

```
ffffffffff80313d24: 48 89 1c 24      mov     %rbx,%rsp)
ffffffffff80313d28: 48 89 6c 24 08   mov     %rbp,0x8(%rsp)
ffffffffff80313d2f: 4c 89 64 24 10   mov     %r12,0x10(%rsp)
ffffffffff803126b3: 48 ba 00 00 00 00 sete    %al
ffffffffff803126ba: 10 00 00
ffffffffff803126bd: 48 c1 e0 03      shl     $0x3,%rax
ffffffffff803126c1: 48 09 d0         or      %rdx,%rax
ffffffffff803126c4: 48 89 44 24 10   mov     %rax,0x10(%rsp)
ffffffffff803126ce: 48 8b 7c 24 10   mov     0x10(%rsp),%rdi
ffffffffff803126d3: 48 89 c5         mov     %rax,%rbp
ffffffffff803126d6: 48 2b 2d e3 13 99 00 sub     10032099(%rip),%rbp
ffffffffff803126dd: e8 fe fb ff ff   callq  ffffffff803122e0 <pft_descend>
ffffffffff80313d40: 0f 84 84 00 00 00 je      ffffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffffffff80313d46: f6 c3 01        test   $0x1,%bl
ffffffffff80313d49: 75 6d          jne    ffffffff80313db8 <alloc_cpumask_var_node+0x98>
ffffffffff80313d4b: 48 8b 3d 96 13 24 00 mov     2364310(%rip),%rdi # ffffffff805550e8 <malloc_sizes+0x68>
ffffffffff80313d52: 89 ea         mov     %ebp,%edx
ffffffffff803126e7: 48 c7 c7 60 1f 56 80 mov     $0xffffffff80561f60,%rdi
ffffffffff80312716: 5d          pop     %rbp
ffffffffff803126d6: 48 2b 2d e3 13 99 00 sub     10032099(%rip),%rbp
ffffffffff803126dd: e8 fe fb ff ff   callq  ffffffff803122e0 <pft_descend>
ffffffffff80313d40: 0f 84 84 00 00 00 je      ffffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffffffff80313d46: f6 c3 01        test   $0x1,%bl
ffffffffff80313d49: 75 6d          jne    ffffffff80313db8 <alloc_cpumask_var_node+0x98>
ffffffffff80313d4b: 48 8b 3d 96 13 24 00 mov     2364310(%rip),%rdi # ffffffff805550e8 <malloc_sizes+0x68>
ffffffffff80313d52: 89 ea         mov     %ebp,%edx
ffffffffff803126e7: 48 c7 c7 60 1f 56 80 mov     $0xffffffff80561f60,%rdi
ffffffffff80312716: 5d          pop     %rbp
ffffffffff80313d40: 0f 84 84 00 00 00 je      ffffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffffffff803126ba: 10 00 00
ffffffffff803126bd: 48 c1 e0 03      shl     $0x3,%rax
ffffffffff803126c1: 48 09 d0         or      %rdx,%rax
ffffffffff803126c4: 48 89 44 24 10   mov     %rax,0x10(%rsp)
ffffffffff803126ce: 48 8b 7c 24 10   mov     0x10(%rsp),%rdi
ffffffffff803126d3: 48 89 c5         mov     %rax,%rbp
ffffffffff803126d6: 48 2b 2d e3 13 99 00 sub     10032099(%rip),%rbp
ffffffffff803126dd: e8 fe fb ff ff   callq  ffffffff803122e0 <pft_descend>
ffffffffff80313d40: 0f 84 84 00 00 00 je      ffffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffffffff80313d46: f6 c3 01        test   $0x1,%bl
ffffffffff80313d49: 75 6d          jne    ffffffff80313db8 <alloc_cpumask_var_node+0x98>
ffffffffff80313d4b: 48 8b 3d 96 13 24 00 mov     2364310(%rip),%rdi # ffffffff805550e8 <malloc_sizes+0x68>
```

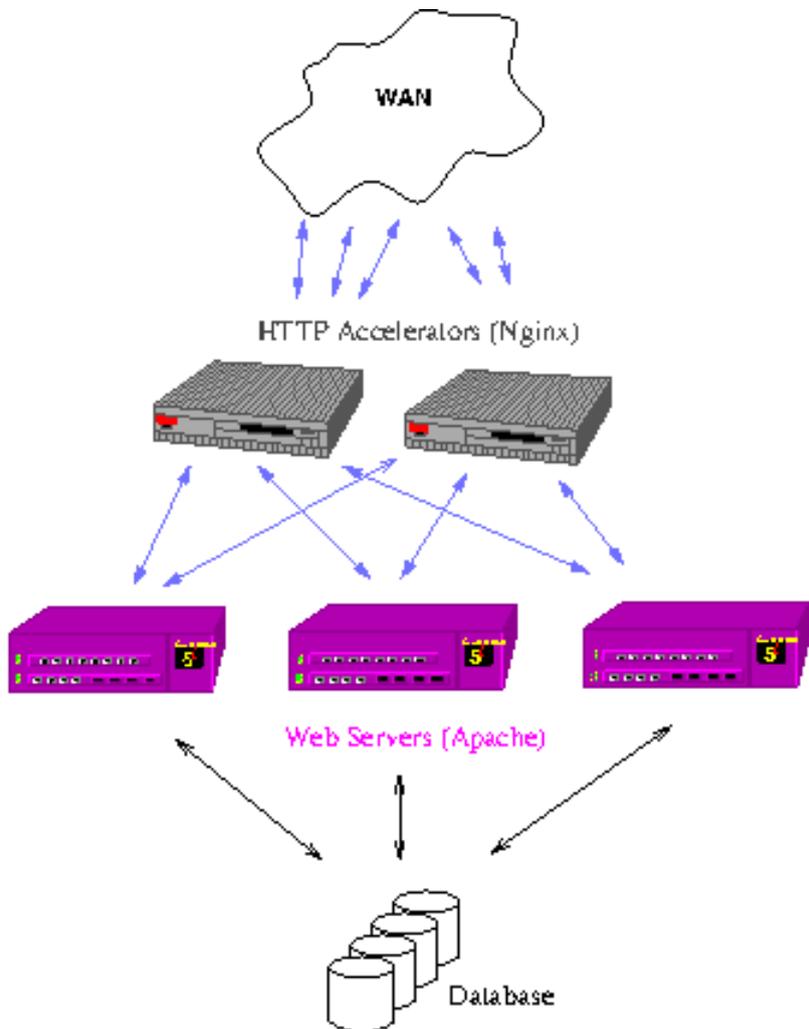
# О DDoS на Web-кластер

Крижановский Александр

*ak@natsys-lab.com*

```
ffffffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffffffff80313d3a: 10 00 00
ffffffffff80313e33: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffffffff803126c1: 4c 09 d0         or     %rdx, %rax
ffffffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi
```

# Распространенный Web-кластер



```
ffffffffff80313d40: 0f 84 84 00 00 00 je     ffffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffffffff80313d46: f6 c3 01        test  $0x1, %b1
ffffffffff80313d49: 75 6d          jne   ffffffff80313db8 <alloc_cpumask_var_node+0x68>
ffffffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # ffffffff805550e5 <malloc_sizes+0x68>
```

# 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

```
ffffff80313d40: 0f 84 84 00 00 00    je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01             test  $0x1,%bl
ffffff80313d49: 75 6d               jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov   2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>
```

# 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

```
ffffff80313d40: 0f 84 84 00 00 00    je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01             test   $0x1,%bl
ffffff80313d49: 75 6d               jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov   2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>
```

```
ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00
ffffff803126bd: 48 c1 e0 03      or     %eax, %eax
ffffff803126c1: 48 09 d0         or     %rdx, %rax
ffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi
```

## 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 полужакрытых (FIN WAIT 2) соединений
- Нулевые окна приема и TCP таймауты

```
ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01        test   $0x1, %bl
ffffff80313d49: 75 6d          jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # fffffff805550eb <malloc_sizes+0x68>
```

```
ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00 00      mov    %eax, 0x0(%rax)
ffffff803126bd: 48 c1 00 00     shl    %eax, %eax
ffffff803126c1: 48 09 d0 00     or    %rdx, %rax
ffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi
```

# 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

```
ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01      test  $0x1, %b1
ffffff80313d49: 75 6d        jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # fffffff805550e5 <malloc_sizes+0x68>
```

# DDoS vs Flash Crowd

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

```
ffffff80313d40: 0f 84 84 00 00 00    je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01             test  $0x1,%bl
ffffff80313d49: 75 6d               jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov   2364310(%rip),%rdi # fffffff805550eb <malloc_sizes+0x68>
```

# 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

```
ffffff80313d24: 48 89 1c 24      mov     %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov     %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov     %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00
ffffff803126bd: 48 c1 e0 03     shl     $16, %eax
ffffff803126c1: 48 09 d0        or     %rdx, %rax
ffffff803126c4: 48 89 44 24 10   mov     %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov     0x10(%rsp), %rdi
```

# 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

```
ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01        test   $0x1, %bl
ffffff80313d49: 75 6d          jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # fffffff805550eb <malloc_sizes+0x68>
```

# 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 статике (**ДИСКОВЫЙ ВВОД-ВЫВОД на акселераторах!**)
- Прикладная логики (PHP, Python, Perl, Ruby и пр.)
- База данных

```
ffffff80313d40: 0f 84 84 00 00 00    je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01             test  $0x1,%bl
ffffff80313d49: 75 6d               jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov   2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>
```

# Эшелонированная защита

- Обрубить вредоносный трафик как можно раньше (drop early)
- **Воронка:** простые сигнатуры обрубить 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)*

```
ffffff80313d40: 0f 84 84 00 00 00    je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01             test   $0x1,%bl
ffffff80313d49: 75 6d               jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>
```

# 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***

```
ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00         mov    %eax, 0x0(%eax)
ffffff803126bd: 48 c1 e0        shl    %eax, %eax
ffffff803126c1: 48 09 d0        or    %rdx, %rax
ffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi
```

# Все Лежит – DDoS?

- => Система отвечает медленно 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.***

```
ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01       test  $0x1,%bl
ffffff80313d49: 75 6d         jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>
```

```
ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00 00 00 00
ffffff803126bd: 48 c1 e1 00 00 00 sar    %rax
ffffff803126c1: 48 09 d0 00 00 00 or     %rax, %rax
ffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi
```

# Logs, Grep, IPtables

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

```
ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01        test  $0x1, %b1
ffffff80313d49: 75 6d          jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # fffffff805550e5 <malloc_sizes+0x68>
```

```

ffffff80313d24: 48 89 1c 24      mov     %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov     %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov     %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete    %al
ffffff803126ba: 10 00 00
ffffff803126bd: 48 c1 e0 03     shl     %eax, %eax
ffffff803126c1: 48 09 d0        or     %edx, %rax
ffffff803126c4: 48 89 44 24 10   mov     %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov     0x10(%rsp), %rdi

```

# Nginx

- Nginx
  - HttpLimitReqModule – лимитирование числа запросов за сессию
  - HttpLimitZoneModule – лимитирование числа соединений
- Apache mod\_evasive
  - *Требуется парсинг HTTP (наиболее горячая точка)*
  - *Блокирование запроса может быть дороже отдачи контента*

## ***Nginx – не средство фильтрации***

```

ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01        test   $0x1, %b1
ffffff80313d49: 75 6d          jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # fffffff805550e5 <malloc_sizes+0x68>

```

# Nginx: Пример Конфигурации

```
reset_timeout_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; }

}
```

```
ffffff80313d40: 0f 84 84 00 00 00    je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01             test   $0x1,%bl
ffffff80313d49: 75 6d               jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>
```

# 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
```

```
ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00 00      shll  $0, %eax
ffffff803126bd: 48 01 00 00 00 00 or     %eax, %eax
ffffff803126c1: 48 09 d0 24 10   or     %rdx, %rax
ffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi
```

# 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

```
ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01      test  $0x1, %bl
ffffff80313d49: 75 6d      jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # fffffff805550e5 <malloc_sizes+0x68>
```

```

ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00         sal   $0, %rax
ffffff803126bd: 48 c1 e0 03     or    %rdx, %rax
ffffff803126c1: 48 09 d0        or    %rdx, %rax
ffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi

```

# 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

```

ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01       test  $0x1, %bl
ffffff80313d49: 75 6d         jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov   2364310(%rip), %rdi # fffffff805550e5 <malloc_sizes+0x68>

```

# 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

```
ffffff80313d40: 0f 84 84 00 00 00    je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01             test  $0x1,%bl
ffffff80313d49: 75 6d               jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov   2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>
```

```

ffffff80313d24: 48 89 1c 24      mov    %rbx,%rsp
ffffff80313d28: 48 89 6c 24 08   mov    %rbp,0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12,0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00
ffffff803126bd: 48 c1 e0 03     shr    %eax
ffffff803126c1: 48 09 d0        or     %rdx,%rax
ffffff803126c4: 48 89 44 24 10   mov    %rax,0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp),%rdi

```

# 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}

```

ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01        test  $0x1,%bl
ffffff80313d49: 75 6d          jne   fffffff80313db8 <alloc_cpumask_var_node+0xb8>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip),%rdi # fffffff805550e5 <malloc_sizes+0x68>

```

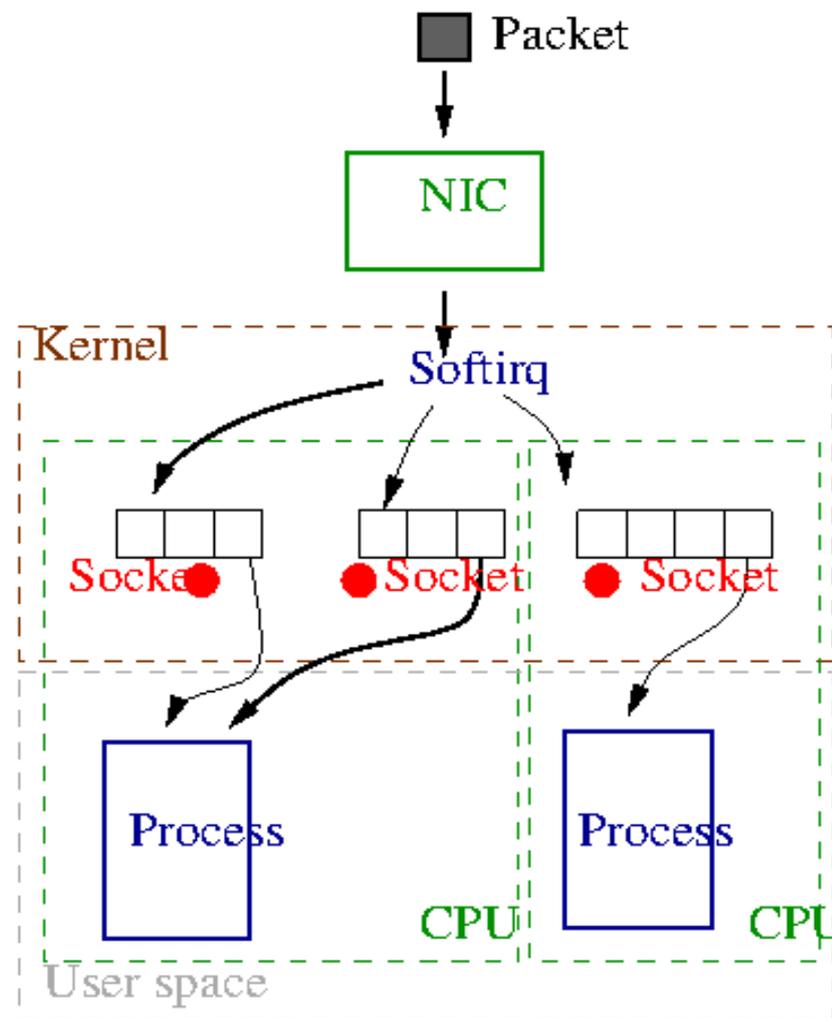
```

ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff80313d33: 48 ba 00 00 00 00 sete   %al
ffffff80313d3a: 10 00           mov    %eax, %eax
ffffff80313d3d: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d41: 48 09 d0        or     %rax, %rax
ffffff80313d44: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff80313d4e: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi

```

# 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



```

ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01        test  $0x1, %bl
ffffff80313d49: 75 6d          jne   fffffff80313db8 <alloc_cpumask_var_node+0x68>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    0x1324963d(%rip), %rdi # fffffff805550eb <malloc_sizes+0x68>

```

```
ffffff80313d24: 48 89 1c 24      mov    %rbx, (%rsp)
ffffff80313d28: 48 89 6c 24 08   mov    %rbp, 0x8(%rsp)
ffffff80313d2f: 4c 89 64 24 10   mov    %r12, 0x10(%rsp)
ffffff803126b3: 48 ba 00 00 00 00 sete   %al
ffffff803126ba: 10 00 00
ffffff803126bd: 48 c1 e0 03
ffffff803126c1: 48 09 d0        or     %rdx, %rax
ffffff803126c4: 48 89 44 24 10   mov    %rax, 0x10(%rsp)
ffffff803126ce: 48 8b 7c 24 10   mov    0x10(%rsp), %rdi
```

# Спасибо!

*ak@natsys-lab.com*

```
ffffff80313d40: 0f 84 84 00 00 00 je     fffffff80313dca <alloc_cpumask_var_node+0xaa>
ffffff80313d46: f6 c3 01        test   $0x1, %b1
ffffff80313d49: 75 6d          jne   fffffff80313db8 <alloc_cpumask_var_node+0x6d>
ffffff80313d4b: 48 8b 3d 96 13 24 00 mov    2364310(%rip), %rdi # fffffff805550e5 <malloc_sizes+0x68>
```