
OWASP GLOBAL AppSec
VIENNA '26
JUN 25-26
25
years
of open source security

OWASP GLOBAL AppSec

VIENNA '26 JUN 25-26

25 years of open source security

WHAT OUR PEN TESTS NEVER FOUND – AND HOW ATTACKERS DID

BY: RAMYA M

AGENDA

- Introduction – The Paradox
- Failure Story: What Was Never Reported
- The Blind Spots: Vulnerability Classes That Pen Tests Commonly Miss
- How Attackers Actually Found Them
- Why Conventional Testing Misses These?
- What Changed After – Making Adversary-Aware Testing
- Tools and Techniques
- Mini-Interactive Exercise
- Conclusion and Key Takeaways
- Q&A

INTRODUCTION: THE PARADOX – PENTESTING VS REALITY

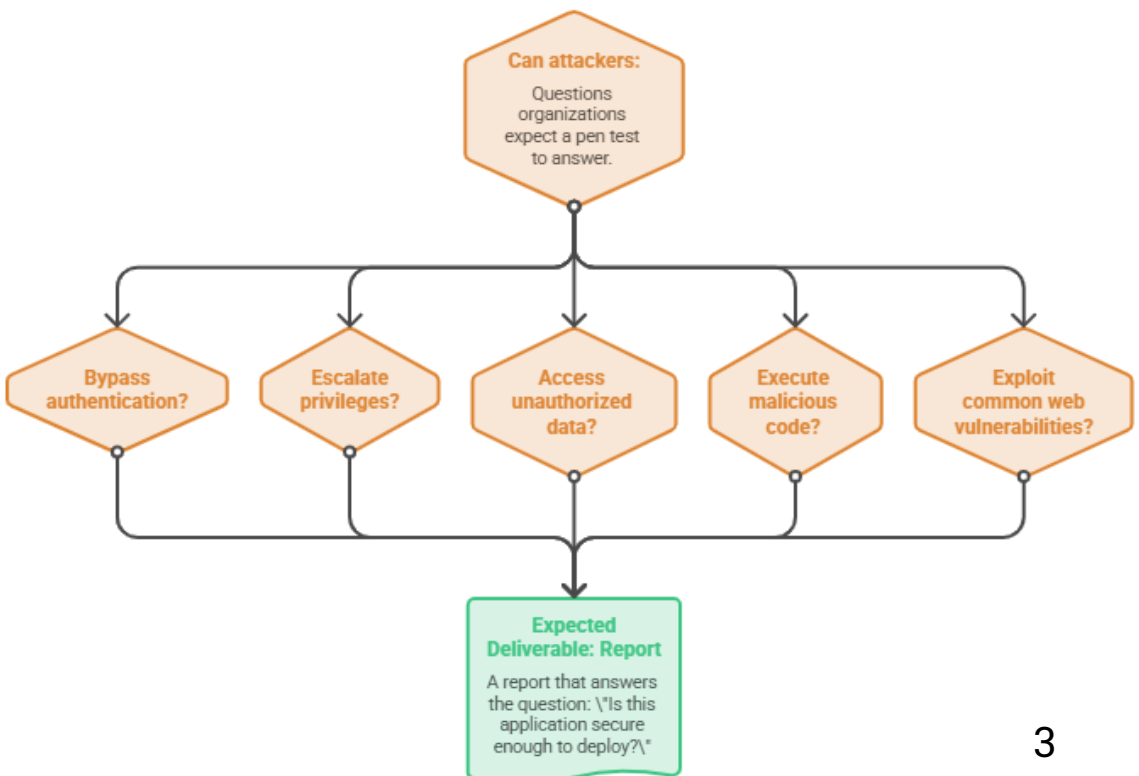
Current Security Posture

Security Vulnerabilities Emerge Over Time

- Threat Model Completed
- Secure Code Review
- Penetration Test
- No Critical Findings
- Security Sign-off

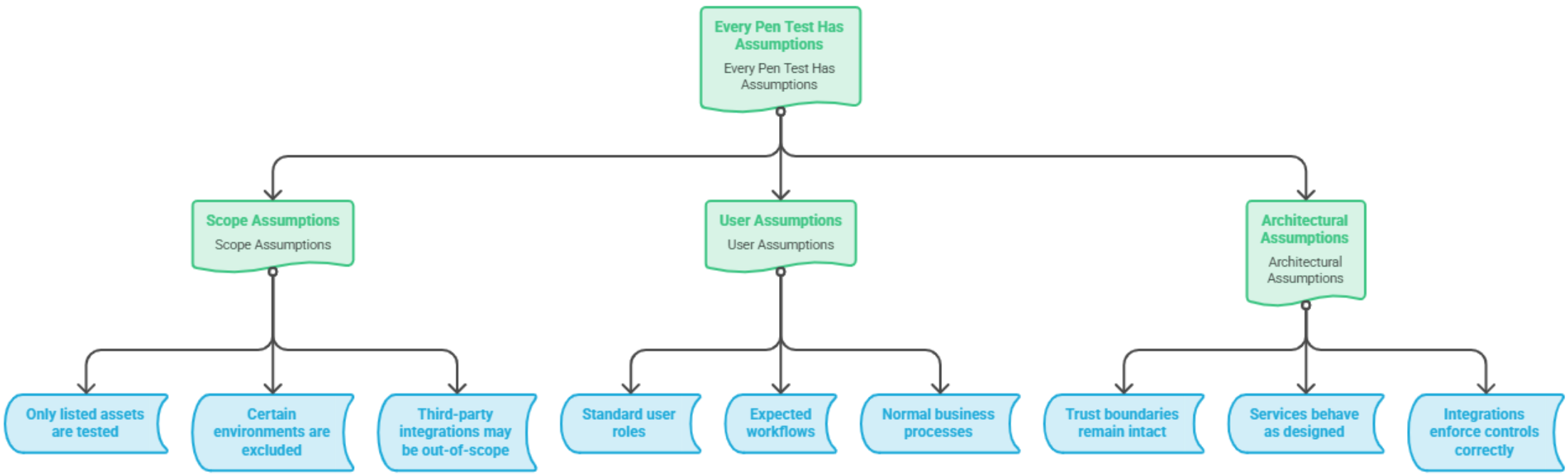
- Account Takeover
- Data Exposure
- Privilege Escalation
- Financial Fraud

Pen Test Expectations and Deliverables



Common Assumptions

Common Assumptions Built Into Pentesting



Time-boxed Pen-testing

The Reality of Time-Boxed Engagements

Pentest Timeline

Activity	Time
Recon	10–20%
Testing	60–70%
Validation	10–15%
Reporting	10–20%

Total Duration: **1–4 Weeks**

Attacker Timeline:
Weeks → Months → Years

How Organizations Measure Pen Test Success

Common Metrics

- Number of Critical finding
- Number of High findings
- OWASP Top 10 coverage
- Remediation closure rate
- Compliance requirements met

The Problem
Security becomes:
Finding Count = Risk

Why "No Critical/High Findings" Creates False Confidence?

"No Critical/High Findings" Does Not Mean

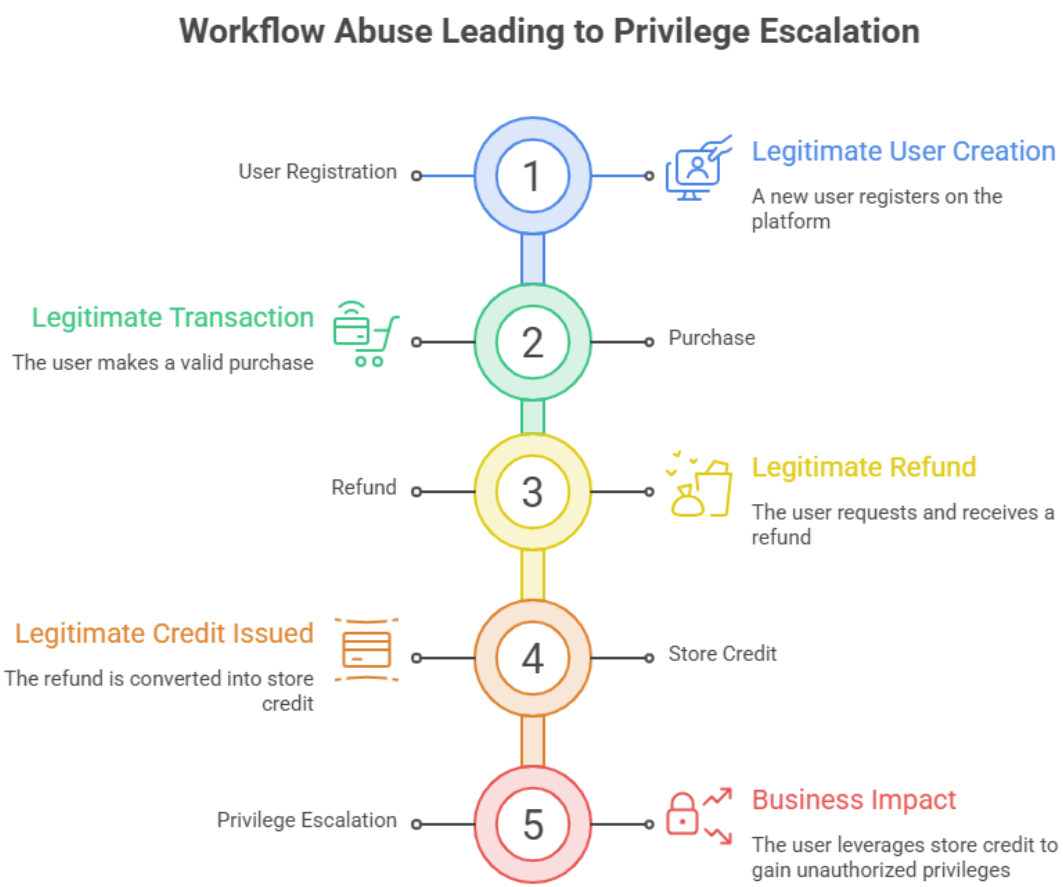
❌ No attack paths exist
❌ No business abuse exists
❌ No trust assumptions can fail
❌ No authorization gaps exist
❌ No architectural weaknesses exist

It Means

✔ No critical vulnerabilities were identified within the tested scope and timeframe

FAILURE STORY:
WHAT WAS NEVER REPORTED
&
THE 4 BLIND SPOTS (Vulnerability Classes That Pen Tests Commonly Miss)

Failure Story - No Vulnerability Required



In this scenario:

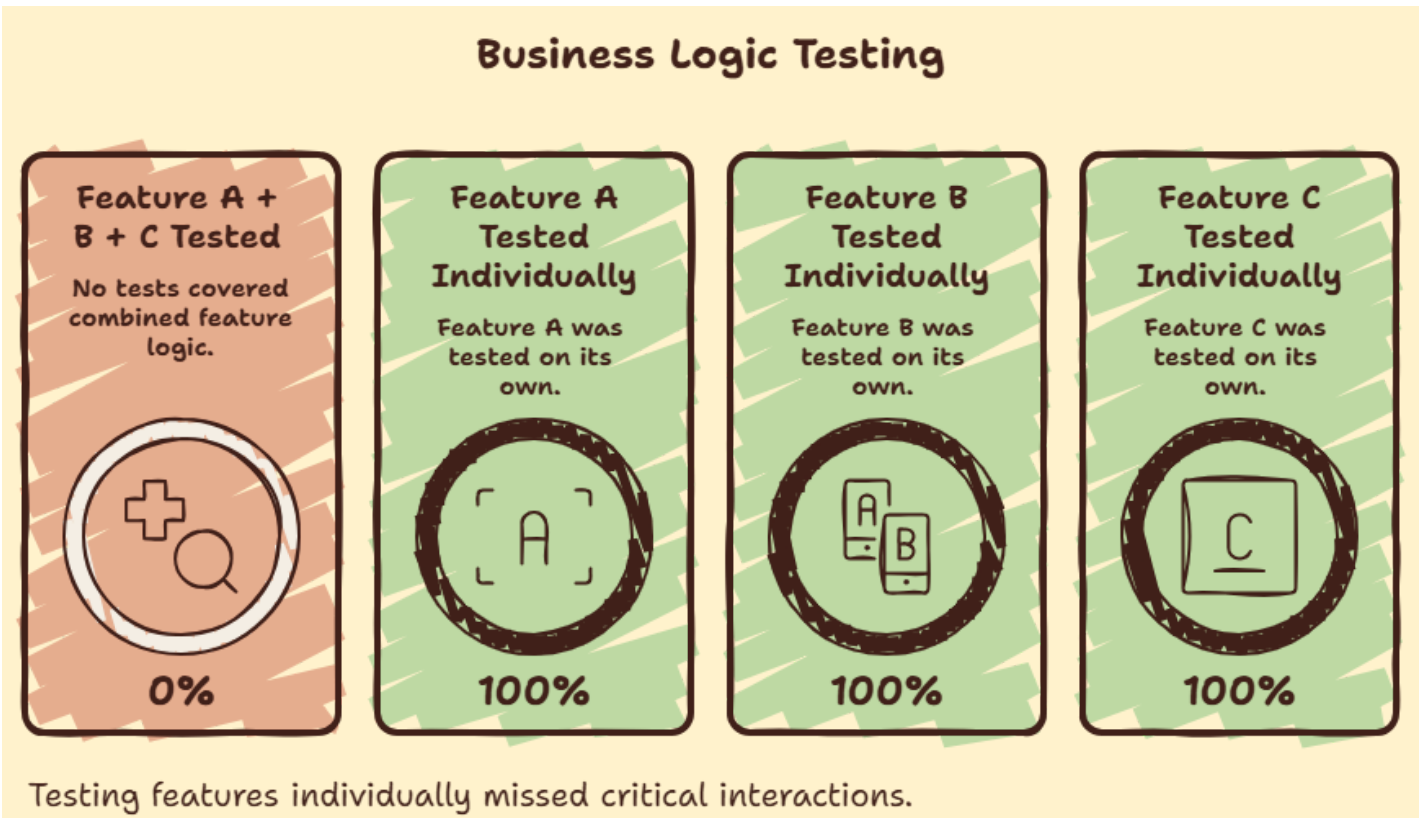
- Every step was legitimate
- Every API behaved correctly
- No control "failed"

Yet business impact occurred.

Lesson: Workflow abuse is not always vulnerability abuse.

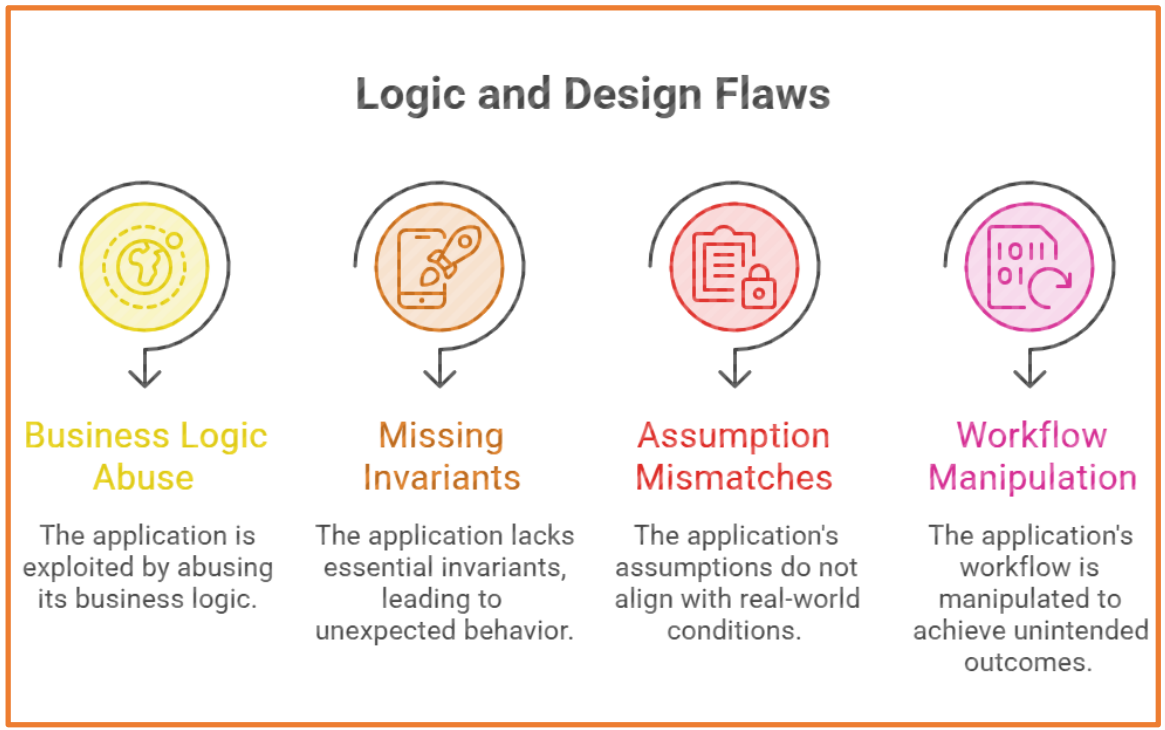
Blind Spot #1: Business Logic Testing

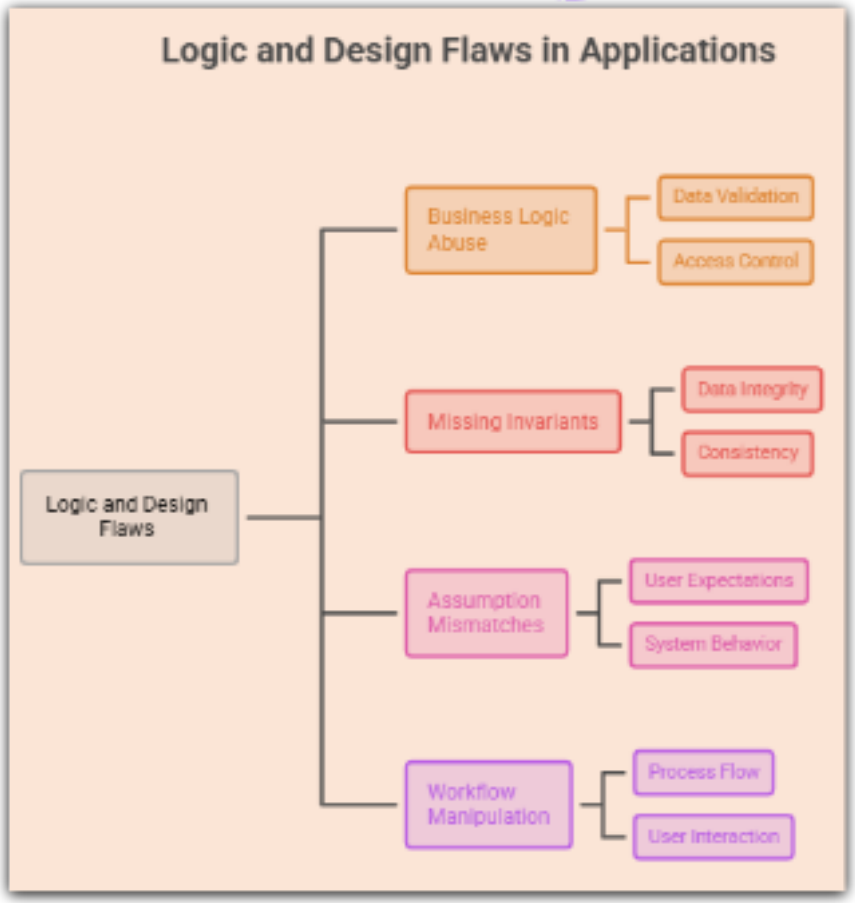
WHAT OUR PEN TESTS NEVER FOUND – AND HOW ATTACKERS DID



Logic & Design Flaws

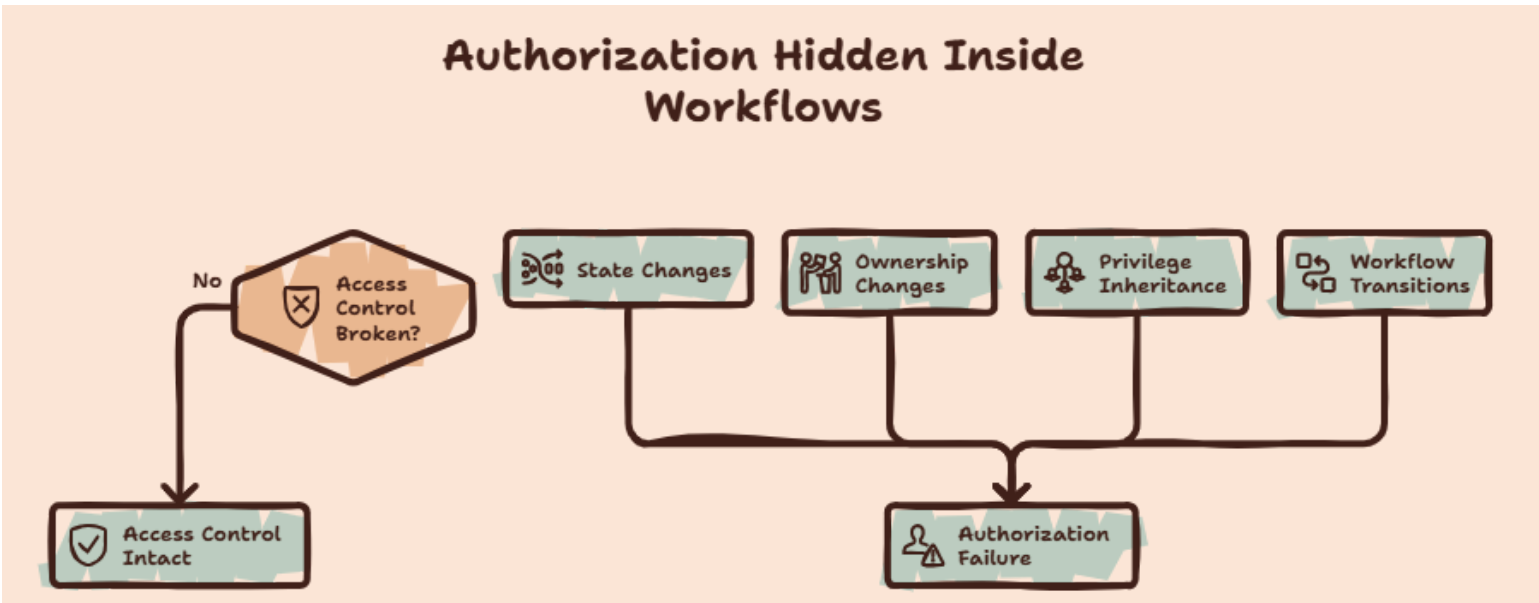
WHAT OUR PEN TESTS NEVER FOUND – AND HOW ATTACKERS DID





Blind Spot #2: Hidden Authorization Gaps

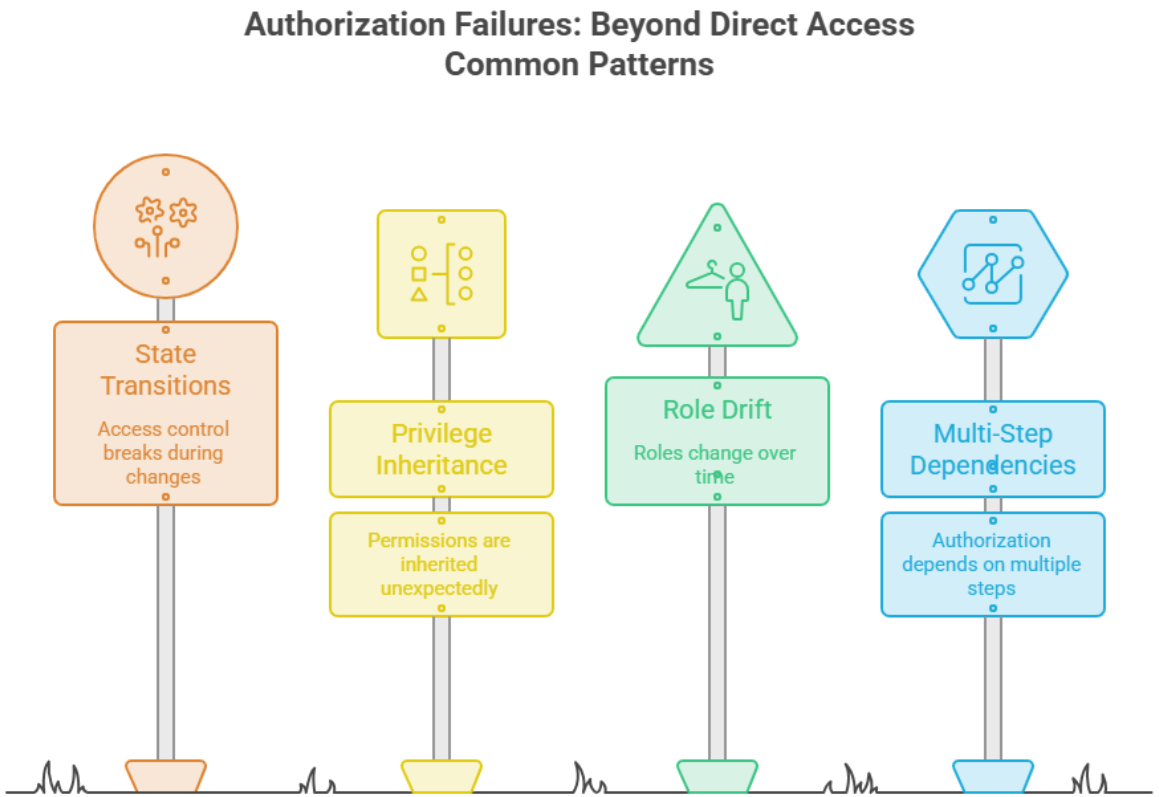
Authorization



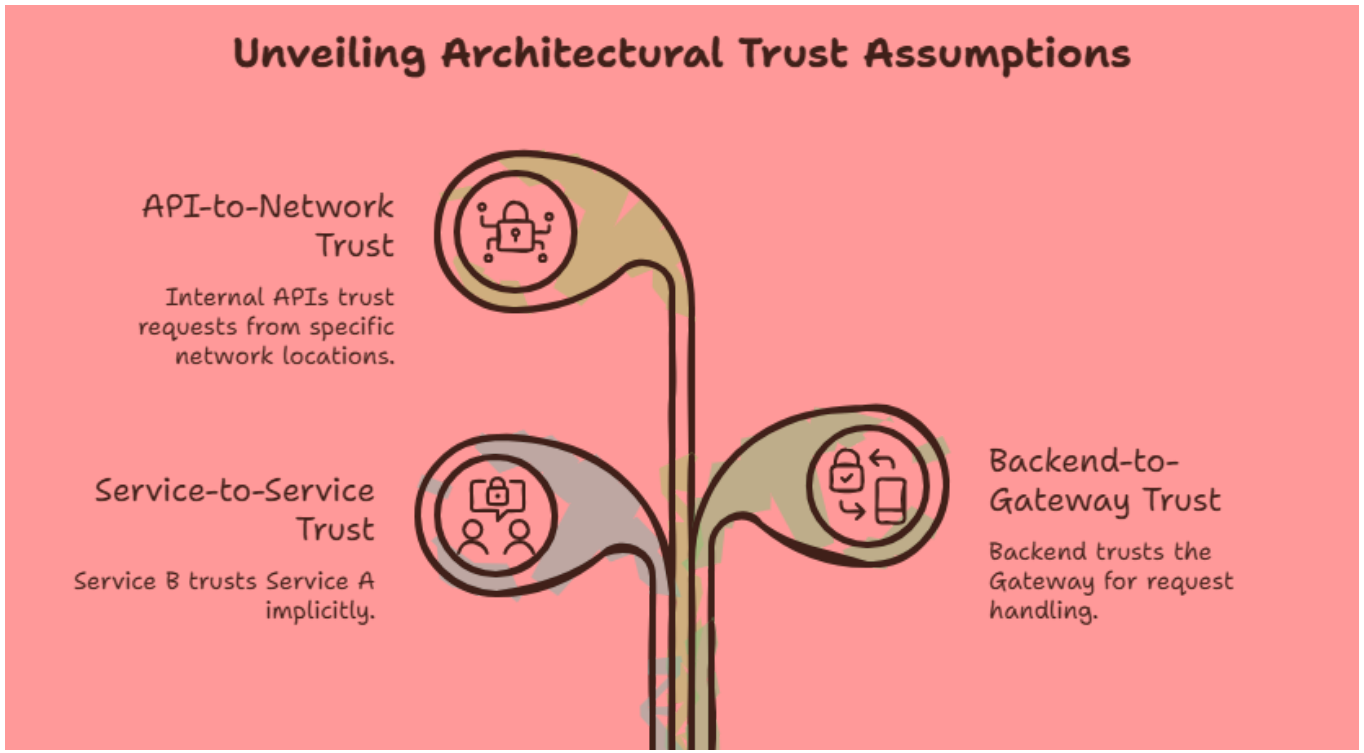
Authorization Failures Hidden in State Transitions

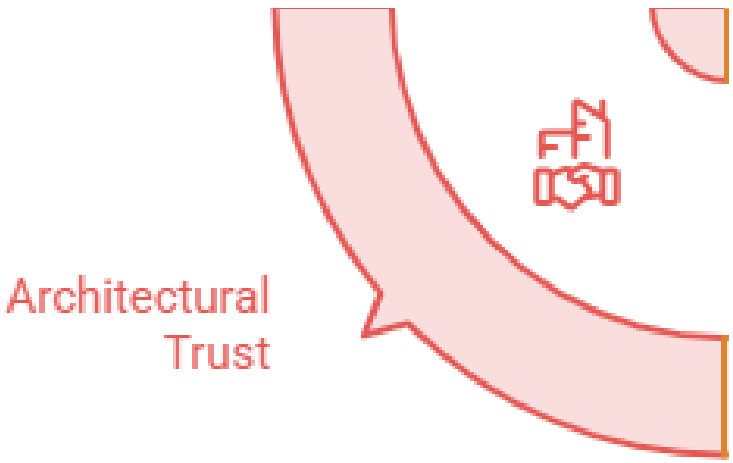
Traditional Testing Asks:
✓ Can User A access User B's resource?

Attackers Ask:
✓ Can User A become User B indirectly?
✓ Can permissions evolve unexpectedly?
✓ What happens during state changes?



Blind Spot #3: Trust Assumptions

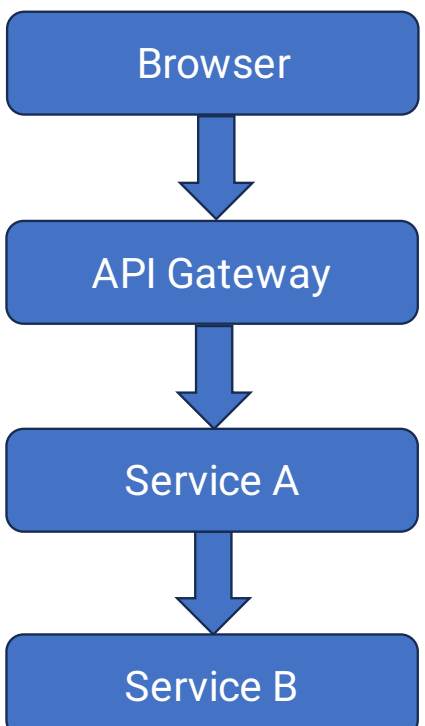


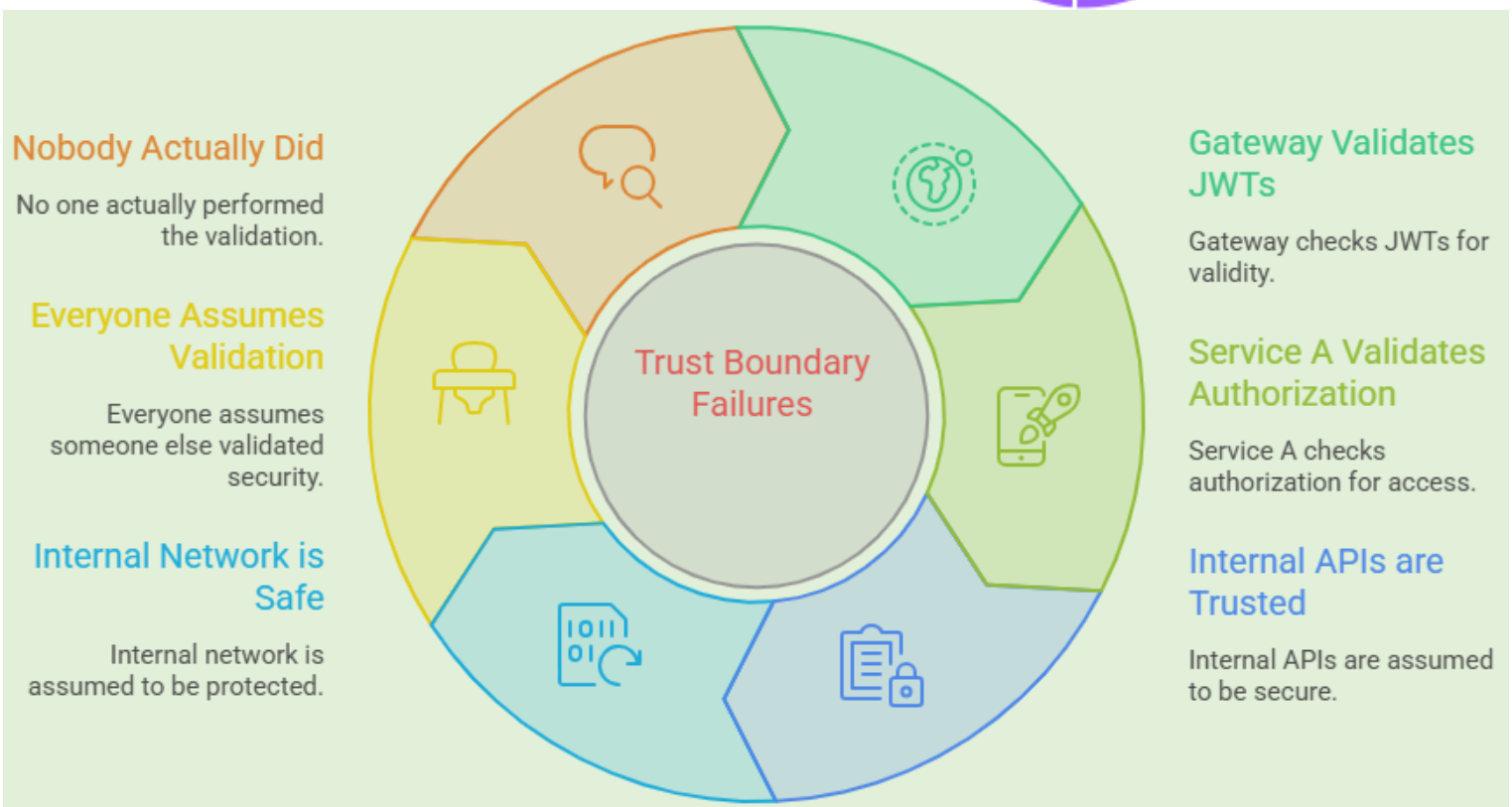


Trust Boundary & Integration Failures

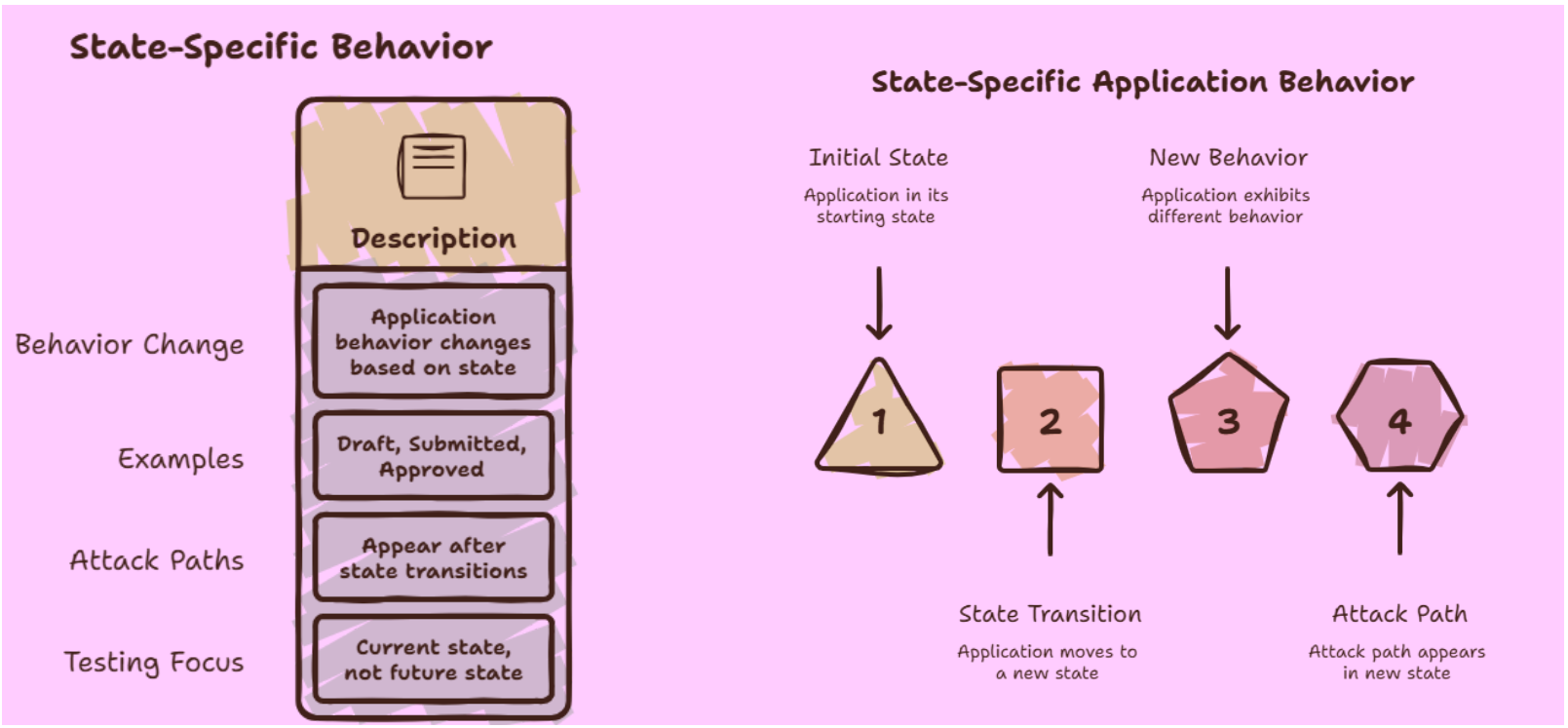
Pen Tests See Endpoints ---------- Attackers See Systems

Example:
Typical Architecture:



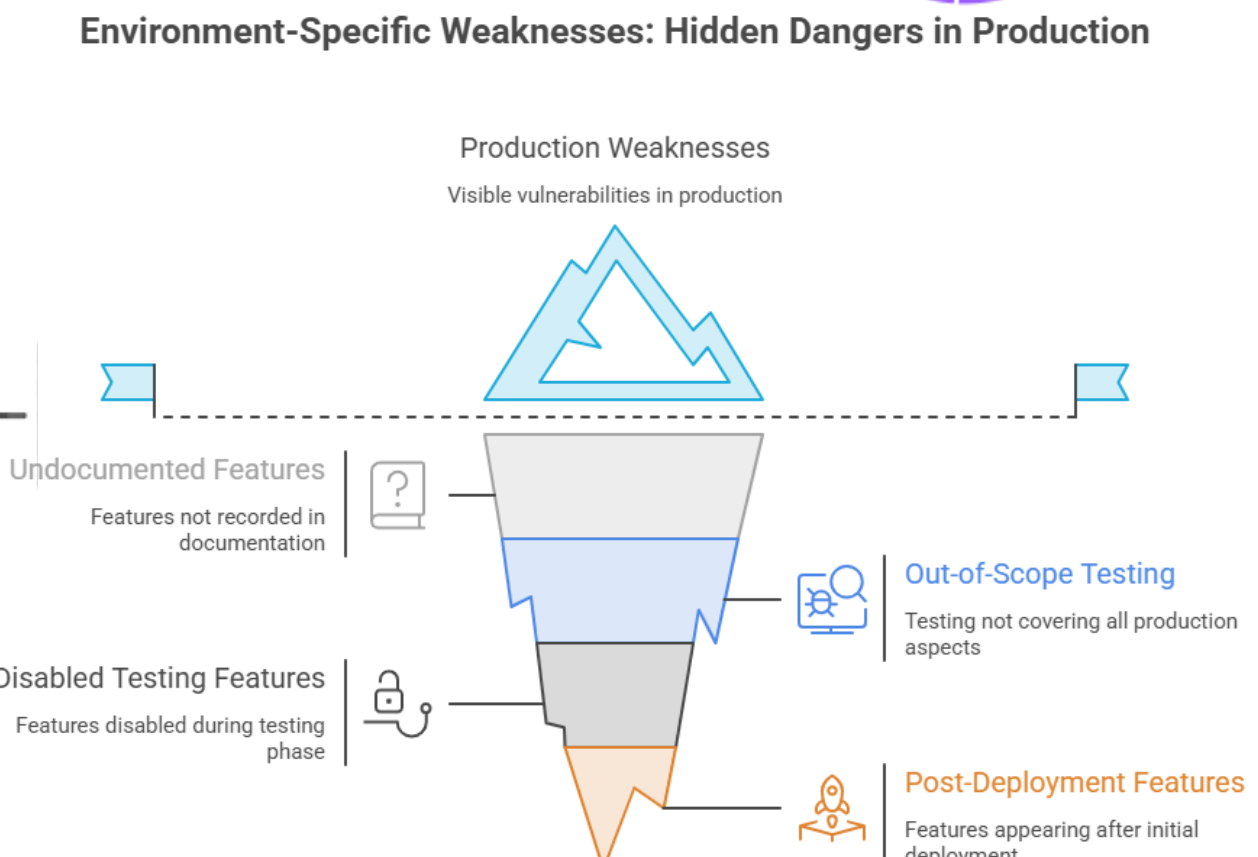


Blind Spot #4: State-Specific Problems



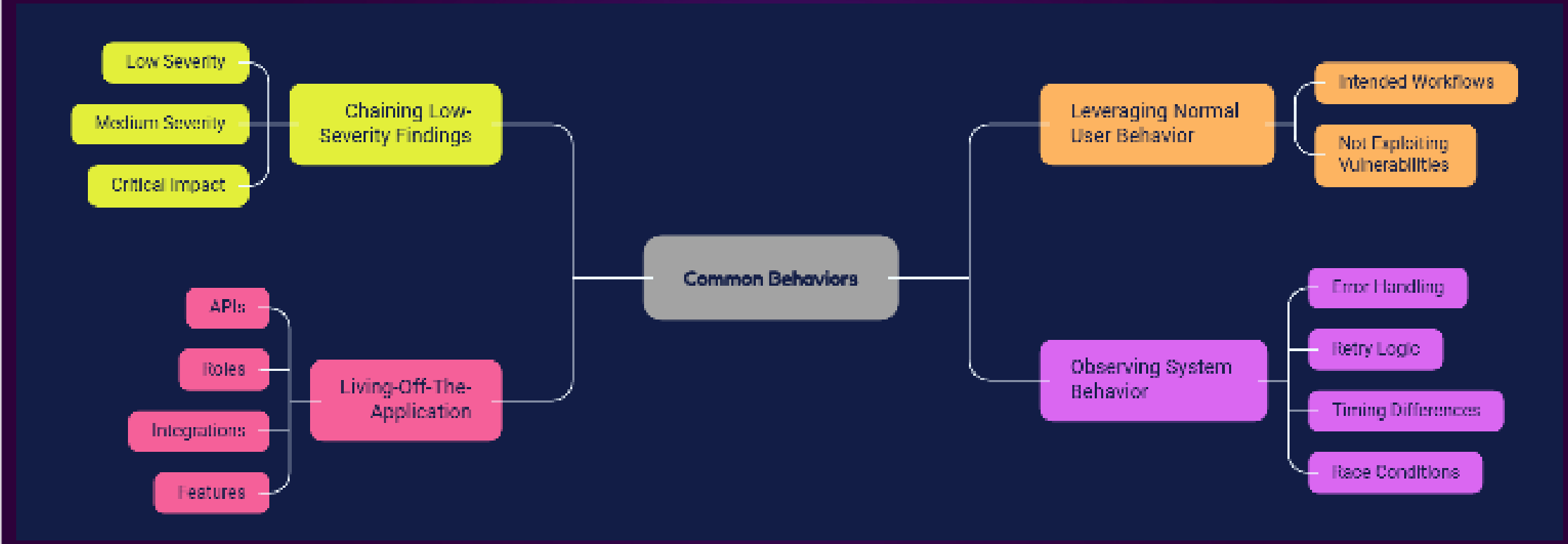
Environment-Specific Weaknesses

- Examples
  - Feature Flags
  - Debug Endpoints
  - Emergency Access Paths
  - Legacy APIs
  - Admin Workflows
  - Migration Utilities
  - Support Functions



HOW ATTACKERS ACTUALLY FOUND THEM?

How Attackers Discover These Paths?



WHY CONVENTIONAL TESTING MISSES THESE VULNERABILITIES?

The Common Answers for the "Why?"

Why did a security incident occur despite a strong security posture?

Unforeseen Vulnerability

A vulnerability was missed during the threat model and pen test.

Human Error

An employee made a mistake that led to the incident.

Sophisticated Attack

The attack was advanced and bypassed existing defenses.

Inadequate Policies

Existing security policies were insufficient to prevent the incident.

Why Do These Attack Paths Rarely Appear in Reports?

Because:

Pen-testing Is Optimized For:

✓ Finding vulnerabilities
✓ Validating controls
✓ Working within scope
✓ Producing actionable reports

Attackers Are Optimized For:

✓ Achieving objectives
✓ Finding pathways
✓ Breaking assumptions
✓ Creating impact

Over-Reliance on OWASP Top 10 Coverage

Focusing solely on the OWASP Top 10 can lead to neglecting other critical vulnerabilities.

- OWASP Top 10 is fundamentally a list of vulnerability categories.
- Attackers don't organize attacks according to vulnerability categories.
- They organize attacks according to objectives.

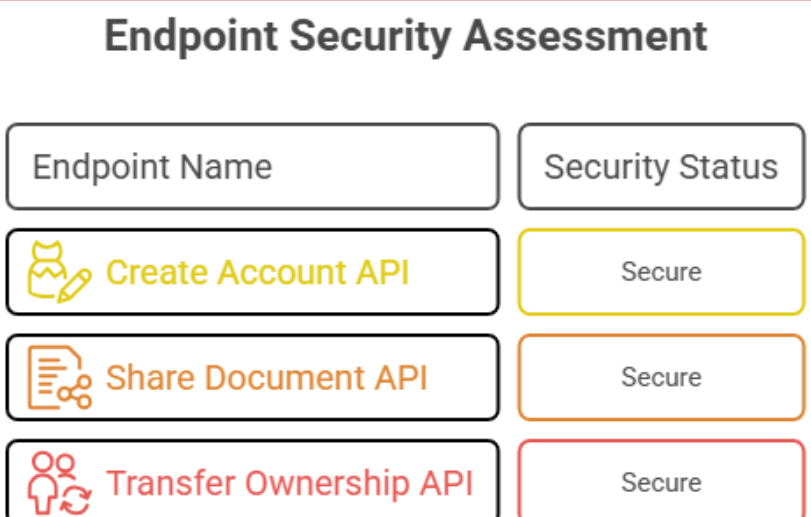
For example:

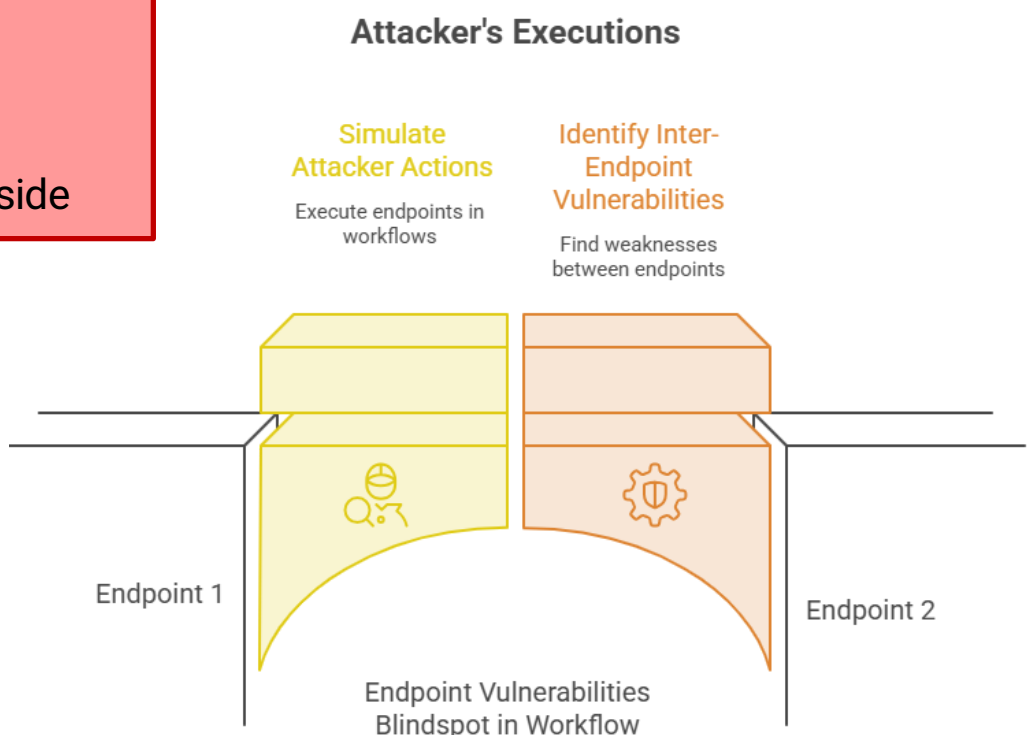
- An attacker wants:
  - Data access
  - Financial gain
  - Privilege escalation
- The path may involve:
  - Business logic
  - Workflow abuse
  - Trust assumptions
- None of which neatly fit a Top 10 category.

Testing Endpoints Instead of Flows

Testing individual endpoints without considering the entire user flow can miss complex attack vectors.

- This is probably the biggest blind spot.
- Many assessments evaluate endpoints independently.
- Individually, everything looks fine.
- But attackers don't execute endpoints in isolation.
- They execute workflows.
- The vulnerability often exists between endpoints, not inside them.





Missing Architectural Context

Understanding the application's architecture and trust boundaries is crucial for effective security testing.

This becomes especially important in modern architectures:

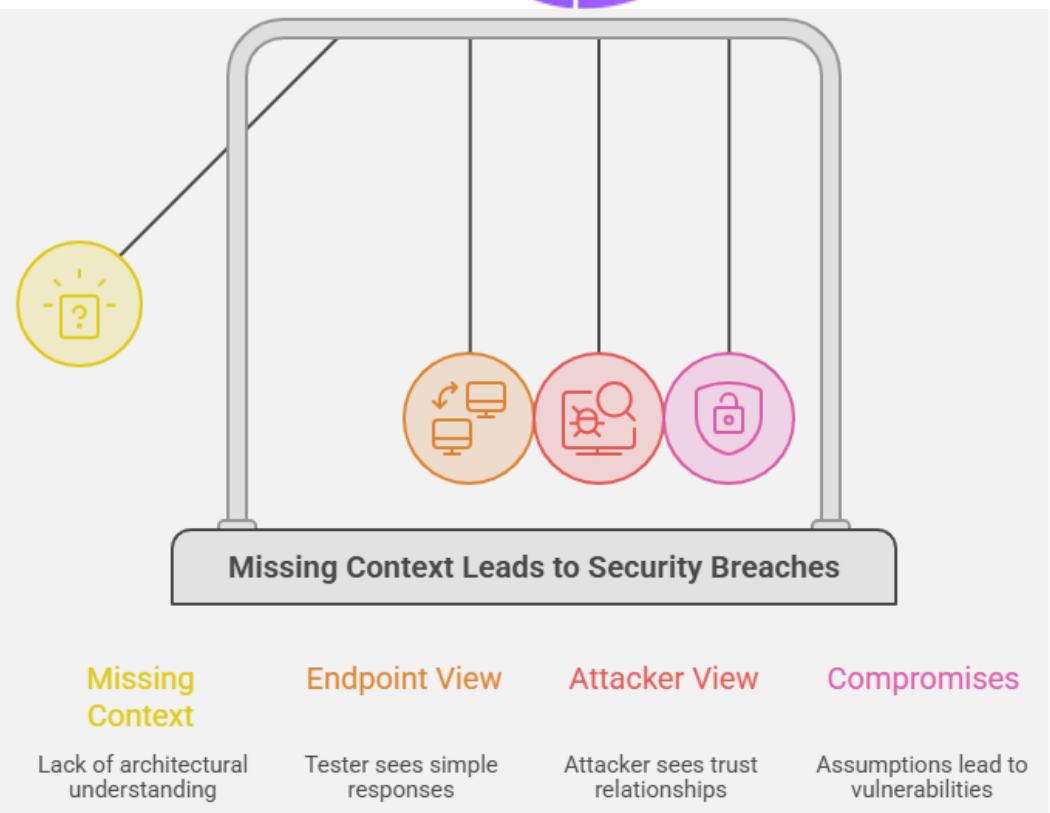
- Microservices
- Partner integrations
- API gateways
- Identity providers
- Service meshes

A tester may see:
Endpoint → Response
An attacker sees:
System → Trust Relationships → Impact

Many compromises occur because of assumptions:

- Gateway validates authorization
- Internal API is trusted
- Service identity cannot be spoofed

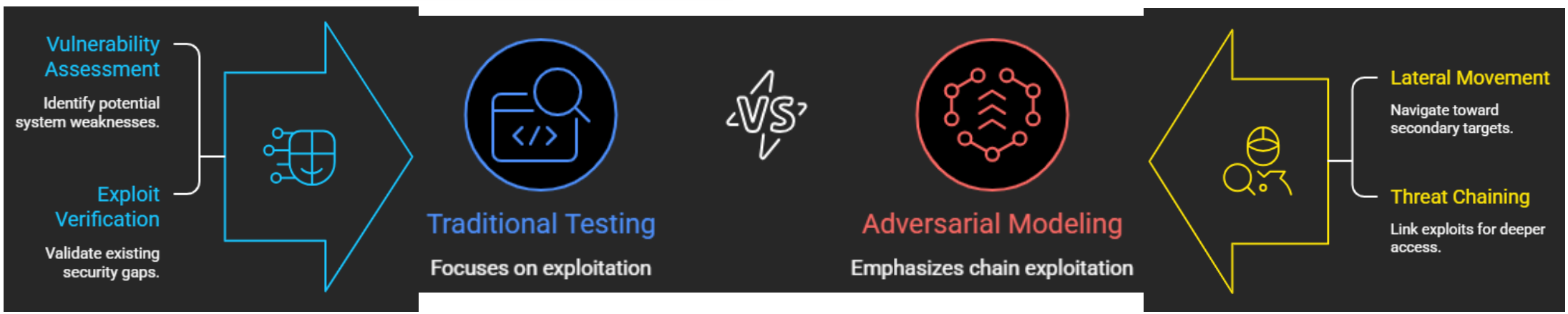
When those assumptions break, attack paths emerge.



Minimal Adversarial Modeling

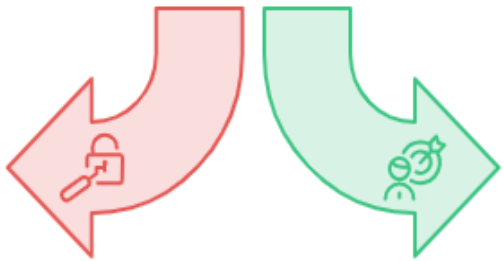
Failing to simulate real-world attacks can result in inadequate security measures.

We test:
"What vulnerabilities exist?"
We rarely test:
"What would an attacker try next?"



WHAT CHANGED: MAKING TESTING ADVERSARY-AWARE

From Vulnerability Testing to Attack-Path Testing



Before

"Can this control be bypassed?"

Focuses on bypassing controls, potentially overlooking achievable objectives.

- Threat Model
- Documentation
- Archive

After

"Can this objective be achieved?"

Emphasizes achieving objectives through abuse cases and pen testing.

- Threat Model
- Abuse Case
- Test Cases
- Pen-test

Example:

Threat Model Finding	Test Scenario
Approval Workflow Abuse	Attempt state-transition bypass
Shared Resource Ownership	Test privilege inheritance
Internal Service Trust	Validate gateway bypass assumptions
Feature Flag Risk	Verify access after feature activation

Five Practical Changes That Worked

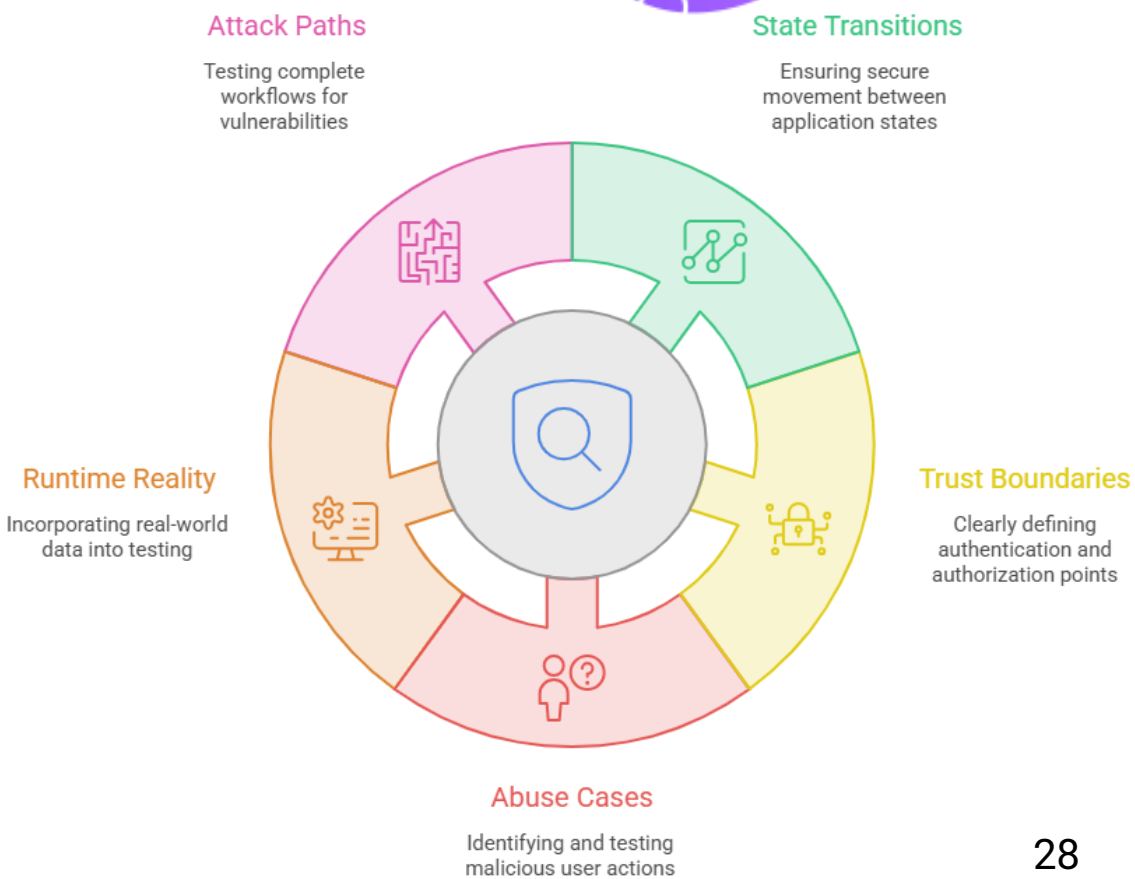
1. Test State Transitions, Not Just States
Can users move between states securely?

2. Map Trust Boundaries Explicitly
Where is authentication?
Where is authorization?
Who trusts whom?

3. Add Abuse Cases
"What could a malicious user do?"
Not just:

4. Use Runtime Reality
Logs
Incidents
Near Misses
Production Learnings

5. Build Attack Paths
Test workflows
Not individual endpoints



TOOLS & TECHNIQUES

Practical Techniques To Change Our Testing

Threat Modeling Artifacts

Use threat modeling artifacts as input for testing.

Authorization Matrix Validation

Validate the authorization matrix to ensure proper access control.

Attack Path Reconstruction

Reconstruct attack paths based on log data.

Manual API Exploration

Manually explore the API to find vulnerabilities.

Lightweight Automation

Automate flow validation with lightweight tools.

Adversary-Aware Testing Toolkit

Technique	Question It Answers
Manual API Exploration	What workflows exist beyond the UI?
Authorization Matrix Validation	Who can do what, and when?
Threat Models as Test Inputs	What attack paths should be tested?
Attack Path Reconstruction	How did the attacker actually move?
Flow Validation Automation	Can dangerous workflows be detected repeatedly?

MINI-INTERACTIVE EXERCISE

Features

✓ Register Account
✓ Buy Flight Ticket
✓ Upgrade Ticket
✓ Generate Boarding Pass QR
✓ Cancel Ticket
✓ Request Refund

Security Review

✓ Pen Test Completed
✓ No Critical Findings

Goal - Gain access to the premium lounge without paying for lounge access.

CONCLUSION & KEY TAKEAWAYS

CONCLUSION

- Penetration testing remains one of the most valuable security activities we perform.
- The challenge is not that penetration tests fail. The challenge is that attackers are solving a different problem.
- While we validate vulnerabilities, attackers search for outcomes. The gap between those two perspectives is where many real-world compromises occur.
- The goal is not to replace penetration testing. The goal is to make testing more adversary-aware by validating attack paths, trust assumptions, and business workflows before attackers do.

Key Takeaways

A Clean Pen Test Report Is Not a Security Guarantee

A penetration test answers:
"What vulnerabilities did we find?"
Attackers ask:
"What outcome can I achieve?"

Many successful attacks exploit workflows, permissions, and assumptions rather than traditional vulnerabilities.

Attackers Exploit Paths, Not Findings

Critical incidents often emerge from:

- Multiple low-severity issues
- Legitimate business workflows
- Trust boundary failures
- State-transition abuse

The individual findings may look harmless.
The attack path is where the real risk exists.

Make Testing Adversary-Aware

Before your next penetration test, ask:

- What are our critical business workflows?
- Where do permissions change?
- What trust assumptions exist?
- How could these actions be chained together?

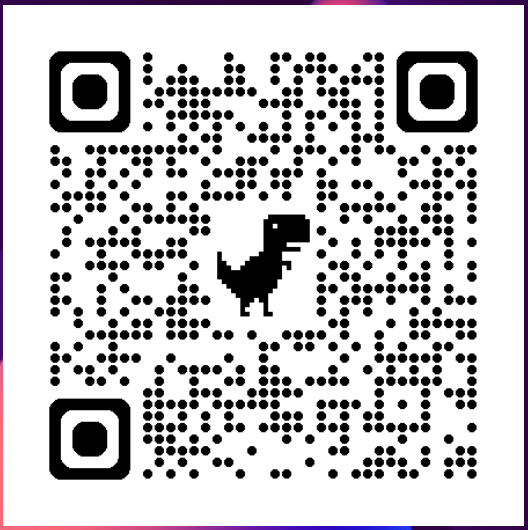
The most valuable test cases often come from threat models, architecture reviews, incidents, and abuse scenarios—not vulnerability lists alone.

OWASP GLOBAL AppSec

VIENNA'26 JUN 25-26

25 years of open source security

THANK YOU!



https://www.linkedin.com/in/ramya-m-482340128/