Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №8**

**Дисциплина: Информационная безопасность**

Работу выполнил: Романов В.В.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шиян В.И.

**Цель:** с помощью систем нагрузочного тестирования определить производительность web-серверов Apache и Nginx, добиться отказа в обслуживании.

**Ход работы:**

#### **Часть 1**

Настроим облако и установим веб-сервер. Для этого в облаке установим 2 виртуальных сервера (VPS), на одном из которых установим Nginx с помощью команды: **sudo apt install nginx**.

Теперь установим ПО для мониторинга атакуемого сервера. Для этого на атакуемом сервере установим следующие утилиты:

1. nload – для мониторинга сетевой активности;
2. iptraf – для анализа сетевого трафика;
3. top – для отслеживания состояния системы.

После этого установим ПО для атакующего сервера. Для этого установим следующие утилиты:

1. **Slowloris** – для медленных атак:

sudo pip3 install slowloris

slowloris <domain/address> -p <port> -s <connections>

1. **Torshammer –** для атак методом POST/GET:

git clone https://github.com/Karlheinzniebuhr/torshammer.git

cd torshammer

python torshammer.py -t <domain/socket> -p <port>

1. **hping3** для выполнения flood-атак:

sudo apt install hping3

hping3 <domain/address> --flood --rand-source --icmp -c 25000

**Мониторинг начального состояния системы:**

Проведем мониторинг состояния атакуемой системы до начала атак (для фиксации базовых метрик). На рисунках 1-2 изображён результат.

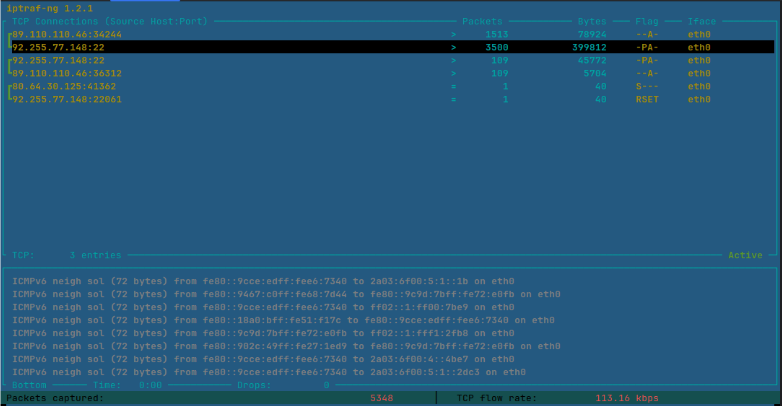


Рисунок 1 – Интерфейс

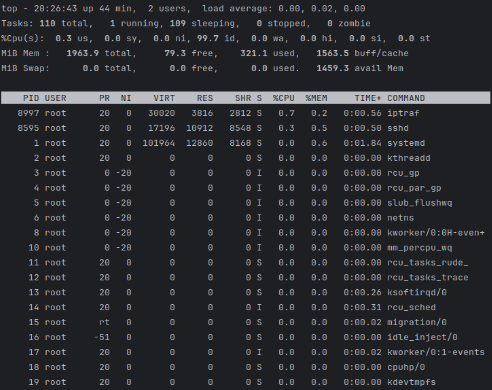


Рисунок 2 – Мониторинг состояния системы

**Атака Slowloris на Nginx:**

Перед атакой увеличим ограничение на количество открытых файлов на атакующем сервере: ulimit -S -n $(ulimit -Hn)

В процессе атаки уровень нагрузки (LA) за последнюю минуту достиг 0.4. На рисунках 3-4 представлен результат.

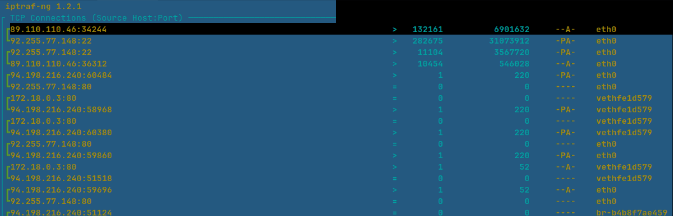


Рисунок 3 – Интерфейс

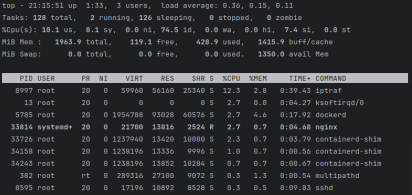


Рисунок 4 – Мониторинг состояния системы

Сервер Nginx был выведен из строя из-за превышения предела worker\_connections. На рисунке 5 представлен результат.

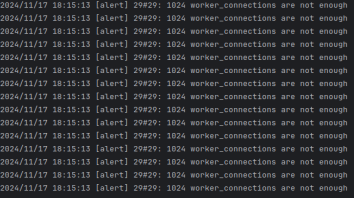


Рисунок 5 – Превышение предела рабочих подключений

Теперь выполним операции по защите сервера. Для этого необходимо:

1. Установить ограничение на длительность соединения (5 секунд) и максимальное число подключений (10) с одного IP.
2. Добавить правила в **iptables**:

iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 30 -j DROP

iptables -I INPUT -p tcp --dport 80 -m state --state NEW -m recent --set

iptables -I INPUT -p tcp --dport 80 -m state --state NEW -m recent --update --sec

На рисунке 6 представлен результат.

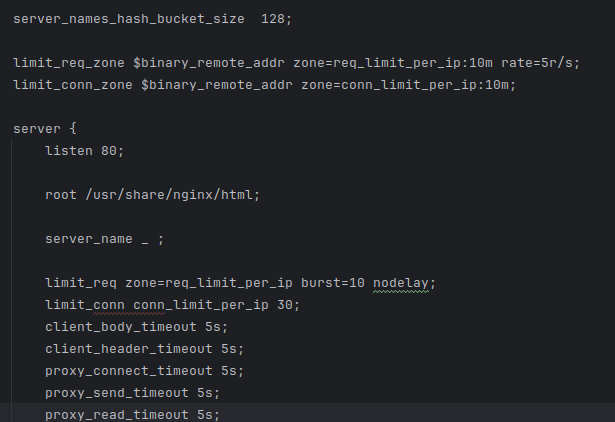


Рисунок 6 – Защита сервера

**Атака hping3 на Nginx (SYN flood):**

Выполним flood-атаку с помощью команды: hping3 -S -p 80 --flood <IP-адрес>.

Результаты атаки: уровень нагрузки (LA) достиг 0.8, HTTP-сервер стал недоступен и зафиксирован высокий уровень TCP-пакетов. На рисунках 7-8 представлен результат.

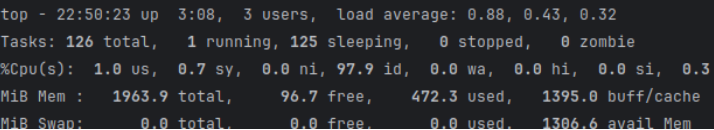


Рисунок 7 – Результаты

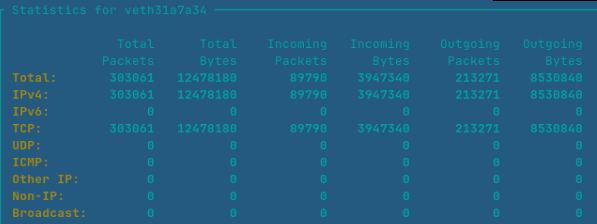


Рисунок 8 – Результаты

### **Часть 2**

Установим веб-сервер. Для этого воспользуемся командой: sudo apt install apache2.

Просмотрим текущие правила iptables. Для этого воспользуемся командой: sudo iptables -L. В выводе отображены 3 цепочки (по умлч. они настроены на политику ACCEPT):

1. INPUT;
2. OUTPUT;
3. FORWARD.

Просмотрим список правил в формате активации команд: sudo iptables -S.

Сбросим текущие правила: sudo iptables -F.

Настроим цепочку INPUT. Для разрешения локального трафика добавим правило: sudo iptables -A INPUT -i lo -j ACCEPT.

Настроим исключения для SSH и HTTP. Для этого разрешим подключения на порты SSH (22) и HTTP (80):

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

Добавим правило для существующих соединений: sudo iptables -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT.

Установим политику по умолчанию:

sudo iptables -P OUTPUT ACCEPT.

sudo iptables -P INPUT DROP.

Просмотрим текущие правила iptables: sudo iptables -L.

**Блокировка нулевых пакетов:**

Было добавлено правило для блокировки нулевых пакетов, которые могут использоваться для разведки уязвимостей: sudo iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP.

**Защита от SYN flood атак:**

Было добавлено правило для блокировки новых соединений с некорректными TCP-флагами: sudo iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP.

**Блокировка XMAS-пакетов:**

Было добавлено правило для защиты от XMAS-атак: sudo iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP.

Проверим защиту от XMAS-сканирования. Для этого с атакующей машины выполнено XMAS-сканирование с использованием **nmap**: sudo nmap -sX <target IP>.

**Сохраним правила iptables**, так как они теряются после перезагрузки. Для этого был установлен пакет **iptables-persistent**: sudo apt install iptables-persistent. Во время установки текущие правила были сохранены для автоматической загрузки после перезагрузки системы.

1. Защитим сервер от разведывательных пакетов XMAS <sudo iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP>. Теперь сервер защищен от некоторых общих атак, которые ищут его уязвимости.
2. Со второй ВМ, на которую установите nmap, проведите XMAS сканирование <sudo nmap -sX>.

На рисунках 9-11 представлен результат.



Рисунок 9 – Результат



Рисунок 10 – Результат

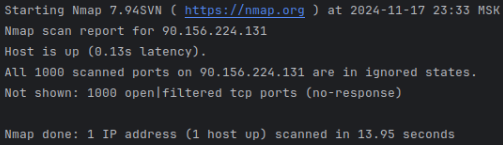


Рисунок 11 – Результат

По умолчанию все не сохраненные правила действуют до следующей перезагрузки сервера; сразу же после перезагрузки не сохраненные правила будут потеряны. Самый простой способ загрузить пакет iptables-persistent.

Во время инсталляции пакет уточнит, нужно ли сохранить текущие правила для дальнейшей автоматической загрузки, если текущие правила были протестированы и соответствуют всем требованиям, их можно сохранить.

### **ЧАСТЬ 3**

**Установка набора LAMP:**

На сервер была установлена связка LAMP (Linux, Apache, MySQL, PHP):

sudo apt-get update

sudo apt-get install tasksel

sudo tasksel

В меню tasksel был выбран LAMP, после чего установка завершилась успешным запуском веб-сервера Apache.

**Установка ModSecurity:**

Выполнена установка модуля ModSecurity для защиты веб-сервера: sudo apt install libapache2-mod-security2.

Проверим успешность установки. Для этого проверим наличие модуля с писке активных: sudo apachectl -M | grep --color security2.

Логирование событий ModSecurity. Логи модификаций записываются в файл: /var/log/apache2/modsec\_audit.log

Настройка конфигурации ModSecurity. Основной файл конфигурации был переименован: sudo mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf

Для применения изменений выполним перезапуск Apache: sudo systemctl restart apache2.

Изменим режим работы ModSecurity. Для этого в конфигурационном файле изменим режим работы с DetectionOnly на On:

sudo nano /etc/modsecurity/modsecurity.conf

Изменим строку ***SecRuleEngine DetectionOnly*** на ***SecRuleEngine On***.

Отключим буферизацию тела ответа. Для уменьшения нагрузки на сервер отключена директива **SecResponseBodyAccess**: SecResponseBodyAccess Off.

Подключим базовый набор правил CRS. Для этого укажем пути для подключения правил CRS: sudo nano /etc/apache2/mods-enabled/security2.conf.

Добавим строки:

Include "/usr/share/modsecurity-crs/\*.conf"

Include "/usr/share/modsecurity-crs/activated\_rules/\*.conf"

Активируем готовые правила. Для этого создадим символические ссылки на выбранные правила:

cd /usr/share/modsecurity-crs/activated\_rules/

sudo ln -s /usr/share/modsecurity-crs/base\_rules/modsecurity\_crs\_30\_http\_policy.conf

sudo ln -s /usr/share/modsecurity-crs/base\_rules/modsecurity\_crs\_49\_generic\_attacks.conf

Перезапустим Apache для применения правил: sudo systemctl restart apache2.

Проверим логи ModSecurity. После активации правил, в каталоге логов Apache можно отслеживать события, связанные с правилами ModSecurity: tail -f /var/log/apache2/modsec\_audit.log.

Это помогает отследить, какие запросы блокируются, и адаптировать правила в случае ложных срабатываний.

Протестируем защиту ModSecurity. Для проверки работы правил ModSecurity можно использовать инструмент **curl** с отправкой запросов, которые соответствуют активным правилам. Например: curl -H "User-Agent: Nmap Scripting Engine" http://<IP-адрес сервера>.

В случае успешного срабатывания правило будет зафиксировано в логах, а запрос отклонен.

Активируем дополнительные правила ModSecurity. Для усиления защиты добавлены дополнительные символические ссылки на правила из других каталогов:

sudo ln -s /usr/share/modsecurity-crs/base\_rules/modsecurity\_crs\_40\_generic\_attacks.conf

sudo ln -s /usr/share/modsecurity-crs/optional\_rules/modsecurity\_crs\_60\_trusted\_hosts.conf

Настроим специфические правила ModSecurity. Для защиты от определенных атак, таких как SQL-инъекции, можно добавлять собственные правила в конфигурационный файл: sudo nano /etc/modsecurity/modsecurity.conf.

**Пример** правила: SecRule ARGS "@detectXSS" "id:12345,deny,status:403,msg:'Cross-site scripting attempt detected'"

Создадим бэкап конфигурации. После завершения настройки ModSecurity и проверки его работоспособности рекомендуется сохранить резервную копию текущей конфигурации: sudo cp /etc/modsecurity/modsecurity.conf /etc/modsecurity/modsecurity.conf.backup.

### **Вывод:** после выполнения всех шагов система защищена на нескольких уровнях:

1. **Firewall:** Блокировка нежелательного трафика через iptables.
2. **ModSecurity:** Применение правил CRS и кастомных конфигураций для фильтрации HTTP-запросов.
3. **Мониторинг:** Использование логов и инструментов для отслеживания активности и выявления атак.