

# **Week 17**

## **Day 81**

### **XSS and CSRF**

Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) are two common and serious web application vulnerabilities that exploit weaknesses in user input handling and session management. Studying these attacks helps in understanding how attackers compromise user data and application security.

#### **1. Cross-Site Scripting (XSS)**

XSS occurs when a web application allows malicious scripts to be injected into trusted web pages. When other users access these pages, the injected script executes in their browsers, allowing attackers to steal sensitive information such as cookies, session tokens, or personal data.

#### **Types of XSS Studied**

##### **i. StoredXSS:**

Malicious scripts are permanently stored on the server (e.g., in databases or comment sections).

##### **ii. ReflectedXSS:**

Malicious scripts are reflected through user input such as URLs or form fields.

##### **iii. DOM-basedXSS:**

Vulnerabilities occur in client-side scripts without server interaction.

## **Practical Work Using Burp Suite**

- i. Intercepted HTTP requests and responses.
- ii. Tested input fields for script injection.
- iii. Analyzed server responses to identify XSS vulnerabilities.
- iv. Verified execution of injected scripts in a controlled environment.

## **2. Cross-Site Request Forgery (CSRF)**

CSRF is an attack where an attacker tricks an authenticated user into performing unwanted actions on a web application without their knowledge. This happens because the application trusts the user's browser session.

### **Examples of CSRF Attacks**

- Unauthorized password change
- Unauthorized fund transfer
- Forced form submissions

## **Practical Work Using Burp Suite**

- i. Analyzed authenticated requests.
- ii. Tested absence or weakness of CSRF tokens.
- iii. Observed how forged requests could be executed using existing user sessions.
- iv. Evaluated session handling and request validation mechanisms.

## **Tool Used: Burp Suite**

Burp Suite was used as an intercepting proxy to:

- Capture and modify HTTP/HTTPS traffic.
- Analyze cookies, headers, and parameters.
- Test session management and input validation.
- Simulate real-world web attacks ethically.

## **Work Description**

The topic of XSS and CSRF was studied in detail as per the cybersecurity syllabus. All practical testing was performed on authorized applications within a controlled lab environment, ensuring ethical and legal compliance. These exercises improved understanding of web attack techniques and secure development practices.

## **Conclusion**

XSS and CSRF attacks pose serious threats to web application security. Through hands-on practice using Burp Suite, vulnerabilities were identified and analyzed, highlighting the importance of input sanitization, secure session management, and anti-CSRF protections.

## **Day 82**

### **Sniffing & Spoofing**

Sniffing and Spoofing are network-level attack techniques used to intercept, analyze, and manipulate data transmitted over a network. These attacks exploit weaknesses in network communication protocols and trust relationships between devices. Understanding these techniques helps cybersecurity professionals identify risks and implement effective network security controls.

#### **1. Sniffing**

Sniffing is the process of capturing and monitoring network traffic to analyze data packets transmitted between devices. Attackers use sniffing to collect sensitive information such as usernames, passwords, session cookies, and unencrypted data.

#### **Types of Sniffing Studied**

##### **i. PassiveSniffing:**

Monitoring network traffic without altering data flow.

##### **ii. ActiveSniffing:**

Intercepting traffic by manipulating network devices or using techniques like ARP poisoning.

#### **Practical Work Using Wireshark**

- i. Captured live network traffic on a local network.
- ii. Analyzed different protocol packets such as HTTP, FTP, DNS, and TCP.
- iii. Identified unencrypted data transmitted over the network.
- iv. Studied packet headers, payloads, and communication flow.

- v. Wireshark helped in understanding how data travels across a network and how insecure protocols expose sensitive information.

## **2. Spoofing**

Spoofing is a technique in which an attacker impersonates a trusted device or identity to deceive systems and users. By forging source information, attackers can redirect traffic, bypass security controls, or launch further attacks.

### **Common Types of Spoofing Studied**

#### **i. ARPSpoofing:**

Sending fake ARP messages to associate the attacker's MAC address with a legitimate IP address.

#### **ii. IPSpoofing:**

Faking the source IP address in network packets.

#### **iii. DNSspoofing:**

Redirecting users to malicious websites by altering DNS responses.

### **Practical Work Using Ettercap**

- i. Performed ARP poisoning in a controlled lab environment.
- ii. Established Man-in-the-Middle (MITM) attacks.
- iii. Intercepted and analyzed network traffic between two systems.
- iv. Demonstrated how spoofing can lead to data theft and session hijacking.

Ettercap provided practical insight into how attackers manipulate network communication.

### **Tools Used**

## **Wireshark**

- Network packet analyzer.
- Used for capturing, filtering, and analyzing traffic.
- Helps identify insecure protocols and suspicious activity.

## **Ettercap**

- Network security tool for MITM attacks.
- Used for sniffing, spoofing, and traffic manipulation.
- Demonstrates real-world network attack techniques.

## **Work Description**

The topic of Sniffing & Spoofing was studied in detail as per the cybersecurity syllabus. All practical tasks were performed on authorized systems within a **controlled lab environment**, ensuring ethical and legal compliance. The exercises improved understanding of network vulnerabilities, traffic interception, and defense mechanisms.

## **Conclusion**

Sniffing and Spoofing attacks highlight the importance of secure network communication. Through hands-on practice using Wireshark and Ettercap, the risks of unencrypted traffic and weak network authentication were clearly understood. This study emphasized the need for encryption, secure protocols, and network monitoring to prevent such attacks.

## **Day 83**

### **Session Hijacking**

Session Hijacking is a web and network security attack in which an attacker takes control of a valid user session by stealing or manipulating the session identifier (session ID). Since web applications use session IDs to maintain user authentication, gaining access to this session allows attackers to impersonate legitimate users without knowing their login credentials.

### **Types of Session Hijacking Studied**

- **ActiveSessionHijacking:**

The attacker takes over an active session and performs actions as the victim.

- **PassiveSessionHijacking:**

The attacker monitors session traffic to collect session information.

- **Man-in-the-Middle(MITM)Attack:**

Intercepting communication between the client and server to capture session data.

### **Tools Used**

#### **1. Burp Suite**

Burp Suite is a web application security testing tool used to analyze and manipulate HTTP/HTTPS traffic.

### **Practical Work Performed Using Burp Suite**

- Configured Burp Suite as an intercepting proxy.
- Captured and analyzed HTTP requests and responses.
- Identified session cookies and authentication tokens.

- Tested session handling weaknesses such as:
  - Insecure cookies
  - Missing HttpOnly and Secure flags
  - Session fixation vulnerabilities
- Modified session cookies to test session reuse and hijacking scenarios.

This practical work helped in understanding how weak session management can lead to account compromise. **2. Ettercap**

Ettercap is a powerful network security tool used for sniffing and Man-in-the-Middle attacks.

### **Practical Work Performed Using Ettercap**

- i. Performed ARP spoofing in a controlled lab setup.
- ii. Established MITM attacks between client and server.
- iii. Intercepted network traffic to capture session data.
- iv. Demonstrated how session cookies can be stolen over unencrypted connections.

Ettercap showed how network-level attacks can lead to session hijacking.

### **Work Description**

The topic of Session Hijacking was studied in detail according to the cybersecurity syllabus. All practical activities were conducted on authorized systems within a **controlled lab environment**, ensuring ethical and legal compliance. The exercises enhanced understanding of session management vulnerabilities and real-world attack techniques.

### **Prevention Techniques Studied**

- Use of HTTPS and SSL/TLS encryption

- Secure and HttpOnly cookie attributes
- Strong session ID generation
- Session timeout and regeneration
- Protection against MITM attacks

## **Conclusion**

Session hijacking is a serious threat to web application security. Through hands-on practice using Burp Suite and Ettercap, vulnerabilities related to session management were identified and analyzed. This study emphasized the importance of secure session handling and encrypted communication to protect user sessions.

## **Day 84**

### **Python for Pentesting**

Python for Pentesting refers to the use of the Python programming language to develop scripts and tools that automate penetration testing and security assessment tasks. Due to its simplicity, extensive libraries, and cross-platform support, Python is widely used by cybersecurity professionals for ethical hacking, vulnerability assessment, and security automation.

Studying Python for pentesting helps in understanding how custom security tools are created and how repetitive security tasks can be automated efficiently.

### **Tools Used**

#### **1. Python3**

Python3 is the latest and most widely used version of Python, offering powerful libraries for security testing.

### **Practical Work Performed Using Python3**

- Learned Python fundamentals relevant to security scripting.
- Developed scripts for:
  - Port scanning
  - Banner grabbing
  - Network communication
  - Password strength testing
- Used Python libraries such as:
  - socket for network connections

- requests for HTTP interactions
- os and subprocess for system-level tasks

These activities demonstrated how Python can be used to automate penetration testing tasks.

## **2. Visual Studio Code (VS Code)**

VS Code is a lightweight and powerful code editor used for writing, editing, and debugging Python scripts.

### **Practical Work Performed Using VS Code**

- Wrote and executed Python scripts efficiently.
- Used debugging features to identify and fix errors.
- Organized code using proper structure and comments.
- Improved productivity through extensions and terminal integration.

VS Code provided an effective development environment for pentesting scripts.

### **Work Description**

The topic of Python for Pentesting was studied in detail as per the cybersecurity syllabus. Practical scripting tasks were performed on authorized systems within a **controlled lab environment**, ensuring ethical and legal compliance. The exercises improved understanding of automation, custom tool development, and real-world security testing techniques.

### **Applications of Python in Pentesting**

- i. Network scanning and reconnaissance
- ii. Web application testing

- iii. Exploit development and testing
- iv. Automation of repetitive security tasks
- v. Log analysis and data extraction

## **Conclusion**

Python plays a crucial role in modern penetration testing. Through hands-on practice using Python3 and VS Code, practical knowledge of security scripting and automation was gained. This study highlighted the importance of programming skills in ethical hacking and cybersecurity.

## **Day 85**

### **Shell Scripting**

Shell Scripting involves writing scripts using the shell (command-line interpreter) to automate system-level tasks in Linux/Unix environments. In cybersecurity, shell scripting is widely used to automate administrative operations, system monitoring, security checks, and penetration testing tasks. It helps security professionals perform tasks efficiently and consistently.

### **Tools Used**

#### **1. Bash Shell**

Bash (Bourne Again Shell) is the most commonly used shell in Linux systems.

### **Practical Work Performed Using Bash Shell**

- Learned basic shell scripting concepts such as:
  - Variables and data types
  - Conditional statements (if, else)
  - Loops (for, while)
  - Functions and script arguments
- Created scripts for:
  - Automating file and directory management
  - Monitoring system processes
  - Checking user permissions
  - Automating repetitive security tasks

These exercises demonstrated how shell scripts can simplify complex system operations.

## 2. Nano Editor

Nano is a lightweight and user-friendly text editor used in terminal environments.

### Practical Work Performed Using Nano Editor

- Created and edited shell script files.
- Added comments and proper formatting for readability.
- Saved and modified scripts efficiently.
- Tested and debugged scripts directly from the terminal.

Nano provided an easy environment for learning and writing shell scripts.

### Work Description

The topic of Shell Scripting was studied in detail as per the cybersecurity syllabus. All practical scripting tasks were performed on authorized systems within a controlled lab environment, ensuring ethical and legal compliance. The hands-on exercises improved understanding of Linux automation and security-focused scripting techniques.

### Applications of Shell Scripting in Cybersecurity

- i. Automating security scans and log analysis
- ii. User and permission management
- iii. System health and resource monitoring
- iv. Backup and recovery automation
- v. Supporting penetration testing workflows

## **Conclusion**

Shell scripting is a powerful skill in cybersecurity. Through practical use of Bash Shell and Nano Editor, hands-on experience was gained in automating system tasks and improving operational efficiency. This study emphasized the importance of scripting for effective system administration and security management.