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

* **Tấn công đánh chặn:** nguồn tài nguyên không sẵn sàng bị chặn.

Ví dụ: A gửi cho B 1 gói tin bị chặn bởi C và gói tin đấy ko đến được tới B. Liên hệ: Availability

* **Tấn công gián đoạn:** dữ liệu được đọc bởi người ko có quyền. Đây là kiểu tấn công bảo mật. Ví dụ: A gửi cho B gói tin bị C xem nhưng gói tin vẫn đến được tới B. Liên hệ: Confidentiality
* **Tấn công sửa đổi:** dữ liệu được sủa đổi bởi người không có quyền. Đây là kiểu tấn công toàn vẹn dữ liệu.

Ví dụ: A gửi cho B 1 gói tin, C chặn gói tin sửa đổi nội dung rồi gửi gói tin đã bị chỉnh sửa đến B. Liên hệ: Integrity

* **Tấn công giả mạo:** Giả mạo người gửi.

Ví dụ: C giả mạo A gửi gói tin đến B. Liên hê: Message Origin Anthentication, Timeliness

**Câu 2: Trình bày các mục tiêu của an ninh mạng? Lấy ví dụ?**

**Có 3 mục tiêu chủ yếu:**

* **Tính bảo mật.** Ví dụ: A và B ko muốn thông điệp của họ bị đọc bởi người khác.
* **Tính toàn vẹn.** Ví dụ: A và B ko muốn thông điệp của họ bị thay đổi bởi người khác.
* **Tính sẵn sàng.** Ví dụ: Đảm bảo máy chủ B luôn nhận được yêu cầu hợp lệ.

**Ngoài ra còn 6 mục tiêu nữa:**

* **Xác thực thực thể.** Ví dụ: A muốn xác nhận thông điệp được gửi từ B chắc chắn từ B gửi.
* **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
* **Chống từ chối:** Ví dụ: A ko thể nào từ chối được thông điệp của mình đã gửi và đã nhận.
* **Cấp phép:** 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.
* **Quyền truy cập:** Ví dụ: giống như phân quyền trong DB

## Câu 3: Khái niệm mã hóa công khai? Sử dụng mã hóa công khai làm chữ kí số?

**Mã hóa khóa công khai** là một dạng [mã hóa](https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a) cho phép người sử dụng trao đổi các [thông tin](https://vi.wikipedia.org/wiki/Th%C3%B4ng_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)) chung [bí mật](https://vi.wikipedia.org/w/index.php?title=B%C3%AD_m%E1%BA%ADt&action=edit&redlink=1) 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 M => 1 giá trị băm
* Giá trị băm dùng khoá private của A để mã hoá => chữ ký điện tử
* Đính kèm chữ ký điện tử với thông điệp M rồi gửi đến B.

Bên B:

* B gói tin của A gửi lấy ra thông điệp M và băm => giá trị băm (1)
* Lấy chữ ký mã hoá theo khoá public của A => giá trị băm (2)
* Nếu giá trị băm (1) bằng băm (2) thì chữ ký được xác thực và ngược lại.

## Câu 4: Trình bày khái niệm firewall, cơ chế kiểm soát, ưu điểm và hạn chế?

Khái niệm Firewall: Firewall là cách thức để ngăn chặn truy cập tới mạng LAN hoặc cổng của mạng LAN. FireWall liên quan đến mục đích an ninh kiểm soát quyền truy cập.

**Cơ chế kiểm soát:**

+ Tất cả lưu lượng từ mạng bên trong ra bên ngoài hoặc ngược lại đều phải đi qua Firewall.

+ Chỉ có lưu lượng được phép, như được định danh bởi chính sách bảo mật cục bộ mới được phép đi qua Firewall

+ Bản thân Firewall phải có khả năng tránh được sự xâm nhập bất hợp pháp, và phải được thiết kế là một hệ thống tin cậy.

**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 sự 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.

## Câu 5: Khái niệm về các phần mềm độc hại, phân loại?

**\* Phần mềm độc hại**: là phần mềm xâm nhập vào hệ thống máy tính mà không có sự hiểu biết hoặc chấp thuận của người dùng và sau đó thực hiện hành động không mong muốn, thường có hại.

**\* Phân loại các phần mềm độc hại:**

**- Phụ thuộc vào chương trình chủ**:

**+ Trapdoors:** cài đặt một điểm truy cập trái phép cho phép người dùng khác truy cập vào máy tính mà không có sự đồng ý của bạn

**+ Logic Bombs:** đoạn mã độc hại được kích hoạt khi gặp tình huống đặc biệt

**+ Trojan Horses:** ẩn mình dưới dạng chương trình hữu ích và có những chức năng người dùng cần, nhưng thực chất lại âm thầm thực hiện hành vi xấu, thậm chí cài đặt virus vào máy

**+ Viruses:** tự sao chép bản thân nó vào các chương trình khác của hệ thống

**- Độc lập lây nhiễm**:

+ **Bacteria**: làm full ổ cứng, hoặc CPU

+ **Worms**: tự nhân bản trong hệ thống hoặc thậm chí lây từ máy tính này sang máy tính khác

## Câu 6: Phân loại firewall? Trình bày về cơ chế Package Filtering?

**Phân loại Firewall:**

Có 3 loại firewall phổ biến

+ **Packet-Filtering Firewall:** là loại firewall phổ biến nhất

+ **Circuit-level gateway**: làm việc ở cấp độ phiên của mô hình lớp mạng OSI

+ **Application-level gateway**: thường được sử dụng cùng với Packet-Filtering Firewall.

**Cơ chế Packet-Filtering Firewall:**

+ Firewall sẽ áp dụng một tập các quy tắc cho mỗi gói tin IP đi đến và sau đó quyết định sẽ chuyển tiếp hay loại bỏ gói tin đó đi.

+ Bộ lọc gói thường được thiết lập bằng danh sách quy tắc dựa vào các trường header IP và TCP.

+ Có 2 chính sách mặc định: discard và forward

**Câu 7: Nêu các hình thức tấn công? Liên hệ với các mục tiêu của an ninh mạng**

Giống câu 1.

## Câu 8: Trình bày về Circuit-level Gateway trong firewall:

- Làm việc ở tầng vận chuyển của mô hình OSI, hoặc nó ở tầng “shim-layer” giữa tầng ứng dụng và tầng vận chuyển của ngăn TCP/IP.

- Nó kiểm soát cơ chế bắt tay của TCP giữa các gói tin để xác định liệu 1 phiên yêu cầu có hợp lệ hay không.

- Là hệ thống tiêu chuẩn độc lập.

- Cổng thường chuyển tiếp các TCP segment từ 1 kết nối đến một kết nối khác mà ko cần xem xét nội dung của nó.

- Chức năng bảo mật bao gồm việc xác định những kết nối nào sẽ được cho phép

- Dùng trong trường hợp người quản lý hệ thống tin tưởng những người dùng nội bộ.

## Câu 9: Trình bày 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 cũng 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

## Câu 10: Khái niệm chữ ký số? Sử dụng mã hóa công khai thực hiện giao dịch có chữ ký số?

\* **Chữ ký số** là thông tin đi kèm theo dữ liệu (văn bản: word, excel, pdf…; hình ảnh; video...) nhằm mục đích xác định người chủ của dữ liệu đó, có thể được cung cấp đến dịch vụ an ninh về toàn vẹn thông điệp.

\* **Sử dụng mã hóa công khai thực hiện giao dịch có chữ ký số:**

Bên A:

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

* A băm thông điệp M => 1 giá trị băm
* Giá trị băm dùng khoá private của A để mã hoá => chữ ký điện tử
* Đính kèm chữ ký điện tử với thông điệp M rồi gửi đến B.

Bên B:

* B gói tin của A gửi lấy ra thông điệp M và băm => giá trị băm (1)
* Lấy chữ ký mã hoá theo khoá public của A => giá trị băm (2)
* Nếu giá trị băm (1) bằng băm (2) thì chữ ký được xác thực và ngược lại.

## Câu 11: 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.

* **Các bước để mã hóa:**

B1: 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.

B2: Xác định 1 khoá là một số nằm trong khoảng từ 0 đến 25. Vẽ bảng thể hiện sự thay đổi của mỗi chữ cái.

B3: Chuyển đổi bản rõ thành bản mã.

* **Lấy ví dụ:**
* Bước 1: Xoá ký tự đặc biệt và chuyển đổi chữ hoa thành chữ thường

Giả sử chúng ta có chuỗi ban đầu: "Hello! How are you?"

Sau bước này, chuỗi sẽ trở thành: "hello how are you"

* Bước 2: Xác định khoá và vẽ bảng thể hiện sự thay đổi của mỗi chữ cái

Khoá được chọn là 5.

Bảng thể hiện sự thay đổi:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Plain | A | B | C | D | E | F | G | H | I | J | K | L | M |
| Cipher | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| Plain | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| Cipher | I | J | K | L | M | N | O | P | Q | R | S | T | U |

* Bước 3: Chuyển đổi bản rõ thành bản mã

Chuỗi "hello how are you" sau khi mã hóa sẽ trở thành "mjqqt mtb fwj dtz".

* Đây là kết quả sau khi áp dụng thuật toán Caesar Cipher với khoá là 5 vào chuỗi ban đầu.

## Câu 12: 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

- 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.

## Câu 13: 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.

**\* Các bước để mã hóa:**

- B1: 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.

- B2: Chọn khoá là một từ.

- B3: Tìm thứ tự của từng ký tự trong khoá tương ứng với bảng chữ cái.

- B4: Ánh xạ bản rõ với ma trận Vigenere Square và thể hiện theo thứ tự của khoá trong bảng chữ cái được bản mã.

\* **Lấy ví dụ:**

* **Bước 1: Xoá ký tự đặc biệt và chuyển đổi chữ hoa thành chữ thường**

Chuỗi ban đầu: "Hello World!"

Chuỗi sau khi xử lý: "helloworld"

* **Bước 2: Chọn khoá là một từ**

Khoá được chọn: "code"

* **Bước 3: Tìm thứ tự của từng ký tự trong khoá tương ứng với bảng chữ cái**

"c" -> 2

"o" -> 14

"d" -> 3

"e" -> 4

* **Bước 4: Ánh xạ bản rõ với ma trận Vigenere Square theo thứ tự của khoá trong bảng chữ cái để có bản mã**

Sử dụng ma trận Vigenere Square:

Ánh xạ chuỗi ban đầu với ma trận Vigenere Square theo thứ tự của từ khoá:

"h" -> "j"

"e" -> "i"

"l" -> "n"

"l" -> "n"

"o" -> "q"

"w" -> "y"

"o" -> "q"

"r" -> "t"

"l" -> "n"

"d" -> "f"

* Do đó, sau khi áp dụng thuật toán Vigenere Cipher với từ khoá "code" vào chuỗi "helloworld", ta thu được chuỗi mã hóa: "jinnyqntf"

## Câu 14: 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

**- Phương pháp mã hóa kinh điển** là 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.

## Câu 15: 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

## Câu 16: 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.

## Câu 17: Trình bày cách thiết lập một bộ khóa trong thuật toán RSA. Sử dụng bộ khóa đó để mã hóa và giải mã một kí 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

## Câu 18: Nêu hiểu biết về hàm băm 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 để tạo ra 1 giá trị băm.

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

- Hàm băm thì băm trên các thông điệp khác nhau thì rất khó tạo ra giá trị băm giống nhau.

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

**\* Sử dụng hàm băm trong đảm bảo tính toàn vẹn**

- A gửi thông điệp M cho B thì A sẽ băm M để được 1 giá trị băm sau đó gán cùng với thông điệp M gửi cho B

- Bên B nhận được thông điệp lấy ra được thông điệp M sau đó băm M và so sánh 2 giá trị băm, nếu trùng khớp thì dữ liệu được toàn vẹn.

## Câu 19: Nêu kiến trúc các mô hình cài đặt firewall?

**- Thứ nhất: Kiến trúc Dual-homed Host**

Được xây dựng dựa trên máy tính dual-home host. Một máy tính được gọi là dual-homed host nếu có ít nhất 2 network interface, có nghĩa là máy đó có gắn 2 card giao tiếp với 2 mạng khác nhau , do đó máy tính này đóng vai trò là router mềm, Kiến trúc dual-home host rất đơn giản, máy dual-home host ở giữa, một bên được nối với internet và bên còn lại nối với mạng nội bộ. Kiến trúc này có các đặc điểm sau:

+ Phải disable chức năng routing của dual-homed host để cấm hoàn toàn thông IP từ ngoài vào

+ Các hệ thống bên ngoài và bên trong dual-home host không liên lạc trực tiếp được với nhau.

+ Dual homed host cung cấp dịch vụ thông qua poxy server hoặc login trực tiếp vào dual –home host.

- **Thứ hai: Kiến trúc Screened Host**

Trong kiến trúc này chức năng bảo mật chính được cung cấp bởi chức năng packet filtering tại screening router. Packet filtering trên screening router được setup sao cho basition host là máy duy nhất trong internal network mà các host trên internet có thể mở kết nối đến Packet filtering cũng cho phép bastion host mở các kết nối ra bên ngoài. Các packet filtering thực hiện công việc sau:

+ Cho phép các interal hosts mở kết nối đến các host trên internet đối với một số dịch vụ được phép

+ Cấm tất cả kết nối từ các interal hosts

- **Thứ ba: Kiến trúc Screened Subnet Host**

Thêm một perimeter network để cô lập interal network với internet.Như vật dù hacker đã tấn công được vào bastion host vẫn còn một rào chắn nữa phải vượt qua interior router. Các lưu thông trong internet network được bảo vệ an toàn cho dù bastion đã bị “ chiếm”. Các dịch vụ nào ít tin cậy và có khả năng dễ bị tấn công thì nên để ở perimeter network.Bastion host là điệm liên lạc cho các kết nối từ ngoài vào :SMTP, FTP,DNS . Còn đối với viêc truy cập các dịch vụ từ internal client đến các server trên internet thì được điều khiển như sau:

Setup packet filtering trên các exterior và interior router để cho phép internal client truy cập các servers bên ngoài 1 cách trực tiếp.

Setup poxy server tren bastion host để cho phép internal clients truy cập các server bên ngoài 1 cách gián tiếp.

## ****Câu 20:**** Trình bày về SQL Injection 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/) 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.

## Câu 21: Trình bày về Cross-Site Scripting (XSS)? Nêu ví dụ và cách khắc phục

* **Cross-Site Scripting (XSS):**
* XSS đưa các tập lệnh vào máy chủ ứng dụng web để tấn công trực tiếp vào khách hàng không đề phòng.
* Các cuộc tấn công XSS xảy ra khi kẻ tấn công lợi dụng web các ứng dụng chấp nhận đầu vào của người dùng mà không xác thực nó và sau đó hiển thị lại cho người dùng.
* Một cuộc tấn công XSS yêu cầu một trang web đáp ứng hai tiêu chí: nó chấp nhận đầu vào của người dùng mà không xác thực nó và nó sử dụng đầu vào đó để phản hồi.
* **Ví dụ:**
* Một cuộc tấn công XSS điển hình là lợi dụng trang web của một blogger mà có chức năng hỏi ý kiến người dùng
* Khi một người dùng không đề phòng truy cập vào trang web của blogger và nhấp chuột vào nhận xét của kẻ tấn công, tập lệnh độc hại sẽ được tải xuống trình duyệt web của nạn nhân nơi nó được thực thi
* **Cách khắc phục:**
* **Chuẩn hóa và kiểm tra đầu vào:** Kiểm tra và xác thực dữ liệu người dùng để ngăn chúng thâm nhập mã độc hại.
* **Sử dụng hàm Escape:** Sử dụng hàm escape như htmlspecialchars để biến đổi dữ liệu người dùng thành HTML entities.
* **Content Security Policy (CSP):** Thiết lập CSP để hạn chế nguồn tài nguyên và scripts được tải trên trang web.
* **Input Validation:** Áp dụng kiểm tra đầu vào để chỉ cho phép dữ liệu hợp lệ.
* **Escape Output:** Trước khi hiển thị dữ liệu người dùng, đảm bảo rằng bạn đã thoát (escape) dữ liệu này.

## Câu 22: Trình bày về XML Injection? Nêu ví dụ và cách khắc phục

* **XML Injection:**
* Ngôn ngữ đánh dấu là một phương pháp thêm chú thích vào văn bản để phần bổ sung có thể được phân biệt với chính văn bản.
* XML là một ngôn ngữ đánh dấu.
* XML được thiết kế để mang dữ liệu thay vì chỉ định cách hiển thị nó.
* XML không có bộ thẻ được xác định trước; thay vào đó, người dùng tự định nghĩa thẻ riêng của họ.
* Cuộc tấn công XML Ịnjection tương tự như cuộc tấn công SQL Ịnjection.
* Kẻ tấn công phát hiện ra một trang web không lọc dữ liệu đầu vào người dùng có thể đưa các thẻ XML và dữ liệu vào cơ sở dữ liệu
* **Ví dụ:**
* Một hệ thống quản lý thông tin sản phẩm trong một cửa hàng trực tuyến sử dụng XML để lưu trữ thông tin sản phẩm, và người dùng có thể tìm kiếm sản phẩm theo tên hoặc ID.
* Người dùng có thể chèn mô tả độc hại vào tài liệu XML và làm thay đổi cách thông tin sản phẩm được hiển thị hoặc thậm chí là làm thay đổi cách hệ thống xử lý dữ liệu này.
* **Cách khắc phục:**
* **Xử lý cẩn thận dữ liệu người dùng:** Escape hoặc loại bỏ các ký tự đặc biệt trước khi sử dụng chúng trong tài liệu XML.
* **Kiểm tra và xác thực đầu vào:** Hạn chế và chỉ chấp nhận dữ liệu hợp lệ từ người dùng.
* **Sử dụng thư viện XML an toàn:** Sử dụng thư viện hoặc framework XML có tính bảo mật để xử lý dữ liệu XML.
* **Tối ưu hóa quyền truy cập:** Giới hạn quyền truy cập vào tài liệu XML để ngăn chặn tấn công từ bên ngoài.
* **Áp dụng CSP và giám sát:** Sử dụng CSP để kiểm soát tài nguyên và hệ thống giám sát để phát hiện các tấn công XML Injection.

## Câu 23: Trình bày về Directory Traversal/Command Injection? Nêu ví dụ về lỗi file upload và cách khắc phục

* **Directory Traversal:**
* Đây là loại tấn công mà kẻ tấn công cố gắng truy cập và điều hướng qua các thư mục mà họ không được phép thông qua việc sử dụng các ký tự đặc biệt hoặc các phép toán thư mục để vượt qua các rào cản bảo mật của hệ thống.
* Kỹ thuật này thường sử dụng '../' để di chuyển lên thư mục cha hoặc các ký tự tương tự.
* **Command Injection:**
* Đây là loại tấn công mà kẻ tấn công chèn các lệnh thực thi vào trong các câu lệnh hoặc truy vấn có thể thực thi trên máy chủ.
* Thông thường, điều này xảy ra khi ứng dụng không kiểm tra hoặc escape cẩn thận dữ liệu đầu vào từ người dùng trước khi sử dụng nó trong các lệnh thực thi.
* **Ví dụ về lỗi file upload:**
* Giả sử có một ứng dụng web cho phép người dùng tải lên tệp hình ảnh và lưu trữ chúng trong thư mục ‘uploads’ và truy vấn đến những hình ảnh đó bằng đường dẫn trên trình duyệt. VD: https://example.com/upload/pic1
* Nếu ứng dụng không kiểm tra cẩn thận dữ liệu tên tệp tải lên, người dùng có thể tải lên một tệp với tên như ‘../../config/database.txt’.
* Nếu không có sự kiểm tra và ngăn chặn, điều này có thể dẫn đến Directory Traversal, cho phép người dùng truy cập vào các thư mục nhạy cảm như config và đọc thông tin nhạy cảm như cấu hình cơ sở dữ liệu.
* **Cách khắc phục:**
* **Kiểm tra tên tệp tải lên:** Đảm bảo rằng tên tệp tải lên không chứa các ký tự đặc biệt hoặc chuỗi có thể dẫn đến Directory Traversal.
* **Hạn chế quyền truy cập:** Xác định và hạn chế quyền truy cập vào các thư mục nhạy cảm bằng cách sử dụng cơ chế phân quyền.
* **Chuyển đổi hoặc loại bỏ dữ liệu người dùng:** Đối với Directory Traversal, chuyển đổi hoặc loại bỏ các ký tự đặc biệt khỏi dữ liệu nhập người dùng trước khi sử dụng.
* **Sử dụng các hàm an toàn:** Sử dụng các hàm có sẵn trong ngôn ngữ lập trình để thực hiện các hoạt động thư mục hoặc file mà không cần sử dụng các chuỗi nguyên thủy nhập từ người dùng.

## Câu 24: Trình bày một số hình thức tấn công phía client: Header manipulation, Cookies

* **Header manipulation:**
* Tiêu đề HTTP bao gồm các trường chứa thông tin về đặc điểm của dữ liệu được truyền
* Kẻ tấn công có thể sửa đổi các tiêu đề HTTP để tạo ra một cuộc tấn công bằng cách sử dụng **header manipulation**.
* **Header manipulation** không phải là một cuộc tấn công thực sự mà là phương tiện mà qua đó các cuộc tấn công khác, chẳng hạn như XSS, có thể được thực hiện
* **Cookies:**
* Một cookie có thể chứa nhiều thông tin khác nhau dựa trên sở thích của người dùng khi truy cập một trang web.
* Tồn tại một số loại cookie khác nhau: Cookie của bên thứ nhất, cookie của bên thứ ba, cookie phiên.
* Cookie của bên thứ nhất có thể bị đánh cắp và sử dụng để mạo danh người dùng.
* Cookie của bên thứ ba có thể được sử dụng để theo dõi thói quen sử dụng trình duyệt hoặc mua bán của người dùng.

## Câu 25: Trình bày hình thức tấn công phía client: Attachments, Session Hijacking

* **Attachments:**
* Tệp đính kèm là các tệp được ghép nối với thư email.
* Các tệp đính kèm độc hại thường được sử dụng để phát tán vi-rút, Trojan, và phần mềm độc hại khác khi chúng được mở.
* **Session Hijacking:**
* **Session Hijacking** (Chiếm quyền điều khiển phiên) là một cuộc tấn công trong đó kẻ tấn công cố gắng mạo danh người dùng bằng cách sử dụng session token của người dùng ấy.

## Câu 26: Trình bày về DoS và các loại tấn công DoS

* **Denial of Service (DoS):**
* Tấn công DoS là một nỗ lực có chủ ý nhằm ngăn chặn người dùng chính chủ truy cập vào một hệ thống bằng cách áp đảo hệ thống đó bằng các yêu cầu.
* Hầu hết các cuộc tấn công DoS ngày nay thực chất là từ chối dịch vụ phân tán (DDoS): thay vì sử dụng một máy tính, DDoS có thể sử dụng hàng trăm hoặc hàng nghìn máy tính zombie trong một mạng botnet để tấn công một thiết bị bằng các yêu cầu.
* **Các loại tấn công DoS:**
* **Ping flood:** Nhiều máy tính nhanh chóng gửi một lượng lớn yêu cầu ICMP, áp đảo máy chủ (cũng như mạng) đến mức nó không thể phản hồi đủ nhanh và sẽ hủy các kết nối hợp pháp tới các máy khách khác và từ chối mọi kết nối mới.
* **Smuft attack:**
  + Kẻ tấn công phát tán yêu cầu ping tới tất cả các máy tính trên mạng máy tính nhưng thay đổi địa chỉ nhận mà từ đó yêu cầu được gửi đến máy tính của nạn nhân.
  + Sau đó, mỗi máy tính sẽ gửi phản hồi tới địa chỉ của nạn nhân khiến máy tính nhanh chóng bị quá tải và sau đó gặp sự cố hoặc không thể truy cập được đối với người dùng hợp lệ.
* **SYN flood:** Kẻ tấn công gửi rất nhiều yêu cầu kết nối đến máy chủ, nhưng không hoàn tất quá trình kết nối. Khi máy chủ cố gắng xác định kết nối cho mỗi yêu cầu này mà không nhận được phản hồi, nó sẽ tiêu tốn tài nguyên và có thể trở nên không thể phục vụ các yêu cầu hợp lệ từ người dùng.

## Câu 27: Trình bày về Interception, Poisoning và các hình thức tấn công chiếm quyền truy cập (Access Rights)

* **Interception:**
* Những kẻ tấn công tạo một bản sao của thông điệp được chuyển trước khi gửi nó đến người nhận. Sau đó, kẻ tấn công có thể gửi tin nhắn gốc đến máy chủ và máy chủ có thể phản hồi. Bây giờ một mối quan hệ đáng tin cậy đã được thiết lập giữa kẻ tấn công và máy chủ.
* Kẻ tấn công có thể bắt đầu thay đổi nội dung của thông điệp và mã lệnh đã sao chép được. Nếu anh ta cuối cùng cũng thực hiện sửa đổi một cách chính xác, máy chủ sẽ phản hồi, cho kẻ tấn công biết anh ta đã thành công.
* **Poisoning:**
* Ngộ độc ARP: Kẻ tấn công có thể sửa đổi địa chỉ MAC trong bộ đệm ARP để địa chỉ IP tương ứng trỏ đến một máy tính khác.
* Ngộ độc DNS là quá trình thay thế địa chỉ DNS để máy tính tự động chuyển hướng sang thiết bị khác
* **Các hình thức tấn công chiếm quyền truy cập:**
* Privilege Escalation (Nâng cấp đặc quyền): khai thác lỗ hổng trong phần mềm để có quyền truy cập vào các tài nguyên mà người dùng thường bị hạn chế truy cập
* Transitive Access (Truy cập bắc cầu): sử dụng mối quan hệ tin cậy giữa ba thành phần (A->B->C) để có được quyền truy cập.

Phần II: Bài tập

## 1. Code thuật toán AES class SymmetricCrypt

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

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

//khóa

**private** SecretKey secretKey;

//khóa theo chuỗi cho trước

**private** SecretKeySpec skeySpec;

//bản sinh khóa theo thuật toán đã chọn

**private** KeyGenerator keyGen;

//bản mã

**private** Cipher cipher;

### 1.1 Sinh khóa đối xứng

**public** SymmetricCrypt()

**throws** NoSuchAlgorithmException

{

keyGen = KeyGenerator.*getInstance*(Algorithm);

secretKey = keyGen.generateKey();

}

### 1.2 Táo khóa đối xứng từ một chuỗi cho trước

**public** SymmetricCrypt(String key)

**throws** NoSuchAlgorithmException

{

skeySpec = **new** SecretKeySpec(key.getBytes(), "AES");

}

**public** SecretKey getSecretKey() {

**return** secretKey;

}

**public** SecretKey getSecretKeySpec() {

**return** skeySpec;

}

### 1.3 Mã hóa thông điệp có 2 tham số: tham số 1 là bản mã, thám số thứ 2 la khóa

### bí mật được tạo ở 1.1

**public** String encryptText(String msg, SecretKey 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")));

}

### 1.4 Mã hóa thông điệp có 2 tham số: tham số 1 là bản mã, thám số thứ 2 la khóa

### bí mật 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, thám số thứ 2 la 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, thám 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 SymetricCrypt 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 {

SymmetricCrypt SC = **new** SymmetricCrypt();

String msg = "Nguyễn Văn Hiền";

String encrypted\_msg =

SC.encryptText(msg, SC.getSecretKey());

System.***out***.println("Plain text: " + msg);

System.***out***.println("Encrypted text: " + encrypted\_msg);

String decrypted\_msg =

SC.decryptText(encrypted\_msg, SC.getSecretKey());

System.***out***.println("Decrypted text: " + decrypted\_msg);

SymmetricCrypt SC2 = **new** SymmetricCrypt("stackjava.com.if");

String msg2 = "Nguyễn Văn Hiền";

String encrypted\_msg2 =

SC.encryptText(msg2, SC2.getSecretKeySpec());

System.***out***.println("Plain text: " + msg2);

System.***out***.println("Encrypted text: " + encrypted\_msg2);

String decrypted\_msg2 =

SC.decryptText(encrypted\_msg2, SC2.getSecretKeySpec());

System.***out***.println("Decrypted text: " + decrypted\_msg2);

}

}

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

public class Digest {

### 2.1. MD5

public String byteArrayToHexString(byte[] array) {

StringBuilder hexString = new StringBuilder();

for (byte b : array) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) {

hexString.append('0');

}

hexString.append(hex);

}

return hexString.toString();

}

public String md5Digest(String msg){

try{

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

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

byte[] digest = md.digest();

return byteArrayToHexString(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());

byte[] digest = md.digest();

return byteArrayToHexString(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 World";

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 = null;

try {

sha256Hash = digest.getHashSHA256(msg);

} catch (Exception e) {

e.printStackTrace();

}

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

}

## 3. Sử dụng (JCA), viết class RSAKeyGen mã hóa, giải mã sử dụng RSA:

public class RSAKeyGen {

private KeyPairGenerator keyGen;

private KeyPair keypair;

private PublicKey publicKey;

private PrivateKey privateKey;

public PublicKey getPublicKey() { return publicKey; }

public void setPublicKey(PublicKey publicKey) { this.publicKey = publicKey; }

public PrivateKey getPrivateKey() { return privateKey; }

public void setPrivateKey(PrivateKey privateKey) {this.privateKey = privateKey;}

### 3.1 Sinh bộ khóa theo thuật toán RSA

public RSAKeyGen(int keyLength){

try{

keyGen = KeyPairGenerator.getInstance("RSA");

keyGen.initialize(keyLength);

keypair = keyGen.generateKeyPair();

setPublicKey(keypair.getPublic());

setPrivateKey(keypair.getPrivate());

}catch (Exception ex){

ex.printStackTrace();

}

}

### 3.2 Lưu bộ khóa ra file

public void writeToFile(String path, byte[] key){

try{

File f = new File(path);

f.getParentFile().mkdir();

FileOutputStream fos = new FileOutputStream(f);

fos.write(key);

fos.flush();

fos.close();

}catch (Exception ex){

ex.printStackTrace();

}

}

}

**\*Viết class RSACryp để sử dụng bộ khóa đã lưu trong file để mã hóa và giải mã theo RSA**

public class RSACryp {

private Cipher cipher;

// Phương thức để ghi dữ liệu vào file

public void writeToFile(String filePath, byte[] data) throws IOException {

File file = new File(filePath);

try (FileOutputStream fos = new FileOutputStream(file)) {

fos.write(data);

fos.flush();

}

}

// Phương thức để đọc dữ liệu từ file

public byte[] readFile(String filePath) throws IOException {

return Files.readAllBytes(new File(filePath).toPath());

}

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

// Chuyển dữ liệu từ file thành PrivateKey

public PrivateKey getPrivateKey(String filePath) {

try {

byte[] keyBytes = Files.readAllBytes(new File(filePath).toPath());

PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keyBytes);

KeyFactory kf = KeyFactory.getInstance("RSA");

return kf.generatePrivate(spec);

} catch (Exception ex) {

ex.printStackTrace();

}

return null;

}

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

A computer screen shot of a code

Description automatically generated

### 3.5 Hàm mã hóa 1 thông điệp cho trc = khóa bm, tham số truyền vào gồm 1 là thông điệp cần mã hóa, 2 là khóa bm

public String encrypt(String msg, PrivateKey key){

try{

cipher = Cipher.getInstance("RSA");

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

return Base64.getEncoder().encodeToString(cipher.doFinal(msg.getBytes("UTF-8")));

}catch (Exception ex){

ex.printStackTrace();

}

return null;

}

### 3.6 Hàm giải mã thông điệp = khóa công khai, tham số truyền vào gồm 1 là bản mã cần giải mã, 2 là khóa công khai.

public String decrypt(String msg, PublicKey publicKey){

try{

cipher = Cipher.getInstance("RSA");

cipher.init(Cipher.DECRYPT\_MODE, publicKey);

return new String(cipher.doFinal(Base64.getDecoder().decode(msg)), "UTF-8");

}catch (Exception ex){

ex.printStackTrace();

}

return null;

}

### 3.7 Sử dụng lớp RSACryp để mã hóa 1 chuỗi và băm chuỗi theo 1 thuật toán băm cho trước rồi lưu cả bản mã và bản băm vào 1 file

public static void main(String[] args) throws IOException {

RSACryp rsaCryp = new RSACryp();

try {

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

RSAKeyGen rsaKeyGen = new RSAKeyGen(2048);

rsaKeyGen.writeToFile("D:\\publicKey.txt", rsaKeyGen.getPublicKey().getEncoded());

rsaKeyGen.writeToFile("D:\\privateKey.txt", rsaKeyGen.getPrivateKey().getEncoded());

String msg = "Tran Huy Hiep";

String encrypted\_msg = rsaCryp.encrypt(msg, rsaCryp.getPrivateKey("D:/privateKey.txt"));

byte[] firstHash = digest.digest(msg.getBytes("UTF-8"));

String hashCiper = encrypted\_msg + ";" Base64.getEncoder().encodeToString(firstHash);

// Chuyển đổi hashCipher thành mảng byte

byte[] hashCipherBytes = hashCiper.getBytes("UTF-8");

// Ghi mảng byte này vào file

rsaCryp.writeToFile("D:/result.txt", hashCipherBytes);

### 3.8 Đọc file chứa bản mã và bản băm của 1 thông điệp, sd lớp RSACryp để giải mã ra thông điệp gốc và băm lại = thuật toán băm cho trước. Sau đó đem SS 2 bản băm để xác nhận tính toàn vẹn.

byte[] dataRead = rsaCryp.readFile("D:\\result.txt");

// Chuyển đổi mảng byte trở lại thành chuỗi để thực hiện các thao tác khác

String dataString = new String(dataRead, "UTF-8");

String result[] = dataString.split(";");

String decrypted\_msg = rsaCryp.decrypt(result[0], rsaCryp.getPublicKey("D:/publicKey.txt"));

byte[] hashAgain = digest.digest(decrypted\_msg.getBytes("UTF-8"));

String hashAgainString = Base64.getEncoder().encodeToString(hashAgain);

System.out.println("Plain Text:" + msg);

System.out.println("Encrypt Text: " + encrypted\_msg);

System.out.println("MD5 hash: " + new String(firstHash));

System.out.println("MD5 hash: " + result[1]);

System.out.println("Decrypt Text: " + decrypted\_msg);

System.out.println("Hash Again: " + hashAgainString);

if(hashAgainString.equals(result[1])){

System.out.println("Same");

} else {

System.out.println("Not same");

}

} catch (NoSuchAlgorithmException | UnsupportedEncodingException e) {

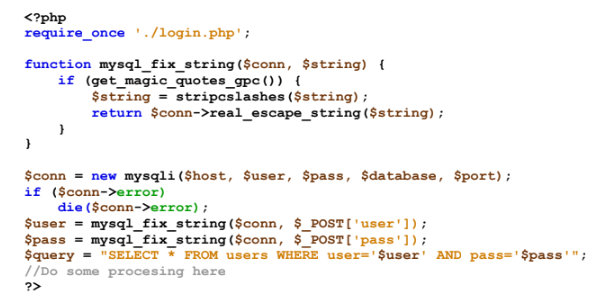
e.printStackTrace();

}

}

## 4. Lập trình mô tả lỗi SQL Injection và 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.

<?php

$con = mysqli\_connect(“localhost”,”root”,””,”sql\_demo”);

If(isset($\_POST[‘username’]) && isset($\_POST[‘password])){

$user = $\_POST[‘username’];

$pass = $\_POST[‘password];

$query = “Select \* from users Where username =? And password = ?”;

$result = $con->prepare($query);

$result->bind\_param(‘ss’,$user,$pass);

$result->execute();

If($row = mysqli\_fetch\_assoc($result)){ Echo “Login success”;}

Else{ echo “Login failed”; }

}

?>

**Bài 3 : Chương trình SSH App-script : (Base System 1: ch11, Base System 2: ch12)**

**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 "$2"

// $ ( số thứ tự được split bởi dấu cách , mình cần lấy đường dẫn **.passwd** được in đậm ) ( đếm từ 0 )

// Lưu ý : $1 nếu system("ls **/challenge/app-script/ch11/.passwd**");

// Lưu ý : $2 nếu system("ls -lA **/challenge/app-script/ch11/.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

//Lưu ý : ch11 là tên chương trình ( ví dụ : system("ls /challenge/app-script/**ch11**/.passwd") )

**B7**: Chạy chương trình **ch11** để hiển thị mật khẩu bằng lệnh

**./ch11**

//Lưu ý: ch11 là tên chương trình ( ví dụ : system("ls /challenge/app-script/**ch11**/.passwd") )

## 5. Mô tả một số ví dụ khai thác lỗi SQL Injection trên các bài CTF của Root-me.org:

**- Input như thế nào để khai thác?**

**- Output khi đó ra sao? => đưa ra nhận định để khai thác bước tiếp theo.**

**Input để khai thác:**

Bài toán yêu cầu đăng nhập vào hệ thống bằng tài khoản và mật khẩu. Truy cập vào trang web, bạn thấy một trường nhập tên đăng nhập và mật khẩu.

**Input để tìm lỗ hổng:**

Thử nhập các giá trị không hợp lệ hoặc các chuỗi như ‘ ' OR 1=1 – ’ vào trường tên đăng nhập hoặc mật khẩu để kiểm tra xem có chấp nhận không.

**Output khi khai thác thành công:**

Nếu hệ thống không được bảo vệ cẩn thận, kỹ thuật nhập ‘ OR 1=1 ’ có thể làm cho truy vấn SQL trở thành luôn đúng (‘TRUE’). Kết quả sẽ cho phép bạn đăng nhập vào hệ thống mà không cần cung cấp thông tin chính xác của tài khoản.

**Nhận định và bước tiếp theo:**

Khi bạn thấy rằng bạn đã có thể đăng nhập vào hệ thống bằng cách sử dụng SQL Injection, bước tiếp theo có thể là khai thác thông tin từ cơ sở dữ liệu, thực hiện các truy vấn, thay đổi dữ liệu hoặc thậm chí kiểm soát hệ thống.

Để tiếp tục khai thác, bạn có thể sử dụng các câu lệnh SQL phức tạp hơn như UNION, SELECT, INSERT, DELETE, hoặc các kỹ thuật khác để thu thập thông tin, thực hiện các hành động không được phép hoặc khai thác lỗ hổng một cách sâu hơn.