BONUS PROJECT

BUILD WEEK 1
CS0424IT

SECURE
SENTINELS

Table Of Contents

SQL Injection

Definizione

Vulnerabilità di sicurezza che consente a un attaccante di interferire con le query SQL che un'applicazione esegue sul database.

Funzionamento

L'attaccante inserisce codice SQL malevolo nei campi di input dell'applicazione, che viene poi eseguito dal database. Questo può portare all'accesso non autorizzato ai dati, alla modifica o cancellazione dei dati, e in alcuni casi, al controllo completo del server di database.

Query sicura

```
SELECT * FROM users WHERE id = 1;
```

Query iniettata

```
SELECT * FROM users WHERE id = 1 OR 1=1;
```

Come mitigare SQL Injection

Query preparate e dichiarazioni precompilate

Query preparate separano la logica SQL dai dati forniti dall'utente. I parametri degli utenti sono trattati come dati, non come codice SQL.

Ulteriori misure di sicurezza

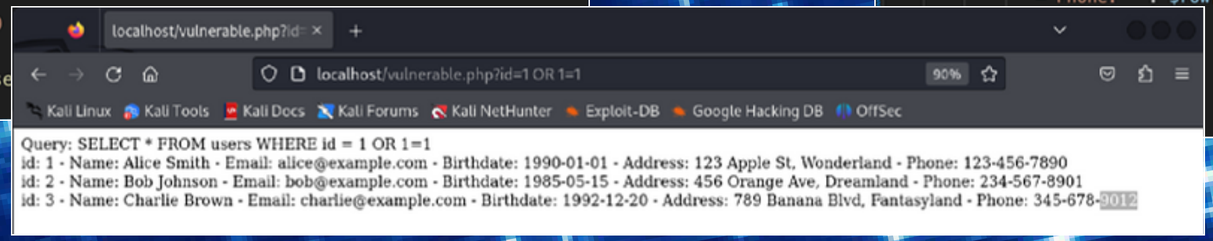
- Principio del Minimo Privilegio.
- Utilizzo di ORM (Object-Relational Mapping) come Hibernate (Java) o SQLAlchemy (Python).
- Monitoraggio e logging delle query per rilevare e rispondere rapidamente a comportamenti sospetti.

Escaping/validazione input

- L'escaping degli input utente assicura che i caratteri speciali vengano trattati come dati e non come parte del comando SQL.
- Minor sicurezza rispetto alle query preparate e dovrebbe essere usato solo come ulteriore misura di sicurezza.
- Validare e filtrare gli input degli utenti per assicurarsi che siano nel formato corretto (es. numeri interi per ID).

Query vulnerabile

```php
<?php
$servername = "localhost";
$username = "root";
$password = "your_mysql_root_password";
$dbname = "secretdb";

// Creazione connessione
$conn = new mysqli($servername, $username, $password, $dbname);

// Controllo connessione
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Ottenere l'input dell'utente
$user_input = $_GET['id'];

// Query SQL vulnerabile
$sql = "SELECT * FROM users WHERE id = " . $user_input;
echo "Query: $sql<br>";  // Aggiunto per debug
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Output dei dati
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"].
             " - Name: " . $row["name"].
             " - Email: " . $row["email"].
             " - Birthdate: " . $row["birthdate"].
             " - Address: " . $row["address"].
             " - Phone: " . $row["phone"].
             "<br>";
    }
} else {
    echo "0
}
$conn->close
?>
```



Query preparata

```php
<?php
$servername = "localhost";
$username = "root";
$password = "your_mysql_root_password";
$dbname = "secretdb";

// Creazione connessione
$conn = new mysqli($servername, $username, $password, $dbname);

// Controllo connessione
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Ottenere l'input dell'utente
$user_input = $_GET['id'];

// Preparazione della query SQL con un segnaposto (?) per il parametro id.
$stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");

// Il metodo bind_param associa il parametro $user_input alla query.
// Il tipo di dato i indica che il parametro è un intero.
$stmt->bind_param("i", $user_input);

// Esecuzione della query
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // Output dei dati
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"].
             " - Name: " . $row["name"].
             " - Email: " . $row["email"].
             " - Birthdate: " . $row["birthdate"].
             " - Address: " . $row["address"].
             " - Phone: " . $row["phone"].
```

Steganografia

Definizione

La scienza di nascondere informazioni all’interno di altri dati in modo tale che la presenza delle informazioni nascoste non sia rilevabile a occhio nudo o attraverso mezzi comuni di analisi.

Funzionamento

Esistono vari metodi per implementare la steganografia, a seconda del tipo di file utilizzato come contenitore.

Steganografia

Nasconde l’esistenza dei dati. Anche se i dati nascosti sono scoperti, potrebbero non essere riconosciuti come tali.

Crittografia

Nasconde il contenuto dei dati cifrandoli, in modo che anche se intercettati, non possano essere compresi senza la chiave di decrittazione.

Metodi steganografici

FILE IMMAGINE

- Least Significant Bit (LSB)
- Mascheramento e Filtro
- Trasformata Discreta del Coseno (DCT)

FILE AUDIO

- LSB nel dominio del tempo
- Mascheramento Echo
- Tecniche di frequenza

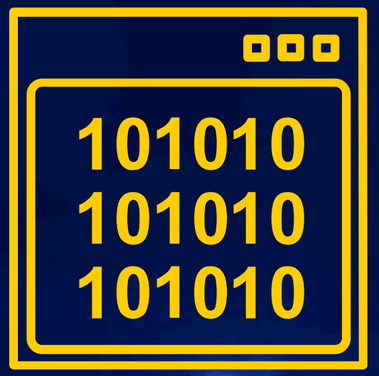
FILE VIDEO

- LSB nei frame video
- Iniezione di frame inutilizzati

FILE DI TESTO

- Uso di spazi invisibili
- Sinonimi e parafrasi

Least Significant Bit



Definizione

- Tecnica di steganografia utilizzata per nascondere informazioni all’interno dei bit meno significativi dei pixel di un’immagine digitale.
- Ogni pixel è rappresentato da tre componenti di colore (rosso, verde, blu), ciascuno con un valore compreso tra 0 e 255 (8 bit).
- Modificando il bit meno significativo di ciascun componente, è possibile nascondere dati senza alterare visibilmente l’immagine.

Implementazione

- Conversione del messaggio in bit
- Modifica dei pixel
- Creazione dell’immagine stenografata

Encoding

```python
from PIL import Image


# Funzione per convertire un messaggio di testo in una stringa di bit
def message_to_bits(message):
    bits = []
    for char in message:
        bits.append(
            bin(ord(char))[2:].zfill(8)
        )  # Converte ogni carattere in una stringa di 8 bit
    return "".join(bits)  # Unisce tutte le stringhe di bit in una singola stringa


# Funzione per incorporare un messaggio nascosto in un'immagine
def embed_message(image_path, output_path, message):
    image = Image.open(image_path)  # Carica l'immagine
    encoded_image = image.copy()  # Crea una copia dell'immagine
    width, height = image.size  # Ottiene la dimensione dell'immagine
    pixels = list(image.getdata())  # Ottiene i dati dei pixel dell'immagine

    message_bits = message_to_bits(message)  # Converte il messaggio in bit
    message_bits += (
        "0" * 8
    )  # Aggiunge un carattere nullo per indicare la fine del messaggio
    bit_index = 0  # Indice per tenere traccia della posizione nel messaggio di bit

    for i in range(width * height):
        if bit_index < len(message_bits):
            pixel = list(
                pixels[i]
            )  # Converte il pixel in una lista per modificare i valori RGB
            for j in range(3):  # Modifica i valori R, G, B
                if bit_index < len(message_bits):
                    pixel[j] = pixel[j] & ~1 | int(
                        message_bits[bit_index]
                    )  # Modifica il bit meno significativo
                    bit_index += 1  # Passa al bit successivo del messaggio
            pixels[i] = tuple(
                pixel
            )  # Riporta il pixel modificato nella lista dei pixel
        else:
            break

    encoded_image.putdata(pixels)  # Aggiorna i dati dei pixel dell'immagine
    encoded_image.save(output_path)  # Salva l'immagine modificata


# Richiede all'utente di inserire il percorso dell'immagine e il messaggio
input_image_path = input("Inserisci il percorso dell'immagine di input: ")
output_image_path = input("Inserisci il percorso dell'immagine di output: ")
message = input("Inserisci il messaggio da nascondere: ")

# Esegue la funzione di incorporamento
embed_message(input_image_path, output_image_path, message)
```

Decoding

```python
from PIL import Image


# Funzione per convertire una stringa di bit in un messaggio di testo
def bits_to_message(bits):
    message = ""
    for i in range(0, len(bits), 8):
        byte = bits[i : i + 8]  # Prende 8 bit alla volta
        if byte == "00000000":  # Il carattere nullo indica la fine del messaggio
            break
        message += chr(int(byte, 2))  # Converte la stringa di bit in un carattere
    return message


# Funzione per estrarre un messaggio nascosto da un'immagine
def extract_message(image_path):
    image = Image.open(image_path)  # Carica l'immagine
    pixels = list(image.getdata())  # Ottiene i dati dei pixel dell'immagine
    width, height = image.size  # Ottiene la dimensione dell'immagine

    bits = ""
    for i in range(width * height):
        pixel = pixels[i]
        for j in range(3):  # Considera i valori R, G, B
            bits += str(
                pixel[j] & 1
            )  # Estrae il bit meno significativo e lo aggiunge alla stringa di bit

    return bits_to_message(bits)  # Converte la stringa di bit in un messaggio di testo


# Richiede all'utente di inserire il percorso dell'immagine
image_path = input("Inserisci il percorso dell'immagine da cui estrarre il messaggio: ")

# Estrae il messaggio nascosto dall'immagine
hidden_message = extract_message(image_path)
print("Messaggio nascosto:", hidden_message)
```

Linguaggi esoterici

Linguaggi di programmazione esoterici sfidano le convenzioni tradizionali e spingono la creatività dei programmatori al limite.

Brainfuck

```
++++++++++
[>+++++++>++++++++++>+++>+
<<<<-]>++.>+.+++++++..+++.>++.
<<+++++++++++++++.>.+++.------.-------
-.>+.>.
```

Cow

```
MoO moO MoO mOo MOO OOM MMM
moO moO MMM mOo mOo moO MMM
mOo MMM moO moO MOO MOo mOo
MoO moO moo mOo mOo moo
```

Ook!

```
Ook. Ook? Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook! Ook?
Ook? Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook? Ook! Ook!
Ook? Ook!
```

Whitespace

Intercal

```
DO ,1 <- #13PLEASE DO ,1 SUB #1 <-
#238DO ,1 SUB #2 <- #108DO ,1 SUB #3
<- #112DO ,1 SUB #4 <- #0DO ,1 SUB #5
<- #64DO ,1 SUB #6 <- #194DO ,1 SUB
#7 <- #48PLEASE DO ,1 SUB #8 <-
#22DO ,1 SUB #9 <- #248DO ,1 SUB #10
<- #168DO ,1 SUB #11 <- #24DO ,1 SUB
#12 <- #16DO ,1 SUB #13 <- #162PLEASE
READ OUT ,1PLEASE GIVE UP
```

Our Team

Simone La Porta

Team Leader

Nicolò Callegaro

Simone Esposito

Grazia Coco

Gianluca Sansone

Alejandro Cristino

Alessio Forli