Catalog

**[ĐỀ CƯƠNG ÔN TẬP 2](#_Toc8659)**

**[1. Trình bày các hình thức tấn công? Nêu ví dụ minh họa? Liên hệ với các mục tiêu của an ninh mạng 2](#_Toc2958)**

**[2. Trình bày các mục tiêu của an ninh mạng? Lấy ví dụ? 2](#_Toc30680)**

**[3. Khái niệm mã hóa công khai? Sử dụng mã hóa công khai làm chữ ký số? 3](#_Toc9850)**

**[4. Trình bày khái niệm firewall, cơ chế kiểm soát, ưu điểm và hạn chế? 3](#_Toc12360)**

**[5. Khái niệm về các phần mềm độc hại, phân loại? 4](#_Toc16686)**

**[6. Phân loại firewall? Trình bày về cơ chế Package Filtering? 4](#_Toc30010)**

**[7. Trình bày về Circuit‐level Gateway trong firewall? 5](#_Toc4046)**

**[8. Trình bày về các loại firewall topology? 5](#_Toc6943)**

**[9. Khái niệm chữ ký số? Trình bày ứng dụng mã hóa công khai để thực hiện giao dịch có chữ ký số? 6](#_Toc10866)**

**[10. Trình bày các bước để mã hóa theo thuật toán Caesar Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa. 6](#_Toc24916)**

**[11. Khi trao đổi thông điệp qua mạng tính toàn vẹn (integrity) được yêu cầu thế nào? Thực hiện yêu cầu toàn vẹn bằng giải pháp nào? 7](#_Toc25609)**

**[12. Trình bày các bước để mã hóa theo thuật toán Vigenere Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa. 7](#_Toc29352)**

**[13. Nêu những hạn chế (nguy cơ bị bẻ khóa) của các phương pháp mã hóa kinh điển. Lấy ví dụ trong trường hợp Caesar Cipher và  Vigenere Cipher 7](#_Toc1272)**

**[14. Trên cơ sở phân tích các hạn chế của các thuật toán mã hóa cổ điển (Caesar Cipher, Vigeniar Cipher), các chuẩn mã hóa tiên tiến được thiết kế như thế nào? Trình bày các phép biến đổi cơ bản hay được sử dụng trong các chuẩn mã hóa tiên tiến. 8](#_Toc21947)**

**[15. Nêu các hạn chế của firewall? Các cơ chế thiết lập firewall? 8](#_Toc1225)**

**[16. Trình bày cách thiết lập một bộ khóa trong thuật toán RSA. Sử dụng bộ khóa đó để mã hóa và giải mã một ký tự tùy chọn 8](#_Toc10829)**

**[17. Nêu những hiểu biết của bạn về hàm băm (Hash function) và sử dụng hàm băm trong đảm bảo tính toàn vẹn của thông điệp? 9](#_Toc7590)**

**[18. Nêu kiến trúc các mô hình cài đặt firewall? 9](#_Toc25628)**

**[19. Trình bày về SQL Injection? Nêu ví dụ và cách khắc phục 9](#_Toc2729)**

**[20. Trình bày về Cross-Site Scripting (XSS)? Nêu ví dụ và cách khắc phục 10](#_Toc22245)**

**[1. Stored XSS: 10](#_Toc27005)**

**[2. Reflected XSS: 10](#_Toc26367)**

**[3. DOM-based XSS: 10](#_Toc15630)**

**[21. Trình bày về XML Injection? Nêu ví dụ và cách khắc phục 11](#_Toc17359)**

**[22. Trình bày về Directory Traversal/Command Injection? Nêu ví dụ về lỗi file upload và cách khắc phục 11](#_Toc30381)**

**[23. Trình bày một số hình thức tấn công phía client: Header manipulation, Cookies 12](#_Toc8928)**

**[1. Header manipulation 12](#_Toc16719)**

**[2. Cookies 12](#_Toc26782)**

**[24. Trình bày hình thức tấn công phía client: Attachments, Session Hijacking 13](#_Toc26578)**

**[1. Attachments: 13](#_Toc21508)**

**[2. Session Hijacking: 13](#_Toc13938)**

**[25. Trình bày về DoS và các loại tấn công DoS 13](#_Toc11465)**

**[26. Trình bày về Interception, Poisioning và các hình thức tấn công chiếm quyền truy cập (Access Rights) 14](#_Toc30929)**

**[1. Sử dụng Java Cryptography Architecture (JCA), viết class SymmetricCryp để thực hiện các chức năng mã hóa và giải mã đối xứng theo thuật toán AES gồm các hàm 15](#_Toc12229)**

[1.1 Sinh khóa đối xứng 15](#_Toc311)

[1.2 Tạo khóa đối xứng từ một chuỗi cho trước 15](#_Toc5027)

[1.3 Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa, tham số thứ 2 là khóa được tạo ở 1.1 15](#_Toc19202)

[1.4 Mã hóa thông điệp có 2  tham số: Tham số 1 là chuỗi cần mã hóa tham số 2 là khóa được tạo ở 1.2  16](#_Toc4138)

[1.5 Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.1 16](#_Toc6489)

[1.6 Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.2 17](#_Toc25742)

[1.7 Sử dụng class SymetricCryp để mã hóa một chuỗi cho trước hoặc giải mã ra bản rõ từ bản mã cho trước. 17](#_Toc15509)

**[2. Sử dụng Java Cryptography Architecture (JCA) để viết các hàm băm thông điệp cho trước theo thuật toán: 17](#_Toc32186)**

[2.1. MD5 17](#_Toc15376)

[2.2. SHA1 18](#_Toc9068)

[2.3. SHA-256 18](#_Toc20270)

**[3. Sử dụng Java Crypto Architecture (JCA), viết chương trình mã hóa và giải mã sử dụng mã hóa công khai RSA:  19](#_Toc2962)**

[3.1 Sinh bộ khóa theo thuật toán RSA  20](#_Toc10202)

[3.2 Lưu bộ khóa ra file 20](#_Toc27991)

[3.3 Hàm đọc file để nạp khóa bí mật, tham số truyền vào là tên file 20](#_Toc25043)

[3.4 Hàm đọc file để nạp khóa công khai, tham số truyền vào là tên file 21](#_Toc14774)

[3.5 Hàm mã hóa một thông điệp cho trước bằng khóa bí mật, tham số truyền vào gồm tham số thứ nhất là thông điệp cần mã hóa, tham số thứ 2 là khóa bí mật 21](#_Toc17787)

[3.6 Hàm giải mã thông điệp bằng khóa công khai, tham số truyền vào gồm tham số thứ nhất là bản mã cần giải mã, tham số thứ 2 là khóa công khai. 21](#_Toc3474)

[3.7 Sử dụng lớp RSACryp để mã hóa một chuỗi và băm chuỗi theo một thuật toán băm cho trước rồi lưu cả bản mã và bản băm vào một file 21](#_Toc16904)

[3.8 Đọc file chứa bản mã và bản băm của một thông điệp, sử dụng lớp RSACryp để giải mã ra thông điệp gốc và băm lại bằng thuật toán băm cho trước. Sau đó đem so sánh hai bản băm để xác nhận tính toàn vẹn 22](#_Toc20901)

**[Bài 4 : Chương trình SSH App-script : (Base System 1: ch11, Base System 2: ch12) 23](#_Toc25154)**

ĐỀ CƯƠNG ÔN TẬP

1. **Trình bày các hình thức tấn công? Nêu ví dụ minh họa? Liên hệ với các mục tiêu của an ninh mạng**

- Có 5 loại tấn công :

+ Tấn công bằng phần mềm độc hại (Malware): Tấn công này thường được thực hiện bằng cách sử dụng phần mềm độc hại để xâm nhập vào hệ thống mục tiêu.Các loại phần mềm độc hại gồm virus, worm, trojan,rootkit,...

+ Tấn công giả mạo (Phishing): Tấn công này thường được thực hiện bằng cách gửi email hoặc nhắn tin giả mạo, yêu cầu người cung cấp thông tin các nhân hoặc thông tin đăng nhập.

+ Tấn công từ chối dịch vụ (Dos và DDoS): Tấn công này thường được thực hiện bằng cách tạo ra lưu lượng truy cập lớn đến hệ thống mục tiêu, gây ra quá tải cho hệ thống mục tiêu và làm cho hệ thống không thể hoạt động được.

+ Tấn công trung gian (Main-in the-middle attack): Tấn công này thường được thực hiện bằng cách giả mạo 1 máy chủ hoặc 1 thiết bị trung gian để đánh cắp thông tin truyền qua mạng.

+ Khai thác lỗ hổng Zero-day(Zero day attack) : Tấn công này thường được thực hiện bằng cách khai thác lỗ hổng bảo mật chưa được biết đến để xâm nhập vào hệ thống mục tiêu.

1. **Trình bày các mục tiêu của an ninh mạng? Lấy ví dụ?**

Có 3 mục tiêu chủ yếu:

* **Confidentiality ( tính bảo mật)**. Ví dụ: A và B ko muốn thông điệp của họ bị đọc bởi người khác.
* **Integrity ( tính toàn vẹn)**. Ví dụ: A và B ko muốn thông điệp của họ bị thay đổi bởi người khác.
* **Availability ( tính khả dụng)**. Ví dụ: Đảm bảo máy chủ B luôn nhận được yêu cầu hợp lệ.

Ngoài ra còn 6 mục tiêu nữa:

* **Entity Authentication (xác thực thực thể)**. Ví dụ: A muốn xác nhận thông điệp được gửi từ B chắc chắn từ B gửi.
* **Message Origin Authentication (xác thực nguồn gốc của thông điệp)**. Ví dụ: A muốn chắc chắn thông điệp được cho là gửi đến từ B là từ B gửi và ngược lại.
* **Timeliness (tính kịp thời)**: Ví dụ: ngăn chặn bên thứ 3 sao chép nội dung cuộc trò chuyện
* **Non‐Repudiation (chống từ chối)**: Ví dụ: A ko thể nào từ chối được thông điệp của mình đã gửi và đã nhận.
* **Authorisation (ủy quyền)**: Ví dụ: một máy tính trên mạng có tài nguyên sẵn sàng cho một tập người dùng, nhưng ko phải tất cả người dùng mạng có thể truy cập được nguồn tài nguyên này. Chỉ những người được ủy quyền mới có thể truy cập.
* **Access Control (kiểm soát truy cập)**: Ví dụ: một nhân viên trong phòng Kế toán có thể có quyền truy cập vào máy chủ Kế toán để xem và chỉnh sửa các báo cáo tài chính, nhưng họ không thể truy cập vào máy chủ của phòng Phát triển Phần mềm để xem mã nguồn của các ứng dụng đang được phát triển.

VD:Chúng ta sử dụng mã hóa để đảm bảo an toàn cho dữ liệu của hệ thống không bị đánh cắp hoặc bị tấn công

1. **Khái niệm mã hóa công khai? Sử dụng mã hóa công khai làm chữ ký số?**

* Mã hóa khóa công khai là một dạng [mã hóa](https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a" \o "Mã hóa) cho phép người sử dụng trao đổi các [thông tin](https://vi.wikipedia.org/wiki/Th%C3%B4ng_tin" \o "Thông tin) mật mà không cần phải trao đổi các [khóa](https://vi.wikipedia.org/wiki/Kh%C3%B3a_(m%E1%BA%ADt_m%C3%A3)" \o "Khóa (mật mã)) chung [bí mật](https://vi.wikipedia.org/w/index.php?title=B%C3%AD_m%E1%BA%ADt&action=edit&redlink=1" \o "Bí mật (trang chưa được viết)) trước đó. Điều này được thực hiện bằng cách sử dụng một cặp khóa có quan hệ toán học với nhau là khóa công khai và khóa cá nhân (hay khóa bí mật).
* Sử dụng mã công khai làm chữ ký số:

Bên A: A muốn gửi thông điệp M và A sử dụng chữ ký số:

+ A băm thông điệp bằng SHA-1, SHA-256, SHA-3 => được **hash value**.

+ **Hash value** dùng khoá private của A để **mã hoá** => **chữ ký số**.

+ Đính kèm chữ ký số với thông điệp M rồi gửi đến B.

Bên B: B nhận gói tin của A gửi lấy ra thông điệp M và **chữ kí số**.

+ Lấy chữ ký số **giải mã** theo khoá public của A => **hash value ban đầu** (1)

+ B băm thông điệp M bằng **hàm băm giống A** thực hiện (đã thống nhất từ trước) => hash value mới (2)

+ Nếu giá trị băm (1) bằng băm (2) thì chữ ký được xác thực và ngược lại.

1. **Trình bày khái niệm firewall, cơ chế kiểm soát, ưu điểm và hạn chế?**

+ Tường lửa (firewall): là 1 phần mềm hoặc phần cứng dùng để ngăng chặn các cuộc tấn công mạng và bảo vệ cho hệ thống máy tính, mạng máy tính, thiết bị điện tử kết nối mạng và các dịch vụ trực tuyến khỏi các mối đe dọa.

+ Cơ chế kiểm soát:

* Kiểm soát truy cập: Tường lửa có thể cấu hình để kiểm soát truy cập vào và ra khỏi mạng nội bộ. Các quy tắc kiểm soát truy cập này có thể đuwocj thiết lập để cho phép hoặc từ chối truy cập từ các địa chỉ IP cụ thể hoặc các loại lưu lượng truy cập cụ thể
* Kiểm soát ứng dụng: Tường lửa có thể được cấu hình để kiểm soát các ứng dụng sử dụng để truy cập vào mạng nội bộ.Các quy tắc kiểm soát ứng dụng này có thể được thiết lập để cho phép hoặc từ chối truy cập từ các ứng dụng cụ thể
* Kiểm soát nội dung: Tường lửa có thể được cấu hình để kiểm soát nội dung lưu lượng truy cập vào ra mạng nội bộ. Các quy tắc kiểm soát này được thiết lập để cho phép hoặc từ chối truy cập từ các loại nội dung cụ thể

**Ưu điểm**:

+ Cung cấp 1 vị trí để giám sát các sự kiện liên quan đến bảo mật

+ Firewall định nghĩa một “choke point” cho quản lý những kết nối của mạng đến internet.

+ Ánh xạ địa chỉ mạng cục bộ thành địa chỉ trên Internet (NAT)

+ Firewall có thể phục vụ như một hệ nền cho IPSet.

**Hạn chế:**

+ Không thể bảo vệ khỏi các cuộc tấn công vượt qua tường lửa.

+ Không thể bảo vệ khỏi các cuộc tấn công nội bộ.

+ Không thể bảo vệ khỏi sự lây truyền của vi rút. Trong hầu hết các trường hợp, việc quét tất cả dữ liệu và tệp được chuyển vào là không thực tế hoặc không thể.

1. **Khái niệm về các phần mềm độc hại, phân loại?**

Phần mềm độc hại là loại phần mềm được thiết kế để truy cập mạng máy tính hay server 1 cách bất hợp pháp.

-Phân loại:

+) Virus: là 1 loại phần phần mềm độc hại có thể sao chép, lây lan sang các máy tính khác.Virus thường dùng để đánh cắp dữ liệu

+) Worm: là loại phần mềm đoọc hại có thể lây lan qua mạng. Worm thường được sử dụng để làm tắc nghẽn mạng hoặc chiếm đoạt quyền hệ thống.

+)Trojan: Trojan là 1 loại phần mềm độc hại được ngụy trang thành 1 chương trình hợp pháp.Trojan thường được sử dụng để đánh cắp dữ liệu hoặc cài phần mềm độc hại khác.

+) Rootkit: có thể che giấu các hoạt động độc hại khỏi các chương trình diệt virus

+)Spyware : Là phần mềm độc hại được thiết kế để theo dõi hoạt động của người dùng. Nó dùng để thu thập dữ liệu ngườicá nhân.

+) Adware: là 1 phần mềm độc hại được thiết kế để hiển thị quảng cáo.

+) Ransomware: là phần mềm tống tiền người dùng để đưa mọi thứ trở về như cũ.

+)Logic bombs:một loại phần mềm độc hại (malware) được chèn vào hệ điều hành hoặc ứng dụng phần mềm để thực hiện một chức năng độc hại sau một thời hạn nhất định hoặc các điều kiện cụ thể được đáp ứng.

+)Back doors:một lối tắt được thêm vào hệ thống hoặc ứng dụng cho phép người dùng truy cập trái phép

+) Zombie and botnet: Phần mềm độc hại được điều khiển từ xa để thực hiện các hành động độc hại.

1. **Phân loại firewall? Trình bày về cơ chế Package Filtering?**

Các loại firewall:

Phân loại Firewall:

Firewall có thể phân loại dựa trên cách hoạt động và phương thức triển khai.

- Phân loại dựa trên cách hoạt động:

* **Packet Filtering Firewall**: (phổ biến nhất) Kiểm tra gói tin dựa trên các quy tắc đã đặt trước.
* **Stateful Inspection Firewall**: Theo dõi trạng thái của kết nối mạng và đưa ra quyết định dựa trên trạng thái đó.
* **Circuit-Level Gateway**: Kiểm tra các phiên TCP để đảm bảo rằng phiên đó được thiết lập một cách hợp lệ.
* **Application-Level Gateway (Proxy Firewall)**: Kiểm tra giao diện ứng dụng và có thể lọc lưu lượng mạng dựa trên các ứng dụng cụ thể.
* **Next-Generation Firewall (NGFW)**: Kết hợp các tính năng của các loại firewall trước đây và thêm các tính năng bảo mật tiên tiến khác.

- Phân loại dựa trên phương thức triển khai:

* **Hardware Firewall**: Được triển khai như một thiết bị vật lý riêng biệt, thường được sử dụng để bảo vệ một mạng toàn diện.
* **Software Firewall**: Được cài đặt trên từng máy tính hoặc thiết bị và bảo vệ chúng khỏi các mối đe dọa.
* **Cloud-Based Firewall**: Được triển khai trên đám mây và cung cấp bảo mật cho các ứng dụng và dữ liệu được lưu trữ trên đám mây.

Cơ chế package filtering :

+) là 1 trong những cơ chế của firewall. Nó kiểm soát lưu lượng mạng dựa trên địa chỉ IP, cổng và giao thức. Mỗi gói tin được kiểm tra và so sánh với các quy tắc được cấu hình trước đó.Nếu gói tin phù hợp với quy tắc thì sẽ chấp nhận và chuyển tiêp.Nếu không nó sẽ từ chối và loại bỏ khỏi mảng. Cơ chế này có ưu điểm là đơn giản , hiệu quả và dễ cấu hình.

1. **Trình bày về Circuit‐level Gateway trong firewall?**

Circuit-level Gateway là 1 loại firewall hoạt động ở tầng phiên. Nó chỉ cho phép kết nối từ địa chỉ IP và cổng cụ thể thông qua firewall.Kiểm tra địa chỉ IP và cổng của máy khách và máy chủ. Nếu hợp lệ thì sẽ kết nối 2 máy. Mọi dữ liệu truyền qua kết nối này thì sẽ được CLG chuyển tiếp mà không cần kiểm tra. Khi kết thúc thì sẽ ngắt kết nối.

Ưu điểm:

+) Nhanh hơn các loại tường lửa khác

+) Chống lại các cuộc tấn công trạng thái và dịch vụ (DOS)

Nhược điểm:

+) Không thể kiểm soát nội dung . Không ngăn chặn các cuộc tấn công SQL Injection

1. **Trình bày về các loại firewall topology?**

Các loại firewall topology

* **Simplest firewall topology**: Đây là cấu trúc đơn giản nhất. Trong hầu hết các trường hợp, firewall là một packet filter firewall.
* **Single Homed Host**: Một Single Homed Host có một packet filter firewall với 1 cổng ứng dụng phía sau firewall. Cổng ứng dụng điều chỉnh quyền truy cập vào máy chủ thông tin. Nó được sử dụng để điều chỉnh truy cập internet bởi các máy nội bộ. Máy chủ thông tin có thể được phép truy cập internet thông qua packet filter firewall, nhưng các thiết bị khác trên mạng nội bộ phải truy cập internet thông qua cổng ứng dụng.
* **Dual Homed Bastion**: Nó có 1 packet filter firewall với cổng ứng dụng phía sau firewall. Máy chủ lưu trữ điều chỉnh quyền truy cập vào máy chủ thông tin. Nó cũng được sử dụng để điều chỉnh truy cập vào internet bởi các máy tính cá nhân mạng nội bộ vì tất cả phải đi qua máy chủ mạng lưu trữ kép. Hệ thống này đảm bảo an ninh hơn vì ngay cả khi firewall bị xâm nhập, mạng nội bộ thì vẫn ẩn.
* **DMZ:** Có 1 packet filter firewall với cổng ứng dụng phía sau firewall. Có 1 packet filter firewall giữa cổng ứng dụng và mạng nội bộ. Khoảng cách giữa 2 packet filter firewall được gọi là DMZ. DMZ chứa tất cả các máy chủ có sẵn công khai của tổ chức. Đây là 1 cấu hình rất an toàn

1. **Khái niệm chữ ký số? Trình bày ứng dụng mã hóa công khai để thực hiện giao dịch có chữ ký số?**

Chữ ký điện tử của một người sử dụng trên một mẩu tin tại một thời điểm xác định được xem như nén mẩu tin về một kích thước cố định và được xác thực. Nó cung cấp các khả năng để

 Kiểm chứng tác giả, ngày và giờ ký;

 Xác thực nội dung mẩu tin;

 Được kiểm chứng bởi bên thứ ba để chống từ chối.

Tạo chữ ký điện tử : sử dụng một hàm băm bản tin và một hàm xác thực có một số khả năng bổ sung như nhận bản băm, thông tin mật của người ký, một số thông tin ngẫu nhiên đặc trưng cho thời điểm ký để tạo ra chữ ký điện tử đính kèm mẩu tin. Đồng thời thuật toán cũng hỗ trợ người nhận kiểm tra chữ ký điện tử kèm với mẩu tin đó.

Ứng dụng:khi A muốn tạo chữ ký RSA, A băm thông điệp, rồi

mã hóa nó bằng khóa riêng của A và đính kèm với thông điệp gửi cho B. Người S

muốn kiểm tra thông điệp có toàn vẹn không và có phải gửi từ A không, sẽ băm thông điệp nhận được, đồng thời giải mã bản băm đính kèm với thông điệp bằng khóa công khai của A.. Và so sánh hai kết quả nhận được ở bước trước, nếu chúng trùng nhau, thì thông điệp không bị sửa và đúng là do A gửi.

1. **Trình bày các bước để mã hóa theo thuật toán Caesar Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa.**

B1: xoá tất các ký tự đặc biệt nằm ngoài bảng chữ cái (ví dụ: số, dấu chấm, dấu phẩy, dấu nháy...) chuyển chữ thường thành chữ hoa

B2: Xác định khóa k.

B2: Chuyển bản rõ thành bản mã bằng cách dịch k kí tự.Mỗi kí tự trong bản rõ sẽ được dịch k kí tự trong bảng chữa cái.

1. **Khi trao đổi thông điệp qua mạng tính toàn vẹn (integrity) được yêu cầu thế nào? Thực hiện yêu cầu toàn vẹn bằng giải pháp nào?**

## Yêu cầu của tính toàn vẹn khi trao đổi thông điệp là đảm bảo thông điệp không bị sửa đổi bởi bất kỳ ai trong quá trình giao dịch.

## Giải pháp:

+ Hàm băm: là công cụ chính được sử dụng để cung cấp tính toàn vẹn. Có hai loại hàm băm:

* Hàm băm không khóa: là những hàm băm không yêu cầu bất kì đầu vào nào ngoài thông điệp được băm
* Hãm băm có khóa: là những hàm yêu cầu khóa cũng như thông điệp được băm

+ Digital Signatures (Chữ Ký Số): Sử dụng chữ ký số để đảm bảo tính toàn vẹn và xác thực. Người gửi sử dụng khóa riêng để tạo chữ ký số và gửi nó cùng với thông điệp. Người nhận sử dụng khóa công khai của người gửi để xác thực chữ ký số và đảm bảo thông điệp không bị thay đổi.

+ Messeage Athentication Codes (MAC): là một đoạn thông tin ngắn được sử dụng để xác thực thư. Giá trị MAC bảo vệ cả tính toàn vẹn dữ liệu của tin nhắn cũng như tính xác thực của nó, bằng cách cho phép người xác minh (người cũng sở hữu khóa bí mật) phát hiện bất kỳ thay đổi nào đối với nội dung tin nhắn.- Giải pháp: sử dụng hàm băm để mã hóa thông tin.

VD: Người dùng có thể sử dụng chữ kí số.Người gửi sẽ sử dụng hàm băm để băm thông điệp rồi gửi. Người dùng sẽ xác minh bằng khóa công khai của người gửi . Nếu xác minh thành công thì có nghĩa là thông điệp đã được gửi bởi người gửi và không bị thay đổi trong quá trình truyền tải.

1. **Trình bày các bước để mã hóa theo thuật toán Vigenere Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa.**

* Bước 1: xoá tất các ký tự đặc biệt nằm ngoài bảng chữ cái (ví dụ: số, dấu chấm, dấu phẩy, dấu nháy...) và chuyển chữ hoa thành chữ thường.

- Bước 2: Chọn khóa là 1 từ .Nếu khóa có độ dài không bằng độ dài của bản rõ thì lặp lại khóa cho đến khi độ dài của khóa bằng độ dài của bản rõ.

* Bước 3: Tìm thứ tự của từng ký tự trong khoá tương ứng với bảng chữ cái.
* Bước 4: Ánh xạ bản rõ với ma trận Vigenere Square and Demonstration theo thứ tự của khoá trong bảng chữ cái được bản mã.
* Ví dụ: Giả sử bạn có thông điệp "HELLO" và khóa "KEY". Ánh xạ ký tự thành số theo bảng chữ cái (ví dụ, 'A' là 0, 'B' là 1, ...)

1. **Nêu những hạn chế (nguy cơ bị bẻ khóa) của các phương pháp mã hóa kinh điển. Lấy ví dụ trong trường hợp Caesar Cipher và  Vigenere Cipher**

* Kỹ thuật đơn giản nên phương pháp mã hóa dữ liệu này có tính an toàn không cao. Việc giữ bí mật thuật toán cũng chỉ đảm bảo ở mức tương đối, nếu trường hợp nó bị rò rỉ ra ngoài hoặc kẻ xấu có thể lần mò giải ra thuật toán thì việc mã hóa trở nên vô nghĩa.
* Ví dụ: Caesar và Vigenere đều dùng phương pháp thay thế 1 ý tự này bằng 1 ký tự khác trong bảng chữ cái nên không gian khóa chính là bảng chữ cái gồm 26 ký tự.

+ Với Caesar Cipher, có thể thử mò 26 vị trí thay đổi ký tự

+ Với Vigenere Cipher, nếu bị lộ từ khóa thì có thể dễ dàng giải mã ngược.

1. **Trên cơ sở phân tích các hạn chế của các thuật toán mã hóa cổ điển (Caesar Cipher, Vigeniar Cipher), các chuẩn mã hóa tiên tiến được thiết kế như thế nào? Trình bày các phép biến đổi cơ bản hay được sử dụng trong các chuẩn mã hóa tiên tiến.**

- **Các chuẩn mã hóa tiên tiến** được thiết kế xoay quanh việc sử dụng “khóa bí mật” để mã hóa dữ liệu, áp dụng kích thước khóa lớn hơn, nhiều khóa hơn, đồng thời thực hiện mã hóa nhiều vòng để làm tăng tính bảo mật và đem lại độ an toàn.

- Phép biến đổi cơ bản hay được sử dụng trong các chuẩn mã hóa tiên tiến:

+ Thay thế + Dịch chuyển

+ Đổi chỗ + Đảo bit

+ Dịch vòng + Phép XOR bit

1. **Nêu các hạn chế của firewall? Các cơ chế thiết lập firewall?**

- Các hạn chế của firewall:

+ Firewall ko thể chống lại các cuộc tấn công ko đi qua nó.

+ Firewall ko thể chống lại các nguy cơ tấn công từ chính bên trong mạng nội bộ mà nó bảo vệ

+ Firewall ko thể bảo vệ chống lại việc chuyển các chương trình hoặc file có virus đi qua nó.

- Các cơ chế thiết lập firewall:

**+** Tất cả truy cập lưu lượng đến hoặc đi, internet đều phải đi qua firewall

**+** Chỉ lưu lượng được ủy quyền, như được xác định bởi chính sách bảo mật cục bộ mới có thể đi qua firewall.

**+** Bản thân firewall không bị xâm nhập. Nó là một hệ thống đáng tin cậy.

1. **Trình bày cách thiết lập một bộ khóa trong thuật toán RSA. Sử dụng bộ khóa đó để mã hóa và giải mã một ký tự tùy chọn**

Cách thiết lập một bộ khóa trong thuật toán RSA:

1. Chọn 2 số nguyên tố lớn khác nhau p, q thỏa mãn điều kiện (lựa chọn ngẫu nhiên và độc lập)

2. Tính tích của p, q: n = p\*q;

3. Tính giá trị hàm số Euler của n: .

4. Tính1 số tự nhiên e sao cho 1<e< và là số nguyên tố cùng nhau với .

5. Chọn số nguyên d, sao cho và gcd (d, e) ≡ 1 (mod )

6. Khóa công khai bao gồm: n và e. Khóa bí mật gồm: n và d.

- Ứng dụng:

1) Chọn p = 11 và q = 3, do đó N = p\*q = 33 ( = 32 < 33 < 64 = )

2) n = (p-1) (q-1) = 20

3) Chọn e = 3 nguyên tố cùng nhau với n

4) Tính nghịch đảo của e trong phép modulo n được d = 7 (3x7 = 21)

5) Khóa công khai KU = (e, N) = (3, 33). Khóa bí mật KR = (d, N) = (7, 33)

6) Mã hóa bản rõ M = 15: C = mod N = 153 mod 33 = 9

7) Giải mã bản mã C = 9: M = mod N = 97 mod 33 = 15

1. **Nêu những hiểu biết của bạn về hàm băm (Hash function) và sử dụng hàm băm trong đảm bảo tính toàn vẹn của thông điệp?**

- Hiểu biết về hàm băm:

+ Một hàm băm thực hiện trên 1 thông điệp đầu vào để tạo ra 1 giá trị băm có độ dài cố định.

+ Các hàm băm phải trả về kết quả giống nhau nếu cùng một dữ liệu đầu vào được sử dụng.

+ Thông điệp được băm được thay đổi nhỏ thì sẽ đưa ra 1 giá trị băm mới khác hoàn toàn nhau.

+ Hàm băm phải có khả năng chống lại va chạm, nghĩa là làm cho khó khăn để tìm ra hai dữ liệu đầu vào khác nhau nhưng có cùng giá trị băm..

+ Rất khó từ giá trị băm để suy ra được thông điệp gốc.

+ Thuật toán băm phổ biến như MD5, SHA-1, SHA-256, SHA-3. Nhưng MD5 và SHA-1 đã không còn an toàn cho nhiều ứng dụng bảo mật.

+ Hàm băm là một công cụ quan trọng trong đảm bảo tính toàn vẹn của thông điệp.

+ Hàm băm có thể được sử dụng trong nhiều ứng dụng khác nhau, bao gồm xác thực thông điệp, kiểm tra tính toàn vẹn của tập tin và dữ liệu.

- Hàm băm có thể được sử dụng để đảm bảo tính toàn vẹn của thông điệp theo cách sau:

* **Trước khi truyền thông điệp, người gửi sử dụng hàm băm để tạo ra giá trị băm của thông điệp.**
* **Người nhận nhận được thông điệp và sử dụng cùng một hàm băm để tạo ra giá trị băm của thông điệp.**
* **Người nhận so sánh giá trị băm do người gửi tạo ra với giá trị băm do mình tạo ra. Nếu hai giá trị băm này khớp nhau, thì thông điệp không bị thay đổi trong quá trình truyền tải.**

1. **Nêu kiến trúc các mô hình cài đặt firewall?**

* **Dual-homed host:** Trong mô hình này, một máy chủ có hai giao diện mạng, một giao diện kết nối với mạng nội bộ và một giao diện kết nối với Internet. Máy chủ này được cấu hình như một firewall, sử dụng các quy tắc để kiểm soát lưu lượng truy cập giữa hai mạng.
* **Screened host:** Mô hình này cũng sử dụng một máy chủ có hai giao diện mạng, nhưng có thêm một máy chủ proxy được cài đặt giữa máy chủ và mạng nội bộ. Máy chủ proxy đóng vai trò là một điểm trung gian cho tất cả lưu lượng truy cập giữa máy chủ và mạng nội bộ. Điều này giúp bảo vệ mạng nội bộ khỏi các mối đe dọa từ Internet.
* **Screened subnet:** Mô hình này sử dụng một mạng con riêng biệt để chứa các máy chủ cần truy cập Internet. Mạng con này được bảo vệ bởi một firewall, chỉ cho phép truy cập từ Internet đến các máy chủ trong mạng con.

1. **Trình bày về SQL Injection? Nêu ví dụ và cách khắc phục**

**- SQL Injection** là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn của các ứng dụng. Được thực hiện bằng cách chèn thêm một đoạn [SQL](https://topdev.vn/blog/sql-la-gi/" \t "_blank) để làm sai lệnh đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ database. **SQL injection** có thể cho phép những kẻ tấn công thực hiện các thao tác như một người quản trị web, trên cơ sở dữ liệu của ứng dụng.

- Cách khắc phục:

+ Cách 1: Vô hiệu hóa các ký tự đặc biệt: loại bỏ các ký tự đặc biệt như dấu ngoặc kép, dấu nháy bằng đặt trước chúng dấu gạch chéo ngược. Hoặc sử dụng phương thức get\_magic\_quotes\_gpc() hoặc sử dụng phương thức real\_escape\_string() khi gọi đến MySQL.

+ Cách 2: Tham số hóa cho câu truy vấn: Không cho phép dữ liệu do người dùng gửi được hiểu là các câu lệnh MySQL. Câu lệnh truy vấn có thể sử dụng tham số ‘?’ như 1 từ thay thế cho dữ liệu.

1. **Trình bày về Cross-Site Scripting (XSS)? Nêu ví dụ và cách khắc phục**

Cross-Site Scripting (XSS) là một loại tấn công web phổ biến, trong đó kẻ tấn công chèn mã JavaScript độc hại vào các trang web hoặc ứng dụng web mà người dùng khác sau đó sẽ thực thi. XSS có thể ảnh hưởng đến người sử dụng cuối, đánh cắp thông tin cá nhân, phiên làm việc hoặc thậm chí kiểm soát tài khoản của họ.

Ví dụ:

1. Stored XSS:

* Kẻ tấn công chèn mã độc hại vào hộp bình luận trên một trang web. Khi người dùng vào những trang web không đáng tin cậy thì đoạn mã sẽ được tải và thực thi

2. Reflected XSS:

* Kẻ tấn công tạo một URL chứa mã độc hại và gửi nó đến người dùng qua email hoặc các phương tiện truyền thông khác.
* Nếu người dùng nhấp vào liên kết, mã độc hại sẽ được thực thi.

3. DOM-based XSS:

* Mã độc hại chèn vào một trang web và tận dụng các sự kiện DOM để thực thi

Ví dụ: Giả sử có một trang web có một hộp thoại tìm kiếm như sau

<input type="text" name="search" id="search">

<button onclick="searchFunction()">Search</button>

Nếu không kiểm tra dữ liệu đầu vào một cách an toàn, kẻ tấn công có thể chèn mã JavaScript độc hại thông qua hộp thoại tìm kiếm:

<script>

alert("XSS Attack - Your sensitive information has been stolen!");

</script>

**Cách khắc phục XSS:**

* **Đầu vào hợp lệ**: Kiểm tra và làm sạch mọi đầu vào từ người dùng trước khi chúng được sử dụng để tạo nội dung trang.
* **Escape các ký tự đặc biệt**: Escape các ký tự như <, >, & để ngăn chặn mã JavaScript không mong muốn được thực thi.
* **Sử dụng Content Security Policy (CSP)**: Thiết lập chính sách an toàn nơi chỉ định từ đâu tài nguyên có thể được tải và giảm khả năng thực thi mã độc hại.
* **HttpOnly và Secure Cookies**: Sử dụng cookies với thuộc tính HttpOnly để ngăn chặn truy cập thông qua mã JavaScript và Secure để chỉ cho phép truy cập qua kết nối an toàn (HTTPS).
* **XSS Protection Header**: Sử dụng header HTTP như X-XSS-Protection để bật cơ chế bảo vệ XSS tích hợp trong trình duyệt.
* **Regular Security Audits**: Thực hiện kiểm tra bảo mật định kỳ để phát hiện và khắc phục các lỗ hổng XSS mới.

1. **Trình bày về XML Injection? Nêu ví dụ và cách khắc phục**

XML Injection là một kỹ thuật tấn công mà kẻ tấn công chèn dữ liệu độc hại vào các tài liệu XML. Điều này có thể dẫn đến việc thực thi các hành động không mong muốn hoặc thu thập thông tin nhạy cảm từ hệ thống.

Ví dụ:

Giả sử có một ứng dụng Web sử dụng XML để truyền thông tin giữa máy khách và máy chủ, nếu người dùng có thể kiểm soát dữ liệu được chèn vào XML, nguy cơ XML injection sẽ tăng lên.

**Cách khắc phục:**

* **Kiểm soát đầu vào**: Hạn chế và kiểm soát đầu vào từ người dùng. Chắc chắn rằng dữ liệu được chèn vào XML được xử lý đúng cách và không chứa các ký tự đặc biệt không mong muốn.
* **Sử dụng API an toàn**: Nếu có thể, sử dụng API an toàn như DOMDocument để xử lý XML thay vì các phương thức xử lý chuỗi.
* **Escape các ký tự đặc biệt**: Trước khi chèn dữ liệu vào XML, escape các ký tự đặc biệt như &, <, >, để đảm bảo chúng không thể được hiểu là các phần của đoạn mã XML.
* **Sử dụng mô hình dữ liệu XML an toàn**: Định nghĩa một mô hình dữ liệu XML an toàn và đảm bảo rằng các yêu cầu đầu vào tuân thủ theo mô hình đó.
* **Kiểm tra và xác thực đầu vào**: Thực hiện kiểm tra và xác thực đầu vào để đảm bảo rằng nó tuân thủ định dạng và kiểu dữ liệu mong đợi.
* **Áp dụng chính sách an toàn**: Sử dụng chính sách an toàn như XML Signature để đảm bảo tính toàn vẹn và nguồn gốc của dữ liệu XML.

1. **Trình bày về Directory Traversal/Command Injection? Nêu ví dụ về lỗi file upload và cách khắc phục**

* Directory Traversal - một loại tấn công trong đó kẻ tấn công cố gắng truy cập các tệp và thư mục mà họ không có quyền truy cập bằng cách sử dụng các ký tự đặc biệt để vượt qua giới hạn của hệ thống tệp và thư mục. Nếu thành công, họ có thể truy cập thông tin nhạy cảm hoặc thậm chí là thực hiện các hành động không mong muốn.
* Command Injection: kẻ tấn công chèn các lệnh hệ thống vào các trường dữ liệu người dùng có thể kiểm soát. Nếu ứng dụng không xử lý đầu vào một cách an toàn, lệnh được chèn có thể được thực thi trên máy chủ.
* Lỗi file upload: Giả sử có một ứng dụng web cho phép người dùng tải lên các tệp hình ảnh và lưu trữ chúng trong một thư mục cố định trên máy chủ. Nếu không kiểm tra đầu vào của người dùng một cách đầy đủ, người tải lên có thể thực hiện tấn công Directory Traversal bằng cách tải lên một tệp có tên như "../../../../../etc/passwd".
* Cách khắc phục:

+ Kiểm Tra và Chuẩn Hóa Đầu Vào

+ Sử Dụng White List:

+ Giới Hạn Quyền Truy Cập: Thiết lập quyền truy cập cho các thư mục và tệp để hạn chế khả năng của kẻ tấn công truy cập thông tin nhạy cảm.

+ Ngăn Chặn Thực Thi Lệnh Hệ Thống

+ Mã Hóa Tên Tệp và Đường Dẫn: Mã hóa tên tệp và đường dẫn trước khi sử dụng chúng để ngăn chặn kẻ tấn công sử dụng các ký tự đặc biệt.

+ Bảo Mật Tốt Hơn Cho File Upload: Sử dụng các thư viện chuyên về xử lý tệp tải lên, và kiểm tra kích thước, loại tệp và nội dung của chúng để đảm bảo tính an toàn.

1. **Trình bày một số hình thức tấn công phía client: Header manipulation, Cookies**
2. Header manipulation

* Header Injection: Kẻ tấn công có thể cố gắng chèn các ký tự đặc biệt như newlines (\n) để thêm các tiêu đề HTTP bổ sung hoặc thay đổi các giá trị tiêu đề có sẵn. Mục tiêu là thay đổi cách mà máy chủ xử lý yêu cầu hoặc phản hồi.
* CORS (Cross-Origin Resource Sharing) Abuse: Kẻ tấn công có thể thực hiện tấn công CORS để lừa đảo trình duyệt để gửi yêu cầu HTTP từ một nguồn không an toàn đến một nguồn an toàn.
* Referer Spoofing: Thay đổi giá trị của trường "Referer" trong yêu cầu để giả mạo nguồn của yêu cầu.
* Cách khắc phục:

+ Kiểm tra và làm sạch dữ liệu đầu vào để ngăn chặn chèn ký tự đặc biệt.

Sử dụng HTTP Strict Transport Security (HSTS) để đảm bảo kết nối an toàn và giữ cho truyền thông chỉ được thực hiện qua kết nối HTTPS.

1. Cookies

* Session Hijacking: Kẻ tấn công có thể đánh cắp cookie phiên (session cookie) của người dùng và sử dụng nó để giả mạo phiên đăng nhập của họ.
* Cookie Poisoning: Thay đổi giá trị của cookie để làm thay đổi cách mà ứng dụng hoặc trang web xử lý thông tin của người dùng.
* Cross-Site Scripting (XSS): Tấn công XSS có thể chèn mã độc hại vào trang web, và khi người dùng truy cập trang đó, mã độc hại có thể đánh cắp cookie hoặc thực hiện các hành động độc hại khác.
* Cách khắc phục:

+ Sử dụng Secure và HttpOnly flags cho các cookie để bảo vệ chúng khỏi tấn công MITM và XSS.

+ Sử dụng token phiên (session token) có hiệu suất và cập nhật nó định kỳ để giảm nguy cơ session hijacking.

1. **Trình bày hình thức tấn công phía client: Attachments, Session Hijacking**
2. Attachments:

* Malicious Attachments: Kẻ tấn công có thể gửi email hoặc tin nhắn chứa các tệp đính kèm độc hại (ví dụ: tệp tin thực thi, malware) đến người dùng. Mục tiêu là khi người dùng mở hoặc tải về tệp đính kèm, mã độc hại được triển khai.
* Malicious Links in Attachments: Kẻ tấn công có thể chèn các liên kết độc hại vào tệp đính kèm. Khi người dùng nhấp vào liên kết, họ có thể bị đưa đến trang web chứa mã độc hại hoặc bị đánh cắp thông tin.
* Drive-by Downloads: Kẻ tấn công có thể chèn mã độc hại vào các tệp đính kèm, như hình ảnh hoặc tệp tải về từ trang web. Khi trình duyệt của người dùng tải về tệp này, mã độc hại có thể được thực thi mà không cần sự tương tác của người dùng.

1. Session Hijacking:

* Man-in-the-Middle (MITM) Attacks: Kẻ tấn công có thể chặn giao tiếp giữa người dùng và máy chủ để đánh cắp session cookies hoặc thông tin đăng nhập. Điều này có thể xảy ra trong các mạng Wi-Fi công cộng hoặc qua các phương tiện truyền thông không an toàn.
* Session Fixation: Kẻ tấn công cố gắng thiết lập hoặc đặt lại session ID của người dùng. Khi người dùng đăng nhập vào trang web, kẻ tấn công có thể lấy được session ID và sử dụng nó để đăng nhập vào tài khoản của người dùng.
* Cross-Site Scripting (XSS): Kẻ tấn công có thể chèn mã JavaScript độc hại vào trang web mà người dùng truy cập. Mã độc hại có thể đánh cắp session cookies khi người dùng truy cập trang web bị tấn công.

1. **Trình bày về DoS và các loại tấn công DoS**

* Dos (Denial of Service) - làm mất khả năng hoặc giảm hiệu suất của một dịch vụ, hệ thống, hay mạng, làm cho người dùng không thể truy cập hoặc sử dụng chúng một cách hiệu quả.
* Mục tiêu của cuộc tấn công DoS là tạo ra môi trường không thể sử dụng, làm cạn kiệt tài nguyên, hoặc làm cho dịch vụ trở nên chậm chạp đến mức không thể chấp nhận được.
* Các loại tấn công:

+ Tấn Công Theo Dòng (Flood Attacks): UDP Flood: Tấn công gửi một lượng lớn gói tin UDP đến một địa chỉ đích, làm cạn kiệt băng thông và tài nguyên mạng. ICMP Flood: Gửi lượng lớn yêu cầu ICMP (ping) đến một máy chủ, gây quá tải và làm giảm hiệu suất hệ thống.

+ Tấn Công Áp Lực (Amplification Attacks): DNS Amplification: Sử dụng yêu cầu DNS với kích thước nhỏ để yêu cầu lớn, khiến cho máy chủ DNS gửi một lượng lớn dữ liệu đến máy chủ đích. NTP Amplification: Sử dụng yêu cầu NTP để gửi lượng lớn dữ liệu đến một máy chủ đích.

+ Tấn Công Áp Sát (Low-and-Slow Attacks): Slowloris Attack: Giữ mở một số kết nối tới máy chủ, nhưng giữ chúng mở rất chậm. Điều này khiến cho máy chủ không thể phục vụ nhiều yêu cầu từ người dùng hợp lệ mới.

+ Tấn Công Tài Nguyên (Resource Exhaustion Attacks): Ping of Death: Gửi gói tin ping với kích thước lớn, khiến cho máy chủ hoặc thiết bị mạng quá tải khi phải xử lý gói tin quá lớn.

+ Tấn Công Mạng Phân Cấp (Layer 7 DoS Attacks): HTTP/HTTPS Flood: Gửi một lượng lớn yêu cầu HTTP/HTTPS đến một máy chủ, làm cạn kiệt tài nguyên và làm chậm dịch vụ.

+ Tấn Công SYN/ACK (SYN/ACK Flood): Gửi một lượng lớn yêu cầu kết nối TCP SYN (synchronize) và không phản hồi với các yêu cầu kết nối ACK (acknowledge), làm cạn kiệt tài nguyên và tạo ra các kết nối giả mạo.

1. **Trình bày về Interception, Poisioning và các hình thức tấn công chiếm quyền truy cập (Access Rights)**
2. **Interception (Chặn Lấy):**

* Packet Sniffing: Kỹ thuật này liên quan đến việc chặn và thu thập gói tin dữ liệu trên mạng. Kẻ tấn công sử dụng công cụ sniffing để đọc thông tin qua mạng, bao gồm cả dữ liệu nhạy cảm như tên đăng nhập và mật khẩu.
* Man-in-the-Middle (MITM) Attacks: Kẻ tấn công chèn mình vào giữa giao tiếp giữa hai bên và có thể theo dõi, thay đổi, hoặc lấy mất thông tin truyền qua mạng. Các tấn công MITM bao gồm ARP Spoofing, DNS Spoofing, và SSL Stripping**.**

1. **Poisoning (Độc Hại):**

* DNS Spoofing/Poisoning: Kẻ tấn công thay đổi bản đồ địa chỉ IP của tên miền trong bảng định tuyến DNS, dẫn đến việc chuyển hướng người dùng đến địa chỉ IP giả mạo. Điều này có thể dẫn đến việc người dùng bị đánh lừa và truy cập trang web giả mạo.
* ARP Spoofing/Poisoning: Kẻ tấn công gửi các gói tin ARP giả mạo để liên kết địa chị MAC của mình với địa chỉ IP của một máy tính khác trong mạng. Điều này có thể dẫn đến việc tất cả lưu lượng mạng đến máy tính đó được chuyển đến kẻ tấn công.

1. **Tấn Công Chiếm Quyền Truy Cập (Access Rights Attacks):**

* Brute Force Attacks: Tấn công chiếm quyền truy cập bằng cách thử tất cả các khả năng của tên đăng nhập và mật khẩu cho đến khi tìm ra thông tin đăng nhập chính xác.
* Phishing: Kẻ tấn công gửi các email hay thông báo giả mạo từ tổ chức hoặc dịch vụ đáng tin cậy để lừa dối người dùng nhập thông tin đăng nhập vào các trang web giả mạo.
* Session Hijacking: Tấn công chiếm quyền truy cập bằng cách đánh cắp hoặc áp đặt phiên đăng nhập của người dùng. Điều này có thể thực hiện thông qua việc sử dụng cookie phiên, tấn công MITM, hoặc tấn công XSS.
* Privilege Escalation: Kẻ tấn công thực hiện các tuyến tấn công để nâng cao quyền đặc quyền của mình từ người dùng bình thường lên thành quản trị viên hoặc quyền hệ thống.
* Zero-Day Exploits: Sử dụng lỗ hổng bảo mật chưa được biết đến (zero-day) để chiếm quyền truy cập vào hệ thống hoặc ứng dụng.

Phần II: Bài tập

1. **Sử dụng Java Cryptography Architecture (JCA), viết class SymmetricCryp để thực hiện các chức năng mã hóa và giải mã đối xứng theo thuật toán AES gồm các hàm**

**public** **class** SymmetricCrypt {

**private** **final** String Algorithm = "AES";

//khóa

**private** SecretKey secretKey;

//khóa theo chuỗi cho trước

**private** SecretKeySpec secretKeySpec;

//bản sinh khóa theo thuật toán đã chọn

**private** KeyGenerator keyGen;

//bản mã

**private** Cipher cipher;

* 1. **Sinh khóa đối xứng**

**public** SymmetricCrypt()

**throws** NoSuchAlgorithmException

{

keyGen = KeyGenerator.*getInstance*(Algorithm);

secretKey = keyGen.generateKey();

}

**1.2 Tạo khóa đối xứng từ một chuỗi cho trước**

public SymmetricCtyptor()  
 throws NoSuchAlgorithmException{  
 //Khởi tạo bộ tạo khóa theo thuật toán đã chọn  
 keyGen=KeyGenerator.getInstance(algorithm);  
 //tạo khóa  
 secretKey = keyGen.generateKey();  
   
 String key="mothaibabonnamsa";  
 secretKeySpec=new SecretKeySpec(key.getBytes(),algorithm);  
}

**1.3 Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa, tham số thứ 2 là khóa được tạo ở 1.1**

public String encryptText(String msg,SecretKey key) throws NoSuchAlgorithmException,  
 InvalidKeyException, NoSuchPaddingException, UnsupportedEncodingException, IllegalBlockSizeException,  
 BadPaddingException

{  
 cipher = Cipher.*getInstance*(Algorithm);

cipher.init(Cipher.***ENCRYPT\_MODE***, key);

**return** Base64.*getEncoder*().encodeToString(

cipher.doFinal(msg.getBytes("UTF-8")));

}

**1.4 Mã hóa thông điệp có 2  tham số: Tham số 1 là chuỗi cần mã hóa tham số 2 là khóa được tạo ở 1.2**

**public** String encryptText(String msg, SecretKeySpec key)

**throws** NoSuchAlgorithmException,

NoSuchPaddingException,

InvalidKeyException,

UnsupportedEncodingException,

IllegalBlockSizeException,

BadPaddingException

{

cipher = Cipher.*getInstance*(Algorithm);

cipher.init(Cipher.***ENCRYPT\_MODE***, key);

**return** Base64.*getEncoder*().encodeToString(

cipher.doFinal(msg.getBytes("UTF-8")));

}

**1.5 Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.1**

**public** String decryptText(String msg, SecretKey key)

**throws** NoSuchAlgorithmException,

NoSuchPaddingException,

InvalidKeyException,

IllegalBlockSizeException,

BadPaddingException,

UnsupportedEncodingException

{

cipher = Cipher.*getInstance*(Algorithm);

cipher.init(Cipher.***DECRYPT\_MODE***, key);

**return** **new** String(cipher.doFinal(

Base64.*getDecoder*().decode(msg)),"UTF-8");

}

**1.6 Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.2**

**public** String decryptText(String msg, SecretKeySpec key)

**throws** NoSuchAlgorithmException,

NoSuchPaddingException,

InvalidKeyException,

IllegalBlockSizeException,

BadPaddingException,

UnsupportedEncodingException

{

cipher = Cipher.*getInstance*(Algorithm);

cipher.init(Cipher.***DECRYPT\_MODE***, key);

**return** **new** String(cipher.doFinal(

Base64.*getDecoder*().decode(msg)),"UTF-8");

}

}

**1.7 Sử dụng class SymetricCryp để mã hóa một chuỗi cho trước hoặc giải mã ra bản rõ từ bản** mã cho trước.

**public** **class** SymmetricApp {

**public** **static** **void** main(String[] args)

**throws** Exception

{

SymmetricCtyptor SC=new SymmetricCtyptor();  
String msg="Nguyễn Văn An";  
String en\_msg=SC.encryptText(msg,SC.getSecretKeySpec());  
System.out.println("Pain text:"+msg);  
System.out.println("Encrypted text:"+en\_msg);  
String de\_msg=SC.decryptText(en\_msg,SC.getSecretKeySpec());  
System.out.println("Decrypted text:"+de\_msg);

}

}

**2. Sử dụng Java Cryptography Architecture (JCA) để viết các hàm băm thông điệp cho trước** theo thuật toán:

**2.1. MD5**

public String md5Digest(String msg){

try{

MessageDigest md = MessageDigest.getInstance("MD5");

md.update(msg.getBytes("UTF-8"),0,msg.length());

return DatatypeConverter.printHexBinary(md.digest());

} catch (Exception ex){

ex.printStackTrace();

}

return null;

}

**2.2. SHA1**

public String sha1Digest(String msg){

try {

MessageDigest md = MessageDigest.getInstance("SHA-1");

md.update(msg.getBytes("UTF-8"),0, msg.length());

return DatatypeConverter.printHexBinary(md.digest());

} catch (Exception ex){

ex.printStackTrace();

}

return null;

}

**2.3. SHA-256**

public String getHashSHA256(String origin) throws Exception {

String sha256;

MessageDigest m = MessageDigest.getInstance("SHA-256");

m.update(origin.getBytes("UTF-8"), 0, origin.length());

sha256 = new BigInteger(1, m.digest()).toString(16);

return sha256;

}

}

public static void main(String[] args){

Digest digest = new Digest();

String msg ="Hello Word";

String md5Hash = digest.md5Digest(msg);

System.out.println("MD5 Hash: " + md5Hash);

String sha1Hash = digest.sha1Digest(msg);

System.out.println("SHA-1 Hash: "+sha1Hash);

String sha256Hash = digest. getHashSHA256 (msg);

System.out.println("SHA-256 Hash: "+sha256Hash);

}

C2:

public byte[] signData(String msg, PrivateKey privateKey) throws Exception {  
 Signature signature = Signature.getInstance("SHA256withRSA");  
 signature.initSign(privateKey);  
 signature.update(msg.getBytes());  
 return signature.sign();  
}  
  
// Phương thức xác minh chữ ký  
public boolean verifySignature(String msg, byte[] signature, PublicKey publicKey) throws Exception {  
 Signature sig = Signature.getInstance("SHA256withRSA");  
 sig.initVerify(publicKey);  
 sig.update(msg.getBytes());  
 return sig.verify(signature);  
}

**3. Sử dụng Java Crypto Architecture (JCA), viết chương trình mã hóa và giải mã sử dụng mã hóa công khai RSA:**

Viết class RSAKeyGen để sinh bộ khóa công khai gồm các hàm

public class KeyGenerator {  
   
 private KeyPairGenerator keyGen;  
   
 private KeyPair keypair;  
  
 private PrivateKey privateKey;  
  
 private PublicKey publicKey;

**3.1 Sinh bộ khóa theo thuật toán RSA**

public KeyGenerator(int keylength)   
 throws NoSuchAlgorithmException {  
 //tạo đối tượng KeyGenerator để sinh khóa  
 keyGen = KeyPairGenerator.getInstance("RSA");  
 //khởi tạo bộ sinh khóa với tham số đầu vào là số byte  
 keyGen.initialize(keylength);  
 //sinh khóa  
 keypair = keyGen.generateKeyPair();  
 privateKey = keypair.getPrivate();  
 publicKey = keypair.getPublic();  
 }

**3.2 Lưu bộ khóa ra file**

public void writeToFile(String path, byte[] key)   
 throws FileNotFoundException, IOException  
{  
 File f = new File(path);  
 //tạo thư mục chung chứa cặp khóa  
 f.getParentFile().mkdir();  
 //lưu khóa ra file tương ứng  
 FileOutputStream fos = new FileOutputStream(f);  
 fos.write(key);  
 fos.flush();  
 fos.close();  
}

Viết class RSACryp để sử dụng bộ khóa đã lưu trong file để mã hóa và giải mã thông điệp theo thuật toán RSA

public class RSACryp {  
 private Cipher cipher;

**3.3 Hàm đọc file để nạp khóa bí mật, tham số truyền vào là tên file**

public PrivateKey getPrivateKey(String filename)   
 throws IOException,   
 NoSuchAlgorithmException,   
 InvalidKeySpecException  
{  
 byte[] keybytes = Files.readAllBytes(  
 new File(filename).toPath());  
 PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keybytes);  
 KeyFactory kf = KeyFactory.getInstance("RSA");  
 return kf.generatePrivate(spec);  
}

**3.4 Hàm đọc file để nạp khóa công khai, tham số truyền vào là tên file**

public PublicKey getPublicKey(String filename)   
 {  
 byte[] keybytes = Files.readAllBytes(  
 new File(filename).toPath());  
 X509EncodedKeySpec spec = new X509EncodedKeySpec(keybytes);  
 KeyFactory kf = KeyFactory.getInstance("RSA");  
 return kf.generatePublic(spec);  
}

**3.5 Hàm mã hóa một thông điệp cho trước bằng khóa bí mật, tham số truyền vào gồm tham số thứ nhất là thông điệp cần mã hóa, tham số thứ 2 là khóa bí mật**

public String encryptText(String msg, PrivateKey key)   
 {  
 cipher = Cipher.getInstance("RSA");  
 cipher.init(Cipher.ENCRYPT\_MODE, key);  
 return Base64.getEncoder().encodeToString(  
 cipher.doFinal(msg.getBytes("UTF-8")));  
}

**3.6 Hàm giải mã thông điệp bằng khóa công khai, tham số truyền vào gồm tham số thứ nhất là bản mã cần giải mã, tham số thứ 2 là khóa công khai.**

public String decryptText(String msg, PublicKey key)   
{  
 cipher = Cipher.getInstance("RSA");  
 cipher.init(Cipher.DECRYPT\_MODE, key);  
 return new String(cipher.doFinal(  
 Base64.getDecoder().decode(msg)),"UTF-8");  
}

**3.7 Sử dụng lớp RSACryp để mã hóa một chuỗi và băm chuỗi theo một thuật toán băm cho trước rồi lưu cả bản mã và bản băm vào một file**

public static void main(String[] args)   
 throws Exception {  
  
 RSACryp AC = new RSACryp ();  
   
 String msg = "Hoàng Minh Tuấn";

/\*----------------Sử dụng RSA--------------\*/  
 //kí số  
 String encrypted\_msg = AC.encryptText(msg,   
 AC.getPrivateKey("C:/PrivateKey"));  
 System.out.println("Plain text: " + msg);  
 System.out.println("Encrypted text: " + encrypted\_msg);  
 //Mã hóa  
 String decrypted\_msg = AC.decryptText(encrypted\_msg,   
 AC.getPublicKey("C:/PublicKey"));  
 System.out.println("Decrypted text: " + decrypted\_msg);

/\*----------------Sử dụng SHA2--------------\*/  
 byte[] signature = AC.signData(msg,  
 AC.getPrivateKey("C:/PrivateKey"));  
 //xác nhận xem có kí thành công hay không  
 boolean isCheck= AC.verifySignature(msg,signature,AC.getPublicKey("C:/PublicKey"));  
 String str=isCheck?"Kí thành công":"kí thất bại";  
 System.out.println(str);  
}

**3.8 Đọc file chứa bản mã và bản băm của một thông điệp, sử dụng lớp RSACryp để giải mã ra thông điệp gốc và băm lại bằng thuật toán băm cho trước. Sau đó đem so sánh hai bản băm để xác nhận tính toàn vẹn**

public static void main(String[] args)   
 throws Exception {  
 // TODO code application logic here  
 RSACrypAC = new RSACryp();  
 File file = new File("C:/text.txt");  
 Scanner scanner = new Scanner(file);  
 String msg="";  
 // Đọc dữ liệu từ tệp  
 while (scanner.hasNextLine()) {  
 // Đọc từng dòng dữ liệu  
 String line = scanner.nextLine();  
 System.out.println(line);  
 msg+=line;  
 }  
 /\*----------------Sử dụng RSA--------------\*/  
 //kí số  
 String encrypted\_msg = AC.encryptText(msg,   
 AC.getPrivateKey("C:/PrivateKey"));  
 System.out.println("Plain text: " + msg);  
 System.out.println("Encrypted text: " + encrypted\_msg);  
 //Mã hóa  
 String decrypted\_msg = AC.decryptText(encrypted\_msg,   
 AC.getPublicKey("C:/PublicKey"));  
 System.out.println("Decrypted text: " + decrypted\_msg);

/\*----------------Sử dụng SHA2--------------\*/  
 byte[] signature = AC.signData(msg,  
 AC.getPrivateKey("C:/PrivateKey"));  
 //xác nhận xem có kí thành công hay không  
 boolean isCheck= AC.verifySignature(msg,signature,AC.getPublicKey("C:/PublicKey"));  
 String str=isCheck?"Kí thành công":"kí thất bại";  
 System.out.println(str);  
}

**Bài 4 : Chương trình SSH App-script : (Base System 1: ch11, Base System 2: ch12)**

**Base System 1: ch11 (system(“ls challenge/app-script/ch11/.passwd”))**

Bước 1: Vì trong thư mục hiện tại ta không có quyền tạo thư mục và file nên ta sẽ tạo một thư mục của riêng ta ở trong /tmp bằng lệnh: mkdir /tmp/lagi

Bước 2: Copy chương trình cat của hệ thống nằm trong /bin vào trong thư mục vừa tạo với tên là ls bằng lệnh

cp /bin/cat /tmp/lagi/ls

Bước 3: Để hệ thống có thể tìm kiếm đến ls trong /tmp/lagi ta thêm đường dẫn tới /tmp/lagi vào biến môi trường $PATH bằng lệnh: export PATH=/tmp/lagi:$PATH

Bước 4: Chạy chương trình ch11 để hiển thị mật khẩu bằng lệnh: ./ch11

**Base System 2:ch12 (system(“ls -lA challenge/app-script/ch12/.passwd”))**

B1 : Tạo thư mục tên là “Tên thư mục” trong thư mục /tmp bằng lệnh :

mkdir /tmp/Tên thư mục

B2 : Tạo một file shell-script với tên là ls trong thư mục vừa tạo bằng trình soạn thảo văn bản nano :

nano /tmp/Tên thư mục/ls

B3 : Sau khi trình soạn thảo nano xuất hiện ta soạn thảo file shell với hai lệnh:

- lệnh thứ nhất khai báo /bin/sh

- lệnh thứ hai là chạy trình đọc file /bin/cat với tham số là tham số thứ hai trên dòng lệnh .

Sau đây là nội dung file ls cần soạn thảo :

#!/bin/sh

/bin/cat "/challenge/app-script/ch12/.passwd"

B4 : Lưu nội dung file ls bằng phím nóng Ctrl – O rồi thoát khỏi nano bằng phím nóng Ctrl – X

B5: Thêm quyền thực thi cho file shell tên là ls vừa soạn thảo bằng lệnh: chmod +x /tmp/Tên thư mục/ls

B6: Để hệ thống có thể tìm kiếm đến ls trong /tmp/Tên thư mục ta thêm đường dẫn tới /tmp/Tên thư mục vào biến môi trường $PATH bằng lệnh

export PATH=/tmp/Tên thư mục:$PATH

B7: Chạy chương trình ch12 để hiển thị mật khẩu bằng lệnh: ./ch12