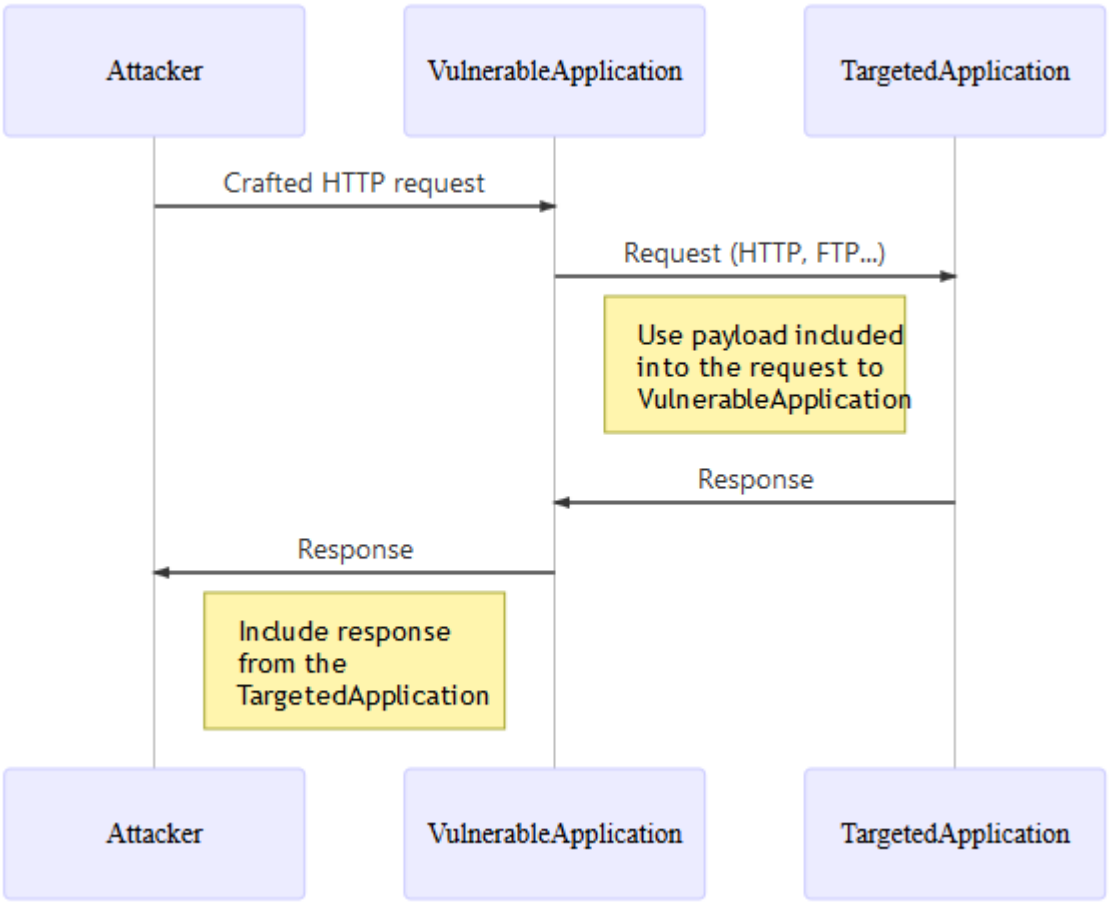
Server-side Request Forgery

Overview

SSRF là 1 lỗ hổng bảo mật web mà cho phép kẻ tấn công khiến ứng dụng bên phía máy chủ thực hiện các yêu cầu Http đến 1 miền tuỳ ý mà kẻ tấn công lựa chọn.
Trong 1 cuộc tấn công thì kẻ tấn công có thể khiến máy chủ tạo kết nối đến các dịch vụ dành riêng cho cơ sở hạ tầng của tổ chức. Trường hợp khác thì chúng có thể kết nối với dữ liệu bên ngoài gây ra rò rỉ các dữ liệu nhạy cảm.



Impact

Khi 1 cuộc tấn công SSRF thành công thì kẻ tấn công có thể thu thập thông tin về các port, ip address, thực thi mã từ xa (RCE), truy cập vào các data hay đọc các file được lưu chữ trên máy chủ.

Reason

Thông thường SSRF xảy ra khi một ứng dụng Web đang thực thi 1 request, trong khi đó kẻ tấn công có thể toàn quyền kiểm soát một phần request đang được gửi đi.
Ví dụ phổ biến nhất đó là khi mà kẻ tấn công có thể kiểm soát tất cả hay 1 phần của URL mà ứng dụng web đưa ra yêu cầu với 1 số các dịch vụ của bên thứ 3.
How to determine
Nhiều lỗ hổng SSRF có thể tương đối dễ phát hiện vì lưu lượng truy cập thoogn thường của ứng dụng liện quan đến các tham số yêu cầu chứa đầy đủ các URL nhưng có 1 số sẽ khó xác định hơn:

+ URLs in requests: Có 1 số các web có thể chỉ đặt 1 phần của URL hay 1 phần hostname vào các request parameter. Các giá trị này sẽ kết hợp với dũ liệu trên máy chủ để tạo thành 1 URL hoàn chỉnh. Nếu các giá trị này dễ bị nhận thấy thì ta có thể nhận định ra lỗ hổng 1 cách dễ dàng. Tuy nhiên khả năng khai thác lỗ hổng ở dạng này có thể bị hạn chế vì ta không thể kiểm soát hoàn toàn được các URL request.
+ URLs with data form: 1 số các ứng dụng truyền dữ liệu cho phép truyền dũ liệu mà trong đấy được phép cả các URL. Ví dụ rõ dàng nhất đó là XML, ta có thể lợi dụng lỗ hổng XXE để thực hiện việc tấn công SSRF.
+ SSRF via the Referer header: 1 số web dùng phần mềm phân tích ở phái máy chủ để theo dõi lượng khách truy nhập. Những software thường ghi lại các Referer header trong các request. Thường thì các phần mềm này sẽ truy cập vào các URL được ghi vào Referer header. Điều này tạo ra cách xác định được các lỗ hổng SSRF 1 cách dễ dàng.

What type of SSRF

1. SSRF attacks against the server itself:

   Trong cuộc tấn công này thì kẻ tấn công thực hiện một yêu cầu Http request quay lại chính máy chủ đang lưu trữ web. Điều này nhằm việc truy cập vào các side nhưng tài khoàn không có đủ quyền truy cập vào chúng.

   VD:

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://localhost/admin
```

2. SSRF attacks against other back-end systems:

   Một loại mối quan hệ tin cậy khác thường phát sinh với SSRF là nơi máy chủ web có thể tương tác với các hệ thống back-end khác mà người dùng không thể truy cập trực tiếp. Các hệ thống này thường có địa chỉ IP riêng không định tuyến được. Vì các hệ thống back-end thường được bảo vệ bởi cấu trúc liên kết mạng, chúng thường có vị trí bảo mật yếu hơn. Trong nhiều trường hợp, hệ thống back-end nội bộ chứa chức năng nhạy cảm có thể được truy cập mà không cần xác thực bởi bất kỳ ai có thể tương tác với hệ thống.

   VD:

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://192.168.0.68/admin
```

3. Circumventing common SSRF defenses

   Ngày nay có 1 số web chứa SSRF đi kèm với các biện pháp bảo vệ nhằm chống lại việc khai thác chúng. Thông thường các biện pháp này có thể bị phá vỡ.

   3.1. SSRF with blacklist-based input filters

   1 số web chặn đầu vào chứa là hostname như: *127.0.0.1, localhot* hay các URL nhạy cảm như */admin*. Ta có thể dùng các bộ lọc khác nhau để phá vỡ điều này.

   - Sử dụng một đại diện IP thay thế `127.0.0.1`, chẳng hạn như `2130706433`, `017700000001`hoặc `127.1`.
   - Đăng ký tên miền của riêng bạn phù hợp với `127.0.0.1`. Bạn có thể sử dụng `spoofed.burpcollaborator.net`cho mục đích này.
   - Làm xáo trộn các chuỗi bị chặn bằng cách sử dụng mã hóa URL hoặc biến thể chữ hoa chữ thường.

   3.2. SSRF with whitelist-based input filters

   Một số web chỉ cho phép đầu vào phù hợp, bắt đầu bằng hoặc chứa whitelist các giá trị được phép. Trong trường hợp này, đôi khi bạn có thể phá vỡ bộ lọc bằng cách khai thác sự mâu thuẫn trong phân tích cú pháp

   - Bạn có thể nhúng thông tin đăng nhập vào URL trước tên máy chủ, bằng cách sử dụng `@`ký tự. Ví dụ: `https://expected-host@evil-host`.
   - Bạn có thể sử dụng `#`ký tự để chỉ ra một phân đoạn URL. Ví dụ: `https://evil-host#expected-host`.
   - Bạn có thể tận dụng hệ thống phân cấp đặt tên DNS để đặt đầu vào bắt buộc vào tên DNS đủ điều kiện mà bạn kiểm soát. Ví dụ: `https://expected-host.evil-host`.
   - Bạn có thể ký tự mã hóa URL để gây nhầm lẫn với mã phân tích cú pháp URL. Điều này đặc biệt hữu ích nếu mã triển khai bộ lọc xử lý các ký tự được mã hóa URL khác với mã thực hiện yêu cầu HTTP phía sau.
   - Bạn có thể sử dụng kết hợp các kỹ thuật này với nhau.

3.3. Bypassing SSRF filters via open redirection

Đôi khi có thể phá vỡ bất kỳ loại phòng thủ dựa trên bộ lọc nào bằng cách khai thác lỗ hổng chuyển hướng mở.

VD:

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://weliketoshop.net/product/nextProduct?
currentProductId=6&path=http://192.168.0.68/admin
```

4. Blind SSRF vulnerabilities

Lỗ hổng SSRF mù phát sinh khi một ứng dụng có thể được tạo ra để đưa ra back-end HTTP request tới một URL được cung cấp, nhưng response từ yêu cầu back-end không được retunred trong phản hồi front-end của web.

Tác động của các lỗ hổng blind SSRF thường thấp hơn các lỗ hổng SSRF được thông báo đầy đủ vì tính chất một chiều của chúng. Chúng không thể bị khai thác bình thường để lấy dữ liệu nhạy cảm từ các hệ thống back-end, mặc dù trong một số tình huống, chúng có thể được khai thác để thực hiện RCE.

Cách đáng tin cậy nhất để phát hiện các lỗ hổng SSRF mù là sử dụng các kỹ thuật ngoài băng tần (OAST). Điều này liên quan đến việc cố gắng kích hoạt một HTTP request đến một hệ thống bên ngoài mà bạn kiểm soát và giám sát các tương tác mạng với hệ thống đó.

How to prevent?

+ Nên tránh sử dụng các chức năng mà người dùng trực tiếp yêu cầu tài nguyên thay cho máy chủ
+ Để ngăn ngừa các lỗ hổng SSRF trong ứng dụng web nên sử dụng các *while list* các *domains* và *protocols* được phép truy cập tài nguyên từ phía máy chủ.

    - Các lược đồ URL không phải HTTP và HTTPS nên được đưa vào danh sách đen
    - Chặn các lược đồ khác nhau không được sử dụng như tệp*: ///, direct: //, feed: //, touch: // và FTP: //*

Reference

- https://portswigger.net/web-security/ssrf
- https://viblo.asia/p/ssrf-la-gi-cach-phat-hien-va-ngan-chan-tan-cong-yeu-cau-gia-mao-tu-phia-may-chu-Eb85op08K2G
- https://viblo.asia/p/server-side-request-forgery-ssrf-RQqKL9D6Z7z
- https://medium.com/@briskinfosec/ssrf-server-side-request-forgery-ae44ec737cb8
- https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html