ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN

VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

한-베정보통신기술대학교

Nhân bản – Phụng sự – Khai phóng

Chapter 3. WEB APPLICATION ATTACK AND SECURITY

3.1. Current Web Application Safety Issues and Trends

- Current Web Application Safety Issues
  - SQL Injection: One of the common types of attacks, where attackers insert malicious SQL statements into queries to access, modify, or delete data.
  - Cross-Site Scripting (XSS): Attack technique where an attacker injects malicious JavaScript code into a website to perform malicious actions against the end user.

3.1. Current Web Application Safety Issues and Trends

- Current Web Application Safety Issues
  - Cross-Site Request Forgery (CSRF): An attack technique where attackers trick users to perform unwanted actions through the use of their authentication permissions
  - Injection Attacks: These include attacks such as Command Injection, XML Injection, and LDAP Injection, where malicious data is inserted into data sources to perform unwanted actions.

3.1. Current Web Application Safety Issues and Trends

- Web Application Security Trends
  - Demand-Based Security (DevSecOps): Integrate security into the development and deployment process to detect and mitigate risks at an early stage
  - API Security: Enhance the security of APIs through the use of standards such as OAuth and OpenID Connect
  - Machine Learning and Web Application Safety: Use machine learning solutions to detect and prevent attacks automatically

3.1. Current Web Application Safety Issues and Trends

- Web Application Security Trends
  - Platform-based security: Focus on securing applications from the infrastructure level, including updating operating systems and supporting software
  - Enhanced security of data: Use strong encryption and access control to protect important data

3.2. Criteria for evaluating a safe web application system

- Integrated Security
  - Access control : Define and implement access controls to ensure only necessary users can access specific resources.
  - Session Management: Ensures that the session management process is performed securely to prevent attacks such as session hijacking
- Data Security
  - Data Encryption: Uses encryption to protect data as it travels over a network and as it is stored in a database.
  - Input Data Testing: Apply input checks to prevent attacks such as SQL injection and Cross-Site Scripting (XSS).

3.2. Criteria for evaluating a safe web application system

- System Security
  - System Update: Make sure that your system and other software components are up to date with the latest security patches.
  - System Control and Monitoring: Establish system controls and monitoring procedures to detect and prevent undesirable activities
- Application Security
  - Security Testing: Perform periodic security testing to detect and fix security vulnerabilities.
  - Attack Protection: Use attack protection solutions to protect against common types of attacks.

3.2. Criteria for evaluating a safe web application system

- Event Management and Logging
  - Secure Logging: Ensures that the system securely logs important events so that it can be used for future monitoring and analysis.
  - Log Analysis: Perform log analysis to detect unusual activity and alert you to security issues

3.2. Criteria for evaluating a safe web application system

- Compliance and Privacy Policy
  - Legal Compliance: Ensure that the system complies with information security laws and regulations.
  - Privacy Policy: Define and maintain a clear and effective privacy policy.
- Source Code Security
  - Source Code Testing: Perform source code testing to detect and fix security vulnerabilities from the source code level.
  - Open Source Security Risk Analysis: If the application uses open source code, make sure that the security risks of the libraries and frameworks used have been analyzed.

3.3. Check for security errors for web applications

- Cross-Site Scripting
  - User Input Testing: Perform testing to ensure that user input data is checked and filtered to prevent malicious scenarios.
  - Check HTTP Only and Secure Flag for Cookies: Make sure that cookies are set with HTTP Only and Secure attributes to protect against XSS attacks.
- SQL Injection Testing
  - Check user input: Perform tests to ensure that SQL queries are checked and clean, and use tricks such as prepared statements and parameterized queries.
  - Check Logs: Check system logs for SQL injection attack attempts

3.3. Check for security errors for web applications

- Cross-Site Request Forgery
  - Anti-CSRF Token Audit: Ensure that all critical operations use Anti-CSRF tokens to prevent CSRF attacks.
  - Check the Referrer Header: Check to see if the referrer header is checked, to ensure that only requests from trusted websites are accepted.
- URL Redirection Testing
  - Check Navigation Parameters: Check if parameters in URLs can be changed to direct users to unwanted pages.
  - Check Navigation Policies: Make sure that navigation policies are tightly set up to prevent unwanted behavior

3.3. Check for security errors for web applications

- Decentralized Testing and Session Management
  - Privilege Check: Ensures that functions can only be accessed by users with corresponding privileges.
  - Check Session Expiration: Check if there is an effective session expiration policy to prevent attacks such as session hijacking
- DDos (Distributed Denial of Service) Attack Testing
  - Test DDoS Protection: Make sure that there are DDoS protections in place, such as using a CDN or anti-DDoS services.
  - Traffic Inspection: Tests your ability to handle heavy traffic to ensure your system is not overloaded

3.3. Check for security errors for web applications

- Test File Upload
  - Check File Types and Extensions: Check if the system checks file types and extensions, to prevent malicious file uploads.
  - Check File Size: Make sure that there is a file size limit to prevent DoS attacks through large file uploads.
- Check API (Application Programming Interface)
  - Check Authentication and Authorization: Ensure that the API requires strict authentication and has an authorization mechanism.
  - Testing Attacks Like Injection and DoS: Check if the API protects against attacks like injection and denial-of-service

3.4. Basic Web Authentication Methods

- Username and Password Authentication
- Token Authentication (Bearer Token)
- OTP (One-Time Password) authentication
- Fingerprint Authentication and Facial Recognition
- USB Port Authentication (Security Key)
- 2-Factor Authentication (2FA) and Multi-Factor Authentication (MFA)
- Biometics Authentication
- Email or phone number authentication

3.5. Web password cracking techniques

- Brute Force Attacks
- Dictionary Attacks
- Phishing Attacks
- Keystroke Logging (Keyloggers)
- Shoulder Surfing
- Rainbow Table Attacks
- Session Hijacking
- Social Engineering
- Zero-Day Exploits

3.6. Attack technique by inserting scripts (XSS)

- Reflected XSS
  - Description: An attacker inserts XSS code directly into the parameters of the URL or web template, and the code is executed immediately when a user visits a link or template containing malicious code.
  - Prevention: Use input filters to check and remove special characters, encoding data before display
- Stored XSS
  - Description: XSS code is inserted into a database or archive file and executed when a user accesses content containing that malicious code.
  - Prevention: Check and clean data before storing, restrict access to data storage.

3.6. Attack technique by inserting scripts (XSS)

- DOM-based XSS
  - Description: Direct attack on the Document Object Model (DOM) of the website by changing the DOM structure to perform malicious actions.
  - Prevention: Perform careful input checks and use encryption functions to avoid DOM attacks.
- Non-Persistent XSS (Reflected)
  - Description: The XSS code only lasts for a specific number of hits or for a short period of time.
  - Prevention: Install malicious code protections such as Content Security Policy (CSP).

3.6. Attack technique by inserting scripts (XSS)

- BeEF (Browser Exploitation Framework)
  - Description: BeEF is a dedicated framework for leveraging XSS attacked browsers, allowing attackers to perform browser control actions.
  - Prevention: Use security technologies such as Content Security Policy to reduce the risk of BeEF attacks
- Phishing Attacks using XSS
  - Description: Use XSS code to display fake websites to trick users into entering login or other sensitive information.
  - Prevention: Use HTTPS, which provides user education about phishing threats.

3.6. Attack technique by inserting scripts (XSS)

- Bypassing Content Security Policy (CSP)
  - Description: An attacker tries to bypass or disable CSP to perform an XSS attack.
  - Prevention: Configure CSP closely, do not disable CSP on the browser, and check the source code for vulnerabilities
- Beacon-based XSS Attacks
  - Description: Use XSS code to create beacon requests to inform the attacker about the state of the user's browser.
  - Prevention: Block or monitor unwanted requests, using Content Security Policy

3.7. Preventive measures

- Content Security Policy (CSP)
  - Description: Establish and enforce CSPs to limit or prevent the execution of unsafe resource types, such as JavaScript snippets from untrusted sources.
  - Implement:
    - Configure the CSP header on the server.
    - Restrict the execution source for the script.
    - Minimize or eliminate the use of unsafe-inline and unsafe-eval.
    - Use script-src, style-src, and other CSP directives to control resource sources and types.

3.7. Preventive measures

- Input Validation
  - Description: Check and clean user input to prevent the insertion of malicious codes.
  - Implement:
    - Use available libraries or frameworks to test input.
    - Only accept valid data and reject invalid data.
    - Perform tests on both server-side and client-side.

3.7. Preventive measures

- Output Encoding
  - Description: Encrypt data before displaying it on the website to prevent the execution of malicious scripts.
  - Implement:
    - Use encoding functions like htmlspecialchars on the server side.
    - Use output encoding libraries on the client side like React or Angular.

3.7. Preventive measures

- HTTP Only and Secure Cookies
  - Description: Use the HttpOnly and Secure attributes in cookie settings to prevent XSS attacks related to cookie information theft.
  - Implement:
    - Set the HttpOnly property to prevent JavaScript from accessing cookies.
    - Use a secure connection (HTTPS) and set the Secure attribute to transmit cookies only over a secure connection.

3.7. Preventive measures

- Security Headers
  - Description: Use security headers to protect your browser from different types of attacks.
  - Implement:
    - Use HTTP headers such as X-Content-Type-Options, X-Frame-Options, and Referrer-Policy.
    - Minimize feedback from the server.

3.7. Preventive measures

- Regular Security Audits và Penetration Testing
  - Description: Perform periodic security testing and source code review tests to detect and remediate security vulnerabilities.
  - Implement:
    - Perform regular security testing, including XSS testing.
    - Use automated and manual testing tools to ensure that the application is free of security vulnerabilities.

3.7. Preventive measures

- Security Education và Awareness
  - Description: Provide information security training to development teams and system administrators to increase awareness and knowledge of XSS threats.
  - Implement:
    - Organize periodic training sessions on information security.
    - Provide educational resources and guidance on security measures.

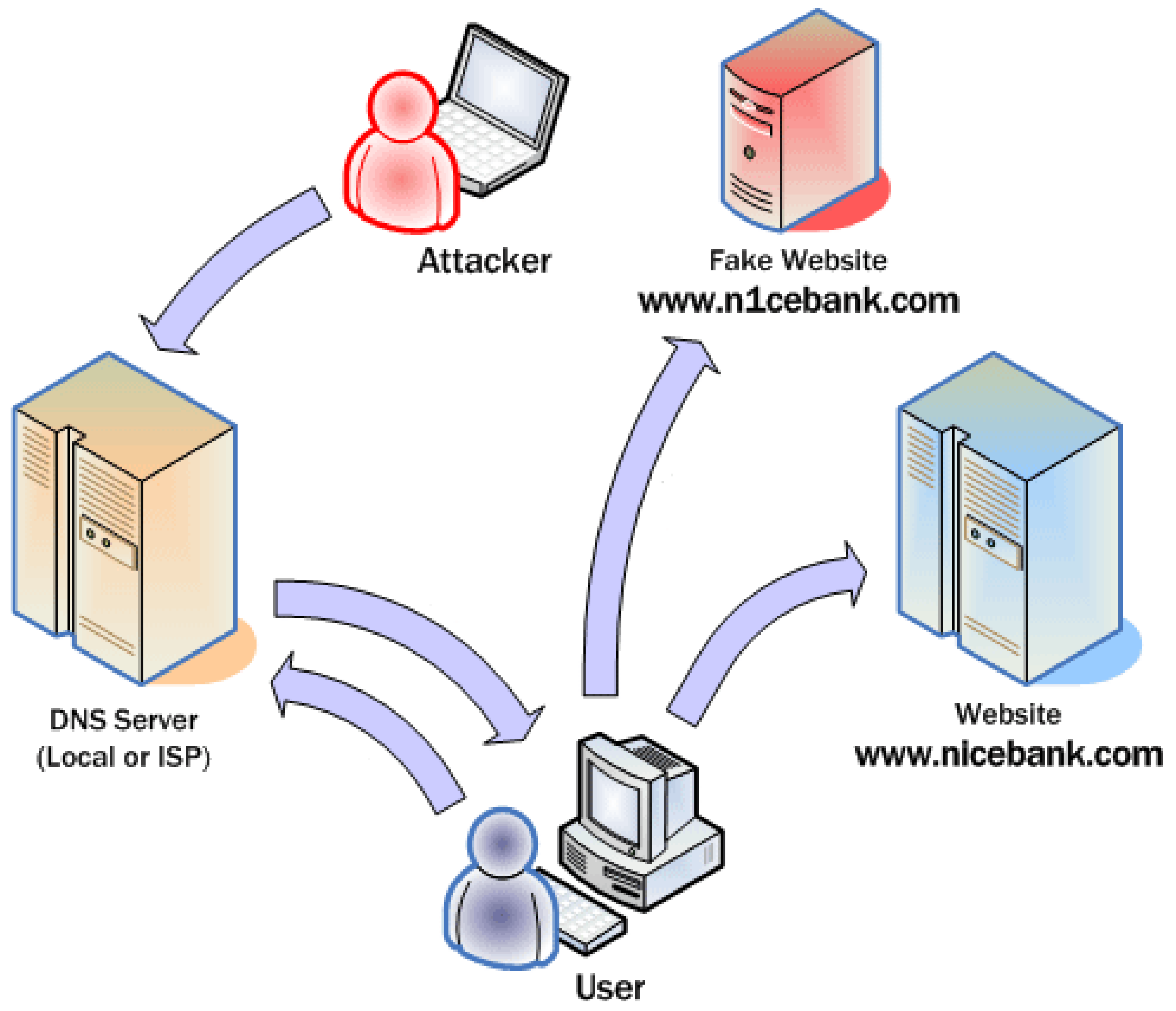
3.7. Preventive measures

- Web Application Firewall (WAF)
  - Description: Use WAF to prevent XSS attacks by monitoring and filtering requests to web applications.
  - Implement:
    - Configure WAF to detect and block XSS attack patterns.
    - Update WAF's privacy policy regularly.

3.8. System error detection

- Security Testing
- Performance Testing
- Data security testing
- URL Redirection Testing
- API Security Testing
- Testing Upload Files

3.9. Building a fake web server



3.10. Web Phishing Attack

- Description of the attack
  - Phishing is a phishing technique where an attacker creates a fake website that looks like the official website of an organization or service. The goal of this attack is usually to steal personal information such as usernames, passwords, credit card information, or other sensitive information from users.

3.10. Web Phishing Attack

- Method
  - Create a fake website: The attacker creates a fake website that looks identical to the target's official website.
  - Phishing email: Attackers send fake emails to many users with phishing content, often accompanied by links to fake websites.
  - Deceiving users: Emails often ask users to take urgent action, such as changing passwords, confirming accounts, or providing personal information.
  - Collection of information: Users are tricked into entering fake websites and entering personal information. This information is then stored or transmitted back to the attacker.

3.10. Web Phishing Attack

- Prevention and recognition
  - Check URL: Always check the URL of the website to make sure that it is official. Avoid clicking links from unsolicited emails or messages.
  - Email verification: Verify the source of the email by checking the sender's email address. Beware of emails that demand immediate action.
  - Kiểm Tra SSL/TLS: In case websites ask to enter personal information, make sure that the connection is protected with SSL/TLS (check is https://).
  - Pay attention to spelling and grammar mistakes: Phishing emails often contain spelling and grammatical errors. Pay attention to these to be aware.

3.10. Web Phishing Attack

- Prevention and recognition
  - Use security tools: Use email and network security tools to block or warn about fake websites and phishing emails.
  - Safe Browsing: Use a secure and up-to-date web browser for universal security features.
  - User education: Increase cybersecurity education so users can recognize and avoid phishing scenarios.
  - Report a problem: Report any fake emails or websites to the relevant organization.

3.11. Prevention and remedy

- Preventive measures
  - Education: Increase cybersecurity education and awareness among employees and end users. They need to know how to identify and avoid being scammed.
  - Email Filtering: Use an email filtering system to prevent and move phishing emails to the trash.
  - Safe Browsing: Use a secure and up-to-date web browser for tight security features.
  - Restrict access: Restrict access to websites and accounts to only necessary people.

3.11. Prevention and remedy

- Preventive measures
  - SSL/TLS Encryption: Requires the use of an SSL/TLS secure connection (https:// check) when sending personal information.
  - Multi-Factor Authentication (MFA): Stimulate the use of MFA to enhance login security.
  - Report immediately: Support and encourage employees and users to immediately report any suspicion of fake emails or websites.

3.11. Prevention and remedy

- Remediation and Handling
  - Notice to the administrator: As soon as signs of a phishing attack are detected, immediately notify the system administrator and security.
  - Access blocking: If a phishing site has been identified, block access to it on all internal networks.
  - Limit damage: If some users have provided personal information, take immediate measures to limit the damage. This may include changing the password and notifying the bank if account information is involved.
  - Log analysis: Check system logs to determine the scope of attacks and review available log data for future prevention.

3.11. Prevention and remedy

- Remediation and Handling
  - Improved security system: Use the experience from the attack to improve security systems, including updating and reviewing security policies.
  - User education: Learn from the incident and educate users on how to recognize and avoid scams.
  - Security System Test: Check and re-evaluate current security measures and implement the necessary updates.
  - Reporting organization: Report the incident to the relevant authorities and organizations so that they can take legal measures and help.

3.12. Vulnerable Open Source Modules and How to Fix…

- Regular updates
- Check the safe configuration
- Authentication and Authorization
- Block malicious file uploads
- Use SSL/TLS
- Periodic security testing
- User education
- Open-nature tracking

Enjoy the Course…!