VULNERABILITY ASSESSMENT AND PENETRATION TESTING(23CY701)

UNIT - 3 NOTES

Managing a Penetration Test

Managing a penetration test involves careful planning, execution, communication, and reporting to ensure the assessment is effective, ethical, and aligned with organizational objectives.

---

1. Planning a Penetration Test

Objectives

Define what the organization wants to achieve:

- Identify security vulnerabilities
- Validate security controls
- Meet compliance requirements
- Assess security posture

Scope Definition

Determine:

- Systems to be tested
- Networks and applications involved
- Physical locations (if applicable)
- Testing limitations

Rules of Engagement (RoE)

Establish:

- Testing schedule
- Authorized activities
- Emergency contacts
- Communication procedures
- Success criteria

Risk Assessment

Evaluate potential business impacts and plan mitigation strategies.

Key Deliverable

📋 **Penetration Testing Plan**

2. Structuring a Penetration Test

Typical Team Structure

👩‍💼 Project Manager

Coordinates activities and communications.

🛡️ Lead Penetration Tester

Oversees technical testing activities.

💻 Security Analysts

Perform assessments and document findings.

🏢 Client Representatives

Provide access and approve testing activities.

---

3. Execution of a Penetration Test

Phase 1: Information Gathering

Collect information about:

- Systems
- Applications
- Networks

- Security controls

Phase 2: Vulnerability Assessment

Identify weaknesses such as:

- Misconfigurations
- Outdated software
- Weak authentication controls

Phase 3: Validation Testing

Safely verify whether identified vulnerabilities are exploitable within the approved scope.

Phase 4: Post-Assessment Analysis

Evaluate:

- Business impact
- Access levels
- Security control effectiveness

Phase 5: Evidence Collection

Document findings with:

- Screenshots
- Logs
- Technical notes

---

4. Information Sharing During a Penetration Test

Effective communication is critical throughout the engagement.

Communication Channels

📞 Regular Status Meetings

Provide progress updates.

📧 Incident Notifications

Report critical findings immediately.

📊 Progress Reports

Share testing status and observations.

🚨 Escalation Procedures

Address unexpected issues or high-risk discoveries.

Information Security Requirements

- Protect sensitive data
- Follow confidentiality agreements
- Limit access to assessment information
- Secure communication channels

Benefits

✅ Improved coordination

✅ Faster response to critical findings

✅ Reduced operational risk

---

5. Reporting the Results of a Penetration Test

The final report is one of the most important deliverables.

Executive Summary

Designed for management:

- Overall security posture
- Major risks identified
- Business impact

Technical Findings

Detailed information about:

- Vulnerabilities discovered
- Evidence collected
- Risk ratings

Risk Assessment

Common severity levels:

🟥 Critical

□High

□Medium

□Low

Recommendations

Provide:

- Remediation steps
- Security improvements
- Prioritized action plans

Retesting Results

Verify whether vulnerabilities have been successfully remediated.

Best Practices

✓ Obtain written authorization before testing

VULNERABILITY ASSESSMENT AND PENETRATION TESTING(23CY701)

✓ Clearly define scope and objectives

✓ Maintain continuous communication

✓ Document all activities and findings

✓ Protect confidential information

✓ Provide actionable recommendations

✓ Conduct retesting after remediation

Basic Linux Exploits

1. Stack Operations

What is a Stack?

A **stack** is a special memory area used by a program to store temporary information while it runs.

Think of it like a **stack of plates**:

🍽 Put a plate on top → **PUSH**

🍽 Remove the top plate → **POP**

The last item placed on the stack is the first one removed.

Why is it Important?

The stack stores:

- Function information
- Temporary data
- Return locations for programs

If the stack is corrupted, a program may crash or behave unexpectedly.

---

2. Buffer Overflows

What is a Buffer?

A **buffer** is a small memory space used to store data temporarily.

Example:

A box can hold **10 balls**.

📦 Capacity = 10 balls

What is a Buffer Overflow?

If you try to put **20 balls** into the same box:

📦 ➜ 💥 Overflow

The extra balls spill outside the box.

Similarly, when a program receives more data than its buffer can hold, it may overwrite nearby memory.

Effects

❌ Program crashes

❌ Data corruption

❌ Security vulnerabilities

---

3. Local Buffer Overflow Exploits

What Does "Local" Mean?

The person must already have access to the computer.

A local vulnerability exists inside a program running on that system.

Example

Suppose a login application accepts only 20 characters.

If it does not properly check input length, very long input may cause:

- Program failure
- Memory corruption
- Unexpected behavior

Why Is It Dangerous?

An attacker with local access may try to misuse such weaknesses to gain higher privileges or disrupt the system.

---

4. Exploit Development Process

Security researchers study vulnerabilities to understand and fix them.

Step 1: Find a Vulnerability

🔍 Discover a weakness in a program.

Step 2: Analyze the Problem

🧠 Understand why the vulnerability exists.

Step 3: Create a Test

🧪 Safely demonstrate the issue in a controlled environment.

Step 4: Assess the Risk

📊 Determine how serious the vulnerability is.

Step 5: Develop a Fix

🔧 Correct the coding error.

Step 6: Test the Fix

✅ Ensure the vulnerability is removed.

Step 7: Release the Patch

📦 Update the software to protect users.

Modern Linux Protections

Linux includes built-in security features:

- ASLR

Randomly changes memory locations to make attacks harder.

- Stack Canaries

Detects unexpected changes in stack memory.

- NX (No Execute)

Prevents programs from running code in unsafe memory areas.

- SELinux/AppArmor

Adds extra security controls to protect the system.

Windows Exploits

What Are Windows Exploits?

A **Windows exploit** is a method used to take advantage of a weakness (vulnerability) in a Windows program or operating system. Security researchers study these weaknesses to help developers fix them and improve security.

1. Compiling and Debugging Windows Programs

Compiling

Compiling means converting source code into a Windows executable file (.exe) that can run on a computer.

Debugging

Debugging means finding and fixing errors in a program. Security researchers use debuggers to understand how a program behaves and why it crashes.

2. Writing Windows Exploits

Security researchers analyze vulnerable programs to understand:

- What causes the vulnerability
- How serious it is
- How it can be fixed

The purpose is to improve security and develop patches.

---

3. Understanding Structured Exception Handling (SEH)

SEH (Structured Exception Handling) is Windows' error-handling system.

When a program encounters a problem, such as:

- Invalid memory access
- Missing files
- Divide-by-zero errors

SEH helps the program manage the error instead of crashing immediately.

---

4. Understanding Windows Memory Protections

Modern Windows versions include built-in security features.

DEP (Data Execution Prevention)

Prevents code from running in memory areas meant only for data.

ASLR (Address Space Layout Randomization)

Randomly changes memory locations, making attacks more difficult.

SafeSEH

Protects the Windows exception-handling process.

SEHOP

Provides additional protection for exception handling.

Stack Cookies

Detects memory corruption and helps stop attacks before they succeed.

---

5. Bypassing Windows Memory Protections

Security experts study how attackers might attempt to overcome security protections. This helps software developers:

- Discover weaknesses
- Improve defenses
- Release security updates
- Better protect users