fuzzing, API testing
- Network entity: Protocol, message, authentication, authorization analysis
- Processes/Companies: OSINT, Social Engineering

Takeaway *Method follows scope: code, network, people. Choose the technique that can actually see the risk.*



Assessment methods are close to software testing, but with a focus on security-related impact - and the slide notes that this is studied extensively in the Robust Software course, so you already have part of the vocabulary. The methods chosen depend heavily on the scope of the assessment, because an application needs different techniques than a network entity or a process.

For applications, the method set is familiar: SAST (static analysis, reading the source), DAST (dynamic analysis, testing the running application), IAST (instrumented analysis, observing the app from inside), SCA (software composition analysis, checking your dependencies for known vulnerabilities), plus fuzzing and API testing. These are the tools of the application-assessment world you will practise in the lab.

For network entities, the focus is protocol, message, authentication and authorisation analysis - sending crafted traffic, inspecting protocol implementations, and testing how authentication and authorisation are enforced. That is the nmap/OpenVAS and protocol-testing world.

For processes and companies, the methods are OSINT (open-source intelligence, gathering public information) and social engineering - the human and organisational angle that no scanner covers. This is a reminder that not every assessment is technical: assessing an organisation's posture includes how its people and processes can be manipulated.

Assessment Strategies - Black Box

Researchers have no information about internal aspects and are presented with a publicly available view

- No source code, no documentation
- Assumes an actor with a specific set of resources

Aims to mimic assessments from outside attackers

- Finds what can be exploited by intruders with no access
 - Usually finds vulnerabilities easier to exploit
- May find alternative paths and use cases (which may present vulnerabilities)

Limits what can be found (reduced coverage)

- Compensating controls (e.g. a firewall) mask the vulnerability -> false negatives: it is still present

Takeaway *Black box is realistic but shallow - its silence is not evidence of absence.*



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

30



universidade
de aveiro

Black box is the assessment strategy where researchers have no information about internal aspects and are presented only with a publicly available view - no source code, no documentation. It assumes an actor with a specific set of resources: a script kiddie, a researcher, a competitor, or a crowd-based effort. The choice of assumed actor changes what you test and what you can claim.

The goal is to mimic an outside attacker, and the slide is honest about what that finds: usually the vulnerabilities that are easier to exploit, plus alternative paths and use cases that may present vulnerabilities - things a defender who knows the system would never think to try. Black box is the most realistic picture of an external attacker.

The limitation is reduced coverage. Because you cannot see inside, you cannot check everything, and - crucially - compensating controls can mask a vulnerability. A firewall that blocks the exploit makes the vulnerability invisible to a black-box test, so you report a false negative: the vulnerability is still present, you just cannot see it. This is the coverage-versus-realism trade-off from the axes slide.

In your lab, black box is how you approach a target you know nothing about: enumerate, fingerprint, probe. Practise writing what you could NOT test as well as what you found - that honesty is what separates a professional report from a lucky one. The next two strategies change only what you are given, not the discipline.

Assessment Strategies - White Box

Researchers are given full documentation and access to systems

- A replica of the production system
- The production system with a limited scope
- The source code and infrastructure code

Aims to find faults and bugs at all scoped domains

- Knowledge available to the tester is maximal; the vantage point is chosen separately
- Finds what can be exploited by: outsiders, insiders, outsiders with lateral movement
- May mimic specific users and roles

Extensive (and expensive) analysis of the domains

- Remedies are known and considered, but vulnerability may still be found

Takeaway *White box finds everything, including what no attacker could ever reach: price and triage it.*



White box flips black box: researchers are given full documentation and access to systems - a replica of production, the production system with limited scope, or the source code and infrastructure code. The goal is to find faults and bugs at all scoped domains, and the knowledge available to the tester is maximal.

The slide makes an important point: in white box, the knowledge is maximal, but the vantage point is chosen separately. Full source code does not mean you test as an outsider or as an insider; you still choose the vantage point. White box can mimic outsiders, insiders, or outsiders with lateral movement, and it can mimic specific users and roles. That separation of knowledge from vantage point is the same four-axis idea from earlier.

The trade-off is cost and realism: white box is an extensive and expensive analysis of the domains, and while the remedies are known and considered, a vulnerability may still be found - because knowing the source does not make it correct. The benefit is coverage: with full knowledge you can check every path, including the ones a black-box tester cannot see.

White box over-states what an attacker can do, as the coverage slide will note - just because you could find a bug with source code does not mean an external attacker could. In your report, be explicit that white-box findings represent what a tester with full knowledge can do, and that the attacker-relevant version may be narrower.

Assessment Strategies - Gray Box

Some information is provided to researchers

- Documentation about the application or systems
- A specific set of credentials

Aims to find faults and bugs at a limited set of scoped domains

- Can mimic a specific user

Takeaway *Credentials are the cheapest coverage you can ask for; grey box is the pragmatic default.*



João Paulo Barraca

Analysis and Exploration of Vulnerabilities

32



universidade
de aveiro

Gray box is the middle ground: researchers are given some information - documentation about the application or systems, or a specific set of credentials - but not everything. The goal is to find faults and bugs at a limited set of scoped domains, and the tester can mimic a specific user.

This is the most common real-world arrangement, and the slide is short because the concept is simple: you are given enough to be effective (credentials, documentation) but not enough to see everything. The credentials let you test authenticated functionality and act as a specific user, which black box cannot reach and white box does not need.

Gray box sits between the extremes on the axes. It gives better coverage than black box (you can reach authenticated paths) with less cost than white box, and it is more realistic than white box because an attacker with stolen credentials is exactly a gray-box actor. That is why most commercial engagements are gray box.

For your lab and your group project, gray box is the natural default: you have credentials and documentation for the target you own, but you do not have source-code-level visibility of everything. Practise being explicit about what you were given and what that lets you claim - the knowledge axis determines the strength of your conclusions.

How good is an assessment? Coverage and error

False positive: reported but not real - wastes remediation effort and erodes trust

False negative: real but not reported - the risk you never see; it is invisible by construction

Coverage: which assets, credentials, layers and attack paths were actually examined

- An assessment over 40 % of the inventory cannot support a statement about the other 60 %
- Credentialed scans raise coverage and cut false positives (patch/configuration state is read, not inferred)

Black box: probably more realistic, low coverage.

White box: high coverage, over-states what an attacker can do

Ask before any report: what could not be looked at, and why?

Takeaway *Quality is coverage plus false-positive/false-negative history - ask what could not be examined.*



This slide is about the quality of an assessment, and it gives you the vocabulary to judge one: false positives, false negatives and coverage. A false positive is something reported that is not real - it wastes remediation effort and erodes trust. A false negative is something real that is not reported - the risk you never see, which is invisible by construction because you cannot report what you did not find.

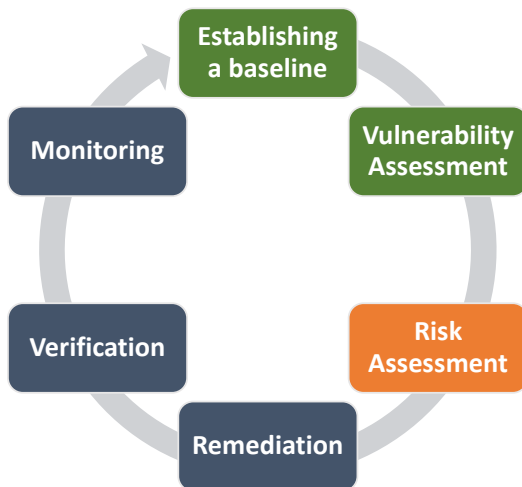
The coverage point is the one to remember: an assessment over 40 percent of the inventory cannot support a statement about the other 60 percent. Coverage is which assets, credentials, layers and attack paths were actually examined. Credentialed scans raise coverage and cut false positives, because with credentials you can read the patch and configuration state instead of inferring it from the outside.

The axes trade-off returns: black box is realistic but low coverage; white box has high coverage but over-states what an attacker can do. Both are honest answers to different questions, and the professional move is to know which you are getting.

The slide closes with the questions to ask before any report: what could not be looked at, and why? And it gives the right way to judge a provider - by inventory completeness, false-positive/false-negative history and retest policy, not by the number of findings. A provider that reports more findings is not necessarily better; one that reports the right findings, honestly bounded, is.

Vulnerability Management Life Cycle - 3/6

Risk Assessment



Risk Assessment

Company takes in consideration the report and assess the risk

- For every asset with vulnerabilities
- Assigns risk levels (typically 3-4): risk combines likelihood and impact, not just vulnerability severity

Risk assessment may take all vulnerabilities found into consideration

- Individual vulnerabilities may be combined into an exploit chain with higher impact

Takeaway Same CVE, opposite risk: this is where your context, assets and exploit chains are priced in.



The third stage of the life cycle is risk assessment: the company takes the assessment report and assesses the risk. For every asset with vulnerabilities it assigns risk levels - typically three or four - and the slide repeats the key point from earlier: risk combines likelihood and impact, not just vulnerability severity. A critical-severity bug on a low-value, unreachable asset is not a critical risk.

This is where the severity-versus-risk distinction becomes a process. The assessment gave you a severity for each vulnerability; the risk assessment converts it into a risk level for the organisation, adding exposure, asset value and the likelihood of exploitation. The output is the prioritisation that drives remediation.

The slide also notes that risk assessment may take all vulnerabilities into consideration together: individual vulnerabilities may combine into an exploit chain with higher impact than any one alone. A low-severity misconfiguration plus a low-severity credential exposure can together be a critical path. This is why you must look at findings as a set, not as isolated items - which is exactly what the worked case at the end of the lecture demonstrates.

For your report, the risk stage is where you justify why you put a finding at a given level. Write the reasoning: severity, exposure, asset value, likelihood, and any chain. A risk level without that reasoning is an opinion, not an assessment.

Documentation

Researchers should carefully document assessments

- Describing the rationale for the assessment, the strategy, the findings
- Essential in cooperation between teams

To understand how the vulnerability was exploited/what the impact may be

- Wrong attitude: we found this, you are not doing your job
- Correct attitude: we found this, that may have caused it, this is the impact, and you can fix it by doing X
 - Clients may not understand the vulnerability, the reason or the impact

Takeaway *Document rationale, evidence and the fix; the tone of the report is part of the deliverable.*



Documentation is the discipline that separates professional work from guesswork, and this slide makes the case: researchers should carefully document assessments - the rationale, the strategy and the findings - because it is essential for cooperation between teams. A finding without documentation cannot be reproduced, verified or fixed.

The slide then makes the point about understanding: documentation exists so that teams can understand how a vulnerability was exploited and what the impact may be. And it names the wrong and right attitudes with unusual directness. The wrong attitude is 'we found this, you are not doing your job.' The correct one is 'we found this, that may have caused it, this is the impact, and you can fix it by doing X.'

Why does the attitude matter so much? Because clients may not understand the vulnerability, the reason or the impact. Your job is not to impress them with what you found; it is to make them able to act. A finding that cannot be understood is a finding that will not be fixed, no matter how severe.

This is directly relevant to your CVE discussion and your software-vulnerabilities project: you will be graded on communication. Write as if the reader knows the system but not the vulnerability - state the cause, the impact, and the fix in that order, and keep the evidence reproducible. That is the professional habit this slide is trying to instil.

Anatomy of a finding (1/2)

ID and title

Risk level and the rationale (CVSS vector + exposure + asset value)

Affected assets: hostname/IP, CPE, owner, environment

- production, test, development

Description: the weakness itself, linked to a CWE so similar findings aggregate

Evidence: the commands/requests and their output that reproduce it

- Sanitized Output!



This slide gives the anatomy of a finding - the structure every good finding should have. Start with an ID and title, then the risk level with its rationale (the CVSS vector plus exposure plus asset value), then the affected assets: hostname or IP, CPE, owner and environment (production, test, development).

The risk rationale matters because it is where you connect severity to your context - the vector alone does not tell the organisation what to do; the exposure and asset value do. And the environment field is not decoration: a vulnerability in a development environment is a different risk from the same one in production, and remediation priority follows.

Then the description: the weakness itself, linked to a CWE so that similar findings aggregate. Linking to a CWE is the standardisation move - it lets the organisation see that ten findings are all the same class of weakness and treat them together. And the evidence: the commands or requests and their output that reproduce it - and the slide stresses sanitised output.

Sanitised output is a security requirement in itself: your evidence must reproduce the finding without exposing real credentials, personal data or internal secrets. If your reproduction dumps a password, you redact it. The evidence is what lets the developer confirm the bug and the fix - without it, the finding is an opinion.

Anatomy of a finding

Impact: what an attacker gains, and the preconditions they need first

Remediation information

- The fix
- Any workaround
- Dependencies
- The change window required

Status: open / pending retest / closed, with retest date and who verified

Write for a decision-maker: state the impact, not only the score

Takeaway *A finding you cannot re-derive or re-test is a rumour - evidence, owner and status make it real.*



The second half of the finding anatomy covers the parts that make a finding actionable. Impact: what an attacker gains, and the preconditions they need first. This is not the CVSS score - it is the concrete consequence in your system: an attacker with this finding can read this database, or pivot to this network, but only if they already have this access.

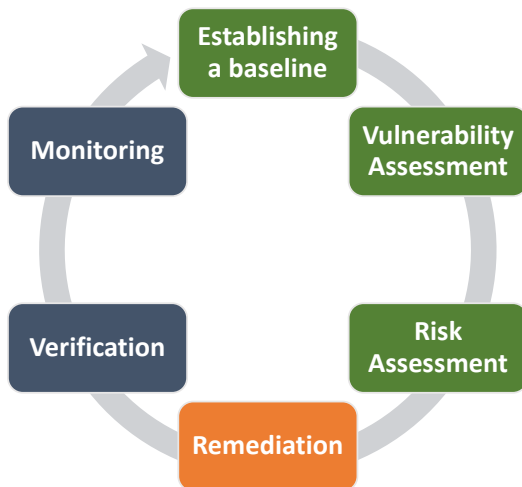
Then remediation information: the fix, any workaround, the dependencies, and the change window required. The change window matters because it is a real operational constraint - a fix that requires a reboot in the middle of the day is different from a config change you can apply instantly. Dependencies matter because the fix may require another team or another vendor.

The status field tracks the finding through the life cycle: open, pending retest, or closed, with a retest date and who verified it. This is the link back to the verification stage - a finding is not done when the fix is applied; it is done when the retest proves it.

And the last sentence is the one that matters most for your career: write for a decision-maker, and state the impact, not only the score. A manager does not act on 'CVSS 9.8'; they act on 'an unauthenticated attacker can read all customer records.' Lead with the impact, and use the score to support it.

Vulnerability Management Life Cycle - 4/6

Remediation



Remediation

The company implements measures to increase the security of its assets

May fix the vulnerability

- Correct software bugs or flaws, update it, change configs
- This capability is not always present

May reduce the impact of a successful exploitation

- Implement mechanisms that reduce impact to a smaller domain
- Implement redundancy and failover

May increase the cost of exploiting the vulnerability

- Deploy firewalls or change its rules
- Increase isolation so that assets are not available in a domain

Takeaway *If you cannot patch, buy time: isolate, add redundancy, raise the cost of exploitation.*



The fourth stage is remediation: the company implements measures to increase the security of its assets. The slide is careful to say 'increase', not 'fix', because remediation is broader than patching - it can fix the vulnerability, reduce the impact of a successful exploitation, or increase the cost of exploiting it.

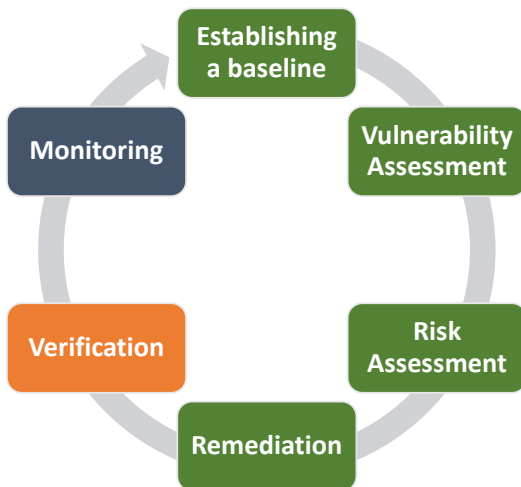
The first option is to fix the vulnerability itself: correct software bugs or flaws, implement specific configurations, or update software and firmware. The slide adds the honest caveat - this capability is not always present. Sometimes there is no patch, sometimes the vendor is gone, sometimes the fix breaks the business function. Knowing when you cannot fix is part of the assessment's value.

The second option is to reduce the impact of a successful exploitation: implement mechanisms that confine the damage to a smaller domain, and add redundancy and failover so a compromise does not mean an outage. This is the compensating-control mindset from the primitives slide - you cannot always stop the exploit, but you can limit what it achieves.

The third is to increase the cost of exploitation: deploy firewalls or change their rules, and increase isolation so that assets are not available in a domain. Every one of these is a legitimate remediation, and a good report gives the client all three options with their costs, because the client's constraints - not the ideal fix - will decide which one they choose.

Vulnerability Management Life Cycle - 5/6

Verification



Verification

Verifies the effectiveness of the remediation

Involves assessing the existence and risk of the vulnerabilities found

- Using the same scope!
- The risk may remain similar if the vulnerability is exploited from another perspective
 - E.g. External vs Internal actor

Takeaway *Retest at the same scope and vantage point, or the fix may only have moved the hole.*



The fifth stage is verification, and the slide makes the point that remediation without verification is a hope, not a control. Verification involves assessing the existence and risk of the vulnerabilities found - using the same scope as the original assessment. Same scope is the operative phrase: you cannot claim a fix worked if you verified only half the systems you originally found vulnerable.

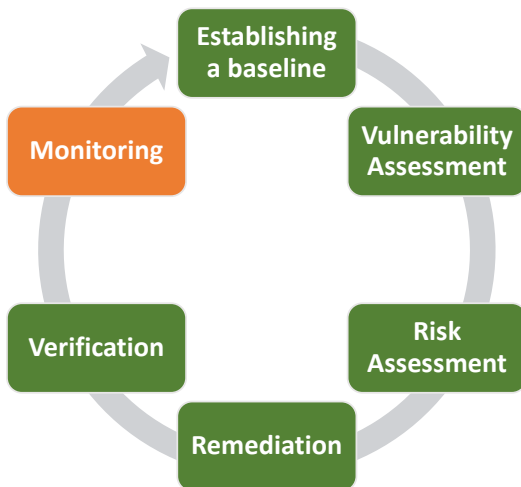
The slide adds a subtle but crucial warning: the risk may remain similar if the vulnerability is exploited from another perspective - for example, an external actor versus an internal actor. You may have fixed the external path, but if the same weakness is reachable from inside, the risk is not gone, only moved. Verification must test the perspectives that matter, not just the one you patched.

This is why the finding status field has a retest date and a verifier: verification is an accountable act, not a hope. The retest pass rate and the reopened-case rate are the metrics that tell you whether remediation actually holds, as you will see on the metrics slide.

Practise verification in your own work: after you 'fix' a finding in a lab target, re-run the exact test that produced it and record that it no longer triggers. That closed loop - find, fix, retest, close - is the professional minimum, and it is exactly what this stage formalises.

Vulnerability Management Life Cycle - 6/6

Monitoring



Monitoring

Deploys mechanisms to detect attempts to exploit the vulnerability

- May consider variations

Involves configuring firewalls, log-analysis systems, IDS (network- and host-based) and profilers

Takeaway Assume the vulnerability is being tried right now - detection is what closes the loop.



The sixth and final stage is monitoring: you deploy mechanisms to detect attempts to exploit the vulnerability, including its variations. This is the stage that converts assessment from a periodic exercise into a continuous one, and it is the answer to the point-in-time limitation from the limitations slide - you cannot test once and walk away.

Monitoring involves configuring firewalls, log-analysis systems, IDS (both network- and host-based), and profilers. The slide's mention of variations is the key: an attacker will not replay the exact exploit you patched; they will adapt it. Monitoring must catch the family of attempts, not just the specific signature you already know.

This is where the earlier limits of the defensive technologies become relevant again - the IDS that detects and alerts but does not block. Monitoring detects attempts and tells you someone is trying; the response is a separate process. In your lab, monitoring is the telemetry you would configure on a target to see when it is probed - which is also how you learn whether your remediation actually held in the wild.

The life cycle is now complete: baseline, assess, remediate, verify, monitor, and back to baseline. The reason it loops is that the inventory, the software and the attacker all change. A system that was clean at assessment time can be compromised before the next assessment - which is why monitoring is not an afterthought but a stage.

Life cycle: exit criteria and metrics

Baseline is done when:

- inventory agreed
- There is one owner per asset
- Risk tier assigned
- Scan windows and credentials agreed

Assessment: coverage of in-scope inventory (%), credential coverage, age of the last scan

Risk assessment: every finding has an owner, a risk level and a target date

Remediation: mean time to remediate per risk level; SLA breaches; fixes vs temporary workarounds

Verification: retest pass rate; reopened-case rate (measures whether fixes actually hold)

Monitoring: recurrence rate, time to detect, share of assets sending telemetry



This slide gives exit criteria and metrics for every stage of the life cycle, and it is the slide that tells you what 'done' means at each point. The baseline is done when the inventory is agreed, there is one owner per asset, risk tiers are assigned, and scan windows and credentials are agreed. If any of those is missing, your baseline is not complete.

Each stage has a measurable output: the assessment is judged by coverage of the in-scope inventory as a percentage, credential coverage, and the age of the last scan; the risk assessment is done when every finding has an owner, a risk level and a target date; remediation is measured by mean time to remediate per risk level, SLA breaches, and the balance of fixes versus temporary workarounds.

The verification and monitoring metrics are where the loop becomes honest: verification is measured by the retest pass rate and the reopened-case rate - which tells you whether fixes actually hold; monitoring is measured by the recurrence rate, time to detect, and the share of assets sending telemetry.

The last two lines are the management lesson: report the trend, not the snapshot - management learns from the curve, not from one number - and keep the definitions constant year on year, because if the definitions change, the curve is meaningless. When you write your own metrics, define them once and keep them stable; that is what makes the numbers comparable over time.

SCAP – Security Content Automation Protocol

A suite of interoperable specifications (and a process) to automate the assessment of a system's security status

- Adoption driven mainly by public-sector procurement; vendor coverage is uneven (strongest on servers/OS configuration)

Some objectives:

- Track system status
- Identify vulnerabilities
- Monitor the system security policies
- Quantify the existing risks
- Common terminology across vendors and environments

Most common in environments with high policy compliance requirements

Takeaway *SCAP makes results comparable and automatable; check vendor coverage before you commit to it.*



SCAP - the Security Content Automation Protocol - is a suite of interoperable specifications and a process for automating the assessment of a system's security status. The slide notes that adoption is driven mainly by public-sector procurement and that vendor coverage is uneven, strongest on servers and OS configuration. That is a realistic caveat: SCAP is a standard, and standards are only as good as their implementations.

The objectives are what SCAP is designed to enable: track system status, identify vulnerabilities, monitor system security policies, quantify existing risks, and use a common terminology across vendors and environments. The last objective is the heart of it - SCAP exists so that a finding described by one vendor's tool means the same thing in another vendor's tool.

The most common environment for SCAP is high policy-compliance environments - government, regulated industries, defence - where you must prove, repeatedly and reproducibly, that a set of systems meets a policy. That is why procurement drives adoption: a buyer cannot evaluate ten vendors' security postures unless they all speak the same language.

For the lecture, remember that SCAP is the umbrella. The individual components - CPE, CVE, CVSS, CCE, XCCDF, OVAL, OCIL - are the pieces that answer specific questions, and you will meet each one in turn. When you see a SCAP data stream later, it is just those pieces packaged together.

SCAP – Security Content Automation Protocol

Enumeration

- CVE: Common Vulnerabilities and Exposures
- CCE: Common Configuration Enumeration
- CPE: Common Platform Enumeration

Languages

- OVAL: Open Vulnerability Assessment Language
- OCIL: Open Checklist Interactive Language
- XCCDF: eXtensible Configuration Checklist Description Format

Metrics

- CVSS: Common Vulnerability Scoring System (FIRST; v3.1 still deployed, v4.0/v4.1 current)

Takeaway *Enumeration, languages, metrics: know which SCAP piece a vendor claim actually refers to.*



This slide is the SCAP component map, and it is worth memorising because it is the answer to a likely exam question. SCAP components fall into three groups. Enumeration: CVE (Common Vulnerabilities and Exposures), CCE (Common Configuration Enumeration) and CPE (Common Platform Enumeration). Languages: OVAL (Open Vulnerability Assessment Language), OCIL (Open Checklist Interactive Language) and XCCDF (eXtensible Configuration Checklist Description Format). Metrics: CVSS (Common Vulnerability Scoring System).

Read the three groups as three different questions. Enumeration answers 'what is it?' - what vulnerability (CVE), what configuration (CCE), what platform (CPE). The languages answer 'how do we express the check?' - OVAL and OCIL express what to assess and how, and XCCDF expresses the policy that ties rules together. Metrics answer 'how bad is it?' - that is CVSS.

The slide notes CVSS is owned by FIRST, with v3.1 still deployed and v4.0 and v4.1 current. That matters for your work because the scoring system you will use on your findings is CVSS, and you need to know which version you are using, since the versions differ in how the score is computed and in which factors they include.

When you write a finding, you are implicitly producing a SCAP-shaped answer: you name the platform (CPE), the vulnerability (CVE), the weakness (CWE), the score (CVSS), and ideally the check that proves it (XCCDF/OVAL). The whole block of slides is teaching you to produce machine-comparable findings instead of prose-only ones.

SCAP – Security Content Automation Protocol



Takeaway SCAP only pays off through content: know whose definitions your engine actually consumes.



This slide is the SCAP puzzle picture - each component as a piece that answers one question: platforms, vulnerabilities, score, configuration, policy and compliance. The picture is the whole point of SCAP: each standard answers one question, and together they cover the assessment process end to end.

Map each piece to the questions: CPE answers 'what platform is this?' CVE answers 'what vulnerability is known?' CVSS answers 'how severe is it?' CCE answers 'what configuration issue exists?' XCCDF answers 'what does my policy require?' OVAL answers 'does this host actually satisfy it?' And the result feeds compliance - 'does this system meet the policy?'

The puzzle framing also explains why the components interoperate: an XCCDF rule references an OVAL definition, which checks a CCE on a platform named by a CPE, scored by CVSS if a vulnerability is involved. None of them works fully alone; they are designed to slot together, which is why you cannot skip one.

Keep the puzzle in mind when you read a SCAP data stream or an OpenSCAP report later: it is the same pieces, packaged. When you run a compliance check in the lab and it produces a pass or fail with an OVAL definition and an XCCDF rule behind it, you are seeing the puzzle assembled.

CPE – Common Platform Enumeration

Know what entities must be addressed in the scope of security

- Consider a company or campus: hundreds of computers with lots of software

Naming convention to describe and enumerate platforms (applications, operating systems and hardware)

- The NVD CPE list currently holds over 1 million platform names
- Small amount of information: name, title, references, metadata (not a description)
- Format (CPE 2.3):

```
<cpe-item name="cpe:/o:microsoft:windows_10-::~::~x64~">  
<title xml:lang="en-US">Microsoft Windows 10 64-bit</title>  
<references>  
  <reference href="https://www.microsoft.com/en-us/windows/features">Product</reference>  
  <reference href="https://www.microsoft.com/en-us/">Vendor</reference>  
</references>  
<meta:item-metadata nvd-id="314192" status="FINAL" modification-date="2015-10-13T18:55:40.893Z"/>  
</cpe-item>
```

language>:<sw_editi
S, h: hardware}. The
everything else.



CPE - Common Platform Enumeration - answers the question 'what entities must be addressed in the scope of security?' It is the naming convention to describe and enumerate platforms: applications, operating systems and hardware. The slide makes the scale concrete: the NVD CPE list holds over a million platform names, because you need a name for every version of every piece of software in scope.

The format (CPE 2.3) is the plumbing:

cpe:2.3:<part>:<vendor>:<product>:<version>:<update>:<edition>:<language>:<sw_edition>:<target_sw>:<target_hw>:<other>, where part is a for application, o for operating system and h for hardware. The legacy cpe:/ form has only seven fields. You do not need to memorise every field, but you do need to recognise a CPE when you see one and to read the fields that matter - vendor, product, version - because those are what matching uses.

The slide's takeaway is the one that ties this to everything else: no CPE in your inventory means no SBOM matching - naming is the plumbing under everything else. A software bill of materials (SBOM) lists the components of your software, and it can only match known vulnerabilities if both sides use the same names - that is CPE.

In your work, the practical use is that CPE is how a finding says which asset it affects. When you write 'this mail server with CPE cpe:2.3:a:vendor:product:1.2.3' you are enabling the automated matching that lets an SBOM scan find every instance of that version across your estate - which is exactly what the worked case at the end will demonstrate.

CCE – Common Configuration Enumeration

Clearly identifies the configuration issues of a platform (CPE)

- That is, the configurations

Publicly available in many cases, but not always.

- Maintained by the content author (e.g. OS vendor); the MITRE/NIST CCE dictionary is no longer expanded - CCE IDs survive as legacy identifiers inside vendor and checklist content

Content

- **CCE Identifier Number - “CCE-2715-1”**
- **Description** - a human-readable description of the configuration issue
- **Conceptual Parameters** – parameters that would need to be specified in order to implement a CCE on a system
- **Associated Technical Mechanisms** – for any given configuration issue there may be one or more ways to implement the desired result
- **References** – pointers to the specific sections of the documents or tools in which the configuration issue is described in detail

Takeaway *Configuration findings need stable IDs; the CCE dictionary is legacy, the practice is not.*



CCE - Common Configuration Enumeration - identifies configuration issues of a platform (a CPE), meaning the configurations themselves. Where CPE names the platform and CVE names the vulnerability, CCE names the specific configuration problem: a service enabled that should be disabled, a permission too broad, a setting that violates policy.

The slide is honest about the state of CCE: it is publicly available in many cases but not always, and it is maintained by the content author (for example the OS vendor). The MITRE/NIST CCE dictionary is no longer expanded - CCE IDs survive as legacy identifiers inside vendor and checklist content. This is the 2026 reality you should know: do not expect a central, always-current CCE dictionary.

The anatomy of a CCE entry is the checklist item: an identifier number (like CCE-2715-1), a human-readable description of the configuration issue, conceptual parameters needed to implement the CCE, the associated technical mechanisms (the ways to implement the desired result), and references to the documents or tools where the issue is described in detail.

Understand CCE as the bridge between a configuration and a check. A configuration finding in your report can be tied to a CCE so that it aggregates with the same issue on other hosts, just as CWE aggregates vulnerabilities. That is what makes configuration findings countable and comparable - without the identifier, they are prose.

CCE – Common Configuration Enumeration

CCE-2715-1

Platform: vista

Date: (C)2012-03-13 (M)2020-08-17

The "reset account lockout counter after" policy should meet minimum requirements.

Parameter: (1) number of minutes

Technical Mechanism: (1) defined by Local or Group Policy

References:

Resource Id	Reference
Old v4 CCE ID	CCE-733
NIST SCAP Windows Vista XCCDF (SCAP-WinVista-XCCDF.xml rev 2007-02-06)	reset-account-lockout-counter



This slide shows CCE in action - an example of a configuration item, the CCE-2715-1 entry you saw named on the previous slide. The point of the visual is to show you what a real CCE entry looks like: an identifier, a description, and the parameters and mechanisms that let you implement it on a system.

Read the example as a template for your own configuration findings. A good configuration finding names the identifier (so it can be matched across hosts), describes the issue in plain language, lists what would need to change to fix it, and gives the mechanism - the command, setting or policy - that implements the fix. That is exactly the anatomy of a finding applied to configuration.

The reason CCE matters at scale is aggregation: if the same configuration issue exists on two hundred hosts, you do not want two hundred prose findings; you want one CCE-referenced finding with a host list. That is what makes remediation measurable - you can count how many hosts are still non-compliant.

For your report, when you find a misconfiguration, try to tie it to a CCE identifier if one exists, or describe it with the same structure if not. The structure - id, description, parameters, mechanism, reference - is the professional habit, whether or not a formal CCE ID is available.

XCCDF - eXtensible Configuration Checklist Description Format

XML-based language to define verifications and fixes associated with a profile

- Profile defines a set of policies and what needs to be verified
- Specific to a CPE (e.g. OS Distribution)

Fixes may include commands and validations

- Run scripts, call APIs
- Specific to an operating system

Takeaway *XCCDF turns policy into something executable:
profile, rule, fix - one platform at a time.*

```
<Profile id="standard">
  <title>Standard System Security Profile</title>
  <select idref="no_direct_root_logins" selected="true"/>
</Profile>

<Group id="root_logins">
  <title>Restrict Root Logins</title>

  <Rule id="no_direct_root_logins" selected="false" severity="medium">
    <title>Direct root Logins Not Allowed</title>
    <fix system="urn:xccdf:fix:script:sh">echo &gt; /etc/securetty</fix>
    <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
      <check-content-ref name="oval:no_direct_root_logins:def:1" href="oval.xml"/>
    </check>
  </Rule>
```



XCCDF - eXtensible Configuration Checklist Description Format - is the XML-based language for defining the verifications and fixes associated with a profile. A profile defines a set of policies and what needs to be verified, and it is specific to a CPE - for example, a particular OS distribution. This is the language that turns policy into something executable.

The slide's mechanics: fixes may include commands and validations - run scripts, call APIs - and they are specific to an operating system. So an XCCDF rule for a Red Hat profile contains not just the check but also the fix command that would bring the system into compliance, and both are tied to that platform.

The takeaway line is the one to remember: XCCDF turns policy into something executable - profile, rule, fix - one platform at a time. A policy document is prose; an XCCDF profile is the same policy expressed so a tool can run the checks and apply the fixes. That is the automation SCAP promises.

Relate it to your own work: when you write a hardening checklist in prose, you are one step away from an XCCDF profile. The step of making it executable - so a tool can verify and fix - is exactly what XCCDF provides, and it is the difference between a document and a control.

OVAL - Open Vulnerability and Assessment Language

XML based language with definitions to validate security controls

- Definitions are referenced by the XCCDF rules that implement a specific policy

Each definition states:

- What to assess: ex: The state of a CCE, or presence of a CPE
- How to assess: ex: How the state(s) is(are) checked
- How to report: ex: What message is provided

```
<definition id="oval:mil.disa.stig.windows:def:177" version="2" class="compliance">
  <metadata>
    <title>BitLocker must be enabled on all fixed drives.</title>
    <affected family="windows">
      <platform>Microsoft Windows 10</platform>
    </affected>
    <description>BitLocker must be enabled on all fixed drives.</description>
  </metadata>
  <criteria operator="AND">
    <criteria test_ref="oval:mil.disa.stig.windows:tst:17700" comment="BitLocker must be enabled on all fixed drives." />
  </criteria>
</definition>
```

Takeaway OVAL answers 'is it true on this host?'; XCCDF decides whether that matters for your policy.



OVAL - Open Vulnerability and Assessment Language - is the XML-based language with definitions used to validate security controls. The slide makes the relationship to XCCDF explicit: OVAL definitions are referenced by the XCCDF rules that implement a specific policy. XCCDF is the policy; OVAL is the machine-observable check underneath it.

Each OVAL definition states three things: what to assess (for example, the state of a CCE or the presence of a CPE), how to assess it (how the state is checked), and how to report it (what message is provided). That is a complete, self-contained test: target, method, and output.

The takeaway is the clean division of labour: OVAL answers 'is it true on this host?', and XCCDF decides whether that matters for your policy. OVAL is the low-level fact-checker - is this package at this version, is this file mode correct, is this service running - and XCCDF is the policy interpreter that combines those facts into a compliance verdict.

For your lab, when you see an OpenSCAP result, the pass or fail you read is the XCCDF verdict, and the evidence behind it is the OVAL definitions that were evaluated. Knowing the difference lets you debug a surprising result: was the policy wrong (XCCDF) or was the host state misread (OVAL)?

OCIL - Open Checklist Interactive Language

For checks that cannot be decided automatically: a question, the expected answer and evaluation logic

Structured as a questionnaire: questions, answer types, decision logic, outcome

- Outcomes: pass / fail / review

Complements XCCDF and OVAL: OVAL decides what is machine-observable; OCIL handles everything else

Used in National Checklist content for procedural and interview-style controls

- Example: "is multi-factor authentication enforced for all administrators?" with no reliable artefact to query
- Supported by SCAP engines (e.g. OpenSCAP can evaluate OCIL questionnaires and produce results)

Practical value: it makes human verification auditable instead of anecdotal

Takeaway Not everything is automatable - OCIL makes human verification auditable instead of anecdotal.



OCIL - Open Checklist Interactive Language - handles the checks that cannot be decided automatically: a question, the expected answer and the evaluation logic. It is structured as a questionnaire - questions, answer types, decision logic and an outcome of pass, fail or review - for the controls that require human judgement.

The slide is clear about how OCIL fits: it complements XCCDF and OVAL. OVAL decides what is machine-observable; OCIL handles everything else. The example is the perfect one: 'is multi-factor authentication enforced for all administrators?' - there may be no reliable artefact to query, so a person must answer, and OCIL makes that human answer auditable.

This is the practical value the slide ends on: it makes human verification auditable instead of anecdotal. Without OCIL, the answer to an interview-style control is a memory or a claim; with OCIL, it is a recorded question with an answer and a pass/fail outcome that can be part of a compliance report.

The example matters for your own work because many real controls are not machine-checkable: whether someone follows a process, whether a policy is actually communicated, whether a backup is genuinely tested. OCIL is how SCAP handles those, and it is a reminder that not everything in security can be automated - some checks require a human being.

SCAP related repositories

CVE: IDs assigned by CNAs (vendors, projects, research groups);

- MITRE is one CNA and a CVE Listing partner;
- Programme sponsored by CISA;
- NVD/NIST aggregates and enriches

CVSS: standard owned and maintained by FIRST.org; NVD publishes scores for some CVEs, but no longer enriches every one (selective triage since 2025/26)

CPE: Provided by NIST

CCE: legacy identifiers carried by vendor/checklist content; the dictionary itself is no longer maintained

XCCDF: military, government and regulated entities (e.g. HIPAA) imposing security requirements

OVAL: provided by OS vendors (Red Hat, SUSE, Debian, Ubuntu) and commercial vendors, via MITRE

Takeaway *Trust the source: who assigns, who enriches, how fresh. Write it into the procurement.*



This slide maps each SCAP component to its real-world owner and source, which tells you where to go for authoritative data. CVE IDs are assigned by CNAs - vendors, projects and research groups; MITRE is one CNA and a CVE Listing partner; the programme is sponsored by CISA; and NVD/NIST aggregates and enriches the data.

CVSS is owned and maintained by FIRST.org, and the slide carries an important 2026 reality: NVD publishes scores for some CVEs but no longer enriches every one - there is selective triage since 2025/26. That means you cannot assume every CVE has an NVD CVSS score; you may have to use the CNA's score or a vendor advisory. This is the funding-gap consequence you will see again.

CPE is provided by NIST. CCE is the legacy case - IDs carried by vendor and checklist content, with the dictionary itself no longer maintained. XCCDF is used mainly by military, government and regulated entities (like HIPAA-covered ones) imposing security requirements. OVAL is provided by OS vendors - Red Hat, SUSE, Debian, Ubuntu - and commercial vendors, via MITRE.

For your CVE discussion and your assessment work, this slide is a practical source map: when you need a score, go to the CNA and FIRST; when you need a platform name, go to NIST CPE; when you need to know how a check is expressed, look at the OS vendor's OVAL content. Knowing who owns the data tells you who to trust and who to ask when it is missing.

Vulnerability data in 2026: what changed

CVE assignment is decentralised

NVD stopped universal enrichment

- 2024 funding gap, selective triage 2025/26
- Many CVEs now arrive without CVSS or CPE
- Prioritisation cannot be outsourced to NVD - use CNA data, vendor advisories, KEV and EPSS

CPE is the bridge to SBOM matching; the CCE dictionary is no longer maintained (legacy IDs only)

SCAP baseline is 1.3 (SP 800-126r3 + 126Ar3)

- SP 800-126r4 and 126Ar4 (SCAP 1.4) are initial public drafts (Dec 2025)

OSCAL is the emerging representation for machine-readable controls - watch it

Ask any supplier: which feed, how fresh, who enriches it, and what happens when it breaks?



This slide is the 2026 state of the vulnerability data ecosystem, and it changes how you should work. The first fact: CVE assignment is decentralised - CNAs (vendors, projects, research groups) assign IDs; MITRE is one CNA and a CVE Listing partner; CISA sponsors the programme. You do not need to request an ID through MITRE for every product anymore; the vendor may assign it itself.

The second fact is the one with the biggest consequence: NVD stopped universal enrichment after a 2024 funding gap, with selective triage through 2025/26. Many CVEs now arrive without a CVSS score or a CPE mapping in NVD. The consequence is the slide's key line: prioritisation cannot be outsourced to NVD - you must use CNA data, vendor advisories, KEV and EPSS.

The rest of the slide is the current landscape: CPE is the bridge to SBOM matching; the CCE dictionary is no longer maintained; the SCAP baseline is version 1.3 (SP 800-126r3 with 126Ar3), with SCAP 1.4 (SP 800-126r4 and 126Ar4) as initial public drafts from December 2025; content ships as SCAP Data Streams; tools are validated by the NCCOE programme; and OSCAL is the emerging representation for machine-readable controls - watch it.

The closing instruction is the practical one to keep: ask any supplier which feed they use, how fresh it is, who enriches it, and what happens when it breaks. A tool is only as good as its data, and in 2026 the data has gaps. Knowing your tool's data source and its failure mode is part of using it responsibly.

Use Case: one finding, end to end

Asset: mail server, CPE `cpe:2.3:a:vendor:product:1.2.3:*:*:*:*:*`

CVE-2024-XXXX affects that version: the vendor CNA publishes CVSS v3.1 9.8 (AV:N/AC:L/PR:N/UI:N/...)

- KEV lists it; EPSS percentile is high -> escalate above the quarterly queue

Baseline content: an XCCDF rule references an OVAL definition -> automated check across every instance

Report: affected hosts, evidence, risk (internet-facing, production mail), fix, workaround, owner, SLA

Verification: same scope, same tool version, open until evidences shows it is fixed

Monitoring: periodic telemetry confirms all instances report

This closes the loop: CPE -> CVE -> CVSS -> KEV/EPSS -> XCCDF/OVAL -> remediation -> verification

Takeaway *The loop is always the same: identify, score, prioritise, fix, verify, watch - in that order.*



This is the worked case that ties the whole lecture together: one finding, end to end. The asset is a mail server, named by CPE - `cpe:2.3:a:vendor:product:1.2.3` - which is the platform naming you saw earlier. CVE-2024-XXXX affects that version, and the vendor's CNA publishes a CVSS v3.1 score of 9.8 with the vector AV:N/AC:L/PR:N/UI:N, meaning remote, low complexity, no privileges, no user interaction.

Then the prioritisation machinery kicks in: KEV lists it (confirmed exploited in the wild) and the EPSS percentile is high, so the finding escalates above the quarterly queue. This is the severity-is-not-a-schedule lesson in action - the 9.8 becomes urgent because it is actually being exploited and exposed, not because of the number alone.

The automated-check step shows the standards working together: a baseline XCCDF rule references an OVAL definition, which gives an automated check across every instance of that mail server. Instead of manually checking each host, the SCAP data stream verifies the whole fleet, and the report lists affected hosts, evidence, risk (internet-facing, production mail), fix, workaround, owner and SLA.

The loop closes with verification and monitoring: the same scope and the same tool version are re-run, and the finding closes only when the evidence shows it is fixed; then telemetry confirms all instances report, and recurrence is tracked monthly. The final line is the summary of everything: CPE to CVE to CVSS to KEV/EPSS to XCCDF/OVAL to remediation to verification - and back to monitoring. Keep this loop as your mental model for the whole course.

Key takeaways

An assessment answers "what is exposed?": it is neither an audit nor a penetration test

- Results are valid at a point in time, while coverage and false negatives bound their quality

Define scope and the four axes (method, vantage point, target, knowledge) before anything else

Never confuse vulnerability severity with organisational risk

Remediation without verification is a hope, not a control

Standards make results comparable and automatable

Priorities move daily: KEV, EPSS, exposure and inventory change

- So monitoring is part of assessment

Get authorisation in writing and document the Rules of Engagement



The key takeaways are important for this lecture. The first is the identity statement: an assessment answers 'what is exposed?' - it is neither an audit (which checks compliance) nor a penetration test (which proves impact). Get that distinction right and you will not misuse the word again.

Define scope and the four axes - method, vantage point, target, knowledge - before anything else, because everything downstream depends on that definition. Results are valid at a point in time, and coverage and false negatives bound their quality - a report that does not state its limitations is a report you should not trust.

Never confuse vulnerability severity with organisational risk: the CVSS number is not the work order; exposure, asset value and likelihood are. Remediation without verification is a hope, not a control - a fix is only done when the retest proves it. And the standards - SCAP with CPE, CVE, CVSS, CCE, XCCDF, OVAL, OCIL - are what make results comparable and automatable.

The final two are the ones that keep you out of trouble: priorities move daily, so KEV, EPSS, exposure and inventory change and monitoring is part of assessment; and get authorisation in writing and document the Rules of Engagement. The first keeps your findings current; the second keeps you legal. Hold both, and you are a professional.