**HTTP Host header attacks**

**HTTP Host header là gì?**

HTTP Host header là 1 header cần thiết trong HTTP/1.1. Nó chỉ định tên miền máy khách muốn truy cập. Ví dụ, khi người dùng truy cập trang https://portswigger.net/web-security, trình duyệt sẽ soạn một yêu cầu chứa Host header như sau:

GET /web-security HTTP/1.1

Host: portswigger.net

Trong một số trường hợp, chẳng hạn khi yêu cầu được chuyển tiếp bởi 1 hệ thống trung gian, giá trị của Host có thể thay đổi trước khi đến hệ thống back-end.

Mục đích của HTTP Host header?

Mục đích của HTTP Host header giúp xác định hệ thống back-end máy khách muốn giao tiếp. Nếu các yêu cầu không chứa Host headers, hoặc Host header có thể làm giả mạo theo cách nào đó, điều này có thể dẫn đến sự cố khi định tuyến các yêu cầu đến ứng dụng mong muốn. Trước đây, sự mơ hồ này không tồn tại vì mỗi địa chỉ IP sẽ chỉ lưu trữ nội dung cho 1 tên miền duy nhất. Ngày nay, do xu hướng ngày càng phát triển của các giải pháp đám mây và outsourcing (gia công) cho nhiều kiến trúc liên quan nên việc có nhiều trang web và ứng dụng có thể truy cập tại cùng một địa chỉ IP ngày càng phổ biến. Cách tiếp cận này ngày càng phổ biến một phần là do cạn kiệt địa chỉ IPv4.

Khi nhiều ứng dụng có thể truy cập qua cùng 1 địa chỉ IP, điều này có thể dẫn đến các tình huống sau:

**Virtual Hosting**

Tình huống này xảy ra khi một máy chủ web lưu trữ nhiều trang web hoặc ứng dụng. Có thể là nhiều trang web với một chủ sở hữu, cũng có thể là nhiều trang web với nhiều chủ sở hữu khác nhau được lưu trữ trên cùng 1 nền tảng. Điều này ít phổ biến hơn trước đây nhưng vẫn xảy ra với một số giải pháp SaaS dựa trên đám mây.

Mặc dù mỗi trang web sẽ có tên miền khác nhau, chúng có thể chia sẻ địa chỉ IP với máy chủ. Các trang web được lưu trữ theo cách này trên cùng một máy chủ được gọi là “virtual hosts”.

Đối với người dùng bình thường truy cập trang web, máy chủ ảo thường không thể phân biệt được với trang web được lưu trữ trên máy chủ chuyên dụng của chính nó.

**Routing traffic via an intermediary**

Một tình huống khác cũng khá phổ biến là khi các trang web được lưu trữ trên các máy chủ back-end khác nhau nhưng tất cả lưu lượng giữa máy khách và máy chủ được chuyển tiếp qua 1 hệ thống trung gian. Có thể dùng 1 con cân bằng tải hoặc reverse proxy. Thiết lập này đặc biệt phổ biến trong trường hợp máy khách truy cập trang web qua CDN (content delivery network).

Trong trường hợp này, mặc dù các trang web được lưu trữ trên các máy chủ back-end khác nhau, tất cả tên miền sẽ phân giải thành 1 địa chỉ IP duy nhất của hệ thống trung gian. Điều này đưa ra một số thách thức tương tự như virtual hosting vì reverse proxy hay cân bằng tải cần biết chính xác back-end để nó chuyển tiếp yêu cầu.

**Cách HTTP Host header giải quyết vấn đề?**

Trong hai tình huống trên, ta dựa vào Host header để chỉ định người nhận mong muốn. Giống như quá trình gửi thư cho ai đó sống ở chung cư. Toàn bộ căn nhà có chung 1 địa chỉ đường nhưng có nhiều căn hộ khác nhau mà mỗi người cần nhận được thư chính xác bằng cách nào đó. Một giải pháp cho vấn đề này chỉ đơn giản là thêm số căn hộ hoặc tên người nhận vào địa chỉ. Trong trường hợp http, host header có mục đích tương tự

Khi một trình duyệt gửi yêu cầu, Url sẽ phân giải thành địa chỉ IP của một máy chủ cụ thể. Khi máy chủ nhận được yêu cầu, nó sẽ xem xét Host header để xác định hệ thống back-end mong muốn và chuyển tiếp yêu cầu.

**Tấn công HTTP Host header**

Các tấn công HTTP Host header khai thác các trang web bị dính lỗ hổng khi nó xử lí giá trị của Host header theo cách không an toàn. Nếu máy chủ tin tưởng Host header, và thất bại trong việc xác thực hoặc chặn nó đúng cách, kẻ tấn công có thể dùng đầu vào này để tiêm payload độc hại lợi dụng hành vi bên phía máy chủ. Các cuộc tấn công tiêm payload trực tiếp vào Host header gọi là “Host header injection”

Các ứng dụng web được đóng gói sẵn thường không biết chúng được triển khai tên miền nào nếu nó không được chỉ định trong 1 tệp cấu hình trong quá trình thiết lập. Khi chúng cần biết tên miền hiện tại, ví dụ, để tạo 1 Url tuyệt đối bao gồm 1 email, chúng có thể lấy tên miền từ Host header:

<a href="https://\_SERVER['HOST']/support">Contact support</a>

Giá trị header có thể sử dụng trong nhiều tương tác giữa các hệ thống khác nhau của trang web.

Vì Host header do người dùng kiểm soát, điều này có thể dẫn đến một số vấn đề. Nếu đầu vào không được xác thực hoặc chặn đúng cách, Host header là hướng đi rất tiềm năng để khai thác nhiều lỗ hổng khác:

* Web cache poisoning
* Business logic flaws in specific functionality
* Routing-based SSRF
* Classic server-side vulnerabilities, such as SQL injection

**Lỗ hổng HTTP Host header sinh ra như thế nào?**

Các lỗ hổng HTTP Host header sinh ra do lập trình viên cho rằng header này không do người dùng kiểm soát. Điều này làm cho máy chủ tin tưởng Host header mà không xác thực hay chặn nó mặc dù kẻ tấn công có thể dễ dàng thay đổi bằng Burp Proxy.

Thậm chí nếu Host header được xử lí an toàn hơn, dựa vào cấu hình của các máy chủ để xử lí các yêu cầu, Host có thể bị ghi đè bằng cách tiêm các headers khác. Chủ sở hữu trang web thường không quan tâm đến các header được hỗ trợ mặc định nên chúng không được xử lí an toàn như nhau.

Thực tế cho thấy các lỗ hổng sinh ra không phải do lập trình kém mà vì cấu hình không an toàn trong hệ thống. Các sự cố cấu hình thường xảy ra vì các trang web tích hợp công nghệ thứ ba trong kiến trúc của chúng mà không cần hiểu rõ các lựa chọn cấu hình và thực thi bảo mật.

**Khai thác lỗ hổng HTTP Host header**

**Cách kiểm tra các lỗ hổng bằng HTTP Host header**

Để xem trang web có bị dính lỗ hổng qua HTTP Host header, bạn sẽ cần burp proxy và kiểm tra bằng công cụ Burp Repeater và Burp Intruder.

**Cung cấp 1 Host header bất kỳ**

Khi thăm dò các lỗ hổng tiêm Host header, bước đầu tiên là kiểm tra xem chuyện gì sẽ xảy ra khi bạn cung cấp 1 tên miền bất kì qua Host header.

Một số proxy lấy địa chỉ IP từ Host header trực tiếp làm cho chúng ta không thể kiểm tra được. Mọi thay đổi bạn làm trên Host header sẽ chỉ khiến yêu cầu được gửi đến 1 địa chỉ IP khác. Tuy nhiên, Burp Suite tách biệt Host header và địa chỉ IP cho phép bạn cung cấp bất kỳ Host header mà bạn muốn, trong khi vẫn đảm bảo yêu cầu được gửi đến mục tiêu đã định.

Thông thường, bạn có thể truy cập trang web mình nhắm đến kể cả khi bạn cung cấp 1 Host header không đúng. Điều này xảy ra vì một số lí do. Ví dụ, một số máy chủ thường được cấu hình mặc định trong trường hợp chúng nhận được các yêu cầu có tên miền mà chúng không nhận ra. Nếu trang web bạn nhắm đến thiết lập mặc định thì bạn may mắn. Trong trường hợp đó, bạn có thể nghiên cứu hành vi của ứng dụng khi thay đổi Host header.

Nói cách khác, vì Host header là 1 thành phần cơ bản của 1 trang web, giả mạo nó thường có nghĩa là bạn sẽ không thể tiếp cận được ứng dụng mình nhắm đến. Máy chủ front-end hoặc cân bằng tải nhận được yêu cầu của bạn có thể không biết nơi chuyển tiếp nên sẽ thông báo lỗi “Invalid Host header”. Điều này có thể xảy ra nếu mục tiêu của bạn được truy cập qua CDN. Trong trường hợp đó, bạn nên chuyển sang một số kĩ thuật được nêu bên dưới.

**Kiểm tra lỗi xác thực**

Thay vì nhận phản hồi “Invalid Host header”, yêu cầu của bạn có thể bị chặn do máy chủ thực thi bảo mật. Ví dụ, một số trang web sẽ xác thực xem Host header có phù hợp với SNI từ TLS handshake. Điều này không có nghĩa là máy chủ miễn nhiễm với tấn công Host header.

Bạn nên tìm hiểu cách trang web phân tích Host header. Điều này có thể bộc lộ những sơ hở để bạn có thể vượt qua được lớp xác thực. Ví dụ, một số thuật toán phân tích sẽ bỏ qua cổng từ Host header, có nghĩa là chỉ tên miền được xác thực. Nếu bạn có thể cung cấp 1 cổng không phải là số, bạn có thể giữ nguyên tên miền để đảm bảo rằng bạn tiếp cận được ứng dụng mình nhắm đến, trong khi tiêm payload qua cổng.

GET /example HTTP/1.1

Host: vulnerable-website.com:bad-stuff-here

Các trang web khác sẽ cố gắng áp dụng logic phù hợp để cho phép tên miền con bất kỳ. Trong trường hợp đó, bạn có thể bỏ qua được toàn bộ lớp xác thực bằng cách đăng kí 1 tên miền bất kỳ kết thúc bằng chuỗi kí tự nằm trong whitelist của trang web:

GET /example HTTP/1.1

Host: notvulnerable-website.com

Thay vào đó, bạn có thể lợi dụng tên miền con không an toàn mà bạn thâm nhập được:

GET /example HTTP/1.1

Host: hacked-subdomain.vulnerable-website.com

**Gửi đi các yêu cầu mơ hồ**

Code xác thực Host header và code dễ bị tấn công thường nằm trong các ứng dụng khác nhau hoặc thậm chí trên các máy chủ riêng biệt. Bằng cách phát hiện và khai thác những điểm khác biệt trong cách chúng ta truy suất Host header, bạn có thể gửi yêu cầu mơ hồ dường như có một máy chủ khác tùy thuộc vào hệ thống nào đang xem xét nó.

Sau đây là một số ví dụ bạn có thể tạo các yêu cầu mơ hồ:

**Trùng lặp Host headers**

Một cách tiếp cận khả thi là thử thêm Host headers trùng lặp. Phải thừa nhận rằng điều này thường sẽ dẫn đến việc yêu cầu của bạn bị chặn. Tuy nhiên, vì trình duyệt khó có thể gửi yêu cầu như vậy, bạn có thể may mắn tìm được lập trình viên không lường trước được trường hợp này. Trong trường hợp này, bạn có thể phát hiện được một số hành vi kỳ quặc của ứng dụng.

Các hệ thống và công nghệ khác nhau sẽ xử lí trường hợp này theo nhiều cách khác nhau nhưng thông thường sẽ có 1 header được ưu tiên hơn, có thể ghi đè giá trị của nó một cách hiệu quả. Khi hệ thống không thống nhất được header nào chính xác, điều này có thể dẫn đến những khác biệt mà bạn có thể khai thác được. Xem xét yêu cầu sau:

GET /example HTTP/1.1

Host: vulnerable-website.com

Host: bad-stuff-here

Giả sử front-end ưu tiên header đầu tiên nhưng back-end lại thích header cuối cùng. Nếu bạn gặp trường hợp trên, bạn có thể dùng header đầu tiên để đảm bảo yêu cầu của bạn được chuyển đến hệ thống mình mong muốn và dùng header thứ hai để truyền payload vào trong code của máy chủ.

**Cung cấp 1 Url tuyệt đối**

Mặc dù yêu cầu thường chỉ định đường dẫn tương đối đối với miền được yêu cầu, nhiều máy chủ cũng cấu hình để hiểu các yêu cầu có Urls tuyệt đối.

Cung cấp 1 url tuyệt đối và 1 Host header có thể dẫn đến sự khác biệt giữa các hệ thống khác nhau. Thông thường, dòng yêu cầu sẽ được ưu tiên khi gửi đi nhưng trong thực tế không phải lúc nào cũng vậy. Bạn có thể khai thác sự khác biệt theo cách giống như Host header trùng lặp.

GET https://vulnerable-website.com/ HTTP/1.1

Host: bad-stuff-here

Lưu ý là bạn cần thử với nhiều giao thức khác nhau. Các máy chủ thường sẽ xử lí khác nhau tùy thuộc vào dòng yêu cầu có chứa http hoặc https url

**Thêm ngắt dòng**

Bạn có thể khám phá hành vi kì quặc bằng cách thụt lề các HTTP header. Một số máy chủ sẽ hiểu header bị thụt lề là 1 dòng được bao bọc, do đó coi nó như giá trị của header trước đó. Một số máy chủ sẽ bỏ qua header bị thụt lề.

Vì xử lí thiếu nhất quán, thường có sự khác biệt giữa các hệ thống khác nhau khi xử lí các yêu cầu. Ví dụ, xem xét yêu cầu sau:

GET /example HTTP/1.1

Host: bad-stuff-here

Host: vulnerable-website.com

Trang web có thể chặn các yêu cầu có nhiều Host headers, nhưng bạn có thể bỏ qua lớp xác thực này bằng cách thụt lề một trong số chúng. Nếu front-end bỏ qua header bị thụt lề, yêu cầu sẽ được xử lí như 1 yêu cầu ban đầu có Host header là vulnerable-website.com. Giả sử back-end bỏ qua khoảng trắng ở đầu và ưu tiên cho header đầu tiên trong trường hợp trùng lặp. Sự khác biệt này cho phép bạn truyền giá trị bất kì qua Host header được bao bọc.

**Ghi đè Host headers**

Thậm chí nếu bạn không thể ghi đè Host header bằng yêu cầu mơ hồ, có thể ghi đè Host header bằng các header khác

Các trang web thường được truy cập qua 1 hệ thống trung gian như cân bằng tải hoặc reverse proxy. Host header mà back-end nhận được là tên miền của hệ thống trung gian.

Bạn có thể dùng X-Forwarded-Host để tiêm đầu vào độc hại đồng thời phá vỡ mọi xác thực trên Host header:

GET /example HTTP/1.1

Host: vulnerable-website.com

X-Forwarded-Host: bad-stuff-here

Mặc dù X-Forwarded-Host là header chính để kiểm tra hành vi này, bạn có thể gặp các headers khác phục vụ cho mục đích giống vậy:

* X-Host
* X-Forwarded-Server
* X-HTTP-Host-Override
* Forwarded

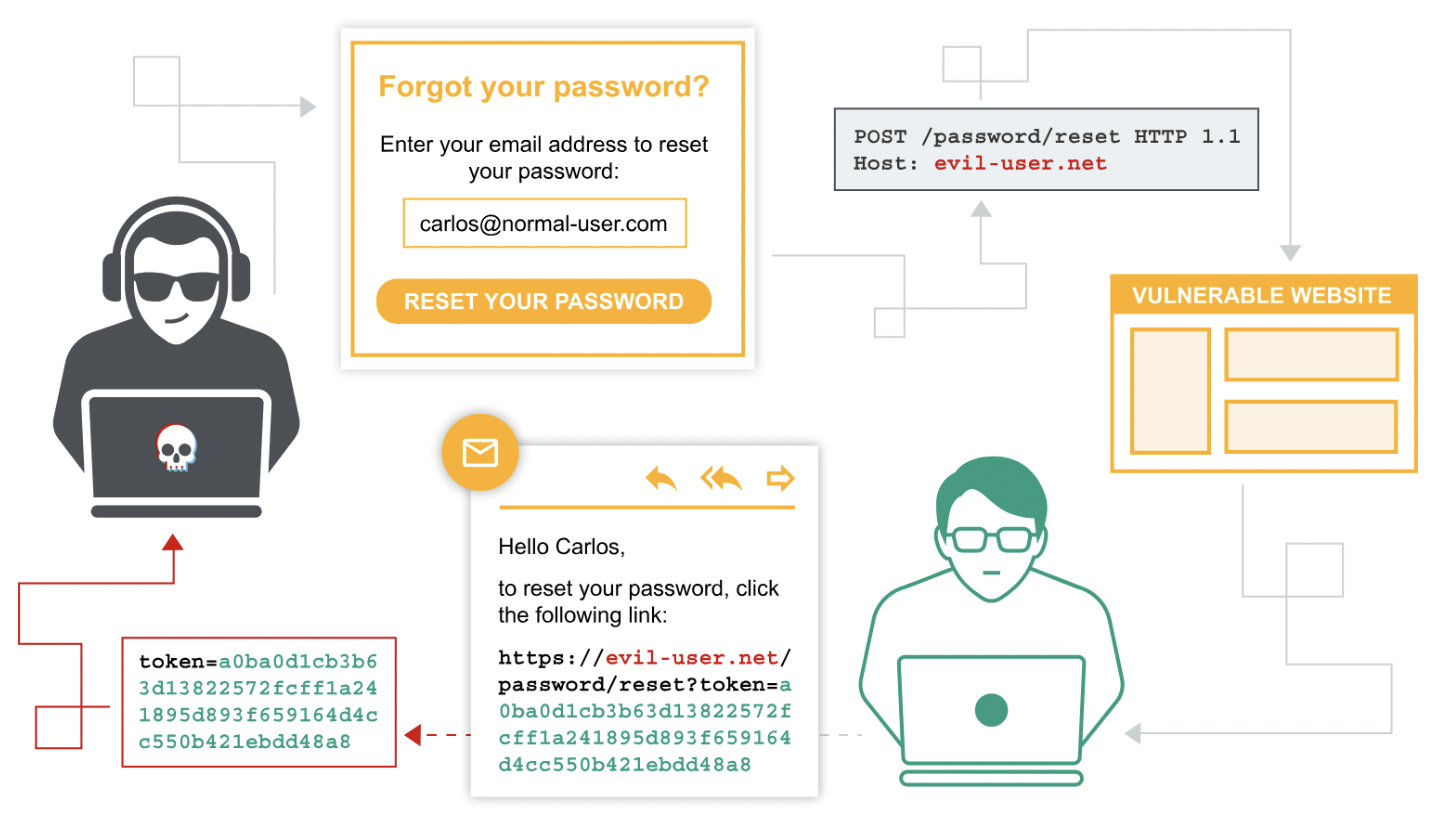
Từ góc độ bảo mật, điều quan trọng cần lưu ý là một số trang web hỗ trợ Header này vì nó được thực thi mặc định bởi công nghệ bên thứ ba.

**Cách khai thác HTTP Host header**

Khi bạn đã xác định được bạn có thể truyền hostname bất kỳ đến ứng dụng, bạn có thể tìm cách khai thác chúng.

**Password reset poisoning**

Password reset poisonging là 1 kĩ thuật mà kẻ tấn công lợi dụng trang web có lỗ hổng trong việc tạo đường dẫn đặt lại mật khẩu trỏ đến miền dưới sự kiểm soát của họ. Hành vi này có thể lợi dụng để đánh cắp tokens bí mật để đặt lại mật khẩu của người dùng và cuối cùng là thâm nhập tài khoản.



**Chức năng đặt lại mật khẩu hoạt động như thế nào?**

Tất cả trang web yêu cầu đăng nhập cũng thực thi chức năng cho phép người dùng đặt lại mật khẩu nếu họ quên chúng. Có nhiều cách để làm việc đó, với mức độ bảo mật và tính thực tiễn khác nhau. Một trong những cách tiếp cận phổ biến nhất là:

1. Người dùng nhập tên hoặc địa chỉ email và gửi yêu cầu đặt lại mật khẩu
2. Trang web kiểm tra người dùng có tồn tại, sau đó tạo 1 mã tạm thời, duy nhất và entropy cao liên kết với tài khoản người dùng ở back-end
3. Trang web gửi email đến người dùng chứa đường dẫn để đặt lại mật khẩu. Mã đặt lại duy nhất của người dùng sẽ có trong tham số của Url:

https://normal-website.com/reset?token=0a1b2c3d4e5f6g7h8i9j

1. Khi người dùng truy cập Url trên, trang web kiểm tra token có hợp lệ và xác định tài khoản nào được đặt lại. Nếu mọi thứ đúng như mong đợi, người dùng sẽ được yêu cầu nhập mật khẩu mới. Cuối cùng, token bị hủy.

Quá trình này đủ đơn giản và bảo mật khi so sánh với những cách khác.Tuy nhiên, tính bảo mật của nó dựa trên nguyên tắc chỉ người dùng truy cập được hộp thư mới nhận được mã đặt lại duy nhất. Password reset poisoning là 1 phương pháp đánh cắp token để thay đổi mật khẩu của người dùng khác.

**Cách thực hiện 1 cuộc tấn công password reset poisoning**

Nếu Url gửi đến người dùng được tạo dựa trên đầu vào có thể kiểm soát được như Host header, có thể thực hiện 1 cuộc tấn công password reset poisoning:

1. Kẻ tấn công lấy được địa chỉ email hoặc tên người dùng và gửi yêu cầu đặt lại mật khẩu thay cho họ. Khi gửi form, họ sẽ chặn yêu cầu HTTP và chỉnh sửa Host header để nó trỏ đến miền mà họ kiểm soát. Ví dụ, chúng ta sẽ dùng evil-user.net
2. Nạn nhân nhận được email đặt lại mật khẩu từ trang web. Tên miền trong Url trỏ đến máy chủ của kẻ tấn công:

https://evil-user.net/reset?token=0a1b2c3d4e5f6g7h8i9j

1. Nếu nạn nhân nhấn vào đường dẫn đó, mã đặt lại mật khẩu sẽ được gửi đến máy chủ của kẻ tấn công
2. Kẻ tấn công có thể truy cập đường dẫn đặt lại mật khẩu hợp lệ

Trong thực tế, để nạn nhân nhấn vào đường dẫn đặt lại mật khẩu, kẻ tấn công phải kích động họ bằng 1 thông báo giả mạo.

Nếu bạn không kiểm soát được đường dẫn đặt lại mật khẩu, bạn có thể dùng Host header để tiêm HTML vào trong những bức thư nhạy cảm. Lưu ý là ứng dụng thư không thực thi Javascript nhưng các kĩ thuật tiêm HTML như tấn công dangling markup có thể áp dụng.