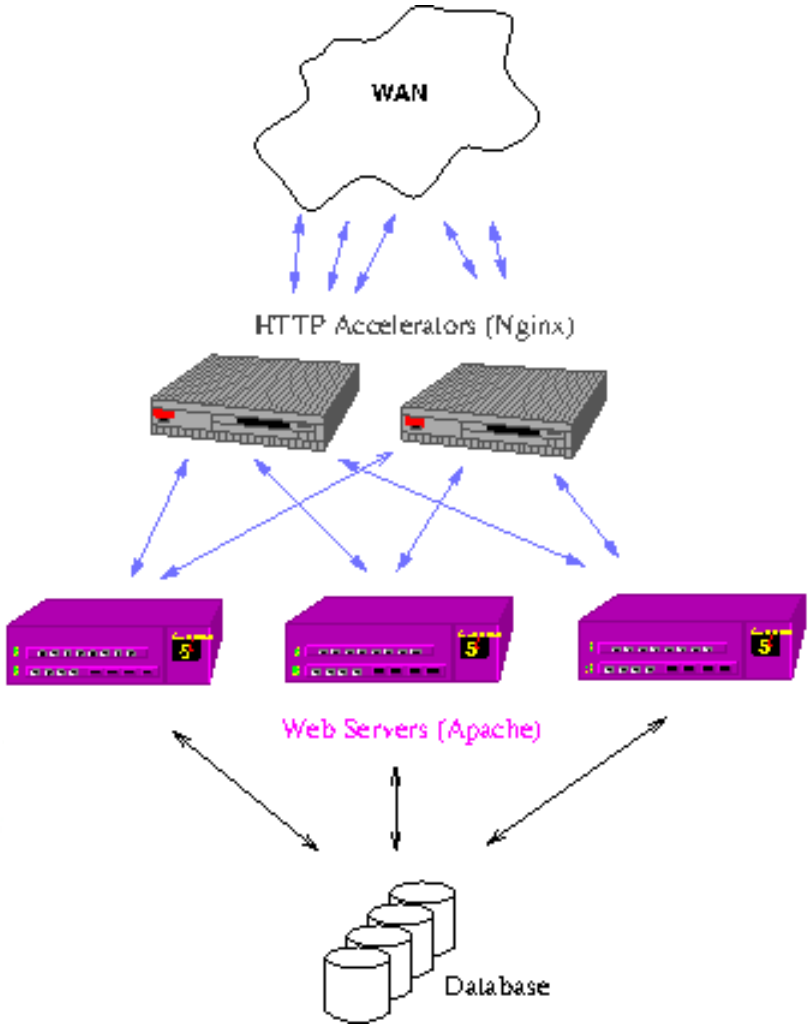
О DDoS на Web-кластер

Крижановский Александр

ak@natsys-lab.com

Распространенный Web-кластер



Типы DDoS на Web-кластер

- **L3**: flood, как правило, просто мусор для забивки канала
  - как правило, поддельные (spoofed) IP-адреса
  - DNS Distributed Reflection Denial of Service (DrDoS)
- **L4**: расходуют ресурсы TCP/IP стека ОС
  - SYN flood
  - Sockstress
- **L7**: использование узких мест прикладных серверов
  - Slow HTTP
  - Запросы к наиболее медленным ресурсам
  - Эмуляция flash crowd

Физический Уровень

- ***Цель:*** *забить канал, неважно чем*
- Различные flood'ы для переполнения канала
- Перекос метрик трафика:
  - Число UDP датаграмм (DNS) и/или ICMP сообщений
  - “мусорный” IP трафик со “странными” значениями транспортного протокола
  - Подозрительные подсети исходящих IP адресов

Уровень ОС

- ***Цель:*** *исчерпать ресурсы ОС*
- Фрагментированный и/или некорректный IP трафик
- SYN flood
- Out-of-order и/или некорректный TCP трафик
- Sockstress (черезмерное число малоактивных открытых сокетов)
- Большое число полузакрытых (FIN WAIT 2) соединений
- Нулевые окна приема и TCP таймауты

Прикладной Уровень

- ***Цель:*** *исчерпать ресурсы прикладных серверов*
- Могут быть отправлены реальными браузерами (JavaScript боты)
- Поля HTTP заголовков (в т.ч. URL) могут выбираться случайно из сотен/тысяч вариантов или генерироваться
- Боты могут “ходить” по ссылкам сайтов и интерпретировать JavaScript
- Целью могут быть ссылки, наиболее активно использующие БД
- Slow HTTP

DDoS vs Flash Crowd

- ***DDoS:*** *отправка большего числа запросов, чем сервис может отбработать*
- Изменение статистических параметров трафика:
  - число пакетов протоколов L3-L4 уровней
  - число SYN/FIN/RST пакетов в еденицу времени на соединение/IP
  - средний размер пакета
  - частота пакетов
  - частота HTTP запросов
  - отношение объема входящего трафика к исходящему
  - отношение HTTP запросов к ответам

Нормальная HTTP Нагрузка

- Узкое место – прикладной код (серверные скрипты, БД)
- Запросы достаточно большие за счет Cookie и URL (~1KB, иногда больше)
- Ответ — Web-страница (десятки килобайт), считанная из файла
- Сотни или тысячи Keep-Alive соединений (к одному HTTP-акселератору), каждое из которых генерирует умеренное число запросов

DDoS Нагрузка

- Исчерпание пропускной способности канала
- Исчерпание ресурсов ОС или прикладных серверов
  - Большое число мелких пакетов
  - Таймауты и расход памяти
- HTTP запросы часто сильно меньше (десятки или сотни байт)
- HTTP ответов сильно меньше, чем получаемых запросов
- Десятки или сотни тысяч соединений
  - много запросов в каждом соединении
  - или постоянная переустановка соединений с каждого хоста

Бутылочные Горлышки

- Отложенные прерывания (*softirq*) и драйвер сетевой карты
- Число поддерживаемых соединений и TCP буферы
- Затраты на установление новых соединений (и *fork()* для виртуального хостинга)
- Разбор HTTP запроса
- Логирование запросов
- Отдача тяжелой статики (**дисковый ввод-вывод на акселераторах!**)
- Прикладная логики (PHP, Python, Perl, Ruby и пр.)
- База данных

Эшелонированная защита

- Обрубать вредоносный трафик как можно раньше (drop early)
- **Воронка:** простые сигнатуры обрубать раньше более сложных

=> Дешевый scale-out

- Быстрые процессоры и память – дешево
- Быстрые диски – не бывает (за разумные деньги)

=> Низкая цена пропуска:

- false negatives всегда >0
- (*но и false positives тоже обычно >0*)

Back-end и Front-end: Apache и Nginx

- Front-end должен быть максимально легким
  - нет операций с диском
  - минимальное число переключений контекстов и копирований
  - минимум логирования

Nginx с тяжелой статикой и/или FastCGI – это Back-end

Все Лежит – DDoS?

=> Система отвечает медлено или не отвечает вообще

=> netstat, grep, tcpdump и пр. - занимают целую вечность

Нужна система мониторинга, показывающая:

- Изменение среднего числа запросов для клиента и соединения
- Желательно изменение числа пакетов и прочие метрики для L3-L7 трафика

В момент DDoS она тоже будет лежать, но начало атаки будет поймано.

Logs, Grep, IPtables

- Fail2ban (*http://fail2ban.org*) - уже готовый logs monitor
- Очень медленно из-за обработки текстовых файлов и большого числа переключений контекстов

Nginx

- Nginx
  - HttpLimitReqModule – лимитирование числа запросов за сессию
  - HttpLimitZoneModule – лимитирование числа соединений
- Apache mod_evasive

- *Требуется парсинг HTTP (наиболее грячая точка)*
- *Блокирование запроса может быть дороже отдачи контента*

Nginx – не средство фильтрации

Nginx: Пример Конфигурации

```
reset_timedout_connection       on;
client_header_timeout           15;
client_body_timeout             15;
send_timeout                    5;
keepalive_timeout               30 15;
limit_req_zone  $binary_remote_addr  zone=qglob:16m   rate=3r/s;
limit_zone cglob $binary_remote_addr 16m;
server {
    limit_conn cglob 32;
    location = / { limit_req   zone=qglob  burst=9 nodelay; }
 }
```

Nginx: Обработка Лимита

```
epoll_wait(12, {{EPOLLIN, ....}}, 512, 500) = 1

recvfrom(3, "GET / HTTP/1.1\r\nHost: ....", 1024, 0, NULL, NULL) =
327

// parse HTTP

write(11, "...limiting requests, excess...", 176) = 176

writev(3, [{"HTTP/1.1 503 Service Temporarily Una....", 200}], 1)
= 200

sendfile(3, 7, [0], 383)              = 383

recvfrom(3, 0xa1bac0, 1024, 0, 0, 0) = -1 EAGAIN

epoll_wait(12, {{EPOLLIN, ....}}, 512, 500) = 1

recvfrom(3, "", 1024, 0, NULL, NULL) = 0

close(3)                              = 0
```

Nginx: Горячие Точки

samples	%	image name	symbol name
496099	1.5719	nginx	ngx_vslprintf
325148	1.0303	nginx	ngx_http_parse_header_line
219699	0.6961	vmlinux	cfq_set_request
202023	0.6401	libc-2.12.so	memcpy
183268	0.5807	libpthread-2.12.so	recv
162710	0.5156	nginx	ngx_linux_sendfile_chain
157504	0.4990	nginx	ngx_http_limit_req_handler

IPTables

Настоящее средство фильтрации

- Connlimit – ограничение числа соединений для IP
- String – фильтрация запросов по специфичному заголовку
- Hashlimit – ограничение по числу запросов для IP
- Recent – ratelimit для динамического списка хостов
- Set (*ipset*) – позволяет эффективно обрабатывать большое число IP

Nginx & IPtables

- IPtables – фильтрация на L3/L4
- Пакет != запрос (особенно при Slow HTTP)

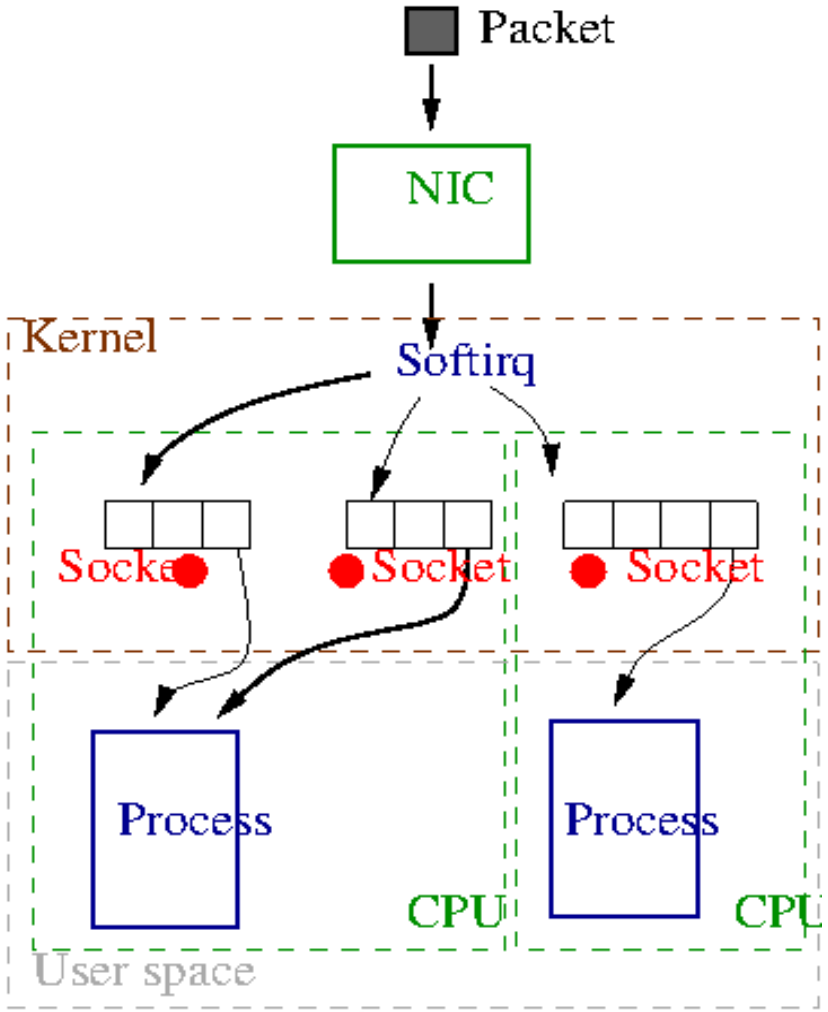
- Модификация HttpLimitReqModule и HttpLimitZoneModule:
  - добавление правил в iptables вместо генерации ответа и логирования
  - Постоянный (persistent) список заблокированных IP
  - LRU вытеснение из списка

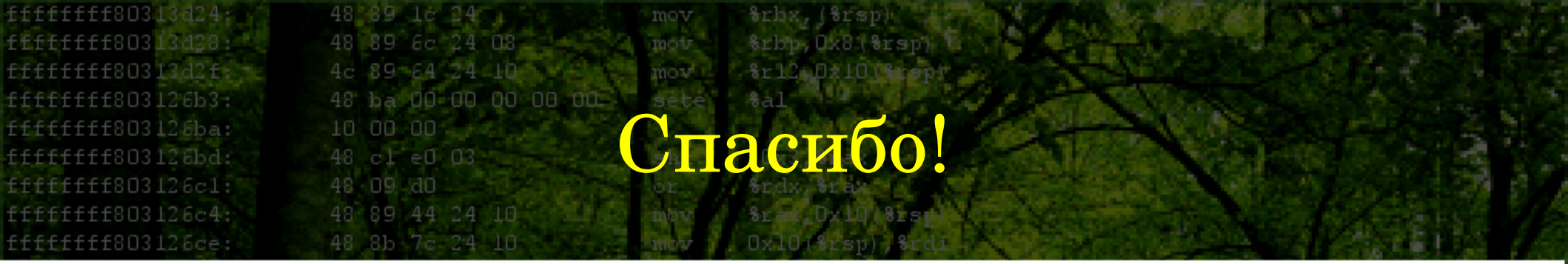
Sysctl

- net.ipv4.tcp_syncookies
- net.core.{somaxconn, tcp_max_syn_backlog}
- net.ipv4.conf.all.rp_filter
- net.ipv4.{tcp_fin_timeout, tcp_tw_recycle}
- net.ipv4.{tcp_mem, tcp_rmem, tcp_wmem}
- net.ipv4.{tcp_keepalive_time, tcp_keepalive_probes, tcp_keepalive_intvl}

Сетевые Прерывания

- MSI-X и smp_affinity для сетевых прерываний
- Receive Steering (linux >= 2.6.35):
  - Packet (RPS) - позволяет «разбрасывать» пакеты по *softirq* на разных ядрах если MSI-X недостаточно
  - Flow (RFS) – отправляет пакеты на CPU прикладного процесса





Спасибо!

ak@natsys-lab.com