Software and System Security 2 - S8 FS25

Raphael Nambiar

Version: 13. Juni 2025

SECURING INFORMATION SYSTEMS

Information System

Definition: Structured set of components to collect, process, store, communicate information

- Applications, services, IT assets
- Software, hardware
- Data, methods, procedures
- People (users, operators)

Information Security Management System (ISMS)

Definition: Structured approach to manage information security

- Risk management framework
- Includes: people, processes, technology
- Goal: keep risks at acceptable levels
- Implemented by management (typically CISO)
- Checklist-style, high abstraction
- Not a technical solution

Security Controls

Definition: Countermeasures to reduce, detect, respond to risks

- **Types:**
  - Preventive - stop incidents (e.g., firewalls, auth)
  - Detective - identify incidents (e.g., IDS)
  - Corrective - limit damage (e.g., backups)
- **Attributes:**
  - Security Property: CIA
  - Function: Identify, Protect, Detect, Respond, Recover
  - Category: People, Physical, Technology, Organizational

ISO 27000 Series

ISO 27001: Lists high-level controls (e.g., disposal, network security)
ISO 27002: Implementation guidance for ISO 27001 controls

- Abstract, generic - industry-independent
- Checklist-like reference
- Example: Malware protection - anti-virus, user training

CIS Controls

Best-practice guidelines whose development started in 2008
Definition: Practical, prioritized controls from real-world attacks

- **Groups:**
  - IG1 - Basic hygiene (SMEs)
  - IG2 - Mid-level, enterprise-grade
  - IG3 - Advanced protection, targeted threats
- **Examples:**
  - CSC 1 - Inventory of devices (active + passive)
  - CSC 2 - Inventory of software (whitelisting)
  - CSC 7 - Continuous vuln. management (scanners, patching)

Measuring Security

Challenge: Measuring security = hard / approximate

- **Methods:**
  - Audits (compliance vs. standards)
  - Penetration testing
  - Risk = Likelihood × Impact
- **Metrics:**
  - % vulnerabilities patched in time (NIST SP 800-55)
  - Ratio blocked/successful malware (ISO 27004)
- **Purpose:**
  - Assess control effectiveness
  - Demonstrate compliance
  - Guide security decisions

Key Takeaways

- Securing systems = people + process + tech
- ISMS / CIS = frameworks, not full solutions
- Controls must be context-specific + prioritized
- Measuring helps track + improve security posture

Threat Landscape

Definition

Definition: Collection of threats in a domain/context

- Focus: Threat types, agents, vectors (not mitigations)
- Supports risk evaluation:
  - Risk = Threat × Vulnerability × Consequence
  - Risk = Likelihood × Impact

Threat Agents

Attributes: Motivation, Resources, Skill, Role

- **Cyber Criminals:** money,secrets, medium-high skill/resources, *-as-a-Service
- **Online Social Hackers:** High social, low-medium tech skill, psychology-based attacks
- **Cyber Spies:** State/corp, espionage, very high skill/resources
- **Employees:** Insider threat, low-medium skill, intentional/unintentional
- **Script Kiddies:** Low skill, use public tools, motive: fun/fame
- **Others:**
  - Hacktivists - political/societal goals
  - Cyber Fighters - nationalists (non-state)
  - Cyber Terrorists - fear/political damage

Cyber Kill Chain

7 Steps of an Attack:

1. Reconnaissance - gather info
2. Weaponization - create exploit + payload
3. Delivery - transmit payload (email, USB...)
4. Exploitation - trigger vuln.
5. Installation - install malware
6. Command & Control - remote channel
7. Actions on Objectives - data theft, damage

Defenders can break the chain at any step.

Security Controls & SIEM

Fundamental Control Principles

- **Least Privilege** – minimum necessary access
- **Fail-Safe Defaults** – deny by default
- **Complete Mediation** – every access checked
- **Separation of Privilege** – multiple conditions for access
- **Least Common Mechanism** – minimize shared components
- **Open Design** – transparency over obscurity
- **Psychological Acceptability** – usability of security
- **Goal:** reduce attack surface, enforce secure defaults

SIEM Overview

Definition: SIEM = Security Information & Event Management

- Collects, normalizes, stores, correlates, and analyzes security data
- Central component of SOC (Security Operations Center)
- Supports detection, alerting, forensic analysis
- Dashboards, queries, incident timelines

SIEM Components

- **Sensors:** Sources that generate security-relevant data for the SIEM
  - **NIDS (Network Intrusion Detection System):** Monitors network traffic for anomalies (e.g., Snort, Suricata)
  - **HIDS (Host Intrusion Detection System):** Monitors system-level activity like file access, login attempts (e.g., OSSEC)
- **Log Collection & Normalization:**

  - Collect logs from various sources (firewalls, servers, applications)
  - Normalize into a common structured format (fields: timestamp, source IP, event type, etc.)
  - Enables correlation and efficient querying
- **Asset Inventory:**
  - List of known systems, owners, IPs, roles, and criticality
  - Provides essential context for alerts and triage
  - Supports prioritization of incidents and reduces false positives
- **Vulnerability Scanner:**
  - Scans systems for known weaknesses (CVEs – Common Vulnerabilities and Exposures)
  - Tools: Nessus, OpenVAS
  - Results feed into SIEM to help prioritize alerts
- **Correlation Engine:**
  - Central logic unit that links related events to detect complex attacks
  - *Simple rule:* 5 failed logins → brute force detection
  - *Complex rule:* new login location + privilege change + file access = suspicious behavior
  - Enables detection of attacker TTPs (Tactics, Techniques, Procedures)

Pyramid of Pain

- Defense model: higher levels = harder for attacker to adapt
- Indicators (low to high): Hashes, IPs, Domains, TTPs
- Goal: detect & disrupt attacker TTPs, not just IOCs

SIEM Lab Summary

Will not be tested in the exam.

Security Testing (Part 1)

Security Testing Methods

Purpose: Identify, assess, and improve security posture
Methods:

- **Vulnerability Scanning** – Automated tools for known vulns (e.g., OpenVAS, LGTM)
- **Penetration Testing** – Manual & tool-assisted attack simulation to find & prove risks
- **Red Teaming** – Simulate real attackers to test detection/response across all layers
- **Purple Teaming** – Red & Blue collaboration to improve detection & response
- **Breach & Attack Simulation (BAS)** – Automated, scripted attack scenarios (e.g., MITRE ATT&CK)
- **Bug Bounty** – Crowdsourced testing (public/private), pay-per-find

Comparison:

- **Scanning:** Known vulns in 3rd-party apps/infrastructure
- **Pentesting:** Custom/web apps, focused scope
- **Red Team:** Test defenses (SOC), full attack paths
- **Purple/BAS:** Improve detection, develop new rules
- **Bug Bounty:** Live targets, continuous findings, public feedback

Penetration Testing

Definition:

- Simulated attack to discover exploitable vulnerabilities and evaluate risk
- NIST: Mimic real-world attacks to bypass security mechanisms

Motivations:

- Uncover weaknesses missed by automated tools
- Validate defense mechanisms & configurations
- Raise awareness, justify security budgets
- Fulfill compliance (e.g., PCI-DSS, HIPAA)

Scope Targets:

- IT Assets – Web apps, networks, infrastructure
- Data – Customer info, credentials
- Physical – Building entry
- Social – Phishing, manipulation

Success Factors: Skills, creativity, tools, lateral thinking

Penetration Testing Methodologies

- **OSSTMM** – Full-spectrum testing, formalized scoring model
- **OWASP Testing Guide** – Web app testing procedures & tools
- **NIST SP 800-115** – General framework, tools, validation
- **PTES** – Practical industry guide (incomplete/outdated)

Other resources: SANS checklists, training materials

Pentest Phases

1. **Pre-engagement** – Define scope, methods, rules, contacts
2. **Intelligence Gathering** – Collect public/recon info
3. **Threat Modeling** – Map potential attack paths
4. **Vulnerability Analysis** – Identify exploitable issues
5. **Exploitation** – Gain access or demonstrate impact
6. **Post-Exploitation** – Lateral movement, persistence
7. **Reporting** – Document findings, risk, mitigation

Pre-Engagement Phase

Scope:

- What systems, techniques, and depth of testing
- Channels: physical, human, network, wireless, telecom
- Define inclusions/exclusions (e.g., „all except billing module")

Rules of Engagement:

- Test windows (e.g., 20:00–06:00), backup constraints
- Use of stealth/evasion (depends on method: black/gray/white box)
- Evidence handling – encrypted, need-to-know access
- **Permission to Test Document** – mandatory, defines scope, 3rd party authorization, liability

Communication:

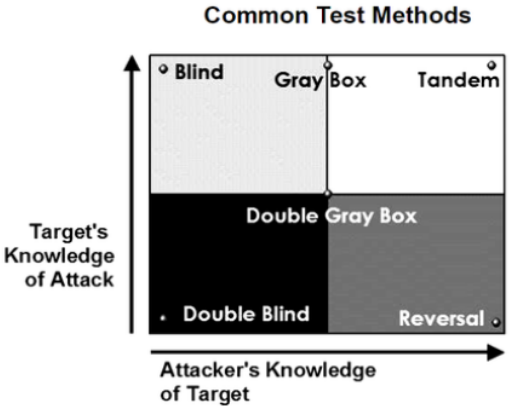
- Define secure channels (e.g., file sharing, IM, phone)
- Emergency contacts for incident handling
- Frequency of status reporting

Pitfalls:

- Clients unclear on purpose (real risk vs checkbox)
- Scope Creep – informal extensions must be managed properly

Common Test Models (OSSTMM)

- **Blind** – Testers get no info (like attackers)
- **Double Blind** – Even defenders don't know
- **Gray Box** – Limited internal info shared
- **White Box** – Full internal info shared
- **Crystal Box / Tandem** – Collaboration with client
- **Reversal / Red Teaming** – Realistic adversary simulation



Evidence Handling

- Avoid storing PII/PHI unless necessary
- Prove access via: screenshots, permission lists, flags
- All data must be encrypted and access-limited

Testing Method Comparison

0.0.1 Vulnerability Scanning

- **What:** Automated scanning for known vulnerabilities using signatures
- **Purpose:** Identify known vulnerabilities early
- **Compliance:** GDPR, HIPAA, PCI-DSS
- **Assets:** Source code, applications, infrastructure
- **Result:** List of potential vulns + risk rating
- **Method:** Tools like OpenVAS, LGTM; fully automated
- **Requirement:** Vulnerability mgmt. capability (triage + patching)
- **Frequency:** Continuous (due to changing signatures and assets)

0.0.2 Classical Penetration Testing

- **What:** Ethical hacking to discover and verify vulnerabilities
- **Purpose:** Find easy-to-moderate vulns + remediation advice
- **Assets:** Limited scope (app, service, system)
- **Result:** Verified vulns, risk scores, how-to-fix
- **Method:** Manual + tools (OWASP Testing Guide)
- **Requirement:** Test environment + vuln mgmt.
- **Frequency:** 1–4×/year or per release cycle

0.0.3 Red Team Testing

- **What:** Realistic attack simulation to test detection/response
- **Purpose:** Measure SOC effectiveness and incident handling
- **Assets:** Broad – physical, human, cyber layers
- **Goal:** Achieve mission (e.g., steal data) without detection
- **Result:** Goal outcome + detailed attack path
- **Method:** Custom attack scenarios (may include social engineering)
- **Requirement:** Mature security org (IR, SOC, controls)
- **Frequency:** Periodic (e.g., annually)

0.0.4 Purple Team Testing

- **What:** Red + Blue collaboration for better detection/prevention
- **Purpose:** Improve SOC rules, detection logic, tuning
- **Assets:** Selected systems, employee targets
- **Result:** Improved detection rules, hardening plans
- **Method:** Controlled attack simulation + feedback loop
- **Requirement:** Cross-team collaboration
- **Frequency:** Periodic (e.g., quarterly)

0.0.5 Breach & Attack Simulation (BAS)

- **What:** Continuous, automated kill chain simulation
- **Purpose:** Evaluate SOC resilience using known attack paths
- **Assets:** Based on selected attack scripts (e.g., MITRE ATT&CK)
- **Result:** Summary of detection/resistance to scripted attacks
- **Method:** Automated platforms (SaaS)
- **Requirement:** Like Purple Team, but with budget for automation
- **Frequency:** Continuous

0.0.6 Bug Bounty Programs

- **What:** Crowdsourced vulnerability testing
- **Purpose:** Discover real-world vulnerabilities
- **Type:** Public (anyone) or Private (invite-only)
- **Assets:** Mostly apps/services with clear rules
- **Result:** Vulnerability reports with PoCs
- **Method:** According to platform rules (HackerOne, Bugcrowd, etc.)
- **Requirement:** Legal setup + risk acceptance
- **Frequency:** Continuous

Penetration Testing (Part II)

Intelligence Gathering

Goal: Collect relevant information from public sources to aid attacks
Types:

- Physical – maps, building layout
- Logical – org charts, partners
- Infrastructure – IPs, domains, hosts
- Documents – metadata, open data leaks
- HUMINT – staff info, social profiles

Levels:

- L1: Automated (compliance-focused)
- L2: Tools + manual (best-practice)
- L3: Manual, stealthy, social-focused (APT-style)

Techniques:

- Passive – undetectable (e.g., Shodan, WHOIS)
- Semi-passive – DNS queries, public info
- Active – detectable (e.g., scanning)

Recon Techniques & Tools

Website Analysis: Org data, staff, emails
Google Dorking:

- Operators: `inurl:`, `intitle:`, `ext:`
- Tools: GHDB, ExploitDB

Domain/IP Discovery:

- WHOIS, SAN certs, Robtex, FindSubdomains
- DNS Tools: `dig`, `nslookup`
- RIR lookup, BGP Toolkit

Passive Tools: Shodan, Censys, Maltego

Scanning

Purpose: Map attack surface – find hosts, ports, services
Nmap:

- `-sS`: SYN scan (stealth)
- `-sT`: TCP connect
- `-sU`: UDP scan
- `-sV`, `-O`, `-A`, `-p-`
- NSE scripts: `--script=banner`, etc.

Network Tools: `traceroute`, `hping3`, `telnet`, `nc`, `openssl`

Footprinting Defenses & HUMINT

Identify: Firewalls, WAFs, IDS

- Tools: Nmap scripts, banner fingerprinting
- Techniques: Packet crafting, evasion, SE

Human Intelligence:

- Social media analysis, username lookup (Knowem, etc.)
- Pretexting, phishing, physical visits

Penetration Testing (Part III)

Threat Modeling

Purpose: Identify vulnerabilities by analyzing system designs and attacker goals.

- **Attacker-Centric:** Map how attackers move from entry points to target assets.

- **Defender-Centric:** Map organizational defenses and simulate attack paths avoiding them.
- **Techniques:** STRIDE, Attack Trees
- **Assets:**
  - Primary: Within test scope (e.g., CRM frontend)
  - Secondary: Outside scope but shared (e.g., employee DB on same server)
- **Threat Relevance:** Secondary assets may alter attacker models (e.g., insiders become relevant).

Attack Patterns and Frameworks

- **CAPEC:** Focused on application-level attacks and training
- **MITRE ATT&CK:** Real-world adversarial behavior, red-team and defense-oriented
- CAPEC and ATT&CK are complementary and cross-referenced

Vulnerability Analysis

Goal: Discover and confirm security issues that can be exploited.
Techniques:

- Scanners: Nmap, Nessus, GVM, sqlmap, XSStrike
- Source code scanners, manual analysis (e.g., CIS Benchmarks)
- Web scanners: Crawl and test input points
- Active fuzzing: E.g., American Fuzzy Lop
- Track findings with attack trees to avoid redundant work

Challenges:

- **False Positives:** Patched systems not reflected in version info
- **False Negatives:** Backported fixes not updating version number
- **Environment Dependent:** Network position, authentication, etc.

Exploitation

Goal: Gain access by leveraging vulnerabilities.
Methods:

- Exploits: SQL injection, buffer overflows, MitM, USB, social engineering
- Select vector based on success/detection probability
- Consider mitigation bypass: DEP, ASLR, AV, WAF

Expertise Levels:

- **Basic:** Use public exploits
- **Advanced:** Modify/tune exploits and payloads
- **Expert:** Discover new vulnerabilities (zero-days), reverse engineering

Post Exploitation

Goal: Assess value of access and maintain control (e.g., lateral movement).
Activities:

- Pivoting, island hopping
- Follow rules of engagement to prevent real harm

Metasploit Framework (MSF)

Purpose: Exploit development and execution platform.
Modules:

- **Exploits:** Execute payloads
- **Payloads:** Single (self-contained), stagers/stages (modular)
- **Meterpreter:** Advanced in-memory post-exploitation agent
- **Auxiliary:** Scanning, info gathering, DoS
- **Post:** System interaction, enumeration, credential dumping

Architecture:

- Ruby-based, modular structure
- msfconsole: Primary CLI interface
- Can integrate with external tools (Nmap, Nessus)

Lab: Exploitation and Metasploit

Goal: Learn practical exploitation using the Metasploit Framework (MSF).
Target Environment:

- Vulnerable Linux machine in virtual lab setup
- Services exposed: SSH, Samba, HTTP

Key Commands:

- `nmap -sS -sV -O -A <IP>` – scan target for open ports and services
- `msfconsole` – launch Metasploit CLI
- `search <keyword>` – find exploits or modules
- `use <module>` – load exploit/module
- `set RHOST <IP>` – set remote host
- `set PAYLOAD <payload>` – select appropriate payload
- `exploit` – execute attack
- `sessions -i <id>` – interact with session
- `getuid`, `sysinfo`, `ps`, `hashdump`, `shell` – post-exploitation

Exploitation Process:

- Scan for vulnerable services (e.g., VSFTPD)
- Search and select matching exploit in Metasploit
- Configure exploit parameters (RHOST, RPORT, payload)
- Launch exploit and gain reverse shell via Meterpreter

Metasploit Modules Used:

- **Exploit:** `exploit/unix/ftp/vsftpd_234_backdoor`
- **Payload:** `linux/x86/meterpreter/reverse_tcp`
- **Auxiliary:** `scanner/portscan/tcp`, `scanner/ftp/ftp_version`

Post-Exploitation Tasks:

- Enumerate users/processes
- Dump password hashes (`hashdump`)
- Launch interactive shell or pivot to further targets

Key Learnings:

- How to map vulnerabilities to working exploits
- Effective use of Meterpreter for post-exploitation
- Importance of version info and accurate scanning