

PHẦN I

CÁCH HOẠT ĐỘNG CỦA BẢO MẬT WEB API



CHUẨN BỊ

KIỂM TRA AN NINH



Thử nghiệm bảo mật API không hoàn toàn phù hợp với
khuôn mẫu của thử nghiệm thâm nhập chung, cũng như
không phù hợp với khuôn mẫu của thử nghiệm bút thử ứng

dụng web. Do quy mô và độ phức tạp của bề mặt tấn công API

của nhiều tổ chức, thử nghiệm thâm nhập API là dịch vụ độc đáo của

riêng tổ chức đó. Trong chương này, tôi sẽ thảo luận về các tính
năng của API mà bạn nên đưa vào thử nghiệm và tài liệu của mình
trước cuộc tấn công. Nội dung trong chương này sẽ giúp bạn đánh giá
số lượng hoạt động cần thiết cho một cam kết, đảm bảo rằng bạn có kế
hoạch thử nghiệm tất cả các tính năng của API mục tiêu và giúp bạn tránh gặp rắc

Thử nghiệm thâm nhập API yêu cầu phạm vi được phát triển tốt hoặc tài khoản về
các mục tiêu và tính năng của những gì bạn được phép thử nghiệm, điều này đảm bảo
khách hàng và người thử nghiệm có sự hiểu biết lẫn nhau về công việc đang được thực hiện.
Phạm vi của một cam kết thử nghiệm bảo mật API phụ thuộc vào một số yếu tố: phương
pháp của bạn, mức độ thử nghiệm, các tính năng mục tiêu, bất kỳ hạn chế nào đối với
thử nghiệm, các yêu cầu báo cáo của bạn và liệu bạn có định tiến hành thử nghiệm khắc
phục hay không.

Nhận ủy quyền

Trước khi bạn tấn công các API, điều cực kỳ quan trọng là bạn phải nhận được một hợp đồng đã ký bao gồm phạm vi cam kết và cấp cho bạn quyền tấn công tài nguyên của khách hàng trong một khung thời gian cụ thể.

Đối với thử nghiệm thâm nhập API, hợp đồng này có thể ở dạng tuyên bố công việc đã ký (SOW) liệt kê các mục tiêu đã được phê duyệt, đảm bảo rằng bạn và khách hàng của bạn đồng ý về dịch vụ mà họ muốn bạn cung cấp. Điều này bao gồm đi đến thỏa thuận về khía cạnh nào của API sẽ được kiểm tra, ngăn chặn việc khai thác bất kỳ loại trừ nào và thiết lập thời gian đã thỏa thuận để thực hiện kiểm tra.

Kiểm tra kỹ xem người ký hợp đồng có phải là đại diện của khách hàng mục tiêu, người có quyền cho phép thử nghiệm hay không. Ngoài ra, hãy đảm bảo rằng các tài sản được kiểm tra thuộc sở hữu của khách hàng; nếu không, bạn sẽ cần rửa sạch và lập lại các hướng dẫn này với chủ sở hữu thích hợp. Hãy nhớ xem xét vị trí nơi khách hàng lưu trữ API của họ và liệu họ có thực sự ở vị trí cho phép kiểm tra cả phần mềm và phần cứng hay không.

Một số tổ chức có thể quá hạn chế với tài liệu xác định phạm vi của họ. Nếu bạn có cơ hội để phát triển phạm vi, tôi khuyên rằng, bằng lời lẽ bình tĩnh của riêng bạn, bạn vui lòng giải thích cho khách hàng của mình rằng tội phạm không có phạm vi hoặc giới hạn. Tội phạm thực sự không xem xét các dự án khác đang tiêu tốn tài nguyên CNTT; họ không tránh mạng con với các máy chủ sản xuất nhạy cảm hoặc quan tâm đến việc hack vào những thời điểm bất tiện trong ngày. Hãy cố gắng thuyết phục khách hàng của bạn về giá trị của việc tham gia ít hạn chế hơn và sau đó làm việc với họ để ghi lại các chi tiết cụ thể.

Gặp gỡ khách hàng, giải thích chính xác những gì sẽ xảy ra, và sau đó ghi lại chính xác điều đó trong hợp đồng, email nhắc nhở hoặc ghi chú. Nếu bạn tuân thủ thỏa thuận bằng văn bản cho các dịch vụ được yêu cầu, thì bạn nên hoạt động hợp pháp và có đạo đức. Tuy nhiên, bạn nên tham khảo ý kiến của luật sư hoặc bộ phận pháp lý của mình để giảm thiểu rủi ro.

Mối đe dọa lập mô hình thử nghiệm API

Mô hình hóa mối đe dọa là quy trình được sử dụng để vạch ra các mối đe dọa đối với nhà cung cấp API. Nếu bạn lập mô hình thử nghiệm thâm nhập API dựa trên mối đe dọa có liên quan, thì bạn sẽ có thể chọn các công cụ và kỹ thuật nhắm vào cuộc tấn công đó. Các thử nghiệm tốt nhất của API sẽ là những thử nghiệm phù hợp với các mối đe dọa thực tế đối với nhà cung cấp API.

Tác nhân đe dọa là kẻ thù hoặc kẻ tấn công API. Đối thủ có thể là bất kỳ ai, từ một thành viên của công chúng tình cờ biết đến API mà không biết hoặc có ít kiến thức về ứng dụng cho đến khách hàng sử dụng ứng dụng, đối tác kinh doanh lừa đảo hoặc người trong cuộc biết khá nhiều về ứng dụng. Để thực hiện thử nghiệm mang lại giá trị cao nhất cho tính bảo mật của API, lý tưởng nhất là vạch ra kẻ thù có thể xảy ra cũng như các kỹ thuật hack của chúng.

Phương pháp kiểm tra của bạn phải trực tiếp theo quan điểm của tác nhân đe dọa, vì quan điểm này sẽ xác định thông tin bạn đang

đưa ra về mục tiêu của bạn. Nếu tác nhân đe dọa không biết gì về API, họ sẽ cần thực hiện nghiên cứu để xác định cách thức mà họ có thể nhắm mục tiêu vào ứng dụng. Tuy nhiên, một đối tác kinh doanh lừa đảo hoặc mối đe dọa nội bộ có thể biết khá nhiều về ứng dụng mà không cần bất kỳ hoạt động do thám nào. Để giải quyết những khác biệt này, có ba cách tiếp cận kiểm thử bút cơ bản: hộp đen, hộp xám và hộp trắng.

Mô hình kiểm thử hộp đen về mối đe dọa của kẻ tấn công cơ hội-ai đó có thể tình cờ gặp tổ chức mục tiêu hoặc API của tổ chức đó. Trong tương tác API hộp đen thực sự, khách hàng sẽ không tiết lộ bất kỳ thông tin nào về bề mặt tấn công của họ cho người kiểm tra. Bạn có thể sẽ bắt đầu cam kết của mình mà không có gì khác ngoài tên của công ty đã ký SOW. Từ đó, nỗ lực thử nghiệm sẽ liên quan đến việc tiến hành trinh sát bằng cách sử dụng trí thông minh nguồn mở (OSINT) để tìm hiểu càng nhiều càng tốt về tổ chức mục tiêu. Bạn có thể phát hiện ra cuộc tấn công bề ngoài của mục tiêu bằng cách sử dụng kết hợp nghiên cứu công cụ tìm kiếm, phương tiện truyền thông xã hội, hồ sơ tài chính công khai và thông tin DNS để tìm hiểu càng nhiều càng tốt về miền của tổ chức. Các công cụ và kỹ thuật cho phương pháp này sẽ được trình bày chi tiết hơn trong Chương 6. Khi bạn đã tiến hành OSINT, bạn nên biên soạn một danh sách các địa chỉ IP, URL và điểm cuối API mục tiêu mà bạn có thể trình bày cho khách hàng xem xét. Khách hàng nên xem danh sách mục tiêu của bạn và sau đó cho phép thử nghiệm.

Thử nghiệm hộp xám là một cam kết có nhiều thông tin hơn nhằm tìm cách phân bổ lại thời gian dành cho việc do thám và thay vào đó đầu tư thời gian đó vào thử nghiệm tích cực. Khi thực hiện kiểm tra hộp xám, bạn sẽ bắt chước một kẻ tấn công có thông tin tốt hơn. Bạn sẽ được cung cấp thông tin chẳng hạn như mục tiêu nào nằm trong và ngoài phạm vi cũng như quyền truy cập vào tài liệu API và có thể là tài khoản người dùng cơ bản. Bạn cũng có thể được phép bỏ qua một số biện pháp kiểm soát bảo mật chu vi mạng nhất định.

Các chương trình tiền thưởng lỗi thường nằm ở đâu đó trên quang phổ giữa thử nghiệm hộp đen và hộp xám. Chương trình tiền thưởng tìm lỗi là một cam kết trong đó một công ty cho phép tin tặc kiểm tra các ứng dụng web của mình để tìm các mối liên hệ dễ bị tổn thương và các phát hiện thành công dẫn đến việc công ty chủ quản cung cấp một khoản thanh toán tiền thưởng cho người tìm thấy. Tiền thưởng lỗi không hoàn toàn là “hộp đen” vì thợ săn tiền thưởng được cung cấp các mục tiêu đã được phê duyệt, các mục tiêu nằm ngoài phạm vi, các loại lỗ hổng được khen thưởng và các loại tấn công được phép. Với những hạn chế này, những người săn tiền thưởng lỗi chỉ bị giới hạn bởi nguồn lực của chính họ, vì vậy họ quyết định lượng thời gian dành cho việc do thám so với các kỹ thuật khác. Nếu bạn quan tâm đến việc tìm hiểu thêm về săn tiền thưởng lỗi, tôi thực sự khuyên bạn nên tham gia Trại huấn luyện tiền thưởng lỗi của Vickie Li (<https://nostarch.com/bug-bounty-bootcamp>).

Trong cách tiếp cận hộp trắng, khách hàng tiết lộ càng nhiều thông tin càng tốt. có thể về các hoạt động bên trong của môi trường của họ. Ngoài thông tin được cung cấp cho kiểm thử hộp xám, thông tin này có thể bao gồm quyền truy cập vào mã nguồn ứng dụng, thông tin thiết kế, bộ công cụ phát triển phần mềm (SDK) được sử dụng để phát triển ứng dụng, v.v. Kiểm thử hộp trắng mô hình hóa mối đe dọa của kẻ tấn công bên trong – một người biết hoạt động bên trong của tổ chức và có quyền truy cập vào mã nguồn thực tế. Bạn càng được cung cấp nhiều thông tin trong tương tác hộp trắng, mục tiêu sẽ càng được kiểm tra kỹ lưỡng hơn.

Quyết định của khách hàng để thực hiện hợp trắng đính hôn, hợp đen, hoặc ở đâu đó ở giữa nên dựa trên mô hình mối đe dọa và thông tin tình báo về mối đe dọa. Bằng cách sử dụng mô hình mối đe dọa, hãy làm việc với khách hàng của bạn để lập hồ sơ về kẻ tấn công có khả năng cao nhất của tổ chức. Ví dụ: giả sử bạn đang làm việc với một doanh nghiệp nhỏ không quan trọng về mặt chính trị; nó không phải là một phần của chuỗi cung ứng cho một công ty quan trọng hơn và không cung cấp dịch vụ thiết yếu. Trong trường hợp đó, sẽ là vô lý nếu cho rằng kẻ thù của tổ chức là một mối đe dọa dai dẳng tiên tiến (APT) được tài trợ tốt giống như một quốc gia. Sử dụng các kỹ thuật của APT chống lại doanh nghiệp nhỏ này sẽ giống như sử dụng máy bay không người lái tấn công một tên trộm vặt. Thay vào đó, để cung cấp cho khách hàng giá trị cao nhất, bạn nên sử dụng mô hình mối đe dọa để tạo ra một mối đe dọa thực tế. Trong trường hợp này, kẻ tấn công có khả năng cao nhất có thể là một cá nhân cơ hội, có kỹ năng trung bình, tình cờ tìm thấy trang web của tổ chức và có khả năng chỉ chạy các khai thác đã công bố đối với các lỗ hổng đã biết. Phương pháp thử nghiệm phù hợp với kẻ tấn công cơ hội sẽ là thử nghiệm hợp đen giới hạn.

Cách hiệu quả nhất để lập mô hình mối đe dọa cho khách hàng là tiến hành một cuộc khảo sát với họ. Cuộc khảo sát sẽ cần tiết lộ phạm vi tiếp xúc chắc chắn của khách hàng với các cuộc tấn công, ý nghĩa kinh tế, sự tham gia chính trị của họ, liệu họ có tham gia vào bất kỳ chuỗi cung ứng nào hay không, liệu họ có cung cấp các dịch vụ thiết yếu hay không và liệu có những động cơ tiềm ẩn nào khác khiến tội phạm muốn hay không. để tấn công họ. Bạn có thể phát triển khảo sát của riêng mình hoặc tổng hợp khảo sát từ các nguồn chuyên nghiệp hiện có như MITRE ATT&CK (<https://attack.mitre.org>) hoặc OWASP (https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html).

Phương pháp thử nghiệm bạn chọn sẽ xác định phần lớn nỗ lực xác định phạm vi còn lại. Vì những người kiểm tra hợp đen được cung cấp rất ít thông tin về phạm vi, nên các mục phạm vi còn lại có liên quan đến kiểm tra hợp xám và hợp trắng.

Những tính năng API nào bạn nên thử nghiệm

Một trong những mục tiêu chính của việc xác định phạm vi tham gia bảo mật API là khám phá khối lượng công việc bạn sẽ phải thực hiện trong quá trình thử nghiệm của mình. Do đó, bạn phải tìm hiểu xem có bao nhiêu điểm cuối API, phương pháp, phiên bản, tính năng, cơ chế xác thực và ủy quyền cũng như mức đặc quyền mà bạn sẽ cần kiểm tra. Mức độ của thử nghiệm có thể được xác định thông qua các cuộc phỏng vấn với khách hàng, đánh giá tài liệu API có liên quan và quyền truy cập vào bộ sưu tập API. Sau khi có thông tin được yêu cầu, bạn sẽ có thể đánh giá xem sẽ mất bao nhiêu giờ để kiểm tra hiệu quả các API của khách hàng.

Thử nghiệm xác thực API

Xác định cách khách hàng muốn xử lý việc kiểm tra người dùng được xác thực và chưa được xác thực. Khách hàng có thể muốn bạn kiểm tra các vai trò và người dùng API khác nhau để xem liệu có lỗ hổng nào xuất hiện ở bất kỳ cấp độ đặc quyền khác nhau nào không. Khách hàng cũng có thể muốn bạn kiểm tra quy trình mà họ sử dụng để xác thực và cấp quyền cho người dùng. Khi nó đến

Điểm yếu của API, nhiều lỗ hổng bất lợi được phát hiện trong xác thực và ủy quyền. Trong tình huống hộp đen, bạn sẽ cần tìm ra quy trình xác thực của mục tiêu và tìm cách được xác thực.

Tường lửa ứng dụng web

Trong tương tác hộp trắng, bạn sẽ muốn biết về bất kỳ tường lửa ứng dụng web (WAF) nào có thể đang được sử dụng. WAF là một cơ chế bảo vệ chung cho các ứng dụng web và API. WAF là một thiết bị kiểm soát lưu lượng mạng tiếp cận API. Nếu WAF đã được thiết lập đúng cách, bạn sẽ nhanh chóng phát hiện ra trong quá trình thử nghiệm khi quyền truy cập vào API bị mất sau khi thực hiện một lần quét đơn giản. WAF có thể rất hữu ích trong việc hạn chế các yêu cầu không mong muốn và dừng kiểm tra bảo mật API trong quá trình thực hiện. WAF hiệu quả sẽ phát hiện tần suất yêu cầu hoặc lỗi yêu cầu và cấm thiết bị thử nghiệm của bạn.

Trong các cam kết hộp xám và hộp trắng, khách hàng có thể sẽ tiết lộ WAF cho bạn, tại thời điểm đó bạn sẽ có một số quyết định để thực hiện. Mặc dù có nhiều ý kiến khác nhau về việc liệu các tổ chức có nên nơi lồng bảo mật để làm cho quá trình thử nghiệm hiệu quả hơn hay không, thì việc bảo vệ an ninh mạng theo lớp là chìa khóa để bảo vệ các tổ chức một cách hiệu quả. Nói cách khác, không ai nên bỏ tất cả trứng vào giỏ WAF. Nếu có đủ thời gian, kẻ tấn công dai dẳng có thể tìm hiểu ranh giới của WAF, tìm ra cách vượt qua nó hoặc sử dụng lỗ hổng zero-day khiến nó không liên quan.

Lý tưởng nhất là khách hàng sẽ cho phép địa chỉ IP tấn công của bạn bỏ qua WAF hoặc điều chỉnh mức độ bảo mật ranh giới điển hình của họ để bạn có thể kiểm tra các biện pháp kiểm soát bảo mật sẽ được hiển thị cho người tiêu dùng API của họ. Như đã thảo luận trước đó, việc lập kế hoạch và quyết định như thế này thực sự là mô hình hóa mối đe dọa. Các thử nghiệm tốt nhất của API sẽ là những thử nghiệm phù hợp với các mối đe dọa thực tế đối với nhà cung cấp API. Để có một thử nghiệm mang lại giá trị cao nhất cho tính bảo mật của API, lý tưởng nhất là vạch ra đối thủ có thể xảy ra và các kỹ thuật hack của chúng. Nếu không, bạn sẽ thấy mình đang thử nghiệm tính hiệu quả của WAF của nhà cung cấp API thay vì hiệu quả của các biện pháp kiểm soát bảo mật API của họ.

Thử nghiệm ứng dụng di động

Nhiều tổ chức có các ứng dụng di động mở rộng cuộc tấn công bề mặt. Hơn nữa, các ứng dụng dành cho thiết bị di động thường dựa vào API để truyền dữ liệu trong ứng dụng và đến các máy chủ hỗ trợ. Bạn có thể kiểm tra các API này thông qua đánh giá mã thủ công, phân tích mã nguồn tự động và phân tích động. Đánh giá mã thủ công liên quan đến việc truy cập mã nguồn của ứng dụng di động và tìm kiếm các lỗ hổng tiềm ẩn. Phân tích mã nguồn tự động cũng tương tự, ngoại trừ nó sử dụng các công cụ tự động để hỗ trợ tìm kiếm các khả năng của lỗ hổng và các tạo tác thú vị. Cuối cùng, phân tích động là kiểm tra ứng dụng trong khi nó đang chạy. Phân tích động bao gồm chặn các yêu cầu API máy khách của ứng dụng dành cho thiết bị di động và phản hồi API máy chủ, sau đó cố gắng tìm ra các điểm yếu có thể bị khai thác.

Kiểm tra tài liệu API

Tài liệu của API là sổ tay mô tả cách sử dụng API và bao gồm các yêu cầu xác thực, vai trò người dùng, ví dụ sử dụng và thông tin điểm cuối API. Tài liệu tốt là điều cần thiết cho sự thành công thương mại của bất kỳ API tự túc nào. Nếu không có tài liệu API hiệu quả, các doanh nghiệp sẽ phải dựa vào đào tạo để hỗ trợ người tiêu dùng của họ. Vì những lý do này, bạn có thể đặt cược rằng API mục tiêu của bạn có tài liệu.

Tuy nhiên, tài liệu này có thể chứa đựng những điểm không chính xác, thông tin lỗi thời và lỗ hổng tiết lộ thông tin. Là một hacker API, bạn nên tìm kiếm tài liệu API của mục tiêu và sử dụng nó để làm lợi thế cho mình. Trong thử nghiệm hộp xám và hộp trắng, kiểm tra tài liệu API phải được đưa vào phạm vi. Việc xem xét tài liệu sẽ cải thiện tính bảo mật của các API mục tiêu bằng cách phát hiện ra các điểm yếu, bao gồm các lỗi logic nghiệp vụ.

Kiểm tra giới hạn tỷ lệ

Giới hạn tỷ lệ là hạn chế về số lượng yêu cầu mà người tiêu dùng API có thể thực hiện trong một khung thời gian nhất định. Nó được thực thi bởi máy chủ web, tường lửa hoặc tường lửa ứng dụng web của nhà cung cấp API và phục vụ hai mục đích quan trọng đối với nhà cung cấp API: nó cho phép kiểm tra tiền từ API và ngăn chặn việc sử dụng quá mức tài nguyên của nhà cung cấp. Vì giới hạn tỷ lệ là một yếu tố thiết yếu cho phép các tổ chức kiểm tra tiền từ API của họ, nên bạn nên đưa nó vào phạm vi của mình trong quá trình tương tác với API.

Ví dụ: một doanh nghiệp có thể cho phép người dùng API bậc miễn phí thực hiện một yêu cầu mỗi giờ. Sau khi yêu cầu đó được thực hiện, người tiêu dùng sẽ không được thực hiện bất kỳ yêu cầu nào khác trong một giờ. Tuy nhiên, nếu người dùng trả phí cho doanh nghiệp này, họ có thể thực hiện hàng trăm yêu cầu mỗi giờ. Nếu không có biện pháp kiểm soát thích hợp, những người tiêu dùng API không trả phí này có thể tìm cách bỏ qua phí và sử dụng bao nhiêu dữ liệu tùy thích.

Thử nghiệm giới hạn tốc độ không giống như thử nghiệm từ chối dịch vụ (DoS). Thử nghiệm DoS bao gồm các cuộc tấn công nhằm phá vỡ các dịch vụ và làm cho hệ thống và ứng dụng không khả dụng với người dùng. Trong khi thử nghiệm DoS nhằm đánh giá mức độ linh hoạt của tài nguyên máy tính của tổ chức, thì thử nghiệm giới hạn tốc độ tìm cách bỏ qua các hạn chế giới hạn số lượng yêu cầu được gửi trong một khung thời gian nhất định. Cố gắng bỏ qua giới hạn tỷ lệ sẽ không nhất thiết gây ra sự gián đoạn cho các dịch vụ. Thay vào đó, việc bỏ qua giới hạn tốc độ có thể hỗ trợ cho các cuộc tấn công khác và thể hiện điểm yếu trong phương pháp kiểm tra tiền từ API của một tổ chức.

Thông thường, một tổ chức xuất bản giới hạn yêu cầu API của mình trong tài liệu API. Nó sẽ đọc một cái gì đó như sau:

Bạn có thể thực hiện các yêu cầu X trong khung thời gian Y. Nếu bạn vượt quá giới hạn này, bạn sẽ nhận được phản hồi Z từ máy chủ web của chúng tôi.

Ví dụ: Twitter giới hạn các yêu cầu dựa trên sự ủy quyền của bạn sau khi bạn được xác thực. Bậc đầu tiên có thể thực hiện 15 yêu cầu cứ sau 15 phút và bậc tiếp theo có thể thực hiện 180 yêu cầu cứ sau 15 phút. Nếu bạn vượt quá giới hạn yêu cầu của mình, bạn sẽ nhận được Lỗi HTTP 420, như thể hiện trong Hình 0-1.



Hình 0-1: Mã trạng thái Twitter HTTP từ <https://developer.twitter.com/en/docs>

Nếu không có đủ biện pháp kiểm soát bảo mật để hạn chế quyền truy cập vào API, nhà cung cấp API sẽ mất tiền do người tiêu dùng gian lận hệ thống, phát sinh thêm chi phí do sử dụng thêm tài nguyên máy chủ và dễ bị tấn công DoS.

Hạn chế và loại trừ

Trừ khi có quy định khác trong tài liệu ủy quyền thử nghiệm thâm nhập, bạn nên cho rằng mình sẽ không thực hiện các cuộc tấn công DoS và DoS phân tán (DDoS). Theo kinh nghiệm của tôi, được ủy quyền để làm như vậy là khá hiếm. Khi thử nghiệm DoS được cấp phép, nó sẽ được nêu rõ ràng trong tài liệu chính thức. Ngoài ra, ngoại trừ một số cam kết mô phỏng đối thủ nhất định, thử nghiệm thâm nhập và kỹ thuật xã hội thường được giữ dưới dạng các hoạt động riêng biệt. Nói như vậy, hãy luôn kiểm tra xem bạn có thể sử dụng các cuộc tấn công kỹ thuật xã hội hay không (chẳng hạn như lừa đảo, vishing và smishing) khi thử nghiệm thâm nhập.

Theo mặc định, không có chương trình tiền thưởng lỗi nào chấp nhận các nỗ lực của kỹ sư xã hội, tấn công DoS hoặc DDoS, tấn công khách hàng và truy cập dữ liệu khách hàng. Trong các tình huống mà bạn có thể thực hiện tấn công người dùng, chương trình thưởng đề xuất tạo nhiều tài khoản và khi có cơ hội liên quan, hãy tấn công tài khoản thử nghiệm của chính bạn.

Ngoài ra, các chương trình hoặc ứng dụng khách cụ thể có thể đánh vào các sự cố đã biết. Một số khía cạnh của API có thể được coi là phát hiện bảo mật nhưng cũng có thể là một tính năng tiện lợi dự kiến. Ví dụ: chức năng quên mật khẩu của bạn có thể hiển thị thông báo cho phép người dùng cuối biết liệu email hoặc mật khẩu của họ có sai hay không; chức năng tương tự này có thể cấp cho kẻ tấn công khả năng sử dụng vũ phu tên người dùng và email hợp lệ. Tổ chức có thể đã quyết định chấp nhận rủi ro này và không muốn bạn kiểm tra nó.

Hãy chú ý đến bất kỳ loại trừ hoặc hạn chế nào trong hợp đồng.

Khi nói đến API, chương trình có thể cho phép thử nghiệm các phần cụ thể của API nhất định và có thể hạn chế các đường dẫn nhất định trong API được phê duyệt. Ví dụ: nhà cung cấp API ngân hàng có thể chia sẻ tài nguyên với bên thứ ba và có thể không có quyền cho phép thử nghiệm. Do đó, họ có thể giải thích rằng bạn có thể tấn công điểm cuối `/api/accounts` nhưng không thể tấn công `/api/shared/accounts`. Ngoài ra, quá trình xác thực của mục tiêu có thể thông qua bên thứ ba mà bạn không được phép tấn công. Bạn sẽ cần phải chú ý đến phạm vi để thực hiện thử nghiệm được ủy quyền hợp pháp.

API đám mây kiểm tra bảo mật

Các ứng dụng web hiện đại thường được lưu trữ trên đám mây. Khi bạn tấn công một ứng dụng web được lưu trữ trên đám mây, bạn thực sự đang tấn công các máy chủ vật lý của các nhà cung cấp đám mây (có thể là Amazon, Google hoặc Microsoft). Mỗi nhà cung cấp dịch vụ đám mây có bộ điều khoản và dịch vụ thử nghiệm thâm nhập riêng mà bạn sẽ muốn làm quen. Kể từ năm 2021, các nhà cung cấp đám mây nhìn chung đã trở nên thân thiện hơn với những người thử nghiệm thâm nhập và rất ít trong số họ yêu cầu gửi giấy phép. Tuy nhiên, một số ứng dụng web và API được lưu trữ trên đám mây sẽ yêu cầu bạn phải xin phép thử nghiệm thâm nhập, chẳng hạn như đối với API Salesforce của một tổ chức.

Bạn phải luôn biết các yêu cầu hiện tại của nhà cung cấp đám mây mục tiêu trước khi tấn công. Danh sách sau đây mô tả các chính sách của các nhà cung cấp phổ biến nhất.

Amazon Web Services (AWS) AWS đã cải thiện đáng kể quan điểm của mình về thử nghiệm thâm nhập. Tại thời điểm viết bài này, AWS cho phép khách hàng của mình thực hiện tất cả các loại thử nghiệm bảo mật, ngoại trừ hoạt động dò tìm vùng DNS, tấn công DoS hoặc DDoS, tấn công DoS hoặc DDoS mô phỏng, tràn ngập cổng, tràn ngập giao thức và tràn ngập yêu cầu. Đối với bất kỳ trường hợp ngoại lệ nào đối với điều này, bạn phải gửi email cho AWS và yêu cầu quyền tiến hành thử nghiệm. Nếu bạn đang yêu cầu một ngoại lệ, hãy đảm bảo bao gồm ngày thử nghiệm, mọi tài khoản và tài sản có liên quan, số điện thoại của bạn và mô tả về cuộc tấn công được đề xuất của bạn.

Google Cloud Platform (GCP) Google chỉ đơn giản tuyên bố rằng bạn không cần xin phép hoặc thông báo cho công ty để thực hiện kiểm tra thâm nhập. Tuy nhiên, Google cũng tuyên bố rằng bạn phải tuân thủ chính sách sử dụng được chấp nhận (AUP) và điều khoản dịch vụ (TOS) và ở trong phạm vi được phép của bạn. AUP và TOS nghiêm cấm các hành động bất hợp pháp, lừa đảo, spam, phân phối các tệp độc hại hoặc phá hoại (chẳng hạn như vi-rút, sâu và ngựa thành Troy) cũng như gián đoạn dịch vụ GCP.

Microsoft Azure Microsoft sử dụng phương pháp thân thiện với tin tặc và không yêu cầu bạn phải thông báo cho công ty trước khi thử nghiệm. Ngoài ra, nó có trang "Quy tắc tham gia kiểm tra thâm nhập" nêu rõ chính xác loại thử nghiệm thâm nhập nào được phép (<https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement>).

Ít nhất là cho đến thời điểm hiện tại, các nhà cung cấp đám mây đang có quan điểm thuận lợi đối với các hoạt động thử nghiệm thâm nhập. Miễn là bạn luôn cập nhật các điều khoản của nhà cung cấp, bạn sẽ hoạt động hợp pháp nếu bạn chỉ kiểm tra các mục tiêu mà bạn được phép tấn công và tránh các cuộc tấn công có thể gây gián đoạn dịch vụ.

Kiểm tra DoS

Tôi đã đề cập rằng các cuộc tấn công DoS thường không cần bàn cãi. Làm việc với khách hàng để hiểu mức độ chấp nhận rủi ro của họ đối với cam kết nhất định. Bạn nên

coi thử nghiệm DOS như một dịch vụ chọn tham gia cho những khách hàng muốn kiểm tra hiệu suất và độ tin cậy của cơ sở hạ tầng của họ. Nếu không, hãy làm việc với khách hàng để xem họ sẵn sàng cho phép những gì.

Các cuộc tấn công DoS là một mối đe dọa lớn đối với tính bảo mật của các API. Một cuộc tấn công DoS cố ý hoặc vô tình sẽ làm gián đoạn các dịch vụ do tổ chức mục tiêu cung cấp, khiến API hoặc ứng dụng web không thể truy cập được. Một sự gián đoạn kinh doanh ngoài kế hoạch như thế này thường là một yếu tố kích hoạt để một tổ chức theo đuổi yêu cầu pháp lý. Do đó, hãy cẩn thận chỉ thực hiện thử nghiệm mà bạn được phép thực hiện!

Cuối cùng, việc khách hàng có chấp nhận thử nghiệm DoS như một phần của phạm vi hay không phụ thuộc vào khẩu vị rủi ro của tổ chức hoặc mức độ rủi ro mà tổ chức sẵn sàng chấp nhận để đạt được mục đích của mình. Hiểu được khẩu vị rủi ro của một tổ chức có thể giúp bạn điều chỉnh thử nghiệm của mình. Nếu một tổ chức tiên tiến và có nhiều niềm tin vào tính bảo mật của mình, thì tổ chức đó có thể rất muốn mạo hiểm. Một cam kết phù hợp với khẩu vị rủi ro lớn sẽ liên quan đến việc kết nối với mọi tính năng và chạy tất cả các khai thác bạn muốn.

Ở phía đối lập của quang phổ là các tổ chức rất sợ rủi ro.

Cam kết cho các tổ chức này sẽ giống như đi trên vỏ trứng. Loại tương tác này sẽ có nhiều chi tiết trong phạm vi: bất kỳ máy nào bạn có thể tấn công sẽ được đánh vắn và bạn có thể cần phải xin phép trước khi chạy một số khai thác nhất định.

Kiểm tra báo cáo và khắc phục

Đối với khách hàng của bạn, khía cạnh có giá trị nhất trong thử nghiệm của bạn là báo cáo bạn gửi để truyền đạt những phát hiện của bạn về hiệu quả của các biện pháp kiểm soát bảo mật API của họ. Báo cáo phải nêu rõ các lỗ hổng mà bạn đã phát hiện ra trong quá trình thử nghiệm và giải thích cho khách hàng cách họ có thể thực hiện khắc phục để cải thiện tính bảo mật cho các API của họ.

Điều cuối cùng cần kiểm tra khi xác định phạm vi là liệu nhà cung cấp API có muốn thử nghiệm khắc phục hay không. Sau khi khách hàng nhận được báo cáo, họ nên cố gắng khắc phục các lỗ hổng API của mình. Thực hiện kiểm tra lại các phát hiện trước đó sẽ xác nhận rằng các lỗ hổng đã được khắc phục thành công.

Việc kiểm tra lại có thể chỉ thăm dò các điểm yếu hoặc có thể là kiểm tra lại toàn bộ để xem liệu có bất kỳ thay đổi nào được áp dụng cho API đưa ra các điểm yếu mới hay không.

Lưu ý về phạm vi tiền thưởng lỗi

Nếu bạn muốn hack chuyên nghiệp, một cách tuyệt vời để đặt chân vào cửa là trở thành một thợ săn tiền thưởng lỗi. Các tổ chức như BugCrowd và HackerOne đã tạo ra các nền tảng giúp mọi người dễ dàng tạo tài khoản và bắt đầu săn bắt. Ngoài ra, nhiều tổ chức chạy các chương trình tiền thưởng lỗi của riêng họ, bao gồm Google, Microsoft, Apple, Twitter và GitHub. Các chương trình này bao gồm nhiều tiền thưởng lỗi API, nhiều trong số đó có các ưu đãi bổ sung. Ví dụ: chương trình tiền thưởng lỗi của Files.com được lưu trữ trên BugCrowd bao gồm tiền thưởng dành riêng cho API, như thể hiện trong Hình 0-2.

Considering the higher business impact of issues affecting the following targets, we are offering a 10% bonus on valid submissions (severity P2-P4) for them:

- app.files.com
- your-assigned-subdomain.files.com
- REST API

Target	P1	P2	P3	P4
your-assigned-subdomain.files.com	up to \$10,000	\$2,500	\$500	\$100
Files.com Desktop Application for Windows or Mac	up to \$2,000	\$1,000	\$200	\$100
app.files.com	up to \$10,000	\$2,500	\$500	\$100
www.files.com	up to \$2,000	\$1,000	\$200	\$100
Files.com REST API	up to \$10,000	\$2,500	\$500	\$100

Hình 0-2: Chương trình tiền thưởng lỗi của Files.com trên BugCrowd, một trong nhiều chương trình khuyến khích các phát hiện liên quan đến API

Trong các chương trình tiền thưởng lỗi, bạn nên chú ý đến hai hợp đồng: điều khoản dịch vụ cho nhà cung cấp tiền thưởng lỗi và phạm vi của chương trình. Vì phạm vi một trong hai hợp đồng này không chỉ có thể dẫn đến việc bị nhà cung cấp tiền thưởng lỗi cấm mà còn gặp rắc rối pháp lý. Điều khoản dịch vụ của nhà cung cấp tiền thưởng sẽ chứa thông tin quan trọng về việc kiểm tiền thưởng, báo cáo kết quả và mối quan hệ giữa nhà cung cấp tiền thưởng, người thử nghiệm, nhà nghiên cứu và tin tặc tham gia và mục tiêu.

Phạm vi sẽ trang bị cho bạn các API mục tiêu, mô tả, số tiền thưởng, quy tắc tương tác, yêu cầu báo cáo và các hạn chế. Đối với tiền thưởng lỗi API, phạm vi thưởng sẽ bao gồm tài liệu API hoặc liên kết đến tài liệu. Bảng 0-1 liệt kê một số cân nhắc chính về tiền thưởng lỗi mà bạn nên hiểu trước khi thử nghiệm.

Bảng 0-1: Cân nhắc kiểm tra tiền thưởng lỗi

mục tiêu	URL được phê duyệt để thử nghiệm và nhận phần thưởng. Hãy chú ý đến các tên miền phụ được liệt kê, vì một số có thể nằm ngoài phạm vi.
Điều khoản tiết lộ Các quy tắc liên quan đến khả năng công bố phát hiện của bạn.	
Loại trừ các URL bị loại trừ khỏi thử nghiệm và phần thưởng.	
Hạn chế thử nghiệm	Hạn chế về các loại lỗ hổng mà tổ chức sẽ khen thưởng. Thông thường, bạn phải có khả năng chứng minh rằng phát hiện của bạn có thể được tận dụng trong một cuộc tấn công trong thế giới thực bằng cách cung cấp bằng chứng khai thác.
Hợp pháp	Các quy định và luật bổ sung của chính phủ áp dụng tùy theo vị trí của tổ chức, khách hàng và trung tâm dữ liệu.

Nếu bạn chưa quen với việc săn lỗi, tôi khuyên bạn nên xem BugCrowd University, nơi có video giới thiệu và trang dành riêng cho thử nghiệm bảo mật API của Sadako (<https://www.bugcrowd.com/resources/webinars/api-bảo-mật-thử-nghiệm-cho-tin-tặc>). Ngoài ra, hãy xem Bug Bounty Bootcamp (Không

Starch Press, 2021), đây là một trong những tài nguyên tốt nhất hiện có để giúp bạn bắt đầu nhận tiền thưởng lỗi. Nó thậm chí còn có một chương về hack API!

Hãy chắc chắn rằng bạn hiểu các phần thưởng tiềm năng, nếu có, của từng loại lỗi hỏng trước khi bạn dành thời gian và công sức cho nó. Ví dụ: tôi đã thấy tiền thưởng lỗi được yêu cầu khai thác hợp lệ giới hạn tỷ lệ mà máy chủ tiền thưởng lỗi coi là thư rác. Xem lại các lần gửi tiết lộ trước đây để xem liệu tổ chức có hiểu chiến hoặc không muốn trả tiền cho những gì có vẻ như là lần gửi hợp lệ hay không. Ngoài ra, hãy tập trung vào những lần gửi thành công đã nhận được tiền thưởng. Người săn lỗi đã cung cấp loại bằng chứng nào và họ đã báo cáo phát hiện của mình như thế nào để tổ chức dễ dàng coi lỗi đó là hợp lệ?

Bản tóm tắt

Trong chương này, tôi đã xem xét các thành phần của phạm vi kiểm tra bảo mật API. Phát triển phạm vi tương tác API sẽ giúp bạn hiểu phương pháp thử nghiệm để triển khai cũng như mức độ tương tác. Bạn cũng nên hiểu rõ những gì có thể và không thể kiểm tra cũng như những công cụ và kỹ thuật nào sẽ được sử dụng trong quá trình tương tác. Nếu các khía cạnh thử nghiệm đã được giải thích rõ ràng và bạn thử nghiệm trong phạm vi các thông số kỹ thuật đó, bạn sẽ sẵn sàng tham gia thử nghiệm bảo mật API thành công.

Trong chương tiếp theo, tôi sẽ đề cập đến chức năng của ứng dụng web mà bạn cần hiểu để biết các API web hoạt động như thế nào. Nếu bạn đã hiểu kiến thức cơ bản về ứng dụng web, hãy chuyển sang Chương 2, nơi tôi trình bày về giải pháp kỹ thuật của API.

1

CÁCH ỨNG DỤNG WEB HOẠT ĐỘNG



Trước khi bạn có thể hack API, bạn phải hiểu các công nghệ hỗ trợ chúng. Trong chương này, tôi sẽ trình bày mọi thứ bạn cần biết về ứng dụng web, bao gồm các khía cạnh cơ bản của Giao thức truyền tải siêu văn bản (HTTP), xác thực và ủy quyền cũng như cơ sở dữ liệu máy chủ web thông thường. Vì các API web được cung cấp bởi các công nghệ này nên việc hiểu những điều cơ bản này sẽ giúp bạn chuẩn bị cho việc sử dụng và hack các API.

Khái niệm cơ bản về ứng dụng web

Các ứng dụng web hoạt động dựa trên mô hình máy khách/máy chủ: trình duyệt web của bạn, máy khách, tạo các yêu cầu về tài nguyên và gửi chúng đến các máy tính được gọi là máy chủ web. Đổi lại, các máy chủ web này gửi tài nguyên đến

các client qua mạng. Thuật ngữ ứng dụng web đề cập đến phần mềm đang chạy trên máy chủ web, chẳng hạn như Wikipedia, LinkedIn, Twitter, Gmail, GitHub và Reddit.

Đặc biệt, các ứng dụng web được thiết kế để tương tác với người dùng cuối. Trong khi các trang web thường ở chế độ chỉ đọc và cung cấp giao tiếp một chiều từ máy chủ web đến máy khách, thì các ứng dụng web cho phép truyền thông theo cả hai hướng, từ máy chủ đến máy khách và từ máy khách đến máy chủ. Ví dụ, Reddit là một ứng dụng web hoạt động như một nguồn cấp tin tức về thông tin lan truyền trên internet. Nếu nó chỉ đơn thuần là một trang web, khách truy cập sẽ được đút cho ăn bất kỳ nội dung nào mà tổ chức đăng sau trang web cung cấp. Thay vào đó, Reddit cho phép người dùng tương tác với thông tin trên trang web bằng cách đăng, ủng hộ, không ủng hộ, bình luận, chia sẻ, báo cáo các bài đăng xấu và tùy chỉnh nguồn cấp tin tức của họ với các subreddits mà họ muốn xem. Những tính năng này phân biệt Reddit với một trang web tĩnh.

Để người dùng cuối bắt đầu sử dụng ứng dụng web, một cuộc trò chuyện phải diễn ra giữa trình duyệt web và máy chủ web. Người dùng cuối bắt đầu cuộc trò chuyện này bằng cách nhập URL vào thanh địa chỉ trình duyệt của họ. Trong phần này, chúng ta sẽ thảo luận về những gì xảy ra tiếp theo.

URL

Có thể bạn đã biết rằng địa chỉ định vị tài nguyên thống nhất (URL) là địa chỉ được sử dụng để định vị các tài nguyên duy nhất trên internet. URL này bao gồm một số thành phần mà bạn sẽ thấy hữu ích khi hiểu khi tạo các yêu cầu API trong các chương sau. Tất cả các URL bao gồm giao thức được sử dụng, tên máy chủ, cổng, đường dẫn và bất kỳ tham số truy vấn nào:

Giao thức://hostname[:port number]/[path]/[?query][parameters]

Giao thức là tập hợp các quy tắc mà máy tính sử dụng để giao tiếp. Các giao thức chính được sử dụng trong URL là HTTP/HTTPS cho các trang web và FTP để truyền tệp.

Cổng, một số chỉ định kênh liên lạc, chỉ được bao gồm nếu máy chủ không tự động giải quyết yêu cầu đến cổng thích hợp. Thông thường, giao tiếp HTTP diễn ra qua cổng 80. HTTPS, phiên bản được mã hóa của HTTP, sử dụng cổng 443 và FTP sử dụng cổng 21. Để truy cập ứng dụng web được lưu trữ trên một cổng không chuẩn, bạn có thể bao gồm số cổng trong URL, như vậy: `https://www.example.com:8443`. (Cổng 8080 và 8443 lần lượt là các cổng thay thế phổ biến cho HTTP và HTTPS.)

Đường dẫn thư mục tệp trên máy chủ web trở đến vị trí của các trang web và tệp được chỉ định trong URL. Đường dẫn được sử dụng trong URL giống như đường dẫn tệp được sử dụng để định vị tệp trên máy tính.

Truy vấn là một phần tùy chọn của URL được sử dụng để thực hiện chức năng như tìm kiếm, lọc và dịch ngôn ngữ của thông tin được yêu cầu. Nhà cung cấp ứng dụng web cũng có thể sử dụng các chuỗi truy vấn để theo dõi một số thông tin nhất định, chẳng hạn như URL đã giới thiệu bạn đến trang web, ID phiên hoặc email của bạn. Nó bắt đầu bằng một dấu chấm hỏi và chứa một chuỗi mà máy chủ được lập trình để xử lý. Cuối cùng, các tham số truy vấn là các giá trị mô tả những gì nên được thực hiện với truy vấn đã cho. Ví dụ: tham số truy vấn `lang=en` sau trang truy vấn?

có thể cho máy chủ web biết rằng nó sẽ cung cấp trang được yêu cầu bằng tiếng Anh. Các tham số này bao gồm một chuỗi khác sẽ được xử lý bởi máy chủ web. Một truy vấn có thể chứa nhiều tham số được phân tách bằng dấu và (&).

Để làm cho thông tin này cụ thể hơn, hãy xem xét URL `https://twitter.com/search?q=hacking&src=typed_query`. Trong ví dụ này, giao thức là `https`, tên máy chủ là `twitter.com`, đường dẫn là tìm kiếm, truy vấn là `?q` (viết tắt của `query`), tham số truy vấn là `hack` và `src=typed_query` là tham số theo dõi. URL này được tạo tự động bất cứ khi nào bạn nhấp vào thanh tìm kiếm trong ứng dụng web Twitter, nhập cụm từ tìm kiếm “hack” và nhấn ENTER. Trình duyệt được lập trình để tạo URL theo cách mà máy chủ web Twitter có thể hiểu được và nó thu thập một số thông tin theo dõi dưới dạng tham số `src`. Máy chủ web sẽ nhận được yêu cầu về nội dung hack và phản hồi với thông tin liên quan đến hack.

Yêu cầu HTTP

Khi người dùng cuối điều hướng đến một URL bằng trình duyệt web, trình duyệt sẽ tự động tạo yêu cầu HTTP cho tài nguyên. Tài nguyên này là thông tin được yêu cầu—thường là các tệp tạo nên một trang web. Yêu cầu được định tuyến qua internet hoặc mạng đến máy chủ web, nơi nó được xử lý ban đầu. Nếu yêu cầu được hình thành đúng cách, máy chủ web sẽ chuyển yêu cầu đến ứng dụng web.

Liệt kê 1-1 hiển thị các thành phần của yêu cầu HTTP được gửi khi xác thực tới `twitter.com`.

```
POST /session2 HTTP/1.13
Máy chủ: twitter.com4
Tác nhân người dùng: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Chấp nhận: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Ngôn ngữ chấp nhận: en-US,en;q=0.5
Chấp nhận mã hóa: gzip, giảm phát
Loại nội dung: ứng dụng/x-www-form-urlencoded
Độ dài nội dung: 444
Cookie: _personalization_id=GA1.2.1451399206.1606701545; dnt=1;

username_or_email%5D=hAPI_hacker&Spassword%5D=NotMyPassword6%217
```

Liệt kê 1-1: Một yêu cầu HTTP để xác thực với `twitter.com`

Các yêu cầu HTTP bắt đầu với phương thức 1, đường dẫn của tài nguyên được yêu cầu 2 và phiên bản giao thức 3. Phương thức, được mô tả trong phần “Phương thức HTTP” ở phần sau của chương này, cho máy chủ biết bạn muốn làm gì. Trong trường hợp này, bạn sử dụng phương thức POST để gửi thông tin đăng nhập của mình đến máy chủ. Đường dẫn có thể chứa toàn bộ URL, đường dẫn tuyệt đối hoặc đường dẫn tương đối của tài nguyên. Trong yêu cầu này, đường dẫn `/sessions` chỉ định trang xử lý các yêu cầu xác thực Twitter.

Các yêu cầu bao gồm một số tiêu đề, là các cặp khóa-giá trị truyền đạt thông tin cụ thể giữa máy khách và máy chủ web. Tiêu đề bắt đầu bằng tên của tiêu đề, theo sau là dấu hai chấm (:) và sau đó là giá trị

của tiêu đề. Tiêu đề Máy chủ 4 chỉ định máy chủ tên miền, twitter.com. Tiêu đề Tác nhân người dùng mô tả trình duyệt và hệ điều hành của khách hàng. Tiêu đề Chấp nhận mô tả loại nội dung mà trình duyệt có thể chấp nhận từ ứng dụng web trong phản hồi. Không phải tất cả các tiêu đề đều được yêu cầu, máy khách và máy chủ có thể bao gồm những tiêu đề khác không được hiển thị ở đây, tùy thuộc vào yêu cầu. Ví dụ: yêu cầu này bao gồm một tiêu đề Cookie, được sử dụng giữa máy khách và máy chủ để thiết lập kết nối trạng thái (thêm về điều này ở phần sau của chương). Nếu bạn muốn tìm hiểu thêm về tất cả các tiêu đề khác nhau, hãy xem trang dành cho nhà phát triển của Mozilla về các tiêu đề (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>).

Bất kỳ thứ gì bên dưới tiêu đề là nội dung thư, là thông tin mà người yêu cầu đang cố gắng xử lý bởi ứng dụng web. Trong trường hợp này, phần thân bao gồm tên người dùng 5 và mật khẩu 6 được sử dụng để xác thực tài khoản Twitter. Một số ký tự trong nội dung được mã hóa tự động. Ví dụ: dấu chấm than (!) được mã hóa thành %21. Mã hóa ký tự là một cách mà ứng dụng web có thể xử lý an toàn các ký tự có thể gây ra sự cố.

Phản hồi HTTP

Sau khi máy chủ web nhận được yêu cầu HTTP, nó sẽ xử lý và phản hồi yêu cầu. Loại phản hồi tùy thuộc vào tính khả dụng của tài nguyên, quyền truy cập tài nguyên của người dùng, tình trạng của máy chủ web và các yếu tố khác. Ví dụ, Liệt kê 1-2 hiển thị phản hồi cho yêu cầu trong Liệt kê 1-1.

HTTP/1.1 302 Tìm thấy2

chính sách bảo mật nội dung: default-src 'none'; kết nối-src 'tự'

địa điểm: <https://twitter.com/>

pragma: không có bộ đệm

máy chủ: tsa_a

đặt cookie: auth_token=8ff3f2424f8ac1c4ec635b4adb52cddf28ec18b8; Tuổi tối đa=157680000; Hết hạn=Thứ

Hai, ngày 01 tháng 12 năm 2025 16:42:40 GMT; Đường dẫn=/; Tên miền=.twitter.com; Chắc chắn; Chỉ HTTP; SameSite=Không có

```
<html><body>Bạn đang bị <a href="https://twitter.com/">chuyển hướng</a>.</body></html>
```

Liệt kê 1-2: Một ví dụ về phản hồi HTTP khi xác thực với twitter.com

Trước tiên, máy chủ web phản hồi với phiên bản giao thức đang sử dụng (trong trường hợp này là HTTP/1.1). HTTP 1.1 hiện là phiên bản tiêu chuẩn của HTTP được sử dụng. Mã trạng thái và thông báo trạng thái 2, được thảo luận chi tiết hơn trong phần tiếp theo, là 302 Đã tìm thấy. Mã phản hồi 302 cho biết rằng khách hàng đã được xác thực thành công và sẽ được chuyển hướng đến trang đích mà khách hàng được phép truy cập.

Lưu ý rằng, giống như các tiêu đề yêu cầu HTTP, có các tiêu đề phản hồi HTTP. Các tiêu đề phản hồi HTTP thường cung cấp cho trình duyệt các hướng dẫn để xử lý các yêu cầu về phản hồi và bảo mật. set-cookie tiêu đề là một dấu hiệu khác cho thấy yêu cầu xác thực đã thành công, vì máy chủ web đã phát hành cookie bao gồm auth_token,

mà khách hàng có thể sử dụng để truy cập các tài nguyên nhất định. Nội dung thông báo phản hồi sẽ theo dòng trống sau tiêu đề phản hồi. Trong trường hợp này, máy chủ web đã gửi một thông báo HTML cho biết rằng máy khách đang được chuyển hướng đến một trang web mới.

Yêu cầu và phản hồi mà tôi đã trình bày ở đây minh họa một cách phổ biến trong đó ứng dụng web hạn chế quyền truy cập vào tài nguyên của nó thông qua việc sử dụng xác thực và ủy quyền. Xác thực web là quá trình chứng minh danh tính của bạn với máy chủ web. Các hình thức xác thực phổ biến bao gồm cung cấp mật khẩu, mã thông báo hoặc thông tin sinh trắc học (chẳng hạn như dấu vân tay). Nếu máy chủ web chấp nhận yêu cầu xác thực, nó sẽ phản hồi bằng cách cung cấp cho người dùng được xác thực quyền truy cập vào một số tài nguyên nhất định. Trong Liệt kê 1-1, chúng ta đã thấy một yêu cầu xác thực tới máy chủ web Twitter đã gửi tên người dùng và mật khẩu bằng yêu cầu POST. Máy chủ web Twitter đã phản hồi yêu cầu xác thực thành công với 302 Found (trong Liệt kê 1-2). Phiên auth_token trong tiêu đề set-cookie được phép truy cập vào các tài nguyên được liên kết với tài khoản Twitter hAPI_hacker.

LƯU Ý

Lượng HTTP được gửi ở dạng văn bản rõ ràng, nghĩa là nó không bị ẩn hoặc mã hóa theo bất kỳ cách nào.

Bất kỳ ai chặn yêu cầu xác thực trong Liệt kê 1-1 đều có thể đọc tên người dùng và mật khẩu. Để bảo vệ thông tin nhạy cảm, các yêu cầu giao thức HTTP có thể được mã hóa bằng Bảo mật tầng vận chuyển (TLS) để tạo giao thức HTTPS.

Mã trạng thái HTTP

Khi máy chủ web phản hồi yêu cầu, nó sẽ đưa ra mã trạng thái phản hồi cùng với thông báo phản hồi. Mã phản hồi báo hiệu cách máy chủ web đã xử lý yêu cầu. Ở cấp độ cao, mã phản hồi xác định xem khách hàng sẽ được phép hay bị từ chối truy cập vào tài nguyên. Nó cũng có thể chỉ ra rằng tài nguyên không tồn tại, có sự cố với máy chủ web hoặc yêu cầu tài nguyên đã cho dẫn đến việc được chuyển hướng đến một vị trí khác.

Liệt kê 1-3 và 1-4 minh họa sự khác biệt giữa phản hồi 200 và một phản hồi 404 tương ứng.

```
HTTP/1.1 200 OK
Máy chủ: tsa_a
Độ dài nội dung: 6552

<!DOCTYPE html>
<html dir="ltr" lang="vi">
[.]
```

Liệt kê 1-3: Một ví dụ về phản hồi 200

```
HTTP/1.1 404 Không tìm thấy
Máy chủ: tsa_a
Độ dài nội dung: 0
```

Liệt kê 1-4: Một ví dụ về phản hồi 404

Phản hồi 200 OK sẽ cung cấp cho khách hàng quyền truy cập vào yêu cầu tài nguyên, trong khi phản hồi 404 Not Found sẽ cung cấp cho khách hàng một số loại trang lỗi hoặc trang trống vì không tìm thấy tài nguyên được yêu cầu.

Vì các API web chủ yếu hoạt động bằng HTTP nên điều quan trọng là phải hiểu các loại mã phản hồi mà bạn sẽ nhận được từ máy chủ web, như được trình bày chi tiết trong Bảng 1-1. Để biết thêm thông tin về các mã phản hồi riêng lẻ hoặc về công nghệ web nói chung, hãy xem Tài liệu web của Mozilla (<https://developer.mozilla.org/en-US/docs/Web/HTTP>). Mozilla đã cung cấp rất nhiều thông tin hữu ích về giải phẫu của các ứng dụng web.

Bảng 1-1: Phạm vi mã phản hồi HTTP

Mã phản hồi	Loại phản hồi	Sự miêu tả
100 giây	Phản hồi dựa trên thông tin	Các phản hồi trong khoảng 100 thường liên quan đến một số loại cập nhật trạng thái xử lý liên quan đến yêu cầu.
những năm 200	phản hồi thành công	Phản hồi trong những năm 200 cho thấy một yêu cầu thành công và được chấp nhận.
những năm 300	chuyển hướng	Phản hồi trong những năm 300 là thông báo chuyển hướng. Điều này thường thấy đối với một yêu cầu tự động chuyển hướng bạn đến chỉ mục/trang chủ hoặc khi bạn yêu cầu một trang từ cổng 80 HTTP đến cổng 443 cho HTTPS.
những năm 400	lỗi máy khách	Các phản hồi trong độ tuổi 400 chỉ ra rằng đã có điều gì đó không ổn từ góc độ khách hàng. Đây thường là loại phản hồi bạn sẽ nhận được nếu bạn đã yêu cầu một trang không tồn tại, nếu có thời gian chờ trong phản hồi hoặc khi bạn bị cấm xem trang.
những năm 500	Lỗi máy chủ	Phản hồi trong những năm 500 là dấu hiệu cho thấy có gì đó không ổn với máy chủ. Chúng bao gồm lỗi máy chủ nội bộ, dịch vụ không khả dụng và phương thức yêu cầu không được công nhận.

Phương thức HTTP

Các phương thức HTTP yêu cầu thông tin từ máy chủ web. Còn được gọi là động từ HTTP, các phương thức HTTP bao gồm GET, PUT, POST, HEAD, PATCH, OPTIONS, TRACE và DELETE.

GET và POST là hai phương thức yêu cầu được sử dụng phổ biến nhất. Các Yêu cầu GET được sử dụng để lấy tài nguyên từ máy chủ web và yêu cầu POST được sử dụng để gửi dữ liệu đến máy chủ web. Bảng 1-2 cung cấp thêm thông tin chuyên sâu về từng phương thức yêu cầu HTTP.

Bảng 1-2: Các phương thức HTTP

Phương pháp	Mục đích
LẤY	Yêu cầu GET cố gắng thu thập tài nguyên từ máy chủ web. Đây có thể là bất kỳ tài nguyên nào, bao gồm trang web, dữ liệu người dùng, video, địa chỉ, v.v. Nếu yêu cầu thành công, máy chủ sẽ cung cấp tài nguyên; nếu không, máy chủ sẽ đưa ra phản hồi giải thích lý do tại sao nó không thể lấy tài nguyên được yêu cầu.
BUU KIẾN	Các yêu cầu POST gửi dữ liệu có trong phần thân của yêu cầu tới máy chủ web. Điều này có thể bao gồm hồ sơ khách hàng, yêu cầu chuyển tiền từ tài khoản này sang tài khoản khác và cập nhật trạng thái chẳng hạn. Nếu một ứng dụng phụ thực hiện cùng một yêu cầu POST nhiều lần, thì máy chủ sẽ tạo ra nhiều kết quả.
ĐẶT	Yêu cầu PUT hướng dẫn máy chủ web lưu trữ dữ liệu đã gửi theo URL được yêu cầu. PUT chủ yếu được sử dụng để gửi tài nguyên đến máy chủ web. Nếu một máy chủ chấp nhận yêu cầu PUT, nó sẽ thêm tài nguyên hoặc thay thế hoàn toàn tài nguyên hiện có. Nếu yêu cầu PUT thành công, một URL mới sẽ được tạo. Nếu cùng một yêu cầu PUT được gửi lại, kết quả sẽ giữ nguyên.
CÁI ĐẦU	Các yêu cầu HEAD tương tự như các yêu cầu GET, ngoại trừ chúng chỉ yêu cầu các tiêu đề HTTP, ngoại trừ nội dung thứ. Yêu cầu này là một cách nhanh chóng để lấy thông tin về trạng thái máy chủ và để xem liệu một URL nhất định có hoạt động hay không.
VÁ	Các yêu cầu PATCH được sử dụng để cập nhật một phần tài nguyên với dữ liệu đã gửi. Các yêu cầu PATCH chỉ khả dụng nếu phản hồi HTTP bao gồm tiêu đề Chấp nhận bản vá.
TÙY CHỌN	Yêu cầu TÙY CHỌN là cách để khách hàng xác định tất cả các yêu cầu các phương pháp được phép từ một máy chủ web nhất định. Nếu máy chủ web phản hồi một yêu cầu TÙY CHỌN, thì nó sẽ phản hồi với tất cả các tùy chọn yêu cầu được phép.
DẤU VẾT	Các yêu cầu TRACE chủ yếu được sử dụng để gỡ lỗi đầu vào được gửi từ máy khách đến máy chủ. TRACE yêu cầu máy chủ lặp lại yêu cầu ban đầu của máy khách, điều này có thể tiết lộ rằng một cơ chế đang thay đổi yêu cầu của máy khách trước khi nó được máy chủ xử lý.
CONNECT	Yêu cầu CONNECT bắt đầu kết nối mạng hai chiều. Khi được phép, yêu cầu này sẽ tạo một đường hầm proxy giữa trình duyệt và máy chủ web.
XÓA BỎ	Các yêu cầu XÓA yêu cầu máy chủ xóa một tài nguyên nhất định.

Một số phương thức là idempotent, có nghĩa là chúng có thể được sử dụng để gửi cùng một yêu cầu nhiều lần mà không làm thay đổi trạng thái của tài nguyên trên máy chủ web. Ví dụ: nếu bạn thực hiện thao tác bật đèn thì đèn sẽ sáng. Khi công tắc đã được bật và bạn thử bật lại công tắc, công tắc vẫn bật—không có gì thay đổi. NHẬN, HEAD, PUT, TÙY CHỌN và XÓA là bình thường.

Mặt khác, các phương thức không idempotent có thể tự động thay đổi kết quả của một tài nguyên trên một máy chủ. Các phương thức không bình thường bao gồm POST, PATCH và CONNECT. POST là phương pháp được sử dụng phổ biến nhất để thay đổi tài nguyên máy chủ web. POST được sử dụng để tạo tài nguyên mới trên máy chủ web, vì vậy nếu yêu cầu POST được gửi 10 lần thì sẽ có 10 tài nguyên mới trên máy chủ web. Ngược lại, nếu một phương thức idempotent như PUT, thường được sử dụng để cập nhật tài nguyên, được yêu cầu 10 lần, thì một tài nguyên sẽ bị ghi đè 10 lần.

XÓA cũng là tạm thời, bởi vì nếu yêu cầu xóa tài nguyên được gửi 10 lần, tài nguyên sẽ chỉ bị xóa một lần. Những lần tiếp theo sẽ không có chuyện gì xảy ra. Các API web thường sẽ chỉ sử dụng POST, GET, PUT, DELETE, với POST là các phương thức không bình thường.

HTTP có trạng thái và không trạng thái

HTTP là một giao thức không trạng thái, nghĩa là máy chủ không theo dõi thông tin giữa các yêu cầu. Tuy nhiên, để người dùng có trải nghiệm lâu dài và nhất quán với ứng dụng web, máy chủ web cần ghi nhớ điều gì đó về phiên HTTP với máy khách đó. Ví dụ: nếu người dùng đăng nhập vào tài khoản của họ và thêm một số mặt hàng vào giỏ hàng, ứng dụng web cần theo dõi trạng thái giỏ hàng của người dùng cuối.

Mặt khác, mỗi khi người dùng điều hướng đến một trang web khác, giỏ hàng sẽ lại trống.

Một kết nối có trạng thái cho phép máy chủ theo dõi hành động, hồ sơ, hình ảnh, sở thích, v.v. Các kết nối trạng thái sử dụng các tệp văn bản nhỏ, được gọi là cookie, để lưu trữ thông tin ở phía máy khách. Cookie có thể lưu trữ cài đặt trang web cụ thể, cài đặt bảo mật và thông tin liên quan đến xác thực. Trong khi đó, máy chủ thường lưu trữ thông tin trên chính nó, trong bộ đệm hoặc trên cơ sở dữ liệu phụ trợ. Để tiếp tục phiên của mình, các trình duyệt bao gồm các cookie được lưu trữ trong các yêu cầu tới máy chủ và khi tấn công các ứng dụng web, kẻ tấn công có thể mạo danh người dùng cuối bằng cách đánh cắp hoặc giả mạo cookie của họ.

Duy trì kết nối trạng thái với máy chủ có các giới hạn về quy mô.

Khi một trạng thái được duy trì giữa máy khách và máy chủ, mối quan hệ đó chỉ tồn tại giữa trình duyệt cụ thể và máy chủ được sử dụng khi trạng thái được tạo. Giả sử, nếu người dùng chuyển từ sử dụng trình duyệt trên một máy tính sang sử dụng trình duyệt trên thiết bị di động của họ, thì máy khách sẽ cần phải xác thực lại và tạo trạng thái mới với máy chủ. Ngoài ra, các kết nối trạng thái yêu cầu máy khách liên tục gửi yêu cầu đến máy chủ. Thách thức bắt đầu nảy sinh khi nhiều máy khách đang duy trì trạng thái với cùng một máy chủ.

Máy chủ chỉ có thể xử lý nhiều kết nối có trạng thái như tài nguyên máy tính của nó cho phép. Điều này được giải quyết dễ dàng hơn nhiều bằng các ứng dụng không trạng thái.

Giao tiếp phi trạng thái loại bỏ nhu cầu về tài nguyên máy chủ

cần thiết để quản lý phiên. Trong giao tiếp không trạng thái, máy chủ không lưu trữ thông tin phiên và mọi yêu cầu không trạng thái được gửi phải chứa tất cả thông tin cần thiết để máy chủ web nhận ra rằng người yêu cầu được phép truy cập các tài nguyên đã cho. Các yêu cầu không trạng thái này có thể bao gồm khóa hoặc một số dạng tiêu đề ủy quyền để duy trì trải nghiệm tương tự như trải nghiệm của kết nối trạng thái. Các kết nối không lưu trữ dữ liệu phiên trên máy chủ ứng dụng web; thay vào đó, họ tận dụng cơ sở dữ liệu phía sau.

Trong ví dụ về giỏ hàng của chúng tôi, một ứng dụng không trạng thái có thể theo dõi nội dung trong giỏ hàng của người dùng bằng cách cập nhật cơ sở dữ liệu hoặc bộ đệm dựa trên các yêu cầu có chứa một mã thông báo bất định. Trải nghiệm của người dùng cuối sẽ giống nhau, nhưng cách máy chủ web xử lý yêu cầu thì hơi khác một chút. Vì trạng thái xuất hiện của chúng được duy trì và khách hàng phát hành

mọi thứ cần thiết trong một yêu cầu nhất định, các ứng dụng không trạng thái có thể mở rộng quy mô mà không lo mất thông tin trong một kết nối có trạng thái. Thay vào đó, bất kỳ số lượng máy chủ nào cũng có thể được sử dụng để xử lý các yêu cầu miễn là tất cả thông tin cần thiết được bao gồm trong yêu cầu và thông tin đó có thể truy cập được trên cơ sở dữ liệu phụ trợ.

Khi hack API, kẻ tấn công có thể mạo danh người dùng cuối bằng cách đánh cắp hoặc giả mạo mã thông báo của họ. Giao tiếp API là không trạng thái-một chủ đề tôi sẽ khám phá chi tiết hơn trong chương tiếp theo.

Cơ sở dữ liệu máy chủ web

Cơ sở dữ liệu cho phép máy chủ lưu trữ và nhanh chóng cung cấp tài nguyên cho máy khách. Ví dụ: bất kỳ nền tảng truyền thông xã hội nào cho phép bạn tải lên các cập nhật trạng thái, ảnh và video chắc chắn đang sử dụng cơ sở dữ liệu để lưu tất cả nội dung đó. Nền tảng truyền thông xã hội có thể tự duy trì các cơ sở dữ liệu đó; cách khác, cơ sở dữ liệu có thể được cung cấp cho nền tảng dưới dạng dịch vụ.

Thông thường, một ứng dụng web sẽ lưu trữ tài nguyên người dùng bằng cách chuyển tài nguyên từ mã giao diện người dùng sang cơ sở dữ liệu phụ trợ. Giao diện người dùng của ứng dụng web, là một phần của ứng dụng web mà người dùng tương tác với, xác định giao diện của nó và bao gồm các nút, liên kết, video và phông chữ của nó.

Mã frontend thường bao gồm HTML, CSS và JavaScript. Ngoài ra, giao diện người dùng có thể bao gồm các khung ứng dụng web như AngularJS, ReactJS và Bootstrap, v.v. Phần phụ trợ bao gồm các công nghệ mà phần đầu cần để hoạt động. Nó bao gồm máy chủ, ứng dụng và bất kỳ cơ sở dữ liệu nào. Các ngôn ngữ lập trình phụ trợ bao gồm JavaScript, Python, Ruby, Go, Golang, PHP, Java, C# và Perl, v.v.

Trong một ứng dụng web an toàn, sẽ không có tương tác trực tiếp giữa người dùng và cơ sở dữ liệu phụ trợ. Truy cập trực tiếp vào cơ sở dữ liệu sẽ loại bỏ một lớp bảo vệ và mở cơ sở dữ liệu cho các cuộc tấn công bổ sung.

Khi hiển thị các công nghệ cho người dùng cuối, nhà cung cấp ứng dụng web mở rộng khả năng tấn công của họ, một số liệu được gọi là bề mặt tấn công.

Hạn chế quyền truy cập trực tiếp vào cơ sở dữ liệu sẽ thu nhỏ kích thước của bề mặt tấn công.

Các ứng dụng web hiện đại sử dụng cơ sở dữ liệu SQL (quan hệ) hoặc cơ sở dữ liệu NoSQL (không quan hệ). Biết được sự khác biệt giữa cơ sở dữ liệu SQL và NoSQL sẽ giúp bạn điều chỉnh các cuộc tấn công tiêm API của mình sau này.

SQL

Cơ sở dữ liệu ngôn ngữ truy vấn có cấu trúc (SQL) là cơ sở dữ liệu quan hệ trong đó dữ liệu được tổ chức trong bảng. Các hàng của bảng, được gọi là bản ghi, xác định loại dữ liệu, chẳng hạn như tên người dùng, địa chỉ email hoặc cấp đặc quyền. Các cột của nó là các thuộc tính của dữ liệu và có thể bao gồm tất cả các tên người dùng, địa chỉ email và cấp đặc quyền khác nhau. Trong Bảng 1-3 đến 1-5, UserID, Tên người dùng, Email và Đặc quyền là các loại dữ liệu. Các hàng là dữ liệu cho bảng đã cho.

Bảng 1-3: Bảng người dùng quan hệ

Tên người dùng	tên tài khoản
111	hAPI_hacker
112	Scuttleph1sh
113	bí ẩn

Bảng 1-4: Bảng email quan hệ

Tên người dùng	E-mail
111	hapi_hacker@email.com
112	scuttleph1sh@email.com
113	bí_ảnhadow@email.com

Bảng 1-5: Bảng đặc quyền quan hệ

Tên người dùng	đặc quyền
111	quản trị viên
112	cộng sự
113	người dùng

Để truy xuất dữ liệu từ cơ sở dữ liệu SQL, một ứng dụng phải tạo một truy vấn SQL. Một truy vấn SQL điển hình để tìm khách hàng có số nhận dạng là 111 sẽ như sau:

```
CHỌN * TỪ Email WHERE UserID = 111;
```

Truy vấn này yêu cầu tất cả các bản ghi từ bảng Email có giá trị 111 trong cột UserID. CHỌN là một câu lệnh được sử dụng để lấy thông tin từ cơ sở dữ liệu, dấu hoa thị là ký tự đại diện sẽ chọn tất cả các cột trong bảng, TỪ được sử dụng để xác định bảng nào sẽ sử dụng và WHERE là mệnh đề được sử dụng để lọc cụ thể kết quả.

Có một số loại cơ sở dữ liệu SQL, nhưng chúng được truy vấn tương tự lớn. Cơ sở dữ liệu SQL bao gồm MySQL, Microsoft SQL Server, PostgreSQL, Oracle và MariaDB, trong số những cơ sở dữ liệu khác.

Trong các chương sau, tôi sẽ đề cập đến cách gửi các yêu cầu API để phát hiện các lỗ hổng tiềm ẩn, chẳng hạn như tiêm nhiễm SQL. SQL injection là một cuộc tấn công ứng dụng web cổ điển đã gây khó khăn cho các ứng dụng web trong hơn hai thập kỷ nhưng vẫn là một phương thức tấn công có thể có trong các API.

NoSQL

Cơ sở dữ liệu NoSQL, còn được gọi là cơ sở dữ liệu phân tán, không liên quan, nghĩa là chúng không tuân theo cấu trúc của cơ sở dữ liệu quan hệ. NoSQL

cơ sở dữ liệu thường là các công cụ nguồn mở xử lý dữ liệu phi cấu trúc và lưu trữ dữ liệu dưới dạng tài liệu. Thay vì các mối quan hệ, cơ sở dữ liệu NoSQL lưu trữ thông tin dưới dạng khóa và giá trị. Không giống như cơ sở dữ liệu SQL, mỗi loại cơ sở dữ liệu NoSQL sẽ có cấu trúc, chế độ truy vấn, lỗi hỏng và cách khai thác riêng. Đây là một truy vấn mẫu sử dụng MongoDB, công ty dẫn đầu thị phần hiện tại cho cơ sở dữ liệu NoSQL:

```
db.collection.find({"UserID": 111})
```

Trong ví dụ này, `db.collection.find()` là một phương pháp được sử dụng để tìm kiếm thông tin về `UserID` trong tài liệu với giá trị là 111. MongoDB sử dụng một số toán tử có thể hữu ích khi biết:

\$eq Khớp các giá trị bằng với một giá trị đã chỉ định

\$gt So khớp các giá trị lớn hơn một giá trị đã chỉ định

\$lt So khớp các giá trị nhỏ hơn một giá trị đã chỉ định

\$ne Khớp tất cả các giá trị không bằng một giá trị đã chỉ định

Các toán tử này có thể được sử dụng trong các truy vấn NoSQL để chọn và lọc thông tin nhất định trong một truy vấn. Ví dụ: chúng ta có thể sử dụng lệnh trước đó mà không cần biết `UserID` chính xác, như sau:

```
db.collection.find({"UserID": {$gt:110}})
```

Câu lệnh này sẽ tìm tất cả `UserID` lớn hơn 110. Hiểu

các toán tử này sẽ hữu ích khi tiến hành các cuộc tấn công NoSQL injection ở phần sau của cuốn sách này.

Cơ sở dữ liệu NoSQL bao gồm MongoDB, Couchbase, Cassandra, IBM Domino, Cơ sở dữ liệu Oracle NoSQL, Redis và Elasticsearch, trong số những cơ sở dữ liệu khác.

Cách API phù hợp với hình ảnh

Một ứng dụng web có thể trở nên mạnh mẽ hơn nếu nó có thể sử dụng sức mạnh của các ứng dụng khác. Giao diện lập trình ứng dụng (API) bao gồm một công nghệ tạo điều kiện giao tiếp giữa các ứng dụng riêng biệt. Đặc biệt, các API web cho phép giao tiếp giữa các máy dựa trên HTTP, cung cấp một phương thức chung để kết nối các ứng dụng khác nhau với nhau.

Khả năng này đã mở ra một thế giới cơ hội cho các nhà cung cấp ứng dụng, vì các nhà phát triển không còn phải là chuyên gia trong mọi khía cạnh của chức năng mà họ muốn cung cấp cho người dùng cuối của họ. Ví dụ: hãy xem xét một ứng dụng trình chiếu. Ứng dụng cần một bản đồ để giúp người lái xe điều hướng các thành phố, phương thức xử lý thanh toán và cách để người lái xe và khách hàng giao tiếp với nhau. Thay vì chuyên về từng chức năng khác nhau này, nhà phát triển có thể tận dụng API Google Maps cho chức năng lập bản đồ, API Stripe để xử lý thanh toán và API Twilio để truy cập tin nhắn SMS. Nhà phát triển có thể kết hợp các API này để tạo ra một ứng dụng hoàn toàn mới.

Tác động ngay lập tức của công nghệ này là gấp đôi. Đầu tiên, nó hợp lý hóa việc trao đổi thông tin. Bằng cách sử dụng HTTP, các API web có thể tận dụng các phương pháp, mã trạng thái và mối quan hệ máy khách/máy chủ được tiêu chuẩn hóa của giao thức, cho phép các nhà phát triển viết mã có thể tự động xử lý dữ liệu. Thứ hai, API cho phép các nhà cung cấp ứng dụng web chuyên môn hóa, vì họ không còn cần phải tạo mọi khía cạnh của ứng dụng web của mình nữa.

API là một công nghệ đáng kinh ngạc với tác động toàn cầu. Tuy nhiên, như bạn sẽ thấy trong các chương sau, chúng đã mở rộng đáng kể bề mặt tấn công của mọi ứng dụng sử dụng chúng trên internet.

Bản tóm tắt

Trong chương này, chúng ta đã đề cập đến các khía cạnh cơ bản của ứng dụng web. Nếu bạn hiểu các chức năng chung của các yêu cầu và phản hồi HTTP, xác thực/ủy quyền và cơ sở dữ liệu, bạn sẽ dễ dàng hiểu được các API web, bởi vì công nghệ cơ bản của các ứng dụng web là công nghệ cơ bản của các API web. Trong chương tiếp theo, chúng ta sẽ xem xét giải phẫu của các API.

Chương này nhằm trang bị cho bạn thông tin vừa đủ để trở nên nguy hiểm với tư cách là một hacker API, chứ không phải với tư cách là nhà phát triển hoặc kiến trúc ứng dụng. Nếu bạn muốn có thêm tài nguyên về các ứng dụng web, tôi khuyên bạn nên sử dụng *The Web Application Hackers Handbook* (Wiley, 2011), *Web Application Security* (O'Reilly, 2020), *Bảo mật web dành cho nhà phát triển* (No Starch Press, 2020) và *The Tangled Web* (No Starch Press, 2011).

2

GIẢI PHẪU CÁC API WEB



Hầu hết những gì người dùng bình thường biết về một ứng dụng web đến từ những gì họ có thể nhìn thấy và nhấp vào trong giao diện người dùng đồ họa (GUI) của trình duyệt web của họ. Dưới phần lớn, các API thực hiện phần lớn công việc. Đặc biệt, các API web cung cấp một cách để ứng dụng sử dụng chức năng và dữ liệu của các ứng dụng khác qua HTTP để cung cấp GUI ứng dụng web bằng hình ảnh, văn bản và video.

Chương này đề cập đến thuật ngữ API phổ biến, các loại, định dạng trao đổi dữ liệu và phương thức xác thực, sau đó liên kết thông tin này với một ví dụ: quan sát các yêu cầu và phản hồi được trao đổi trong quá trình tương tác với API của Twitter.

Cách API Web hoạt động

Giống như các ứng dụng web, API web dựa vào HTTP để tạo điều kiện thuận lợi cho mối quan hệ máy khách/máy chủ giữa máy chủ của API (nhà cung cấp) và hệ thống hoặc người đưa ra yêu cầu API (người tiêu dùng).

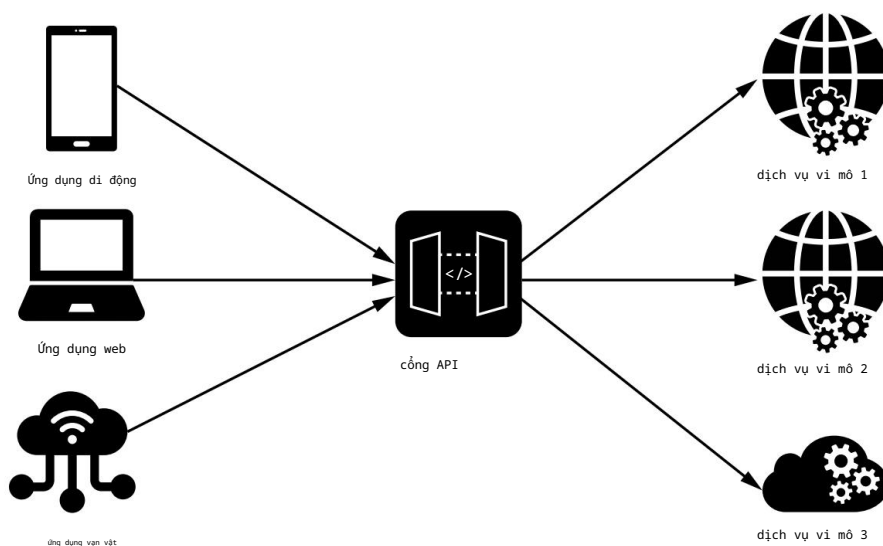
Người tiêu dùng API có thể yêu cầu tài nguyên từ điểm cuối API, đây là URL để tương tác với một phần của API. Mỗi ví dụ sau đây là một điểm cuối API khác nhau:

```
https://example.com/api/v3/users/
https://example.com/api/v3/customers/
https://example.com/api/updated_on/
https://example.com/api/state/1/
```

Tài nguyên là dữ liệu được yêu cầu. Tài nguyên đơn lẻ là một đối tượng duy nhất, chẳng hạn như `/api/user/{user_id}`. Bộ sưu tập là một nhóm tài nguyên, chẳng hạn như `/api/profiles/users`. Một bộ sưu tập con đề cập đến một bộ sưu tập trong một tài nguyên cụ thể. Ví dụ: `/api/user/{user_id}/settings` là điểm cuối để truy cập tập hợp con cài đặt của một người dùng (đơn lẻ) cụ thể.

Khi người tiêu dùng yêu cầu tài nguyên từ nhà cung cấp, yêu cầu sẽ chuyển qua cổng API, đây là thành phần quản lý API hoạt động như một điểm vào ứng dụng web. Ví dụ, như trong Hình 2-1, người dùng cuối có thể truy cập các dịch vụ của ứng dụng bằng rất nhiều thiết bị, tất cả đều được lọc qua cổng API. Sau đó, cổng API sẽ phân phối các yêu cầu tới bất kỳ dịch vụ siêu nhỏ nào cần thiết để đáp ứng từng yêu cầu.

Cổng API lọc các yêu cầu không hợp lệ, giám sát lưu lượng truy cập đến và định tuyến từng yêu cầu đến dịch vụ hoặc vi dịch vụ phù hợp. Cổng API cũng có thể xử lý các biện pháp kiểm soát bảo mật như xác thực, ủy quyền, mã hóa khi truyền bằng SSL, giới hạn tốc độ và cân bằng tải.



Hình 2-1: Kiến trúc vi dịch vụ mẫu và cổng API

Microservice là một phần mô-đun của ứng dụng web xử lý một chức năng cụ thể . Microservices sử dụng API để truyền dữ liệu và kích hoạt hành động. Ví dụ: một ứng dụng web có cổng thanh toán có thể có một số tính năng khác nhau trên một trang web: tính năng thanh toán, tính năng ghi thông tin tài khoản khách hàng và tính năng gửi biên nhận qua email khi mua hàng. Thiết kế phụ trợ của ứng dụng có thể là nguyên khối, nghĩa là tất cả các dịch vụ đều tồn tại trong một ứng dụng hoặc nó có thể có kiến trúc vi dịch vụ, trong đó mỗi dịch vụ hoạt động như một ứng dụng độc lập của riêng nó.

Người tiêu dùng API không nhìn thấy thiết kế phụ trợ, chỉ thấy các điểm cuối mà họ có thể tương tác và các tài nguyên mà họ có thể truy cập. Những điều này được nêu rõ trong hợp đồng API, đây là tài liệu mà con người có thể đọc được mô tả cách sử dụng API và cách bạn có thể mong đợi nó hoạt động. Tài liệu API khác với tổ chức này với tổ chức khác nhưng thường bao gồm mô tả về các yêu cầu xác thực, cấp độ quyền của người dùng, điểm cuối API và các tham số yêu cầu bắt buộc. Nó cũng có thể bao gồm các ví dụ sử dụng. Từ quan điểm của một hacker API, tài liệu có thể tiết lộ điểm cuối nào sẽ gọi dữ liệu khách hàng, khóa API nào bạn cần để trở thành quản trị viên và thậm chí cả các lỗi logic nghiệp vụ.

Trong hộp sau, tài liệu API GitHub cho /applications/

Điểm cuối {client_id}/grants/{access_token} , lấy từ <https://docs.github.com/vi/nghi-ngoi/tham-chieu/ung-dung>, là một ví dụ về tài liệu chất lượng.

THU HỒI MỘT GR ANT CHO MỘT ĐƠN			
Chủ sở hữu ứng dụng OAuth có thể thu hồi khoản trợ cấp cho ứng dụng OAuth của họ và một người dùng cụ thể.			
XÓA /applications/{client_id}/grant/{access_token}			
THÔNG SỐ			
Tên	Kiểu	TRONG	Sự miêu tả
chấp nhận	sợi dây		tiêu đề Cài đặt thành ứng dụng/ vnd.github.v3+json được khuyến dùng.
client_id	sợi dây	con đường	ID ứng dụng khách của ứng dụng GitHub của bạn.
truy cập thẻ	sợi dây	thân hình	Yêu cầu. Mã thông báo truy cập OAuth được sử dụng để xác thực với API GitHub.

Tài liệu về điểm cuối này bao gồm mô tả về mục đích của yêu cầu API, phương thức yêu cầu HTTP sẽ sử dụng khi tương tác với điểm cuối API và bản thân điểm cuối, / applications, theo sau là các biến.

Từ viết tắt CRUD, viết tắt của Tạo, Đọc, Cập nhật, Xóa, mô tả các hành động và phương pháp chính được sử dụng để tương tác với các API. Tạo là quá trình tạo bản ghi mới, được thực hiện thông qua yêu cầu POST. Đọc là truy xuất dữ liệu, được thực hiện thông qua yêu cầu GET. Cập nhật là cách các bản ghi hiện có được sửa đổi mà không bị ghi đè và được thực hiện bằng các yêu cầu POST hoặc PUT. Xóa là quá trình xóa các bản ghi, có thể được thực hiện bằng POST hoặc DELETE, như minh họa trong ví dụ này. Lưu ý rằng CRUD chỉ là phương pháp hay nhất và các nhà phát triển có thể triển khai API của họ theo những cách khác. Do đó, khi bạn học cách hack API sau này, chúng tôi sẽ kiểm tra ngoài các phương pháp CRUD.

Theo quy ước, dấu ngoặc nhọn có nghĩa là một biến nhất định là cần thiết trong các tham số đường dẫn. Biến {client_id} phải được thay thế bằng ID của khách hàng thực và biến {access_token} phải được thay thế bằng mã thông báo truy cập của riêng bạn. Mã thông báo là những gì nhà cung cấp API sử dụng để xác định và ủy quyền các yêu cầu cho người tiêu dùng API đã được phê duyệt. Các tài liệu API khác có thể sử dụng dấu hai chấm hoặc dấu ngoặc vuông để biểu thị một biến (ví dụ: / api/v2/customers/ hoặc /api/:collection/:client_id).

Phần "Tham số" đưa ra các yêu cầu xác thực và ủy quyền để thực hiện các hành động được mô tả, bao gồm tên của từng giá trị tham số, loại dữ liệu cần cung cấp, vị trí bao gồm dữ liệu và mô tả giá trị tham số.

Các loại API Web tiêu chuẩn

API có các loại tiêu chuẩn, mỗi loại khác nhau về quy tắc, chức năng và mục đích. Thông thường, một API nhất định sẽ chỉ sử dụng một loại, nhưng bạn có thể gặp phải các điểm cuối không khớp với định dạng và cấu trúc của các điểm cuối khác hoặc hoàn toàn không khớp với một loại tiêu chuẩn. Có thể nhận ra các API điển hình và không điển hình sẽ giúp bạn biết những gì mong đợi và kiểm tra với tư cách là một hacker API. Hãy nhớ rằng, hầu hết các API công khai được thiết kế để tự phục vụ, do đó, một nhà cung cấp API cụ thể sẽ thường cho bạn biết loại API mà bạn sẽ tương tác.

Phần này mô tả hai loại API chính mà chúng ta sẽ tập trung vào trong suốt cuốn sách này: API RESTful và GraphQL. Các phần sau của cuốn sách, cũng như các phòng thí nghiệm của cuốn sách, chỉ bao gồm các cuộc tấn công chống lại API RESTful và GraphQL.

API RESTful

Chuyển trạng thái đại diện (REST) là một tập hợp các ràng buộc kiến trúc cho các ứng dụng giao tiếp bằng các phương thức HTTP. Các API sử dụng các ràng buộc REST được gọi là API RESTful (hoặc chỉ REST).

REST được thiết kế để cải thiện nhiều điểm kém hiệu quả của các API cũ hơn, chẳng hạn như Giao thức truy cập đối tượng đơn giản (SOAP). Ví dụ: nó hoàn toàn dựa vào việc sử dụng HTTP, điều này giúp nó có khả năng tiếp cận người dùng cuối nhiều hơn. Các API REST chủ yếu sử dụng các phương thức HTTP GET, POST, PUT và DELETE để thực hiện CRUD (như được mô tả trong phần “Cách API Web hoạt động”).

Thiết kế RESTful phụ thuộc vào sáu ràng buộc. Những ràng buộc này là “nên” thay vì “phải”, phản ánh thực tế rằng REST về cơ bản là một bộ hướng dẫn cho kiến trúc dựa trên tài nguyên HTTP:

1. Giao diện thống nhất: API REST phải có giao diện thống nhất. TRONG
nói cách khác, thiết bị khách yêu cầu không quan trọng; thiết bị di động, thiết bị IoT (internet vạn vật) và máy tính xách tay đều phải có khả năng truy cập máy chủ theo cùng một cách.
2. Máy khách/máy chủ: Các API REST phải có kiến trúc máy khách/máy chủ.
Khách hàng là người tiêu dùng yêu cầu thông tin và máy chủ là nhà cung cấp thông tin đó.
3. Không trạng thái: API REST không yêu cầu giao tiếp trạng thái.
API REST không duy trì trạng thái trong quá trình giao tiếp; cứ như thể mỗi yêu cầu là yêu cầu đầu tiên mà máy chủ nhận được. Do đó, người tiêu dùng sẽ cần cung cấp mọi thứ mà nhà cung cấp sẽ cần để hành động theo yêu cầu. Điều này có lợi là giúp nhà cung cấp không phải ghi nhớ người tiêu dùng từ yêu cầu này sang yêu cầu khác.
Người tiêu dùng thường cung cấp mã thông báo để tạo trải nghiệm giống như trạng thái.
4. Có thể lưu vào bộ nhớ cache: Phản hồi từ nhà cung cấp API REST sẽ cho biết liệu phản hồi có thể lưu vào bộ nhớ cache hay không. Bộ nhớ đệm là phương pháp tăng thông lượng yêu cầu bằng cách lưu trữ dữ liệu thường được yêu cầu ở phía máy khách hoặc trong bộ đệm của máy chủ. Khi một yêu cầu được thực hiện, trước tiên, máy khách sẽ kiểm tra bộ nhớ cục bộ của nó để biết thông tin được yêu cầu. Nếu nó không tìm thấy thông tin, nó sẽ chuyển yêu cầu đến máy chủ, máy chủ này sẽ kiểm tra bộ nhớ cục bộ của nó để tìm thông tin được yêu cầu. Nếu dữ liệu cũng không có ở đó, yêu cầu có thể được chuyển đến các máy chủ khác, chẳng hạn như máy chủ cơ sở dữ liệu, nơi có thể truy xuất dữ liệu.

Như bạn có thể tưởng tượng, nếu dữ liệu được lưu trữ trên máy khách, máy khách có thể ngay lập tức truy xuất dữ liệu được yêu cầu mà máy chủ không phải trả chi phí xử lý. Điều này cũng áp dụng nếu máy chủ đã lưu trữ một yêu cầu. Yêu cầu càng đi xuống phía dưới chuỗi để truy xuất dữ liệu, chi phí tài nguyên càng cao và càng mất nhiều thời gian. Làm cho các API REST có thể lưu vào bộ nhớ cache theo mặc định là một cách để cải thiện hiệu suất và khả năng mở rộng tổng thể của REST bằng cách giảm thời gian phản hồi và sức mạnh xử lý của máy chủ. Các API thường quản lý bộ nhớ đệm bằng cách sử dụng các tiêu đề giải thích khi nào thông tin được yêu cầu sẽ hết hạn từ bộ đệm.
5. Hệ thống nhiều lớp: Máy khách có thể yêu cầu dữ liệu từ điểm cuối mà không cần biết về kiến trúc máy chủ bên dưới.
6. Mã theo yêu cầu (tùy chọn): Cho phép gửi mã cho khách hàng để chấp hành.

REST là một kiểu chứ không phải là một giao thức, vì vậy mỗi API RESTful có thể khác nhau. Nó có thể có các phương thức được kích hoạt ngoài CRUD, các bộ yêu cầu xác thực riêng, tên miền phụ thay vì đường dẫn cho điểm cuối, các yêu cầu giới hạn tốc độ khác nhau, v.v. Hơn nữa, các nhà phát triển hoặc một tổ chức có thể gọi API của họ là "RESTful" mà không tuân thủ tiêu chuẩn, điều đó có nghĩa là bạn không thể mong đợi mọi API bạn gặp phải đáp ứng tất cả các yêu cầu. Ràng buộc REST.

Liệt kê 2-1 cho thấy một yêu cầu REST API GET khá điển hình được sử dụng để tìm hiểu xem có bao nhiêu chiếc gối trong kho của một cửa hàng. Liệt kê 2-2 cho thấy phản hồi của nhà cung cấp.

NHẬN /api/v3/inventory/item/gối HTTP/1.1
MÁY CHỦ: rest-shop.com
Tác nhân người dùng: Mozilla/5.0
Chấp nhận: ứng dụng/json

Liệt kê 2-1: Một yêu cầu API RESTful mẫu

HTTP/1.1 200 OK
Máy chủ: RESTfulServer/0.1
Kiểm soát bộ đếm: không lưu trữ
Loại nội dung: ứng dụng/json

```
{
  "mục": {
    "id": "00101",
    "tên": "gối",
    "đếm": 25
    "giá": {
      "Tiền tệ: USD",
      "giá trị": "19,99"
    }
  },
}
```

Liệt kê 2-2: Một phản hồi API RESTful mẫu

Yêu cầu API REST này chỉ là một yêu cầu HTTP GET tới URL được chỉ định. Trong trường hợp này, yêu cầu truy vấn kho hàng của cửa hàng để tìm gối. Nhà cung cấp phản hồi bằng JSON cho biết ID, tên và số lượng mặt hàng trong kho của mặt hàng đó. Nếu có lỗi trong yêu cầu, nhà cung cấp sẽ phản hồi bằng mã lỗi HTTP trong phạm vi 400 cho biết điều gì đã xảy ra.

Một điều cần lưu ý: cửa hàng rest-shop.com đã cung cấp tất cả thông tin mà họ có về tài nguyên "gối" trong phản hồi của mình. Nếu ứng dụng của người tiêu dùng chỉ cần tên và giá trị của chiếc gối, thì người tiêu dùng sẽ cần lọc ra những thông tin bổ sung. Lượng thông tin được gửi lại cho người tiêu dùng hoàn toàn phụ thuộc vào cách nhà cung cấp API đã lập trình API của họ.

API REST có một số tiêu đề phổ biến mà bạn nên làm quen. Chúng giống hệt với các tiêu đề HTTP nhưng thường thấy trong các yêu cầu API REST hơn so với các loại API khác, vì vậy chúng có thể giúp bạn xác định các API REST. (Tiêu đề, quy ước đặt tên và trao đổi dữ liệu cho mật được sử dụng thường là các chỉ báo tốt nhất về loại API.) Các phần phụ sau đây trình bày chi tiết một số tiêu đề API REST phổ biến mà bạn sẽ xem

sang.

Ủy quyền

Tiêu đề ủy quyền được sử dụng để chuyển mã thông báo hoặc thông tin đăng nhập tới trình cung cấp API. Định dạng của các tiêu đề này là ủy quyền: <type> <token/credentials>.

Ví dụ: hãy xem tiêu đề ủy quyền sau:

Ủy quyền: Người mang Ab4dtok3n

Có nhiều loại ủy quyền khác nhau. Cơ bản sử dụng thông tin đăng nhập được mã hóa base64. Bearer sử dụng mã thông báo API. Cuối cùng, AWS-HMAC-SHA256 là loại ủy quyền AWS sử dụng khóa truy cập và khóa bí mật.

Loại nội dung

Các tiêu đề Loại nội dung được sử dụng để chỉ ra loại phương tiện được truyền. Các tiêu đề này khác với các tiêu đề Chấp nhận, cho biết loại phương tiện bạn muốn nhận; Tiêu đề Loại nội dung mô tả phương tiện bạn đang gửi.

Dưới đây là một số tiêu đề Loại nội dung phổ biến cho API REST:

application/json Được sử dụng để chỉ định Ký hiệu đối tượng JavaScript (JSON) làm loại phương tiện. JSON là loại phương tiện phổ biến nhất cho API REST.

application/xml Được sử dụng để chỉ định XML làm loại phương tiện.

application/x-www-form-urlencoded Định dạng trong đó các giá trị được gửi được mã hóa và phân tách bằng dấu và (&) và dấu bằng (=) được sử dụng giữa các cặp khóa/giá trị.

Tiêu đề phần mềm trung gian (X)

Tiêu đề X-<anything> được gọi là tiêu đề phần mềm trung gian và có thể phục vụ mọi loại mục đích. Chúng cũng khá phổ biến bên ngoài các yêu cầu API.

Thời gian phản hồi X có thể được sử dụng làm phản hồi API để cho biết thời gian xử lý phản hồi. X-API-Key có thể được sử dụng làm tiêu đề ủy quyền cho các khóa API. X-Powered-By có thể được sử dụng để cung cấp thêm thông tin về các dịch vụ phụ trợ. X-Rate-Limit có thể được sử dụng để cho người tiêu dùng biết họ có thể thực hiện

bao nhiêu yêu cầu trong một khung thời gian nhất định. X-RateLimit Remaining có thể cho người tiêu dùng biết có bao nhiêu yêu cầu còn lại trước khi họ vi phạm việc thực thi giới hạn tỷ lệ muộn. (Còn nhiều nữa, nhưng bạn hiểu ý rồi.)

Tiêu đề phần mềm trung gian X-<anything> có thể cung cấp nhiều thông tin hữu ích cho người tiêu dùng API cũng như tin tức.

MÃ HÓA DỮ LIỆU

Như chúng ta đã đề cập trong Chương 1, các yêu cầu HTTP sử dụng mã hóa như một phương pháp để đảm bảo rằng các giao tiếp được xử lý đúng cách. Các ký tự khác nhau có thể gây ra sự cố đối với các công nghệ mà máy chủ sử dụng được gọi là ký tự xấu. Một cách để xử lý các ký tự xấu là sử dụng lược đồ mã hóa định dạng thư theo cách loại bỏ chúng. Các lược đồ mã hóa phổ biến bao gồm mã hóa Unicode, mã hóa HTML, mã hóa URL và mã hóa base64. XML thường sử dụng một trong hai dạng mã hóa Unicode: UTF-8 hoặc UTF-16.

Khi chuỗi "hAPI hacker" được mã hóa bằng UTF-8, nó sẽ trở thành như sau:

```
\x68\x41\x50\x49\x20\x68\x61\x63\x68\x65\x72
```

Đây là phiên bản UTF-16 của chuỗi:

```
\u{68}\u{41}\u{50}\u{49}\u{20}\u{68}\u{61}\u{63}\u{6b}\u{65} \u{72}
```

Cuối cùng, đây là phiên bản mã hóa base64:

```
aEFQSSSB0YWNrZXI=
```

Việc nhận biết các lược đồ mã hóa này sẽ hữu ích khi bạn bắt đầu kiểm tra các yêu cầu và phản hồi cũng như gặp phải dữ liệu được mã hóa.

GraphQL

Viết tắt của Graph Query Language, GraphQL là một đặc điểm kỹ thuật cho API cho phép khách hàng xác định cấu trúc của dữ liệu họ muốn yêu cầu từ máy chủ. GraphQL là RESTful, vì nó tuân theo sáu ràng buộc của API REST. Tuy nhiên, GraphQL cũng có cách tiếp cận lấy truy vấn làm trung tâm, bởi vì nó được cấu trúc để hoạt động tương tự như ngôn ngữ truy vấn cơ sở dữ liệu như Ngôn ngữ truy vấn có cấu trúc (SQL).

Như bạn có thể thu thập từ tên của đặc tả, GraphQL lưu trữ tài nguyên trong một cấu trúc dữ liệu đồ thị. Để truy cập API GraphQL, bạn sẽ truy cập theo cách đánh máy vào URL nơi API được lưu trữ và gửi yêu cầu được ủy quyền có chứa tham số truy vấn dưới dạng nội dung của yêu cầu POST, tương tự như sau:

```
truy vấn {
  người dùng {
    tên tài khoản
    email
  }
}
```

Trong ngữ cảnh phù hợp, truy vấn này sẽ cung cấp cho bạn tên người dùng, ID và email của các tài nguyên được yêu cầu. Phản hồi của GraphQL cho truy vấn này sẽ giống như sau:

```
{
  "dữ liệu": {
    "người dùng": {
      "tên người dùng": "hapi_hacker",
      "id": 1111,
      "email": "hapihacker@email.com"
    }
  }
}
```

GraphQL cải thiện các API REST điển hình theo nhiều cách. Vì các API REST dựa trên tài nguyên, nên có khả năng sẽ xảy ra trường hợp người tiêu dùng cần thực hiện một số yêu cầu để có được tất cả dữ liệu họ cần. Mặt khác, nếu người tiêu dùng chỉ cần một giá trị cụ thể từ nhà cung cấp API, người tiêu dùng sẽ cần lọc ra dữ liệu dư thừa. Với GraphQL, người tiêu dùng có thể sử dụng một yêu cầu duy nhất để có được dữ liệu chính xác mà họ muốn. Đó là bởi vì, không giống như API REST, nơi máy khách nhận bất kỳ dữ liệu nào mà máy chủ được lập trình để trả về từ điểm cuối, bao gồm cả dữ liệu họ không cần, API GraphQL cho phép máy khách yêu cầu các trường cụ thể từ một tài nguyên.

GraphQL cũng sử dụng HTTP, nhưng nó thường phụ thuộc vào một điểm nhập duy nhất (URL) bằng phương thức POST. Trong một yêu cầu GraphQL, phần thân của yêu cầu POST là những gì nhà cung cấp xử lý. Ví dụ: hãy xem yêu cầu GraphQL trong Liệt kê 2-3 và phản hồi trong Liệt kê 2-4, mô tả yêu cầu kiểm tra kho hàng của cửa hàng để tìm các đồ họa.

ĐĂNG /graphql HTTP/1.1
 MÁY CHỦ: graphql-shop.com
 Ủy quyền: Người chịu ab4dt0k3n

```
{truy vấn1 {
  kho2 (mục:"Thẻ đồ họa", id: 00101) {
    tên
    trườg3{
    giá
    Số lượg} } }
}
```

Liệt kê 2-3: Một yêu cầu GraphQL ví dụ

HTTP/1.1 200 OK
 Loại nội dung: ứng dụng/json
 Máy chủ: GraphQLServer

```
{
  "dữ
  liệu": { "hàng tồn kho": { "tên": "Thẻ đồ họa",
    "trường":4[
    {
      "giá": "999,99"
      "số lượng": 25 } ] } }
}
```

Liệt kê 2-4: Một phản hồi GraphQL ví dụ

Như bạn có thể thấy, tải trọng truy vấn trong phần thân chỉ định thông tin cần thiết. Nội dung yêu cầu GraphQL bắt đầu với thao tác truy vấn 1, thao tác này tương đương với yêu cầu GET và được sử dụng để lấy thông tin từ API. Nút GraphQL mà chúng tôi đang truy vấn, "khoảng không quảng cáo" 2, còn được gọi là loại truy vấn gốc. Các nút, tương tự như các đối tượng, được tạo thành từ các trường 3, tương tự như các cặp khóa/giá trị trong REST. Sự khác biệt chính ở đây là chúng tôi có thể chỉ định các trường chính xác mà chúng tôi đang tìm kiếm. Trong ví dụ này, chúng tôi đang tìm kiếm các trường "giá" và "số lượng". Cuối cùng, bạn có thể thấy rằng phản hồi GraphQL chỉ cung cấp các trường được yêu cầu cho các đồ họa cụ thể 4. Thay vì lấy ID mặt hàng, tên mặt hàng và thông tin không cần thiết khác, truy vấn chỉ giải quyết các trường cần thiết.

Nếu đây là API REST, có thể cần phải gửi yêu cầu đến các điểm cuối khác nhau để lấy số lượng và sau đó là nhãn hiệu của các đồ họa, nhưng với GraphQL, bạn có thể tạo truy vấn cho thông tin cụ thể mà bạn đang tìm kiếm từ một điểm cuối duy nhất.

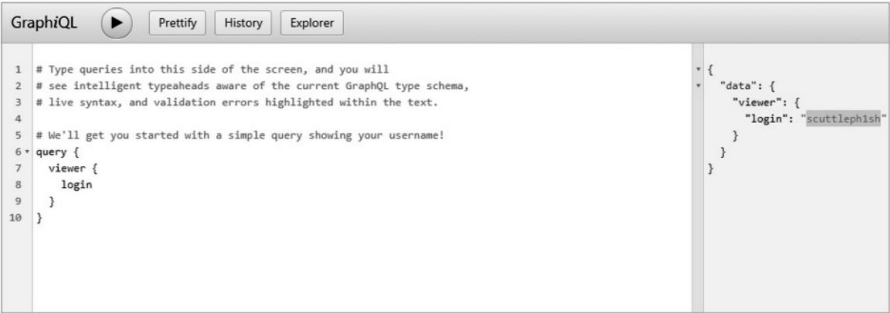
GraphQL vẫn hoạt động bằng cách sử dụng CRUD, điều này thoạt nghe có vẻ khó hiểu vì nó dựa trên các yêu cầu POST. Tuy nhiên, GraphQL sử dụng ba thao tác trong yêu cầu POST để tương tác với API GraphQL: truy vấn, đột biến và đăng ký. Truy vấn là thao tác lấy dữ liệu (đọc).

Đột biến là một thao tác được sử dụng để gửi và ghi dữ liệu (tạo, cập nhật và xóa). Subscribe là thao tác dùng để gửi dữ liệu (đọc) khi có sự kiện xảy ra. Đăng ký là một cách để các máy khách GraphQL nghe các bản cập nhật trực tiếp từ máy chủ.

GraphQL sử dụng lược đồ, là tập hợp dữ liệu có thể được truy vấn với dịch vụ nhất định. Có quyền truy cập vào lược đồ GraphQL cũng tương tự như có quyền truy cập vào bộ sưu tập API REST. Lược đồ GraphQL sẽ cung cấp cho bạn thông tin bạn cần để truy vấn API.

Bạn có thể tương tác với GraphQL bằng trình duyệt nếu có GraphQL IDE, chẳng hạn như GraphiQL, (xem Hình 2-2).

Nếu không, bạn sẽ cần một ứng dụng khách GraphQL chẳng hạn như Postman, Apollo Client, GraphQL-Request, GraphQL-CLI hoặc GraphQL-Compose. Trong các chương sau, chúng ta sẽ sử dụng Postman làm ứng dụng khách GraphQL của mình.



Hình 2-2: Giao diện GraphQL cho GitHub

SOAP: ĐỊNH DẠNG API HƯỚNG HÀNH ĐỘNG

Giao thức truy cập đối tượng đơn giản (SOAP) là một loại API hướng hành động dựa trên XML. SOAP là một trong những API web cũ hơn, ban đầu được phát hành dưới dạng XML RPC vào cuối những năm 1990, vì vậy chúng tôi sẽ không đề cập đến nó trong cuốn sách này.

Mặc dù SOAP hoạt động trên HTTP, SMTP, TCP và UDP, nhưng nó được thiết kế chủ yếu để sử dụng qua HTTP. Khi SOAP được sử dụng qua HTTP, tất cả các yêu cầu đều được thực hiện bằng cách sử dụng HTTP POST. Ví dụ: hãy xem yêu cầu SOAP mẫu sau:

```
POST /Inventory HTTP/1.1 Host:
www.soap-shop.com Content-
Type: application/soap+xml; bộ ký tự=utf-8 Độ dài nội dung:
nnn

<?xml version="1.0"?>

1<soap:Envelope
2xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
3soap:encodingStyle="http://www.w3.org/2003/05/soap- mã hóa">
4
5  3<soap:Body xmlns:m="http://www.soap-shop.com/inventory">
6    <m:GetInventoryPrice>
7      <m:InventoryName>ThebestSOAP</m:InventoryName>
8    </m:GetInventoryPrice> </
9  </soap:Body>
10
11</xà phòng:Phong bì>
```

Phản hồi SOAP tương ứng trông như thế này:

```
HTTP/1.1 200 OK
Loại nội dung: ứng dụng/xà phòng+xml; bộ ký tự=utf-8 Độ dài
nội dung: nnn
```

(còn tiếp)

```
<?xml version="1.0"?>

<xà phòng:Phong bì
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
xà phòng:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.soap-shop.com/inventory">
4<xà phòng:Lỗi>
<faultcode>soap:VersionMismatch</faultcode>
    <chuỗi lỗi, xml:lang='vi">
        Tên không khớp với bản ghi hàng tồn kho
    </faultstring>
</xà phòng:Lỗi>
</soap:Body>

</xà phòng:Phong bì>
```

Các thông báo API SOAP được tạo thành từ bốn phần: phong bì 1 và tiêu đề 2 là cần thiết, phần thân 3 và lỗi 4 là tùy chọn. Phong bì là một thẻ XML ở phần đầu của một thông điệp báo hiệu rằng thông báo đó là một thông báo SOAP. Tiêu đề có thể được sử dụng để xử lý một tin nhắn; trong ví dụ này, tiêu đề yêu cầu Kiểu nội dung cho phép nhà cung cấp SOAP biết loại nội dung được gửi trong yêu cầu POST (ứng dụng/

xà phòng + xml). Vì các API tạo điều kiện giao tiếp giữa máy với máy, nên về cơ bản, các tiêu đề tạo thành một thỏa thuận giữa người tiêu dùng và nhà cung cấp liên quan đến các kỳ vọng trong yêu cầu. Tiêu đề là một cách để đảm bảo rằng người tiêu dùng và nhà cung cấp hiểu nhau và nói cùng một ngôn ngữ. Phần thân là trọng tâm chính của thông báo XML, nghĩa là nó chứa dữ liệu được gửi đến ứng dụng. Lỗi là một phần tùy chọn của phản hồi SOAP có thể được sử dụng để cung cấp thông báo lỗi.

Thông số API REST

Sự đa dạng của API REST đã nhường chỗ cho các công cụ và tiêu chuẩn hóa khác lấp đầy một số khoảng trống. Thông số kỹ thuật API hoặc ngôn ngữ mô tả là các khuôn khổ giúp tổ chức thiết kế API của họ, tự động tạo tài liệu nhất quán mà con người có thể đọc được và do đó giúp nhà phát triển và người dùng biết điều gì sẽ xảy ra đối với chức năng và kết quả của API. Nếu không có thông số kỹ thuật, sẽ có rất ít hoặc không có sự thống nhất giữa các API. Người tiêu dùng sẽ phải tìm hiểu cách định dạng tài liệu của từng API và điều chỉnh ứng dụng của họ để tương tác với từng API.

Thay vào đó, người tiêu dùng có thể lập trình ứng dụng của họ để nhập các thông số kỹ thuật khác nhau và sau đó dễ dàng tương tác với bất kỳ API nào bằng thông số kỹ thuật đã cho. Nói cách khác, bạn có thể coi thông số kỹ thuật giống như ổ cắm điện gia đình của các API. Thay vì có một ổ cắm điện duy nhất cho mọi thiết bị gia dụng, việc sử dụng một định dạng nhất quán duy nhất trong nhà cho phép bạn mua một chiếc máy nướng bánh mì và cắm nó vào ổ cắm trên bất kỳ bức tường nào mà không gặp bất kỳ rắc rối nào.

Đặc tả OpenAPI 3.0 (OAS), trước đây được gọi là Swagger, là một trong những đặc tả hàng đầu cho API RESTful. OAS giúp tổ chức và quản lý các API bằng cách cho phép các nhà phát triển mô tả các điểm cuối, tài nguyên, hoạt động cũng như các yêu cầu xác thực và ủy quyền. Sau đó, họ có thể tạo tài liệu API mà con người và máy có thể đọc được, được định dạng là JSON hoặc YAML. Tài liệu API nhất quán tốt cho nhà phát triển và người dùng.

Ngôn ngữ lập mô hình API RESTful (RAML) là một cách khác để liên tục tạo tài liệu API. RAML là một đặc tả mở chỉ hoạt động với YAML để định dạng tài liệu. Tương tự như OAS, RAML được thiết kế để lập tài liệu, thiết kế, xây dựng và kiểm tra API REST. Để biết thêm thông tin về RAML, hãy xem repo GitHub raml-spec (<https://github.com/raml-org/raml-spec>).

Trong các chương sau, chúng ta sẽ sử dụng ứng dụng khách API có tên là Postman để nhập thông số kỹ thuật và truy cập tức thì vào các khả năng của API của tổ chức.

Định dạng trao đổi dữ liệu API

API sử dụng một số định dạng để tạo thuận lợi cho việc trao đổi dữ liệu. Ngoài ra, thông số kỹ thuật sử dụng các định dạng này để lập tài liệu API. Một số API, như SOAP, yêu cầu một định dạng cụ thể, trong khi những API khác cho phép khách hàng chỉ định format để sử dụng trong nội dung yêu cầu và phản hồi. Phần này giới thiệu ba định dạng phổ biến: JSON, XML và YAML. Sự quen thuộc với các định dạng thay đổi dữ liệu sẽ giúp bạn nhận ra các loại API, chức năng của API và cách chúng xử lý dữ liệu.

JSON

Ký hiệu đối tượng JavaScript (JSON) là định dạng trao đổi dữ liệu chính mà chúng tôi sẽ sử dụng xuyên suốt cuốn sách này vì nó được sử dụng rộng rãi cho các API. Nó tổ chức dữ liệu theo cách con người có thể đọc được và dễ dàng phân tích cú pháp bởi các ứng dụng; nhiều ngôn ngữ lập trình có thể biến JSON thành các kiểu dữ liệu mà chúng có thể sử dụng.

JSON đại diện cho các đối tượng dưới dạng các cặp khóa/giá trị được phân tách bằng dấu phẩy, trong một cặp dấu ngoặc nhọn, như sau:

```
{
  "firstName": "James",
  "lastName": "Lovell",
  "chuyến đi đến mặt trăng": 2,
  "isAstronaut": đúng,
  "walkedOnMoon": sai,
  "comment" : "Đây là một bình luận",
  "tàu vũ trụ": ["Gemini 7", "Gemini 12", "Apollo 8", "Apollo 13"],
  "sách": [
    {
      "title": "Lost Moon",
      "thể loại": "phi hư cấu"
    }
  ]
}
```

Mọi thứ nằm giữa dấu ngoặc nhọn đầu tiên và dấu ngoặc nhọn cuối cùng được coi là một sự vật. Trong đối tượng có một số cặp khóa/giá trị, chẳng hạn như "firstName": "James", "lastName": "Lovell" và "tripsToTheMoon": 2. Mục nhập đầu tiên của cặp khóa/giá trị (ở bên trái) là khóa, một chuỗi mô tả cặp giá trị và thứ hai là giá trị (ở bên phải), là một loại dữ liệu nào đó được đại diện bởi một trong các loại dữ liệu được chấp nhận (chuỗi, số, giá trị Boolean, null, một mảng, hoặc một đối tượng khác). Ví dụ: lưu ý giá trị Boolean sai cho mảng "walkedOnMoon" hoặc "tàu vũ trụ" được bao quanh bởi dấu ngoặc vuông. Cuối cùng, đối tượng lồng nhau "sách" chứa tập hợp các cặp khóa/giá trị của chính nó. Bảng 2-1 mô tả các loại JSON chi tiết hơn.

JSON không cho phép nhận xét nội tuyến, do đó, bất kỳ loại comment nào giống như nhận xét thông báo phải diễn ra dưới dạng một cặp khóa/giá trị như "bình luận" : "Đây là một bình luận". Ngoài ra, bạn có thể tìm nhận xét trong tài liệu API hoặc phản hồi HTTP.

Bảng 2-1: Các loại JSON

Kiểu	Mô tả Bất	Ví dụ
Dãy	kỳ sự kết hợp nào của các ký tự trong dấu ngoặc kép.	{ "Phương châm": "Hack hành tinh", "Uống": "Choáng", "Người dùng": "Dao cạo" }
Số	Các số nguyên cơ bản, phân số, số âm và số mũ. Lưu ý rằng nhiều mục được phân tách bằng dấu phẩy.	{ "số_1" : 101, "số_2" : -102, "số_3" : 1.03, "số_4" : 1.0E+4 }
Giá trị Boolean	Đúng hoặc sai.	{ "quản trị viên": sai, "privesc" : đúng }
Vô giá trị	Không có giá trị.	{ "giá trị": không }
Mảng	Một bộ sưu tập có thứ tự các giá trị. Tập hợp các giá trị được làm tròn trong dấu ngoặc vuông ([]) và các giá trị được phân tách bằng dấu phẩy.	{ "uid" : ["1", "2", "3"] }
Các đối tượng	Một tập hợp các cặp giá trị không có thứ tự được chèn giữa các dấu ngoặc nhọn ({}). Một đối tượng có thể chứa nhiều cặp khóa/giá trị tuple.	{ "quản trị viên": sai, "giá trị cốt lõi", "privesc" : đúng, "uid" : 101, "lỗ hổng" : "galore" }

Để minh họa các loại này, hãy xem các cặp khóa/giá trị sau trong dữ liệu JSON được tìm thấy trong phản hồi API Twitter:


```
{
  "id":1278533978970976256, 1
  "id_str":"1278533978970976256", 2
  "full_text":"1984: William Gibson xuất bản cuốn tiểu thuyết đầu tay của mình, Neuromancer. Đó là
  một câu chuyện khoa học viễn tưởng về Henry Case, một hacker máy tính đã sa lưới được một anh chàng
  bí ẩn tên là Armitage trao cho cơ hội chuộc lỗi. Không gian mạng. Hacking. Thực tế ảo. The matrix.
  Hacktivism. Phải đọc. https://t.co/R9hm2L0KQi",
  "cắt ngắn": sai 3
}
```

Trong ví dụ này, bạn sẽ có thể xác định số 1278533978970976256 1, các chuỗi giống như chuỗi cho khóa "id_str" và "full_text" 2 và giá trị Boolean 3 cho "cắt ngắn".

XML

Định dạng Ngôn ngữ đánh dấu mở rộng (XML) đã xuất hiện được một thời gian và có thể bạn sẽ nhận ra nó. XML được đặc trưng bởi các thẻ mô tả mà nó sử dụng để bọc dữ liệu. Mặc dù các API REST có thể sử dụng XML, nhưng hầu hết nó chỉ được liên kết với các API SOAP. API SOAP chỉ có thể sử dụng XML làm trao đổi dữ liệu.

Twitter JSON mà bạn vừa thấy sẽ giống như sau nếu được chuyển đổi thành XML:

```
<?xml phiên bản="1.0" mã hóa="UTF-8" ?> 1
<gốc> 2
  <id>1278533978970976300</id>
  <id_str>1278533978970976256</id_str>
  <full_text>1984: William Gibson xuất bản cuốn tiểu thuyết đầu tay, Thần kinh học. Đó là một câu chuyện
  về khoa học viễn tưởng về Henry Case, một hacker máy tính đã sa lưới, người được một anh chàng bí ẩn tên
  là Armitage trao cơ hội chuộc lỗi. Không gian mạng. hack. Thực tế ảo. Ma trận. Hacktivism.
  Phải đọc. https://t.co/R9hm2L0KQi </full_text>
  <cắt ngắn>sai</cắt ngắn>
</root>
```

XML luôn bắt đầu bằng một prolog chứa thông tin về Phiên bản XML và mã hóa được sử dụng 1.

Tiếp theo, các phần tử là phần cơ bản nhất của XML. Một phần tử là bất kỳ thẻ XML hoặc thông tin nào được bao quanh bởi các thẻ. Trong ví dụ trước, <id>1278533978970976300</id>, <id_str>1278533978</id_str>, <full_text>, </full_text> và <truncate>>false</truncate> đều là các phần tử. XML phải có một phần tử gốc và có thể chứa các phần tử con. Trong ví dụ này, phần tử gốc là <root> 2. Các phần tử con là các thuộc tính XML. Một ví dụ về phần tử con là phần tử <BookGenre> trong ví dụ sau:

```
<Thư việnSách>
  <BookGenre>Khoa học viễn tưởng</BookGenre>
</LibraryBooks>
```

Nhận xét trong XML được bao quanh bởi hai dấu gạch ngang, như sau: <!--Ví dụ về nhận xét XML-->.

Sự khác biệt chính giữa XML và JSON là các thẻ mô tả, mã hóa ký tự và độ dài của JSON: XML mất nhiều thời gian hơn để truyền tải cùng một thông tin, con số không lồ là 565 byte.

YAML

Một hình thức trao đổi dữ liệu nhẹ khác được sử dụng trong API, YAML là từ viết tắt lặp đi lặp lại viết tắt của YAML Ain't Markup Language. Nó được tạo ra dưới dạng định dạng dễ đọc hơn cho con người và máy tính để trao đổi dữ liệu.

Giống như tài liệu JSON, YAML chứa các cặp khóa/giá trị. Giá trị có thể là bất kỳ loại dữ liệu YAML nào, bao gồm số, chuỗi, Booleans, giá trị null và chuỗi. Ví dụ: hãy xem dữ liệu YAML sau:

```
---
id: 1278533978970976300
id_str: 1278533978970976256
#Nhận xét về Thần kinh
full_text: "1984: William Gibson xuất bản cuốn tiểu thuyết đầu tay của mình, Neuromancer. Đó là một câu chuyện khoa học viễn tưởng về Henry Case, một hacker máy tính đã sa lưới được một anh chàng bí ẩn tên là Armitage đề nghị một cơ hội chuộc lỗi. Không gian mạng. Hacking. Thực tế ảo. Ma trận. Hacktivism. Phải đọc. https://t.co/R9hm2LOKQi"
cắt ngắn: sai
...
```

Bạn sẽ nhận thấy rằng YAML dễ đọc hơn nhiều so với JSON. YAML tài liệu bắt đầu với

```
---
```

và kết thúc với

```
...
```

thay vì với dấu ngoặc nhọn. Ngoài ra, dấu ngoặc kép xung quanh chuỗi là tùy chọn. Ngoài ra, các URL không cần phải được mã hóa bằng dấu gạch chéo ngược. Cuối cùng, YAML sử dụng thụt đầu dòng thay vì dấu ngoặc nhọn để thể hiện lồng nhau và cho phép nhận xét bắt đầu bằng #.

Thông số kỹ thuật API thường sẽ được định dạng dưới dạng JSON hoặc YAML, vì những định dạng này con người dễ hiểu. Chỉ với một vài khái niệm cơ bản trong đầu, chúng ta có thể xem xét một trong hai định dạng này và hiểu điều gì đang diễn ra; tương tự như vậy, máy móc có thể dễ dàng phân tích thông tin.

Nếu bạn muốn xem thêm hoạt động của YAML, hãy truy cập <https://yaml.org>. Toàn bộ trang web được trình bày ở định dạng YAML. YAML là đệ quy tất cả các cách xuống.

Xác thực API

API có thể cho phép người tiêu dùng truy cập công khai mà không cần xác thực, nhưng khi API cho phép truy cập vào dữ liệu nhạy cảm hoặc độc quyền, nó sẽ sử dụng một số hình thức xác thực và ủy quyền. Quá trình xác thực của API phải xác thực rằng người dùng là người mà họ tuyên bố là và ủy quyền

quy trình sẽ cấp cho họ khả năng truy cập dữ liệu mà họ được phép truy cập. Phần này bao gồm nhiều phương thức xác thực và ủy quyền API. Các phương pháp này khác nhau về mức độ phức tạp và bảo mật, nhưng tất cả đều hoạt động trên một nguyên tắc chung: người tiêu dùng phải gửi một số loại thông tin đến nhà cung cấp khi đưa ra yêu cầu và nhà cung cấp phải liên kết thông tin đó với người dùng trước khi cấp hoặc từ chối quyền truy cập. đến một tài nguyên.

Trước khi chuyển sang xác thực API, điều quan trọng là phải hiểu xác thực là gì. Xác thực là quá trình chứng minh và xác minh danh tính. Trong ứng dụng web, xác thực là cách bạn chứng minh với máy chủ web rằng bạn là người dùng hợp lệ của ứng dụng web nói trên. Thông thường, điều này được thực hiện thông qua việc sử dụng thông tin đăng nhập, bao gồm một ID duy nhất (chẳng hạn như tên người dùng hoặc email) và mật khẩu. Sau khi khách hàng gửi thông tin đăng nhập, máy chủ web sẽ so sánh thông tin được gửi với thông tin đăng nhập mà nó đã lưu trữ. Nếu thông tin đăng nhập được cung cấp khớp với thông tin đăng nhập được lưu trữ, máy chủ web sẽ tạo phiên người dùng và cấp cookie cho khách hàng.

Khi phiên kết thúc giữa ứng dụng web và người dùng, máy chủ web sẽ hủy phiên và xóa cookie máy khách được liên kết.

Như đã mô tả trước đó trong chương này, API REST và GraphQL ít trạng thái hơn, vì vậy khi người tiêu dùng xác thực các API này, không có phiên nào được tạo giữa máy khách và máy chủ. Thay vào đó, người tiêu dùng API phải chứng minh danh tính của họ trong mọi yêu cầu được gửi đến máy chủ web của nhà cung cấp API.

Xác thực cơ bản

Hình thức xác thực API đơn giản nhất là xác thực cơ bản HTTP, trong đó người tiêu dùng bao gồm tên người dùng và mật khẩu của họ trong tiêu đề hoặc phần thân của yêu cầu. API có thể chuyển tên người dùng và chuyển từ tới nhà cung cấp ở dạng văn bản thuần túy, chẳng hạn như tên người dùng:mật khẩu hoặc có thể mã hóa thông tin đăng nhập bằng cách sử dụng thứ gì đó như base64 để tiết kiệm dung lượng (ví dụ: dXNlcm5hbWU6cGFzc3dvcmQK).

Mã hóa không phải là mã hóa và nếu dữ liệu được mã hóa base64 bị bắt, dữ liệu đó có thể dễ dàng được giải mã. Ví dụ: bạn có thể sử dụng dòng lệnh Linux để mã hóa base64 tên người dùng: mật khẩu và sau đó giải mã kết quả được mã hóa:

```
$ echo "tên người dùng:mật khẩu"|base64
dXNlcm5hbWU6cGFzc3dvcmQK
$ echo "dXNlcm5hbWU6cGFzc3dvcmQK"|base64 -d
tên người dùng: mật khẩu
```

Như bạn có thể thấy, xác thực cơ bản không có bảo mật vốn có và com hoàn toàn phụ thuộc vào các điều khiển bảo mật khác. Kẻ tấn công có thể xâm phạm xác thực cơ bản bằng cách nắm bắt lưu lượng HTTP, thực hiện tấn công trung gian, lừa người dùng cung cấp thông tin đăng nhập của họ thông qua các chiến thuật kỹ thuật xã hội hoặc thực hiện tấn công vũ phu trong đó họ thử nhiều tên người dùng và mật khẩu khác nhau cho đến khi họ tìm thấy một số mà làm việc.

Vì các API thường không trạng thái nên những API chỉ sử dụng xác thực cơ bản yêu cầu người tiêu dùng cung cấp thông tin xác thực trong mọi yêu cầu. Thay vào đó, nhà cung cấp API thường sử dụng xác thực cơ bản một lần cho yêu cầu đầu tiên, sau đó cấp khóa API hoặc một số mã thông báo khác cho tất cả các yêu cầu khác.

Khóa API

Khóa API là các chuỗi duy nhất mà nhà cung cấp API tạo và cấp quyền truy cập xác thực cho người tiêu dùng được phê duyệt. Sau khi người tiêu dùng API có khóa, họ có thể đưa khóa đó vào các yêu cầu bất cứ khi nào được nhà cung cấp chỉ định. Nhà cung cấp thường sẽ yêu cầu người tiêu dùng chuyển khóa trong tham số chuỗi truy vấn, tiêu đề yêu cầu, dữ liệu nội dung hoặc dưới dạng cookie khi họ đưa ra yêu cầu.

Các khóa API thường trông giống như các chuỗi số bán ngẫu nhiên hoặc ngẫu nhiên và chữ cái. Ví dụ: hãy xem khóa API có trong chuỗi truy vấn của URL sau:

```
/api/v1/users?apikey=ju574n3x4mp134p1k3y
```

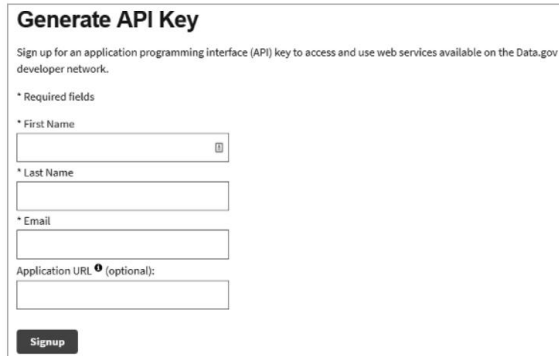
Sau đây là khóa API được bao gồm dưới dạng tiêu đề:

```
"Bi mật API": "17813fg8-46a7-5006-e235-45be7e9f2345"
```

Cuối cùng, đây là khóa API được chuyển vào dưới dạng cookie:

```
Cookie: API-Key= 4n07h3r4p1k3y
```

Quá trình lấy khóa API phụ thuộc vào nhà cung cấp. Ví dụ, API của NASA yêu cầu người tiêu dùng đăng ký API với tên, địa chỉ email và URL ứng dụng tùy chọn (nếu người dùng đang lập trình một ứng dụng để sử dụng API), như trong Hình 2-3.



Generate API Key

Sign up for an application programming interface (API) key to access and use web services available on the Data.gov developer network.

* Required fields

* First Name

* Last Name

* Email

Application URL (optional):

Signup

Hình 2-3: Biểu mẫu của NASA để tạo khóa API

Khóa kết quả sẽ giống như thế này:

```
roS6SmRjLdxZzrNSAkxjCdb6WodSda2G9zc2Q7sK
```

Nó phải được chuyển dưới dạng tham số URL trong mỗi yêu cầu API, như sau:

```
api.nasa.gov/planetary/apod?api_key=roS6SmRjLdxZzrNSAkxjCdb6WodSda2G9zc2Q7sK
```

Đối với một số lý do, khóa API có thể an toàn hơn xác thực cơ bản. Khi các khóa đủ dài, phức tạp và được tạo ngẫu nhiên, chúng có thể cực kỳ khó đoán đối với kẻ tấn công hoặc dùng vũ lực.

Ngoài ra, các nhà cung cấp có thể đặt ngày hết hạn để giới hạn khoảng thời gian mà các khóa hợp lệ.

Tuy nhiên, các khóa API có một số rủi ro liên quan mà chúng ta sẽ ưu tiên đề cập ở phần sau của cuốn sách này. Vì mỗi nhà cung cấp API có thể có hệ thống riêng để tạo khóa API, nên bạn sẽ tìm thấy các trường hợp trong đó khóa API được tạo dựa trên dữ liệu người dùng. Trong những trường hợp này, tin tặc API có thể đoán hoặc giả mạo khóa API bằng cách tìm hiểu về người tiêu dùng API. Các khóa API cũng có thể bị lộ trên internet trong các kho lưu trữ trực tuyến, để lại trong các nhận xét về mã, bị chặn khi chuyển qua các kết nối không được mã hóa hoặc bị đánh cắp thông qua lừa đảo.

Mã thông báo web JSON

Mã thông báo web JSON (JWT) là một loại mã thông báo thường được sử dụng trong xác thực dựa trên mã thông báo API. Nó được sử dụng như sau: Người tiêu dùng API xác thực với nhà cung cấp API bằng tên người dùng và mật khẩu. Nhà cung cấp tạo JWT và gửi lại cho người tiêu dùng. Người tiêu dùng thêm JWT được cung cấp vào tiêu đề ủy quyền trong tất cả các yêu cầu API.

JWT bao gồm ba phần, tất cả đều được mã hóa base64 và tách biệt được đánh giá theo thời gian: tiêu đề, tải trọng và chữ ký. tiêu đề bao gồm thông tin về thuật toán được sử dụng để ký tải trọng. Tải trọng là dữ liệu có trong mã thông báo, chẳng hạn như tên người dùng, đầu thời gian và nhà phát hành. Chữ ký là thông điệp được mã hóa và mã hóa được sử dụng để xác thực mã thông báo.

Bảng 2-2 cho thấy một ví dụ về những phần này, không được mã hóa để dễ đọc, cũng như mã thông báo cuối cùng.

LƯU Ý ường chữ ký không phải là mã hóa theo nghĩa đen của HMACSHA512...; đúng hơn, chữ ký được tạo bằng cách gọi hàm mã hóa HMACSHA512(), được chỉ định bởi "alg": "HS512", trên tiêu đề và tải trọng được mã hóa, sau đó mã hóa kết quả.

Bảng 2-2: Các thành phần JWT

Nội dung thành phần	
tiêu đề	<pre>{ "alg": "HS512", "typ": "JWT" }</pre>
Khối hàng	<pre>{ "phụ": "1234567890", "name": "hAPI Hacker", "iat": 1516239022 }</pre>
Chữ ký	<pre>HMACSHA512(base64UrlEncode(tiêu đề) + "." + base64UrlEncode(tải trọng), Mật khẩu siêu bí mật)</pre>
JWT	<pre>eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODk wIiwibmFtZSI6ImhBUEkgSGFja2VyIiwiaWF0IjoxNTE2MjM5MDIyfQ.zsUjG DbBjqI-bJbaUmvUdKaGSEvR0KfNjy9K6TckK55sd97AMdPDLxUZwsneff401ZwQ ikhgPm7HHlXN4jm0Q</pre>

JWT nói chung là an toàn nhưng có thể được triển khai theo những cách sẽ làm tổn hại đến tính bảo mật đó. Các nhà cung cấp API có thể triển khai JWT không sử dụng mã hóa, điều đó có nghĩa là bạn sẽ chỉ cần một lần giải mã base64 để có thể xem nội dung bên trong mã thông báo. Một tin tặc API có thể giải mã một mã thông báo như vậy, giả mạo nội dung và gửi lại cho nhà cung cấp để có quyền truy cập, như bạn sẽ thấy trong Chương 10. Khóa bí mật JWT cũng có thể bị đánh cắp hoặc đoán bởi vũ lực.

HMAC

Mã xác thực thông báo dựa trên hàm băm (HMAC) là phương thức xác thực API chính được Amazon Web Services (AWS) sử dụng. Khi sử dụng HMAC, nhà cung cấp tạo một khóa bí mật và chia sẻ nó với người tiêu dùng. Khi người tiêu dùng tương tác với API, hàm băm HMAC được áp dụng cho khóa bí mật và dữ liệu yêu cầu API của người tiêu dùng. Băm kết quả (còn được gọi là thông báo tóm tắt) được thêm vào yêu cầu và gửi đến nhà cung cấp. Nhà cung cấp tính toán HMAC, giống như người tiêu dùng đã làm, bằng cách chạy thông báo và khóa thông qua hàm băm, sau đó so sánh giá trị băm đầu ra với giá trị do khách hàng cung cấp. Nếu giá trị băm của nhà cung cấp khớp với giá trị băm của người tiêu dùng, thì người tiêu dùng được phép đưa ra yêu cầu. Nếu các giá trị không khớp, thì có thể khóa bí mật của khách hàng không chính xác hoặc thông báo đã bị giả mạo.

Tính bảo mật của thông báo tóm tắt phụ thuộc vào độ mạnh mật mã của hàm băm và khóa bí mật. Các cơ chế băm mạnh hơn thường tạo ra các hàm băm dài hơn. Bảng 2-3 cho thấy cùng một thông báo và khóa được băm bằng các thuật toán HMAC khác nhau.

Bảng 2-3: Thuật toán HMAC

thuật toán	đầu ra băm
HMAC-MD5	f37438341e3d22aa11b4b2e838120dcf
HMAC-SHA1	4c2de361ba8958558de3d049ed1fb5c115656e65
HMAC-SHA256	be8e73ffbd9a953f2ec892f06f9a5e91e6551023d1942ec7994fa1a78a5ae6bc
HMAC-SHA512	6434a354a730f888865bc5755d9f498126d8f67d73f32ccd2b775c47c91ce26b66dfa59c25aed7f4a6bcb4786d3a3c6130f63ae08367822af3f967d3a7469e1b

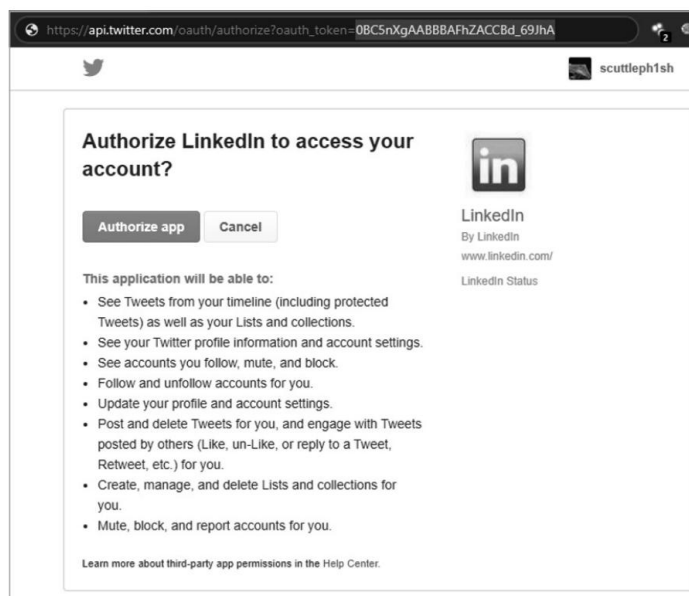
Bạn có thể có một số cảnh báo về việc sử dụng SHA1 hoặc MD5. Khi viết cuốn sách này, hiện tại không có lỗ hổng bảo mật nào ảnh hưởng đến HMAC-SHA1 và HMAC-MD5, nhưng các chức năng này yếu hơn về mặt đồ họa mã hóa so với SHA-256 và SHA-512. Tuy nhiên, các chức năng an toàn hơn cũng chậm hơn. Việc lựa chọn sử dụng hàm băm nào phụ thuộc vào việc ưu tiên hiệu suất hoặc bảo mật.

Như với các phương pháp xác thực trước đó, tính bảo mật của HMAC phụ thuộc vào người tiêu dùng và nhà cung cấp giữ bí mật khóa riêng tư. Nếu khóa bí mật bị xâm phạm, kẻ tấn công có thể mạo danh nạn nhân và giành quyền truy cập trái phép vào API.

OAuth 2.0

OAuth 2.0, hay chỉ OAuth, là một tiêu chuẩn ủy quyền cho phép các dịch vụ khác nhau truy cập vào dữ liệu của nhau, thường sử dụng API để hỗ trợ giao tiếp giữa dịch vụ với dịch vụ.

Giả sử bạn muốn tự động chia sẻ các tweet trên Twitter của mình trên LinkedIn. Trong mô hình của OAuth, chúng tôi sẽ coi Twitter là nhà cung cấp dịch vụ và LinkedIn là ứng dụng hoặc ứng dụng khách. Để đăng các tweet của bạn, LinkedIn sẽ cần có quyền truy cập thông tin Twitter của bạn. Vì cả Twitter và LinkedIn đã triển khai OAuth, thay vì cung cấp thông tin đăng nhập của bạn cho nhà cung cấp dịch vụ và người tiêu dùng mỗi khi bạn muốn chia sẻ thông tin này trên các nền tảng, bạn chỉ cần truy cập cài đặt LinkedIn của mình và ủy quyền cho Twitter. Làm như vậy sẽ đưa bạn đến `api.twitter.com` để ủy quyền cho LinkedIn truy cập tài khoản Twitter của bạn (xem Hình 2-4).

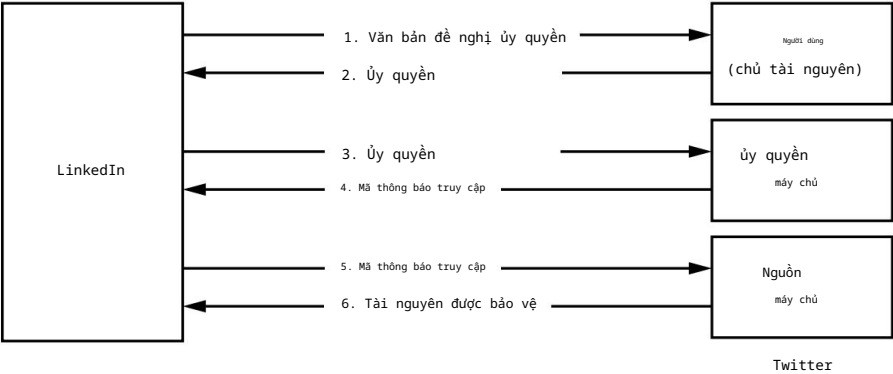


Hình 2-4: Yêu cầu ủy quyền OAuth của LinkedIn-Twitter

Khi bạn cho phép LinkedIn truy cập các bài đăng trên Twitter của mình, Twitter sẽ tạo mã thông báo truy cập có giới hạn, dựa trên thời gian cho LinkedIn. Sau đó, LinkedIn cung cấp mã thông báo đó cho Twitter để đăng thay mặt bạn và bạn không cần phải cung cấp cho LinkedIn thông tin đăng nhập Twitter của mình.

Hình 2-5 hiển thị quy trình OAuth chung. Người dùng (chủ sở hữu tài nguyên) cấp cho ứng dụng (máy khách) quyền truy cập vào dịch vụ (máy chủ ủy quyền), dịch vụ tạo mã thông báo và sau đó ứng dụng sử dụng mã thông báo để trao đổi dữ liệu với dịch vụ (cũng là máy chủ tài nguyên).

Trong ví dụ LinkedIn-Twitter, bạn là chủ sở hữu tài nguyên, LinkedIn là ứng dụng/ứng dụng khách và Twitter là máy chủ ủy quyền và tài nguyên máy chủ.



Hình 2-5: Minh họa quy trình OAuth

OAuth là một trong những hình thức ủy quyền API đáng tin cậy nhất. Tuy nhiên, trong khi bổ sung tính bảo mật cho quy trình ủy quyền, nó cũng mở rộng bề mặt tấn công tiềm ẩn—mặc dù các lỗi thường liên quan nhiều hơn đến cách nhà cung cấp API triển khai OAuth hơn là với chính OAuth. Các nhà cung cấp API triển khai OAuth kém có thể tự đặt mình vào một loạt các cuộc tấn công như tiêm mã thông báo, sử dụng lại mã ủy quyền, giả mạo yêu cầu trên nhiều trang web, chuyển hướng không hợp lệ và lừa đảo.

Không có chứng thực

Như trong các ứng dụng web nói chung, có rất nhiều trường hợp hợp lệ khi một API hoàn toàn không có xác thực. Nếu một API không xử lý dữ liệu nhạy cảm và chỉ cung cấp thông tin công khai, thì nhà cung cấp có thể đưa ra trường hợp không cần xác thực.

API đang hoạt động: Khám phá API của Twitter

Sau khi đọc chương này và chương trước, bạn sẽ hiểu các thành phần khác nhau chạy bên dưới GUI của ứng dụng web. Bây giờ chúng ta hãy làm cho những khái niệm này trở nên cụ thể hơn bằng cách xem xét kỹ API của Twitter. Nếu bạn mở trình duyệt web và truy cập URL <https://twitter.com>, yêu cầu ban đầu kích hoạt một loạt giao tiếp giữa máy khách và máy chủ. Trình duyệt của bạn tự động sắp xếp các hoạt động truyền dữ liệu này, nhưng bằng cách sử dụng proxy web như Burp Suite, mà chúng ta sẽ thiết lập trong Chương 4, bạn có thể thấy tất cả các yêu cầu và phản hồi đang hoạt động.

Giao tiếp bắt đầu với loại lưu lượng HTTP điển hình được mô tả trong Chương 1:

1. Khi bạn đã nhập một URL vào trình duyệt của mình, trình duyệt sẽ tự động gửi một yêu cầu HTTP GET tới máy chủ web tại twitter.com:

```

NHẬN/HTTP/1.1
Máy chủ: twitter.com

Tác nhân người dùng: Mozilla/5.0
Chấp nhận: văn bản/html
--snip--
Bánh quy: [...]

```

2. Máy chủ ứng dụng web Twitter nhận yêu cầu và phản hồi yêu cầu GET bằng cách đưa ra phản hồi 200 OK thành công:

```

HTTP/1.1 200 OK

kiểm soát bộ đệm: không có bộ đệm, không lưu trữ, phải xác thực lại
kết nối: đóng

chính sách bảo mật nội dung: content-src 'self'
loại nội dung: văn bản/html; bộ ký tự = utf-8
máy chủ: tsa_a
--snip--
x-powered-by: Express
x-thời gian phản hồi: 56

<!DOCTYPE html>
<html dir="ltr" lang="vi">
--snip--

```

Tiêu đề phản hồi này chứa trạng thái của kết nối HTTP, hướng dẫn máy khách, thông tin phần mềm trung gian và thông tin liên quan đến cookie. Hướng dẫn máy khách cho trình duyệt biết cách xử lý thông tin được yêu cầu, chẳng hạn như dữ liệu lưu vào bộ nhớ đệm, chính sách bảo mật nội dung và hướng dẫn về loại nội dung đã được gửi. Tải lượng thực tế bắt đầu ngay bên dưới thời gian phản hồi x; nó cung cấp cho trình duyệt HTML cần thiết để hiển thị trang web.

Bây giờ, hãy tưởng tượng rằng người dùng tra cứu "hack" bằng thanh tìm kiếm của Twitter. Điều này khởi động một yêu cầu POST tới API của Twitter, như được hiển thị bên dưới. Twitter có thể tận dụng các API để phân phối các yêu cầu và cung cấp liên tục các tài nguyên được yêu cầu cho nhiều người dùng.

```

POST /1.1/jot/client_event.json?q=hacking HTTP/1.1
Máy chủ: api.twitter.com

Tác nhân người dùng: Mozilla/5.0
--snip--
Ủy quyền: Người mang AAAAAAAAAAAAAAAAAA...
--snip--

```

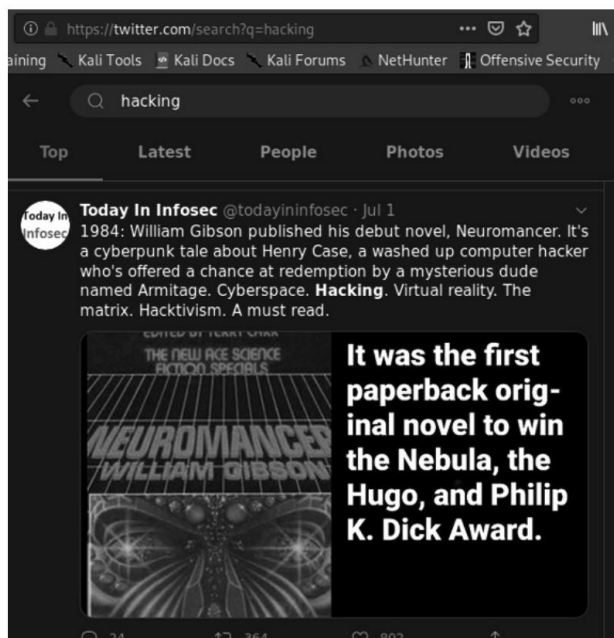
Yêu cầu POST này là một ví dụ về API Twitter truy vấn dịch vụ web tại `api.twitter.com` cho cụm từ tìm kiếm "hack". API Twitter

phản hồi bằng JSON chứa kết quả tìm kiếm, bao gồm các tweet và thông tin về từng tweet như đề cập của người dùng, thẻ bắt đầu bằng # và thời gian đăng:

```
"đã_tạo_at": [...]  
"id":1278533978970976256  
"id_str": "1278533978970976256"  
"full-text": "1984: William Gibson xuất bản cuốn tiểu thuyết đầu tay..."  
"cắt ngắn": sai,  
--snip--
```

Thực tế là API Twitter dường như tuân thủ CRUD, quy ước đặt tên API, mã thông báo để ủy quyền, ứng dụng/x-www-form-urlencoded và JSON dưới dạng trao đổi dữ liệu cho thấy khá rõ ràng rằng API này là API RESTful.

Mặc dù nội dung phản hồi được định dạng theo cách dễ đọc, nhưng nó được trình duyệt xử lý để hiển thị dưới dạng trang web mà con người có thể đọc được. Trình duyệt hiển thị kết quả tìm kiếm bằng cách sử dụng chuỗi từ yêu cầu API. Sau đó, phản hồi của nhà cung cấp sẽ điền vào trang các kết quả tìm kiếm, hình ảnh và thông tin liên quan đến mạng xã hội như lượt thích, tin nhắn lại, nhận xét (xem Hình 2-6).



Hình 2-6: Kết quả hiển thị từ yêu cầu tìm kiếm Twitter API

Từ quan điểm của người dùng cuối, toàn bộ tương tác xuất hiện liền mạch: bạn nhấp vào thanh tìm kiếm, nhập truy vấn và nhận kết quả.

Bản tóm tắt

Trong chương này, chúng tôi đã đề cập đến thuật ngữ, các bộ phận, loại và kiến trúc hỗ trợ của API. Bạn đã biết rằng API là giao diện để tương tác với các ứng dụng web. Các loại API khác nhau có các quy tắc, chức năng và mục đích khác nhau, nhưng chúng đều sử dụng một số loại định dạng để trao đổi dữ liệu giữa các ứng dụng. Họ thường sử dụng các sơ đồ xác thực và ủy quyền để đảm bảo người tiêu dùng chỉ có thể truy cập các tài nguyên mà họ được cho là.

Hiểu những khái niệm này sẽ giúp bạn chuẩn bị để tự tin tấn công các thành phần tạo nên API. Khi bạn tiếp tục đọc, hãy tham khảo chương này nếu bạn gặp phải các khái niệm API khiến bạn bối rối.

3

CÁC LỖI KHÓA API PHỔ BIẾN



Hiểu các lỗi hỏng phổ biến sẽ giúp bạn xác định các điểm yếu khi bạn đang thử nghiệm API. Trong chương này, tôi đề cập đến hầu hết các lỗi hỏng có trong danh sách Top 10 bảo mật API của Dự án bảo mật ứng dụng web mở (OWASP), cùng với hai điểm yếu hữu ích khác: lỗi hỏng tiết lộ thông tin và logic nghiệp vụ. Tôi sẽ mô tả từng lỗi hỏng, tầm quan trọng của nó và các kỹ thuật được sử dụng để khai thác nó. Trong các chương sau, bạn sẽ có kinh nghiệm tìm kiếm và khai thác nhiều lỗi hỏng này.

API BẢO MẬT CỦA OWASP VÀ TOP 10

OWASP là một tổ chức phi lợi nhuận tạo ra nội dung và công cụ miễn phí nhằm bảo mật các ứng dụng web. Do các mối quan hệ dễ bị tổn thương API ngày càng phổ biến, OWASP đã phát hành Top 10 Bảo mật API OWASP, danh sách 10 lỗ hổng API phổ biến nhất, vào cuối năm 2019. Hãy xem dự án, được dẫn dắt bởi các chuyên gia bảo mật API Inon Shkedy và Erez Yalon, tại <https://owasp.org/www-project-api-security>. Trong Chương 15, tôi sẽ chứng minh cách các khả năng của lỗ hổng được mô tả trong Top 10 Bảo mật API của OWASP đã bị khai thác trong các vụ vi phạm lớn và phát hiện tiền thưởng lỗi. Chúng tôi cũng sẽ sử dụng một số công cụ OWASP để tấn công các API trong Phần II và III của cuốn sách.

Công bố thông tin

Khi một API và phần mềm hỗ trợ của nó chia sẻ thông tin nhạy cảm với những người dùng không có đặc quyền, thì API đó có lỗ hổng tiết lộ thông tin. Thông tin có thể được tiết lộ trong các phản hồi API hoặc các nguồn công khai như kho lưu trữ mã, kết quả tìm kiếm, tin tức, phương tiện truyền thông xã hội, trang web của mục tiêu và các thư mục API công khai.

Dữ liệu nhạy cảm có thể bao gồm bất kỳ thông tin nào mà kẻ tấn công có thể tận dụng để tạo lợi thế cho chúng. Ví dụ: một trang web đang sử dụng API WordPress có thể vô tình chia sẻ thông tin người dùng với bất kỳ ai điều hướng đến đường dẫn API `/wp-json/wp/v2/users`, đường dẫn này trả về tất cả tên người dùng WordPress hoặc "sên". Chẳng hạn, hãy xem yêu cầu sau:

NHẬN `https://www.sitename.org/wp-json/wp/v2/users`

Nó có thể trả về dữ liệu này:

```
[{"id":1,"name":"Quản trị viên", "slug":"admin"}],
{"id":2,"name":"Vincent Valentine", "slug":"Vincent"}]
```

Sau đó, những con sên này có thể được sử dụng để cố gắng đăng nhập với tư cách là người dùng đã tiết lộ bằng một cuộc tấn công vũ phu, nhờ thông tin xác thực hoặc rải mật khẩu. (Chương 8 mô tả chi tiết các cuộc tấn công này.)

Một vấn đề tiết lộ thông tin phổ biến khác liên quan đến thông báo dài dòng. Thông báo lỗi giúp người tiêu dùng API khắc phục sự cố tương tác của họ với API và cho phép nhà cung cấp API hiểu các vấn đề với ứng dụng của họ. Tuy nhiên, nó cũng có thể tiết lộ thông tin nhạy cảm về tài nguyên, người dùng và kiến trúc cơ bản của API (chẳng hạn như phiên bản của máy chủ web hoặc cơ sở dữ liệu). Ví dụ: giả sử bạn cố gắng xác thực API và nhận được thông báo lỗi, chẳng hạn như "ID người dùng được cung cấp không tồn tại".

Tiếp theo, giả sử bạn sử dụng một email khác và thông báo lỗi thay đổi thành "incorrect password". Điều này cho bạn biết rằng bạn đã cung cấp ID người dùng hợp pháp cho API.

Tìm kiếm thông tin người dùng là một cách tuyệt vời để bắt đầu giành quyền truy cập vào một API. Các thông tin sau cũng có thể bị lợi dụng trong một cuộc tấn công: gói phần mềm, thông tin hệ điều hành, nhật ký hệ thống và lỗi phần mềm. Nói chung, bất kỳ thông tin nào có thể giúp chúng tôi tìm ra các lỗ hổng nghiêm trọng hơn hoặc hỗ trợ khai thác đều có thể được coi là lỗ hổng tiết lộ thông tin.

Thông thường, bạn có thể thu thập nhiều thông tin nhất bằng cách tương tác với API điểm cuối và phân tích phản hồi. Phản hồi API có thể tiết lộ thông tin trong các tiêu đề, tham số và lỗi chi tiết. Các nguồn thông tin hữu ích khác là tài liệu API và các tài nguyên được thu thập trong quá trình trình sát. Chương 6 đề cập đến nhiều công cụ và kỹ thuật được sử dụng để phát hiện việc tiết lộ thông tin API.

Cấp phép đối tượng bị hỏng

Một trong những lỗ hổng phổ biến nhất trong API là ủy quyền mức đối tượng bị hỏng (BOLA). Các lỗ hổng BOLA xảy ra khi nhà cung cấp API cho phép người tiêu dùng API truy cập vào các tài nguyên mà họ không được phép truy cập. Nếu điểm cuối API không có kiểm soát truy cập cấp đối tượng, thì điểm cuối đó sẽ không thực hiện kiểm tra để đảm bảo người dùng chỉ có thể truy cập tài nguyên của riêng họ. Khi các điều khiển này bị thiếu, Người dùng A sẽ có thể yêu cầu thành công tài nguyên của Người dùng B.

API sử dụng một số loại giá trị, chẳng hạn như tên hoặc số, để xác định các đối tượng khác nhau. Khi chúng tôi phát hiện ra các ID đối tượng này, chúng tôi nên kiểm tra xem liệu chúng tôi có thể tương tác với tài nguyên của những người dùng khác khi chưa được xác thực hoặc được xác thực với tư cách là một người dùng khác hay không. Chẳng hạn, hãy tưởng tượng rằng chúng tôi chỉ được phép truy cập người dùng Cloud Strife. Chúng tôi sẽ gửi yêu cầu GET ban đầu tới <https://bestgame.com/api/v3/users?id=5501> và nhận được phản hồi sau:

```
{
  "id": "5501",
  "first_name": "Đám mây",
  "last_name": "Xung đột",
  "liên kết": "https://www.bestgame.com/user/strife.buster.97",
  "tên": "Cuộc xung đột trên mây",
  "dob": "31-01-1997",
  "tên người dùng": "strife.buster.97"
}
```

Điều này không gây ra vấn đề gì vì chúng tôi được phép truy cập thông tin của Cloud. Tuy nhiên, nếu chúng tôi có thể truy cập thông tin của người dùng khác, thì có một vấn đề lớn về ủy quyền.

Trong tình huống này, chúng tôi có thể kiểm tra các sự cố này bằng cách sử dụng một số nhận dạng khác gần với ID của Cloud là 5501. Giả sử chúng tôi có thể lấy thông tin về người dùng khác bằng cách gửi yêu cầu cho <https://bestgame.com/api/v3/users?id=5502> và nhận được phản hồi sau:

```
{
  "id": "5502",
  "first_name": "Zack",
```

```

    "last_name": "Công bằng",
    "liên_kết": " https://www.bestgame.com/user/shinra-number-1",
    "name": "Zack Fair",
    "dob": "2007-09-13",
    "tên người dùng": "shinra-số-1"
  }

```

Trong trường hợp này, Cloud đã phát hiện ra BOLA. Lưu ý rằng đối tượng có thể dự đoán ID không nhất thiết chỉ ra rằng bạn đã tìm thấy BOLA. Để ứng dụng dễ bị tổn thương, ứng dụng phải không xác minh được rằng một người dùng nhất định chỉ có thể truy cập tài nguyên của chính họ.

Nói chung, bạn có thể kiểm tra BOLA bằng cách hiểu cách cấu trúc các tài nguyên của API và cố gắng truy cập các tài nguyên mà lẽ ra bạn không thể truy cập. Bằng cách phát hiện các mẫu trong đường dẫn và tham số API, bạn sẽ có thể dự đoán các tài nguyên tiềm năng khác. Các thành phần in đậm trong các yêu cầu API sau đây sẽ thu hút sự chú ý của bạn:

```

NHẬN/api/tài nguyên/1
NHẬN /user/account/find?user_id=15
POST /công ty/tài khoản/Apple/số dư
ĐĂNG /admin/pwreset/tài khoản/90

```

Trong những trường hợp này, bạn có thể đoán các nguồn tài nguyên tiềm năng khác, như sau, bằng cách thay đổi các giá trị in đậm:

```

NHẬN/api/tài nguyên/3
NHẬN/người dùng/tài khoản/tìm?user_id=23
ĐĂNG /công ty/tài khoản/Google/số dư
ĐĂNG /admin/pwreset/tài khoản/111

```

Trong các ví dụ đơn giản này, bạn đã thực hiện một cuộc tấn công bằng cách chỉ thay thế các mục in đậm bằng các số hoặc từ khác. Nếu bạn có thể truy cập thành công thông tin mà lẽ ra bạn không được phép truy cập, thì bạn đã phát hiện ra lỗ hổng BOLA.

Trong Chương 9, tôi sẽ trình bày cách bạn có thể dễ dàng làm mờ các tham số như `user_id=` trong đường dẫn URL và sắp xếp các kết quả để xác định xem có tồn tại lỗ hổng BOLA hay không. Trong Chương 10, chúng tôi sẽ tập trung vào việc tấn công các lỗ hổng ủy quyền như BOLA và BFLA (ủy quyền mức chức năng bị hỏng, sẽ được thảo luận sau trong chương này). BOLA có thể là một lỗ hổng API treo thấp mà bạn có thể dễ dàng khám phá bằng cách sử dụng tính năng nhận dạng mẫu và sau đó kích hoạt nó bằng một vài yêu cầu. Những lần khác, việc khám phá có thể khá phức tạp do sự phức tạp của ID đối tượng và các yêu cầu được sử dụng để lấy tài nguyên của người dùng khác.

Xác thực người dùng bị hỏng

Xác thực người dùng bị hỏng đề cập đến bất kỳ điểm yếu nào trong quy trình xác thực API. Các lỗ hổng này thường xảy ra khi nhà cung cấp API không triển khai cơ chế bảo vệ xác thực hoặc triển khai cơ chế không chính xác.

Xác thực API có thể là một hệ thống phức tạp bao gồm một số quy trình với rất nhiều khả năng xảy ra lỗi. Một vài thập kỷ trước, chuyên gia bảo mật Bruce Schneier đã nói: “Tương lai của các hệ thống kỹ thuật số là sự phức tạp và sự phức tạp là kẻ thù tồi tệ nhất của bảo mật”. Như chúng ta đã biết từ sáu ràng buộc của API REST được thảo luận trong Chương 2, API RESTful được coi là không trạng thái. Để không trạng thái, nhà cung cấp không cần phải ghi nhớ người tiêu dùng từ yêu cầu này sang yêu cầu khác. Để ràng buộc này hoạt động, API thường yêu cầu người dùng trải qua quy trình đăng ký để nhận được mã thông báo duy nhất. Sau đó, người dùng có thể bao gồm mã thông báo trong các yêu cầu để chứng minh rằng họ được phép thực hiện các yêu cầu đó.

Do đó, quy trình đăng ký được sử dụng để lấy mã thông báo API, xử lý mã thông báo và hệ thống tạo mã thông báo đều có thể có các điểm yếu riêng. Ví dụ : để xác định xem quy trình tạo mã thông báo có yếu hay không, chúng tôi có thể thu thập mẫu mã thông báo và phân tích chúng để tìm điểm tương đồng. Nếu quy trình tạo mã thông báo không dựa vào mức độ ngẫu nhiên hoặc entropy cao, thì có khả năng chúng ta sẽ có thể tạo mã thông báo của riêng mình hoặc chiếm đoạt mã thông báo của người khác.

Xử lý mã thông báo có thể là lưu trữ mã thông báo, phương thức truyền mã thông báo qua mạng, sự hiện diện của mã thông báo được mã hóa cứng, v.v. Chúng tôi có thể phát hiện các mã thông báo được mã hóa cứng trong các tệp nguồn JavaScript hoặc nắm bắt chúng khi chúng tôi phân tích một ứng dụng web. Khi chúng tôi đã chiếm được mã thông báo, chúng tôi có thể sử dụng nó để có quyền truy cập vào các điểm cuối đã ẩn trước đó hoặc để vượt qua sự phát hiện. Nếu nhà cung cấp API gán danh tính cho mã thông báo, thì chúng tôi sẽ lấy danh tính đó bằng cách chiếm quyền điều khiển mã thông báo bị đánh cắp.

Các quy trình xác thực khác có thể có tập hợp lỗi riêng bao gồm các khía cạnh của hệ thống đăng ký, chẳng hạn như đặt lại mật khẩu và các tính năng xác thực đa yếu tố. Ví dụ: hãy tưởng tượng tính năng đặt lại từ mật khẩu yêu cầu bạn cung cấp địa chỉ email và mã gồm sáu chữ số để đặt lại mật khẩu của bạn. Chà, nếu API cho phép bạn thực hiện bao nhiêu yêu cầu tùy thích, thì bạn chỉ phải thực hiện một triệu yêu cầu để đoán mã và đặt lại mật khẩu của bất kỳ người dùng nào. Mã gồm bốn chữ số sẽ chỉ yêu cầu 10.000 yêu cầu.

Đồng thời theo dõi khả năng truy cập các tài nguyên nhạy cảm mà không được xác thực; Khóa API, mã thông báo và thông tin đăng nhập được sử dụng trong URL; thiếu các hạn chế về tỷ lệ khi xác thực; và thông báo lỗi dài dòng. Ví dụ: mã được cam kết với kho lưu trữ GitHub có thể tiết lộ khóa API quản trị viên được mã hóa cứng:

```
"oauth_client":
[{"client_id": "12345-abcd",
"client_type": "quản trị viên",
"api_key": "AIzaSyDrbTFCeb5k0yPSfL2heqdF-N19XoLxdw"}]
```

Do tính chất không trạng thái của API REST, khóa API được hiển thị công khai tương đương với việc khám phá tên người dùng và mật khẩu. Bằng cách sử dụng khóa API bị lộ, bạn sẽ đảm nhận vai trò được liên kết với khóa đó. Trong Chương 6, chúng ta sẽ sử dụng các kỹ năng trinh sát của mình để tìm các khóa bị lộ trên internet.

Trong Chương 8, chúng ta sẽ thực hiện nhiều cuộc tấn công chống lại xác thực API, chẳng hạn như bỏ qua xác thực, tấn công brute-force, nhồi thông tin xác thực và nhiều loại tấn công chống lại mã thông báo.

Tiếp xúc dữ liệu quá mức

Việc lộ dữ liệu quá mức xảy ra khi một điểm cuối API phản hồi với nhiều thông tin hơn mức cần thiết để thực hiện một yêu cầu. Điều này thường xảy ra khi nhà cung cấp mong đợi người tiêu dùng API lọc kết quả; nói cách khác, khi người tiêu dùng yêu cầu thông tin cụ thể, nhà cung cấp có thể phản hồi với tất cả các loại thông tin, giả sử sau đó người tiêu dùng sẽ xóa bất kỳ dữ liệu nào họ không cần khỏi phản hồi. Khi lỗ hổng này xuất hiện, nó có thể tương đương với việc hỏi tên của ai đó và yêu cầu họ trả lời bằng tên, ngày sinh, địa chỉ email, số điện thoại và thông tin nhận dạng của mọi người khác mà họ biết.

Ví dụ: nếu một người tiêu dùng API yêu cầu thông tin về tài khoản người dùng của họ và cũng nhận được thông tin về các tài khoản người dùng khác, thì API sẽ tiết lộ quá nhiều dữ liệu. Giả sử tôi đã yêu cầu thông tin tài khoản của riêng mình với yêu cầu sau:

NHẬN /api/v3/account?name=Cloud+Strife

Bây giờ giả sử tôi đã nhận được JSON sau đây trong phản hồi:

```
{
  "id": "5501",
  "first_name": "Đám mây",
  "last_name": "Xung đột",
  "đặc quyền": "người dùng",
  "tiêu biểu": [

    "tên": "Don Corneo",
    "id": "2203"
    "email": "dcorn@gmail.com",
    "đặc quyền": "siêu quản trị viên"
    "quản trị viên": đúng
    "hai_factor_auth": sai,
  ]
}
```

Tôi đã yêu cầu thông tin tài khoản của một người dùng và nhà cung cấp đã phản hồi với thông tin về người đã tạo tài khoản của tôi, bao gồm tên đầy đủ của quản trị viên, số ID của quản trị viên và liệu quản trị viên có bật xác thực hai yếu tố hay không.

Việc lộ dữ liệu quá mức là một trong những lỗ hổng API tuyệt vời mà bỏ qua mọi biện pháp kiểm soát bảo mật tại chỗ để bảo vệ thông tin nhạy cảm và trao tất cả cho kẻ tấn công trên đĩa bạc chỉ vì chúng đã sử dụng API. Tất cả những gì bạn cần làm để phát hiện mức độ phơi nhiễm dữ liệu quá mức là kiểm tra các điểm cuối API mục tiêu của bạn và xem lại thông tin được gửi để phản hồi.

Thiếu tài nguyên và giới hạn tỷ lệ

Một trong những lỗi hỏng quan trọng hơn cần kiểm tra là thiếu tài nguyên và giới hạn tốc độ. Giới hạn tỷ lệ đóng một vai trò quan trọng trong việc kiểm tiền và tính khả dụng của API. Không giới hạn số lượng yêu cầu mà người tiêu dùng có thể thực hiện, cơ sở hạ tầng của nhà cung cấp API có thể bị quá tải bởi các yêu cầu. Quá nhiều yêu cầu mà không có đủ tài nguyên sẽ dẫn đến hệ thống của nhà cung cấp gặp sự cố và không khả dụng—trạng thái từ chối dịch vụ (DoS) .

Bên cạnh khả năng DoS-ing API, kẻ tấn công vượt qua giới hạn tốc độ có thể gây thêm chi phí cho nhà cung cấp API. Nhiều nhà cung cấp API kiểm tiền từ API của họ bằng cách giới hạn yêu cầu và cho phép khách hàng trả phí yêu cầu thêm thông tin. RapidAPI, ví dụ, cho phép 500 yêu cầu mỗi tháng miễn phí nhưng 1.000 yêu cầu mỗi tháng cho khách hàng trả tiền. Một số nhà cung cấp API cũng có cơ sở hạ tầng tự động thay đổi quy mô theo số lượng yêu cầu. Trong những trường hợp này, số lượng yêu cầu không giới hạn sẽ dẫn đến chi phí cơ sở hạ tầng tăng đáng kể và dễ ngăn chặn.

Khi thử nghiệm một API được cho là có giới hạn tốc độ, điều đầu tiên bạn nên kiểm tra xem giới hạn tốc độ có hoạt động không và bạn có thể làm như vậy bằng cách gửi một loạt yêu cầu tới API. Nếu giới hạn tốc độ đang hoạt động, bạn sẽ nhận được một số loại phản hồi thông báo rằng bạn không còn có thể thực hiện các yêu cầu bổ sung, thường ở dạng mã trạng thái HTTP 429.

Sau khi bạn bị hạn chế thực hiện các yêu cầu bổ sung, đã đến lúc thử xem giới hạn tỷ lệ được thực thi như thế nào. Bạn có thể bỏ qua nó bằng cách thêm hoặc xóa tham số, sử dụng ứng dụng khác hoặc thay đổi địa chỉ IP của mình không? Chương 13 bao gồm các biện pháp khác nhau để cố gắng bỏ qua giới hạn tốc độ.

Ủy quyền cấp chức năng bị hỏng

Ủy quyền mức chức năng bị hỏng (BFLA) là một lỗi hỏng trong đó người dùng của một vai trò hoặc nhóm có thể truy cập chức năng API của một vai trò hoặc nhóm khác. Các nhà cung cấp API thường sẽ có các vai trò khác nhau đối với các loại tài khoản khác nhau, chẳng hạn như người dùng chung, người bán, đối tác, quản trị viên, v.v. BFLA xuất hiện nếu bạn có thể sử dụng chức năng của một cấp hoặc nhóm đặc quyền khác. Nói cách khác, BFLA có thể là một động thái ngang, trong đó bạn sử dụng các chức năng của một nhóm có đặc quyền tương tự hoặc có thể là một bước leo thang đặc quyền, trong đó bạn có thể sử dụng các chức năng của một nhóm có đặc quyền hơn. Các chức năng API đặc biệt thú vị để truy cập bao gồm các chức năng xử lý thông tin nhạy cảm, tài nguyên thuộc nhóm khác và chức năng quản trị như quản lý tài khoản người dùng.

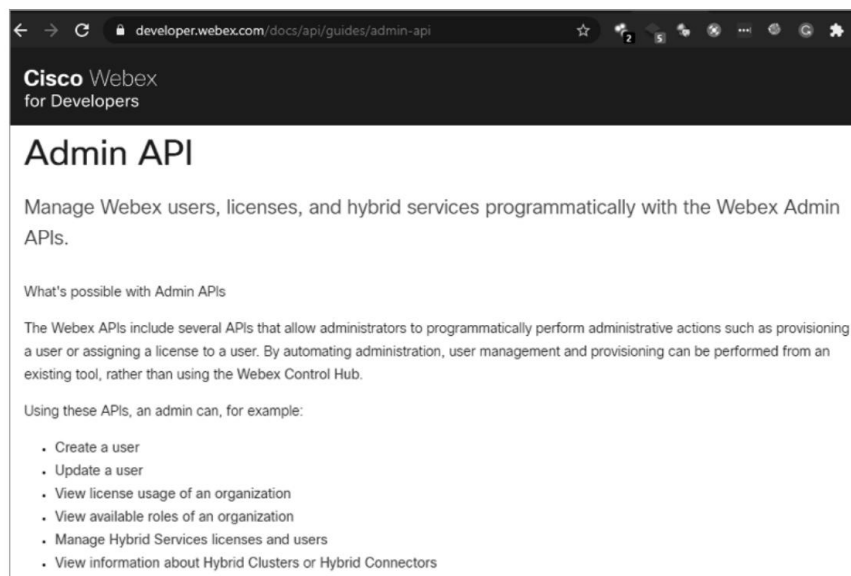
BFLA tương tự như BOLA, ngoại trừ thay vì vấn đề ủy quyền liên quan đến việc truy cập tài nguyên, đó là vấn đề ủy quyền để thực hiện các hành động. Ví dụ: hãy xem xét một API ngân hàng để bị tấn công. Khi có lỗi hỏng BOLA trong API, bạn có thể truy cập thông tin của các tài khoản khác, chẳng hạn như lịch sử thanh toán, tên người dùng, địa chỉ email và số tài khoản. Nếu có lỗi hỏng BFLA, bạn có thể chuyển tiền và thực sự cập nhật thông tin tài khoản. BOLA là về truy cập trái phép, trong khi BFLA là về các hành động trái phép.

Nếu một API có các cấp hoặc vai trò đặc quyền khác nhau, thì API đó có thể sử dụng các điểm cuối khác nhau để thực hiện các hành động đặc quyền. Ví dụ: ngân hàng có thể sử dụng điểm cuối `/user/account/balance` cho người dùng muốn truy cập thông tin tài khoản của họ và điểm cuối `/admin/account/{user}` cho quản trị viên muốn truy cập thông tin tài khoản người dùng. Nếu ứng dụng không có các kiểm soát truy cập được triển khai đúng cách, chúng tôi sẽ có thể thực hiện các hành động quản trị, chẳng hạn như xem chi tiết tài khoản đầy đủ của người dùng, chỉ bằng cách thực hiện các yêu cầu quản trị.

Không phải lúc nào API cũng sử dụng điểm cuối quản trị cho chức năng quản trị. Thay vào đó, chức năng này có thể dựa trên các phương thức yêu cầu HTTP chẳng hạn như GET, POST, PUT và DELETE. Nếu nhà cung cấp không hạn chế các phương thức HTTP mà người tiêu dùng có thể sử dụng, thì chỉ cần thực hiện một yêu cầu trái phép bằng một phương thức khác có thể chỉ ra lỗ hổng BFLA.

Khi tìm kiếm BFLA, hãy tìm bất kỳ chức năng nào bạn có thể sử dụng để tạo lợi thế cho mình, bao gồm thay đổi tài khoản người dùng, truy cập tài nguyên người dùng và giành quyền truy cập vào các điểm cuối bị hạn chế. Ví dụ: nếu API cung cấp cho đối tác khả năng thêm người dùng mới vào nhóm đối tác nhưng không hạn chế chức năng này đối với nhóm cụ thể, thì bất kỳ người dùng nào cũng có thể tự thêm họ vào bất kỳ nhóm nào. Hơn nữa, nếu chúng tôi có thể tự thêm mình vào một nhóm, thì rất có thể chúng tôi sẽ có thể truy cập tài nguyên của nhóm đó.

Cách dễ nhất để khám phá BFLA là tìm tài liệu API quản trị và gửi yêu cầu với tư cách là người dùng không có đặc quyền để kiểm tra các chức năng và khả năng của quản trị viên. Hình 3-1 hiển thị tài liệu API Quản trị viên Webex công khai của Cisco, tài liệu này cung cấp danh sách các hành động hữu ích để thử nếu bạn đang thử nghiệm Cisco Webex.



Hình 3-1: Tài liệu API quản trị viên Webex của Cisco

Là người dùng không có đặc quyền, hãy đưa ra các yêu cầu trong phần quản trị, chẳng hạn như cố gắng tạo người dùng, cập nhật tài khoản người dùng, v.v. Nếu có các biện pháp kiểm soát truy cập, bạn có thể sẽ nhận được phản hồi HTTP 401 Trái phép hoặc 403 Bị cấm. Tuy nhiên, nếu bạn có thể thực hiện yêu cầu thành công, bạn đã phát hiện ra lỗ hổng BFLA.

Nếu tài liệu API cho các hành động đặc quyền không có sẵn, bạn sẽ cần khám phá hoặc thiết kế ngược các điểm cuối được sử dụng để thực hiện các hành động đặc quyền trước khi thử nghiệm chúng; thêm về điều này trong Chương 7. Khi bạn đã tìm thấy các điểm cuối quản trị, bạn có thể bắt đầu đưa ra các yêu cầu.

chuyển nhượng hàng loạt

Chuyển nhượng hàng loạt xảy ra khi người tiêu dùng API bao gồm nhiều tham số hơn trong yêu cầu của họ so với ứng dụng dự định và ứng dụng thêm các tham số này vào các biến mã hoặc đối tượng nội bộ. Trong tình huống này, người tiêu dùng có thể chỉnh sửa thuộc tính đối tượng hoặc leo thang đặc quyền.

Ví dụ: một ứng dụng có thể có chức năng cập nhật tài khoản mà người dùng chỉ nên sử dụng để cập nhật tên người dùng, mật khẩu và địa chỉ của họ. Nếu người tiêu dùng có thể bao gồm các tham số khác trong yêu cầu liên quan đến tài khoản của họ, chẳng hạn như cấp đặc quyền tài khoản hoặc thông tin nhạy cảm như số dư tài khoản và ứng dụng chấp nhận các tham số đó mà không cần kiểm tra chúng dựa trên danh sách trắng các hành động được phép, thì người tiêu dùng có thể tận dụng lợi thế của điểm yếu này để thay đổi các giá trị này.

Hãy tưởng tượng một API được gọi để tạo một tài khoản với các tham số cho "Người dùng" và "Mật khẩu":

```
{
  "Người dùng": "scuttlephish",
  "Mật khẩu": "GreatPassword123"
}
```

Trong khi đọc tài liệu API về quy trình tạo tài khoản, giả sử bạn phát hiện ra rằng có một khóa bổ sung, "isAdmin", mà người tiêu dùng có thể sử dụng để trở thành quản trị viên. Bạn có thể sử dụng công cụ như Postman hoặc Burp Suite để thêm thuộc tính vào yêu cầu và đặt giá trị thành true:

```
{
  "Người dùng": "scuttlephish",
  "Mật khẩu": "GreatPassword123",
  "isAdmin": đúng
}
```

Nếu API không làm sạch đầu vào yêu cầu, nó sẽ dễ bị tấn công hàng loạt và bạn có thể sử dụng yêu cầu được cập nhật để tạo tài khoản quản trị viên. Ở phần phụ trợ, ứng dụng web dễ bị tấn công sẽ thêm thuộc tính khóa/giá trị, {"isAdmin": "true"}, vào đối tượng người dùng và đặt người dùng đó tương đương với quản trị viên.

Bạn có thể phát hiện ra các lỗ hổng chuyển nhượng hàng loạt bằng cách tìm những điều thú vị tham số trong tài liệu API và sau đó thêm các tham số đó vào yêu cầu. Tìm kiếm các tham số liên quan đến thuộc tính tài khoản người dùng, chức năng quan trọng và hành động quản trị. Việc chặn các yêu cầu và phản hồi API cũng có thể tiết lộ các tham số đáng để thử nghiệm. Ngoài ra, bạn có thể đoán các tham số hoặc làm mờ chúng trong các yêu cầu API. (Chương 9 mô tả nghệ thuật làm mờ.)

Cấu hình sai bảo mật

Cấu hình sai bảo mật bao gồm tất cả các lỗi mà nhà phát triển có thể mắc phải trong cấu hình bảo mật hỗ trợ của API. Nếu cấu hình sai bảo mật đủ nghiêm trọng, nó có thể dẫn đến việc lộ thông tin nhạy cảm hoặc chiếm toàn bộ hệ thống. Ví dụ: nếu cấu hình bảo mật hỗ trợ của API cho thấy một lỗ hổng chưa được vá, thì có khả năng kẻ tấn công có thể tận dụng một khai thác đã xuất bản để dễ dàng “pwn” API và hệ thống của nó.

Cấu hình sai bảo mật thực sự là một tập hợp các điểm yếu bao gồm các tiêu đề được định cấu hình sai, mã hóa chuyển tiếp được định cấu hình sai, sử dụng tài khoản mặc định, chấp nhận các phương thức HTTP không cần thiết, thiếu vệ sinh đầu vào và thông báo lỗi dài dòng.

Việc thiếu vệ sinh đầu vào có thể cho phép kẻ tấn công tải các khoản thanh toán độc hại lên máy chủ. API thường đóng một vai trò quan trọng trong việc tự động hóa các quy trình, vì vậy hãy tưởng tượng có thể tải lên các tải trọng mà máy chủ tự động xử lý thành một định dạng có thể được thực thi hoặc thực thi từ xa bởi người dùng cuối không nghi ngờ. Ví dụ: nếu một điểm cuối tải lên được sử dụng để chuyển các tệp đã tải lên tới một thư mục web, thì điểm cuối đó có thể cho phép tải lên tập lệnh. Điều hướng đến URL chứa tệp có thể khởi chạy tập lệnh, dẫn đến truy cập trình bao trực tiếp vào máy chủ web. Ngoài ra, việc thiếu vệ sinh đầu vào có thể dẫn đến hành vi không mong muốn trên một phần của ứng dụng. Trong Phần III, chúng tôi sẽ làm mờ các đầu vào API trong nỗ lực khám phá các lỗ hổng như cấu hình sai bảo mật, quản lý nội dung không phù hợp và các điểm yếu của phần tìm.

Các nhà cung cấp API sử dụng các tiêu đề để cung cấp cho người tiêu dùng hướng dẫn xử lý các yêu cầu bảo mật và phản hồi. Các tiêu đề được định cấu hình sai có thể dẫn đến tiết lộ thông tin nhạy cảm, tấn công hạ cấp và tấn công kịch bản chéo trang. Nhiều nhà cung cấp API sẽ sử dụng các dịch vụ bổ sung cùng với API của họ để nâng cao các chỉ số liên quan đến API hoặc để cải thiện tính bảo mật. Việc các dịch vụ bổ sung đó thêm tiêu đề vào các yêu cầu về số liệu và có thể đóng vai trò là một mức độ đảm bảo nào đó đối với người tiêu dùng là điều khá phổ biến. Ví dụ: lấy phản hồi sau:

```
HTTP/200 OK
--snip--
X-Powered-By: VulnService 1.11
Bảo vệ X-XSS: 0
Thời gian phản hồi X: 566,43
```

Tiêu đề X-Powered-By tiết lộ công nghệ phụ trợ. Tiêu đề như thế này một người sẽ thường quảng cáo chính xác dịch vụ hỗ trợ và phiên bản của nó. Bạn có thể sử dụng thông tin như thế này để tìm kiếm các khai thác được xuất bản cho phiên bản phần mềm đó.

X-XSS-Protection chính xác như thế này: một tiêu đề có nghĩa là để ngăn chặn các cuộc tấn công kịch bản chéo trang (XSS). XSS là một loại lỗ hổng tiêm nhiễm phổ biến, trong đó kẻ tấn công có thể chèn tập lệnh vào trang web và lừa người dùng cuối nhấp vào các liên kết độc hại. Chúng tôi sẽ đề cập đến XSS và tập lệnh API chéo (XAS) trong Chương 12. Giá trị Bảo vệ X-XSS bằng 0 cho biết không có biện pháp bảo vệ nào được áp dụng và giá trị bằng 1 cho biết rằng bảo vệ đã được bật. Tiêu đề này và những tiêu đề khác giống như vậy, tiết lộ rõ ràng liệu có kiểm soát an ninh hay không.

Tiêu đề X-Response-Time là phần mềm trung gian cung cấp số liệu sử dụng. Trong ví dụ trước, giá trị của nó đại diện cho 566,43 mili giây. Tuy nhiên, nếu API không được định cấu hình đúng cách, tiêu đề này có thể hoạt động như một kênh phụ được sử dụng để hiển thị các tài nguyên hiện có. Ví dụ : nếu tiêu đề X-Response-Time có thời gian phản hồi nhất quán cho các bản ghi không tồn tại, nhưng tăng thời gian phản hồi của nó đối với một số bản ghi khác, thì đây có thể là dấu hiệu cho thấy những bản ghi đó tồn tại. Đây là một ví dụ:

Không tìm thấy HTTP/Người dùngA 404

--snip--

Thời gian phản hồi X: 25,5

Không tìm thấy HTTP/Người dùngB 404

--snip--

Thời gian phản hồi X: 25,5

Không tìm thấy HTTP/UserC 404

--snip--

Thời gian phản hồi X: 510,00

Trong trường hợp này, UserC có giá trị thời gian phản hồi gấp 20 lần thời gian phản hồi của các tài nguyên khác. Với kích thước mẫu nhỏ này, thật khó để kết luận chắc chắn rằng UserC có tồn tại hay không. Tuy nhiên, hãy tưởng tượng bạn có một mẫu gồm hàng trăm hoặc hàng nghìn yêu cầu và biết X-Response trung bình -Giá trị thời gian cho một số tài nguyên hiện có và không tồn tại. Chẳng hạn, giả sử bạn biết rằng một tài khoản không có thật như /user/account/thisdefinitelydoesnotexist876 có Thời gian phản hồi X trung bình là 25,5 ms. Bạn cũng biết rằng tài khoản /user/account/1021 hiện tại của bạn nhận được Thời gian phản hồi X là 510,00. Nếu sau đó bạn đã gửi yêu cầu cưỡng bức tất cả các số tài khoản từ 1000 đến 2000, thì bạn có thể xem lại kết quả và xem số tài khoản nào dẫn đến thời gian phản hồi tăng đáng kể.

Bất kỳ API nào cung cấp thông tin nhạy cảm cho người tiêu dùng nên sử dụng Bảo mật tầng vận chuyển (TLS) để mã hóa dữ liệu. Ngay cả khi API chỉ được cung cấp nội bộ, riêng tư hoặc ở cấp đối tác, việc sử dụng TLS, giao thức mã hóa lưu lượng HTTPS, là một trong những cách cơ bản nhất để đảm bảo rằng các yêu cầu và phản hồi API được bảo vệ khi được truyền qua mạng. . Mã hóa quá cảnh bị định cấu hình sai hoặc thiếu có thể khiến người dùng API chuyển thông tin API nhạy cảm ở dạng văn bản rõ ràng qua các mạng, trong trường hợp đó

kẻ tấn công có thể nắm bắt các phản hồi và yêu cầu bằng một cuộc tấn công trung gian (MITM) và đọc chúng một cách rõ ràng. Kẻ tấn công cần có quyền truy cập vào cùng một mạng với người mà chúng đang tấn công và sau đó chặn lưu lượng mạng bằng bộ phân tích giao thức mạng như Wireshark để xem thông tin được truyền đạt giữa người tiêu dùng và nhà cung cấp.

Khi một dịch vụ sử dụng thông tin xác thực và tài khoản mặc định và mặc định đã biết, kẻ tấn công có thể sử dụng các thông tin xác thực đó để đảm nhận vai trò của tài khoản đó. Điều này có thể cho phép họ có quyền truy cập vào thông tin nhạy cảm hoặc chức năng quản trị, có khả năng dẫn đến sự xâm phạm của các hệ thống hỗ trợ.

Cuối cùng, nếu nhà cung cấp API cho phép các phương thức HTTP không cần thiết, thì sẽ có nguy cơ gia tăng là ứng dụng sẽ không xử lý các phương thức này đúng cách hoặc sẽ dẫn đến việc tiết lộ thông tin nhạy cảm.

Bạn có thể phát hiện một số cấu hình sai bảo mật này bằng các trình quét lỗ hổng ứng dụng web như Nessus, Qualys, OWASP ZAP và Nikto. Các trình quét này sẽ tự động kiểm tra thông tin về phiên bản máy chủ web, tiêu đề, cookie, cấu hình mã hóa chuyển tiếp và các tham số để xem liệu các biện pháp bảo mật dự kiến có bị thiếu hay không. Bạn cũng có thể kiểm tra các cấu hình sai bảo mật này theo cách thủ công, nếu bạn biết mình đang tìm gì, bằng cách kiểm tra tiêu đề, chứng chỉ SSL, cookie và tham số.

tiêm

Lỗi chèn tồn tại khi một yêu cầu được chuyển đến cấu trúc cơ sở hạ tầng hỗ trợ của API và nhà cung cấp API không lọc đầu vào để loại bỏ các ký tự không mong muốn (một quy trình được gọi là khử trùng đầu vào). Kết quả là, cơ sở hạ tầng có thể xử lý dữ liệu từ yêu cầu dưới dạng mã và chạy nó. Khi loại lỗ hổng này xuất hiện, bạn sẽ có thể tiến hành các cuộc tấn công tiêm nhiễm như tiêm nhiễm SQL, tiêm nhiễm NoSQL và tiêm nhiễm hệ thống.

Trong mỗi cuộc tấn công tiêm nhiễm này, API sẽ phân phối trực tiếp tải trọng chưa được làm sạch của bạn tới hệ điều hành đang chạy ứng dụng hoặc cơ sở dữ liệu của nó. Do đó, nếu bạn gửi một tải trọng chứa các lệnh SQL tới một API để bị tổn thương sử dụng cơ sở dữ liệu SQL, thì API sẽ chuyển các lệnh tới cơ sở dữ liệu, cơ sở dữ liệu này sẽ xử lý và thực hiện các lệnh. Điều tương tự cũng xảy ra với cơ sở dữ liệu NoSQL để bị tổn thương và các hệ thống bị ảnh hưởng.

Thông báo lỗi chi tiết, mã phản hồi HTTP và API không mong muốn hành vi đều có thể là manh mối mà bạn có thể đã phát hiện ra một lỗ hổng tiêm nhiễm. Ví dụ: giả sử bạn gửi `OR 1=0--` làm địa chỉ trong quy trình đăng ký tài khoản. API có thể chuyển trực tiếp tải trọng đó tới cơ sở dữ liệu SQL phụ trợ, nơi câu lệnh `OR 1=0` sẽ không thành công (vì 1 không bằng 0), gây ra một số lỗi SQL:

```
ĐĂNG /api/v1/đăng ký HTTP 1.1
```

```
Máy chủ: example.com
```

```
--snip--
```

```
{
```

```
"Tên": "hAPI",
```



```
"Tên": "Hacker",
"Địa chỉ":      "HỒI 1=0--",
}
```

Một lỗi trong cơ sở dữ liệu phụ trợ có thể hiển thị dưới dạng phản hồi cho người tiêu dùng. Trong trường hợp này, bạn có thể nhận được phản hồi như "Lỗi: Bạn gặp lỗi trong cú pháp SQL của mình. . . ." Bất kỳ phản hồi trực tiếp nào từ cơ sở dữ liệu hoặc hệ thống hỗ trợ đều là dấu hiệu rõ ràng cho thấy có lỗ hổng tiềm ẩn.

Các lỗ hổng tiềm ẩn thường được bổ sung bởi các lỗ hổng khác như vệ sinh đầu vào kém. Trong ví dụ sau, bạn có thể thấy một cuộc tấn công chèn mã sử dụng yêu cầu API GET để tận dụng một tham số truy vấn yếu. Trong trường hợp này, tham số truy vấn yếu chuyển bất kỳ dữ liệu nào trong phần truy vấn của yêu cầu trực tiếp đến hệ thống bên dưới mà không cần làm sạch dữ liệu đó trước:

```
NHẬN http://10.10.78.181:5000/api/v1/resources/books?show=/etc/passwd
```

Nội dung phản hồi sau đây cho thấy rằng điểm cuối API đã bị thao túng để hiển thị tệp /etc/passwd của máy chủ lưu trữ, tiết lộ người dùng trên hệ thống:

```
gốc:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/dev:/usr/sbin/nologin
đồng bộ hóa:x:4:65534:sync:/bin:/bin/sync
trò chơi:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
thư:x:8:8:mail:/var/mail:/usr/sbin/nologin
tin tức:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

Việc tìm ra các lỗ hổng tiềm ẩn yêu cầu kiểm tra cẩn thận các điểm cuối API, chú ý đến cách API phản hồi, sau đó tạo các yêu cầu cố gắng thao túng các hệ thống phụ trợ. Giống như các cuộc tấn công duyệt thư mục, các cuộc tấn công tiêm chích đã xuất hiện trong nhiều thập kỷ, vì vậy có nhiều biện pháp kiểm soát bảo mật tiêu chuẩn để bảo vệ các nhà cung cấp API khỏi chúng. Tôi sẽ trình bày các phương pháp khác nhau để thực hiện các cuộc tấn công tiêm chích, mã hóa lưu lượng và vượt qua các kiểm soát tiêu chuẩn trong Chương 12 và 13.

Quản lý tài sản không phù hợp

Việc quản lý tài sản không phù hợp diễn ra khi một tổ chức để lộ các API đã ngừng hoạt động hoặc vẫn đang được phát triển. Như với bất kỳ phần mềm nào, các phiên bản API cũ có nhiều khả năng chứa lỗ hổng hơn vì chúng không còn được vá và nâng cấp nữa. Tương tự như vậy, các API vẫn đang được phát triển thường không an toàn như các đối tác API sản xuất của chúng.

Quản lý tài sản không phù hợp có thể dẫn đến các lỗ hổng khác, chẳng hạn như lộ dữ liệu quá mức, tiết lộ thông tin, chỉ định hàng loạt, giới hạn tỷ lệ không phù hợp và tiêm API. Đối với những kẻ tấn công, điều này có nghĩa là

phát hiện lỗi hỏng quản lý nội dung không phù hợp chỉ là bước đầu tiên để tiếp tục khai thác API.

Bạn có thể phát hiện ra việc quản lý nội dung không phù hợp bằng cách chú ý đến tài liệu API lỗi thời, nhật ký thay đổi và lịch sử phiên bản trên các kho lưu trữ. Ví dụ: nếu tài liệu API của một tổ chức chưa được cập nhật cùng với các điểm cuối của API, tài liệu đó có thể chứa các tham chiếu đến các phần của API không còn được hỗ trợ. Các tổ chức thường bao gồm thông tin về phiên bản trong tên điểm cuối của họ để phân biệt giữa các phiên bản cũ hơn và mới hơn, chẳng hạn như /v1/, /v2/, /v3/, v.v. Các API vẫn đang trong quá trình phát triển thường sử dụng các đường dẫn như /alpha/, /beta/, /test/, /uat/ và /demo/. Nếu bạn biết rằng một API hiện đang sử dụng `apiv3.org/admin` nhưng một phần của tài liệu API đề cập đến `apiv1.org/admin`, thì bạn có thể thử kiểm tra các điểm cuối khác nhau để xem liệu `apiv1` hoặc `apiv2` có còn hoạt động hay không. Ngoài ra, nhật ký thay đổi của tổ chức có thể tiết lộ lý do tại sao v1 được cập nhật hoặc ngừng hoạt động. Nếu bạn có quyền truy cập vào v1, bạn có thể kiểm tra những điểm yếu đó.

Ngoài việc sử dụng tài liệu, bạn có thể phát hiện ra các lỗi hỏng quản lý tài sản không phù hợp thông qua việc sử dụng các yêu cầu phỏng đoán, làm mờ hoặc vũ phu. Theo dõi các mẫu trong tài liệu API hoặc lược đồ đặt tên đường dẫn, sau đó đưa ra các yêu cầu dựa trên các giả định của bạn.

Lỗi hỏng logic kinh doanh

Lỗi hỏng logic nghiệp vụ (còn được gọi là lỗi hỏng logic nghiệp vụ hoặc BLF) là các tính năng dự kiến của ứng dụng mà kẻ tấn công có thể sử dụng với mục đích xấu. Ví dụ: nếu API có tính năng tải lên không xác thực tải trọng được mã hóa, thì người dùng có thể tải lên bất kỳ tệp nào miễn là tệp đó đã được mã hóa. Điều này sẽ cho phép người dùng cuối tải lên và thực thi mã tùy ý, bao gồm cả tải trọng độc hại.

Các lỗi hỏng kiểu này thường xuất phát từ giả định rằng người tiêu dùng API sẽ tuân theo chỉ dẫn, đáng tin cậy hoặc chỉ sử dụng API theo một cách nhất định. Trong những trường hợp đó, tổ chức về cơ bản phụ thuộc vào niềm tin như một biện pháp kiểm soát an ninh bằng cách mong đợi người tiêu dùng hành động nhân từ. Thật không may, ngay cả những người tiêu dùng API tốt bụng cũng mắc lỗi có thể dẫn đến việc ứng dụng bị xâm phạm.

Sự cố rò rỉ API của đối tác Experian vào đầu năm 2021 là một ví dụ điển hình về lỗi tin cậy API. Một đối tác Experian nhất định được ủy quyền sử dụng API của Experian để thực hiện kiểm tra tín dụng, nhưng đối tác này đã thêm chức năng kiểm tra tín dụng của API vào ứng dụng web của họ và vô tình làm lộ tất cả các yêu cầu cấp đối tác cho người dùng. Một yêu cầu có thể bị chặn khi sử dụng ứng dụng web của đối tác và nếu yêu cầu đó bao gồm tên và địa chỉ, API Experian sẽ phản hồi với điểm tín dụng và các yếu tố rủi ro tín dụng của cá nhân đó. Một trong những nguyên nhân hàng đầu của lỗi hỏng logic nghiệp vụ này là Experian đã tin tưởng đối tác sẽ không tiết lộ API.

Một vấn đề khác với niềm tin là thông tin đăng nhập, chẳng hạn như khóa API, mã thông báo và mật khẩu, liên tục bị đánh cắp và rò rỉ. Khi thông tin đăng nhập của người tiêu dùng đáng tin cậy bị đánh cắp, người tiêu dùng có thể trở thành sói trong bầy cừu.

quần áo và làm cho havoc. Nếu không có các biện pháp kiểm soát kỹ thuật mạnh mẽ, các lỗ hổng logic kinh doanh thường có thể có tác động đáng kể nhất, dẫn đến việc khai thác và thỏa hiệp.

Bạn có thể tìm kiếm tài liệu API để biết các dấu hiệu nhận biết lỗ hổng logic nghiệp vụ. Những tuyên bố như sau sẽ thấp sáng bóng đèn trên đầu bạn:

“Chỉ sử dụng tính năng X để thực hiện chức năng Y.”

“Không làm X với điểm cuối Y.”

"Chỉ quản trị viên mới nên thực hiện yêu cầu X."

Những tuyên bố này có thể cho biết rằng nhà cung cấp API tin tưởng rằng bạn sẽ không thực hiện bất kỳ hành động không được khuyến khích nào như đã hướng dẫn. Khi bạn tấn công API của họ, hãy đảm bảo không tuân theo các yêu cầu đó để kiểm tra sự hiện diện của các biện pháp kiểm soát bảo mật.

Một lỗ hổng logic nghiệp vụ khác xuất hiện khi các nhà phát triển cho rằng người tiêu dùng sẽ chỉ sử dụng trình duyệt để tương tác với ứng dụng web và sẽ không nắm bắt các yêu cầu API diễn ra ở hậu trường. Tất cả những gì cần thiết để khai thác loại điểm yếu này là chặn các yêu cầu bằng một công cụ như Burp Suite Proxy hoặc Postman, sau đó thay đổi yêu cầu API trước khi nó được gửi đến nhà cung cấp. Điều này có thể cho phép bạn nắm bắt các khóa API được chia sẻ hoặc sử dụng các tham số có thể tác động tiêu cực đến tính bảo mật của ứng dụng.

Ví dụ, hãy xem xét một cổng xác thực ứng dụng web mà người dùng thường sử dụng để xác thực tài khoản của họ. Giả sử ứng dụng web đã đưa ra yêu cầu API sau:

```
ĐĂNG /api/v1/đăng nhập HTTP 1.1
Máy chủ: example.com
--snip--
UserId=hapihacker&password=arealpassword&MFA=true
```

Có khả năng chúng ta có thể bỏ qua xác thực đa yếu tố bằng cách thay đổi tham số MFA thành false.

Việc kiểm tra các lỗi logic nghiệp vụ có thể là một thách thức vì mỗi doanh nghiệp ness là duy nhất. Các máy quét tự động sẽ gặp khó khăn trong việc phát hiện các sự cố này vì các lỗi này là một phần mục đích sử dụng của API. Bạn phải hiểu cách hoạt động của doanh nghiệp và API, sau đó xem xét cách bạn có thể sử dụng các tính năng này để tạo lợi thế cho mình. Nghiên cứu logic kinh doanh của ứng dụng với tư duy đối nghịch và thử phá vỡ mọi giả định đã được đưa ra.

Bản tóm tắt

Trong chương này, tôi đã đề cập đến các lỗ hổng API phổ biến. Điều quan trọng là phải làm quen với các lỗ hổng này để bạn có thể dễ dàng nhận ra chúng, tận dụng chúng trong quá trình tương tác và báo cáo chúng

trở lại tổ chức để ngăn bọn tội phạm kéo khách hàng của bạn lên tiêu đề.

Bây giờ bạn đã quen thuộc với các ứng dụng web, API và những điểm yếu của chúng, đã đến lúc chuẩn bị máy hack của bạn và bạn rộn với bàn phím.