GNU/Linux

информация по Linux.

- Утилиты
  - kill
  - top
  - Iptables
- Диагностика проблем
  - Диагностика проблем производительности (ОС) Linux
  - Диагностика сетевых проблем

Утилиты

Утилиты

kill

Утилита `kill` в операционных системах UNIX и UNIX-подобных системах (например, Linux) используется для завершения процессов. Она позволяет отправлять сигналы процессам, которые могут приводить к их завершению или изменению поведения.

Завершение процесса по имени: Чтобы убить процесс по его имени, вы можете воспользоваться командой `pkill`. Например, чтобы завершить все процессы с именем "myprocess", вы можете выполнить следующую команду:

```
kill myprocess
```

отправит сигнал завершения (`SIGTERM`) всем процессам с указанным именем.

ершение дерева процессов: Чтобы прибить дерево процессов (все процессы, зависящие определенного процесса), вы можете использовать опцию `-TERM` (или `-15`) команды `kill`. ример, предположим, что у вас есть процесс с идентификатором `PID`, который запускает гие процессы. Вы можете завершить все процессы, зависящие от этого процесса, с ощью следующей команды:

```
ill -TERM -- -PID
```

есь `-PID` означает все процессы, зависящие от процесса с указанным идентификатором.

гналы, используемые в команде `kill`: В команду `kill` можно передавать различные гналы, которые определяют, как воздействовать на процессы. Некоторые пространенные сигналы включают:

- `SIGTERM` (15): Это стандартный сигнал завершения, который по умолчанию отправляется командой `kill`. Он позволяет процессу корректно завершиться.
- `SIGKILL` (9): Этот сигнал немедленно завершает процесс, не давая ему возможность выполнить какие-либо действия предварительной обработки.
- `SIGHUP` (1): Этот сигнал обычно используется для перезапуска процесса или обновления его конфигурации.
- `SIGINT` (2): Этот сигнал отправляется при нажатии комбинации клавиш `Ctrl+C`. Он используется для прерывания выполнения процесса в терминале.
- `SIGSTOP` (19) и `SIGCONT` (18): Эти сигналы приостанавливают и возобновляют выполнение процесса соответственно. `SIGSTOP` приостанавливает процесс без его завершения, а `SIGCONT` возобновляет его выполнение.

редать сигнал можно так:

```
kill -SIGTERM PID
#или
kill -15 PID
```

Если не указать сигнал явно, по умолчанию используется `SIGTERM`.

Утилиты

top

Основная информация тут:

```
an top
```

вод утилиты:

```
op - 16:51:17 up 18:34,  1 user,  load average: 0.21, 0.08, 0.02
asks: 138 total,   1 running, 137 sleeping,   0 stopped,   0 zombie
Cpu(s): 40.9 us, 13.9 sy,  0.0 ni, 37.2 id,  7.4 wa,  0.0 hi,  0.3 si,  0.3 st
iB Mem :    969.4 total,     69.3 free,    407.4 used,    492.8 buff/cache
iB Swap:   1024.0 total,    447.1 free,    576.9 used.    363.6 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
169868 metro     20   0  381192  53296  42492 S  28.9   5.4   0:00.87 php-fpm8.1
169870 metro     20   0  401344  63972  48748 S  12.0   6.4   0:00.36 php-fpm8.1
   639 mysql     20   0 1287396 120080  17796 S   6.6  12.1   9:51.15 mysqld
 83570 www-data  20   0   55640  10244   6488 S   4.7   1.0   0:04.10 nginx
   491 message+  20   0    8904   4484   3816 S   0.3   0.5   0:04.69 dbus-daemon
 82798 www-data  20   0  765256  12868   6812 S   0.3   1.3   0:00.76 apache2
 82799 www-data  20   0  765096  12128   6728 S   0.3   1.2   0:00.20 apache2
168816 root      20   0   10476   4008   3308 R   0.3   0.4   0:00.54 top
     1 root      20   0  166528   8552   5772 S   0.0   0.9   0:18.58 systemd
```

манда `top` представляет собой интерактивную утилиту в Linux, предназначенную для ниторинга системы и отображения информации о процессах. Вот объяснение каждого лбца, который отображается при использовании `top`:

- `PID`: Идентификатор процесса (Process ID).
- `USER`: Имя пользователя, которому принадлежит процесс.
- `PR`: Приоритет процесса.
- `NI`: Индекс приоритета процесса.
- `VIRT`: Общий объем виртуальной памяти, занимаемой процессом.
- `RES`: Резидентный объем памяти, используемой процессом.
- `SHR`: Объем разделяемой памяти, используемой процессом.
- `S`: Состояние процесса (D - спящий, R - работает, S - заблокирован, Z - зомби).
- `%CPU`: Процент использования CPU процессом.
- `%MEM`: Процент использования памяти процессом.

- `TIME+`: Общее время, затраченное процессом на выполнение.
- `COMMAND`: Имя команды или исполняемого файла, связанного с процессом.

Load average - "load average" (средняя нагрузка) представляет собой метрику, которая показывает среднюю загрузку системы за определенный период времени. Она измеряется в виде трех чисел, разделенных запятыми, например, "0.21, 0.08, 0.02". Каждое число представляет нагрузку системы за последние 1, 5 и 15 минут соответственно.

Load average отражает количество процессов, находящихся в очереди на исполнение (включая активные процессы и ожидающие процессы), и это включает как процессы, использующие CPU, так и процессы, заблокированные вводом-выводом. Если средняя нагрузка равна числу CPU или меньше, то система работает в пределах своих возможностей. Если средняя нагрузка превышает количество CPU (например, 200% для системы с 1 CPU), это означает, что есть процессы, ожидающие своей очереди на выполнение, и система испытывает перегрузку.

В системе с 1 CPU нагрузка может быть больше 100%, так как нагрузка отображает не только использование CPU, но и общую активность процессов в системе, включая ожидание ввода-вывода и другие факторы. Когда нагрузка превышает 100% для системы с одним CPU, это означает, что процессы активно конкурируют за вычислительные ресурсы процессора, и некоторые процессы могут испытывать задержки в выполнении.

Например: Предположим, у вас есть система с одним CPU и средняя нагрузка составляет 200%. Это означает, что количество активных и ожидающих процессов превышает вычислительные ресурсы процессора, и процессы конкурируют между собой за доступ к процессору. В результате некоторые процессы могут испытывать задержки в выполнении, и система может работать медленнее.

Когда в системе возникают проблемы с высокой нагрузкой и нехваткой ресурсов, следует проанализировать процессы, использующие большую часть ресурсов, и решить проблему путем оптимизации настроек, улучшения алгоритмов, добавления ресурсов или остановки проблемных процессов. Остановка процесса, который является источником проблемы, может быть одним из вариантов решения, но перед этим следует убедиться, что это безопасно и не повредит работе системы или другим процессам.

Вывод утилиты `top` может выглядеть следующим образом, если в системе с 1 CPU наблюдается нагрузка в 200%:

```
op - 13:15:28 up 10 days,  2:30,  2 users,  load average: 2.00, 1.12, 1.21
asks: 193 total,   2 running, 191 sleeping,   0 stopped,   0 zombie
Cpu(s): 100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
iB Mem :   7862.4 total,    170.3 free,   5285.9 used,   2406.2 buff/cache
iB Swap:   2048.0 total,    896.0 free,   1152.0 used.   1445.9 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1234 username  20   0   23456   6789   4321 R  100.0  0.1   0:10.00 high_load_process
```

```
  5678 username  20   0   12345   1234    567 S    0.0  0.0   0:02.00 another_process
```

В приведенном примере `load average` составляет `2.00`, что указывает на то, что система испытывает перегрузку, превышающую вычислительные ресурсы единственного CPU. В столбце `%CPU` для процесса с PID 1234 отображается значение 100.0, что говорит о том, что этот процесс активно использует всю доступную мощность процессора, что приводит к нагрузке в 200% для системы с 1 CPU.

Обычно, в выводе команды `top`, показатель `%CPU` для каждого процесса отражает процент использования CPU этим процессом от общей вычислительной мощности системы. В системе с одним CPU максимальное значение `%CPU` для процесса составит 100%. Однако, в случае, когда средняя нагрузка системы (load average) превышает количество доступных CPU, можно сделать вывод, что система испытывает перегрузку и некоторые процессы активно конкурируют за вычислительные ресурсы, превышая 100% использования CPU.

Таким образом, в примере, когда показатель "load average" равен 2.00, это означает, что средняя нагрузка системы составляет 200% от вычислительных ресурсов единственного CPU. Это свидетельствует о том, что процессы находятся в активной конкуренции за ресурсы, и система испытывает перегрузку.

Iptables

Инфо:

Iptables - это инструмент для управления файрволом в операционной системе Linux. Он позволяет настраивать правила безопасности и контролировать трафик сети.

В Ubuntu iptables является утилитой уровня ядра и доступен по умолчанию. В Ubuntu 22 (и других версиях Ubuntu) установка iptables не требуется, так как она уже включена в ядро Linux и доступна изначально.

Iptables предоставляет мощные возможности фильтрации и маршрутизации пакетов в Linux. Он работает на уровне ядра и используется для настройки правил файрвола и сетевых политик.

Применение:

1. Просмотр текущих правил:

```
ptables -L
```

имер вывода:

```
hain INPUT (policy ACCEPT)
arget     prot opt source               destination
CCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ROP       tcp  --  anywhere             anywhere             tcp dpt:http

hain FORWARD (policy ACCEPT)
arget     prot opt source               destination

hain OUTPUT (policy ACCEPT)
arget     prot opt source               destination
```

```
епочки (Chain):
hain INPUT: Цепочка, которая обрабатывает входящий трафик (пакеты, направленные к системе).
```

```
Chain FORWARD: Цепочка, которая обрабатывает пересылаемый трафик (пакеты, которые должны быть переданы через систему).
Chain OUTPUT: Цепочка, которая обрабатывает исходящий трафик (пакеты, отправляемые системой).

Столбцы для каждой цепочки:
target: Цель, то есть действие, которое будет выполнено с пакетом, если он соответствует правилу.
prot: Протокол, к которому относится правило (например, TCP, UDP).
opt: Опции, связанные с протоколом или действием.
source: Исходный адрес или сеть, откуда ожидается входящий трафик.
destination: Целевой адрес или порт, куда направляется трафик.

Из примера. В цепочке INPUT есть два правила:
Первое правило разрешает входящий TCP-трафик на порт SSH (22) с любого источника.
Второе правило блокирует входящий TCP-трафик на порт HTTP (80) с любого источника.
Цепочка FORWARD не содержит правил.
Цепочка OUTPUT не содержит правил.
```

В iptables есть несколько возможных состояний для параметра `target` в правилах. Некоторые из наиболее распространенных состояний `target` включают:

1. `ACCEPT`: Пакет будет принят и разрешен.
2. `DROP`: Пакет будет отброшен и удален, без отправки какого-либо уведомления отправителю.
3. `REJECT`: Пакет будет отброшен и отправлено уведомление отправителю о том, что пакет был отклонен.
4. `LOG`: Пакет будет принят, но информация о пакете будет записана в системный журнал или файл журнала для целей мониторинга и отладки.
5. `DNAT`: Изменение адреса назначения (Destination NAT) позволяет перенаправить пакеты на другой адрес и/или порт.
6. `SNAT`: Изменение адреса источника (Source NAT) позволяет заменить исходный адрес пакета на другой адрес и/или порт.
7. `MASQUERADE`: Вид NAT, который заменяет исходный адрес источника на адрес интерфейса, через который пакет покидает систему.
8. `REDIRECT`: Перенаправление пакетов на локальный порт или сокет.
9. `MARK`: Пометка пакетов для дальнейшей обработки другими инструментами или правилами.

Каждое состояние `target` в iptables выполняет определенное действие с пакетом, определенным правилом. Выбор подходящего состояния `target` зависит от ваших потребностей и требований конкретной ситуации.

2. Очистка существующих правил:

```
iptables -F
```

3. Закрыть \ открыть доступа к порту:

```
ptables -A INPUT -p tcp --dport <порт> -j DROP
ptables -A INPUT -p tcp --dport <порт> -j ACCEPT
```

4. Разрешение доступа только для определенных IP-адресов (вайтлист):

```
ptables -A INPUT -p tcp --dport <порт> -s <IP-адрес> -j ACCEPT
```

имер:

```
ptables -A INPUT -p tcp --dport 51821 -s 192.168.1.100 -j ACCEPT
```

манда добавляет правило в цепочку INPUT, которая разрешает входящий TCP-трафик на т 51821 только от указанного IP-адреса.

5. Сохранение и загрузка правил iptables:

```
ptables-save > /etc/iptables/rules.v4 #записать изменения в файл
ptables-restore < /etc/iptables/rules.v4 #загрузить правила в iptables
```

Дополнительно:

бы правила iptables из файла `/etc/iptables/rules.v4` автоматически применялись при грузке системы, можно использовать инструменты настройки системы для запуска манды `iptables-restore` при старте.

пример можно настроить службу `netfilter-persistent` или через `systemd`.

стройка `netfilter-persistent`:

```
pt install netfilter-persistent
etfilter-persistent reload #применить правила iptables без перезагрузки системы
```

тозагрузка netfilter-persistent при старте системы:

```
ystemctl enable netfilter-persistent
```

стройка через `systemd`:

```
vim /etc/systemd/system/iptables.service
```

Добавить в файл службы текст:

```
Unit]
escription=Load iptables rules
fter=network.target

Service]
ype=oneshot
xecStart=/sbin/iptables-restore /etc/iptables/rules.v4

Install]
antedBy=multi-user.target
```

лее можно загрузить и активировать службу, она будет запускаться при старте системы:

```
udo systemctl daemon-reload
udo systemctl enable iptables.service
```

я применения правил без перезапуска системы можно выполнить:

```
ystemctl start iptables.service
```

лучить список правил с номерами:

```
ptables -L INPUT --line-numbers
```

алить правило по номеру:

```
ptables -D INPUT <номер>
```

Диагностика проблем

Диагностика проблем производительности (ОС) Linux

Диагностика в ОС Linux является важной задачей для обеспечения эффективной работы системы. Такие проблемы могут возникать по разным причинам, включая неправильную настройку, недостаток ресурсов, проблемы с программным обеспечением или нагрузкой на систему.

Почему возникают проблемы производительности:

- Неправильная настройка параметров системы, таких как ядра, сети или памяти.
- Недостаточные ресурсы, такие как процессорное время, память или пропускная способность дисков.
- Наличие вредоносных программ или других проблемных процессов.
- Высокая нагрузка на систему из-за интенсивной работы или слишком большого количества активных процессов.

Утилиты для диагностики проблем производительности:

- top: отображает текущие активные процессы, их использование CPU и памяти, а также общую нагрузку на систему.
- htop: альтернатива top с расширенными функциями и интерактивным интерфейсом.
- vmstat: предоставляет информацию о системных ресурсах, включая использование CPU, памяти, дисков и сети.
- iostat: отображает статистику ввода/вывода дисковых устройств, помогая выявить проблемы с производительностью дисков.
- sar: собирает и анализирует системные данные, включая загрузку процессора, использование памяти, активность сети и другие параметры.
- strace: позволяет отслеживать системные вызовы и сигналы, что может помочь выявить проблемные процессы.

Влияние проблем производительности:

- Снижение отзывчивости системы и приложений.
- Увеличение времени отклика пользовательских запросов.
- Задержки и прерывания в работе процессов и сервисов.

- Потеря данных или некорректная обработка информации.

Например, в гипотетической системе Linux наблюдаются проблемы с недостатком ресурсов, и один из процессов расходует все ресурсы ОС.

1. Нужно использовать утилиту `top` для определения процесса, потребляющего большую часть ресурсов.
2. В окне `top` будет список активных процессов, их использование CPU, памяти и другие параметры. Отсортируйте процессы по использованию ресурсов, нажав клавишу `Shift` + `P`.
3. Обратите внимание на процесс, который потребляет большую часть ресурсов, таких как CPU или память. Возможно, это будет видно в столбцах `%CPU` и `%MEM`.
4. Определите идентификатор процесса (PID) проблемного процесса. Он обычно указывается в левой части окна `top`. Запишите этот PID для дальнейшего использования.
5. После определения проблемного процесса, введите команду `kill <PID>`, чтобы остановить его. Например, если PID проблемного процесса равен 1234, выполните следующую команду:

   ```
   ill 1234
   ```

6. Дождитесь некоторого времени, чтобы убедиться, что процесс завершился. Вы можете повторно запустить команду `top`, чтобы проверить использование ресурсов системы.
7. Если проблема все еще продолжается или процесс не останавливается, вы можете использовать команду `kill -9 <PID>`, которая принудительно прекратит выполнение процесса. Это следует использовать только в крайних случаях, когда обычная команда `kill` не срабатывает.

Вывод top гипотетической ОС:

```
op - 13:15:28 up 10 days,  2:30,  2 users,  load average: 0.78, 1.12, 1.21
asks: 193 total,   2 running, 191 sleeping,   0 stopped,   0 zombie
Cpu(s): 12.3 us,  4.5 sy,  0.0 ni, 82.7 id,  0.5 wa,  0.0 hi,  0.0 si,  0.0 st
iB Mem :   7862.4 total,    170.3 free,   5285.9 used,   2406.2 buff/cache
iB Swap:   2048.0 total,    896.0 free,   1152.0 used.   1445.9 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
1234 username  20   0   23456   6789   4321 R  50.0   0.1   0:10.00 problematic_process
5678 username  20   0   12345   1234    567 S   5.0   0.0   0:02.00 another_process
```

гом примере показаны заголовки столбцов, а также несколько строк, представляющих личные процессы. Вы можете видеть идентификатор процесса (PID), пользователя ER), использование CPU (%CPU), использование памяти (%MEM), а также другую ормацию о процессах. Проблемный процесс обычно отличается по высокому ользованию CPU или памяти.

Диагностика сетевых проблем

Диагностика сетевых проблем в ОС Linux включает исследование и анализ различных аспектов сети для определения и устранения проблем.

Возможные причины проблем:

- Неправильная конфигурация сетевых настроек.
- Проблемы с подключением к сети или устройством сетевого интерфейса.
- Конфликты IP-адресов или другие проблемы сетевого стека.
- Недоступность сетевых ресурсов, таких как серверы или маршрутизаторы.

Базовые утилиты для диагностики сетевых проблем:

- `ping`: Позволяет проверить доступность узла в сети и оценить время отклика.
- `traceroute` или `tracepath`: Определяют маршрут пакетов к заданному узлу и показывают промежуточные хопы.
- `ifconfig` или `ip`: Позволяют просмотреть и настроить сетевые интерфейсы, IP-адреса и другие параметры.
- `netstat` или `ss`: Предоставляют информацию о сетевых соединениях, портах и маршрутах.
- `tcpdump`: Позволяет захватывать и анализировать сетевой трафик для обнаружения проблем.
- `nslookup` или `dig`: Предоставляют информацию о DNS-запросах и разрешении имён.
- `iptables` или `ufw`: Позволяют управлять правилами брандмауэра и фильтровать сетевой трафик.
- `mtr`: `mtr` (My Traceroute) - это комбинированный инструмент, который объединяет функции `ping` и `traceroute`. Он предоставляет непрерывный мониторинг сети и позволяет определить проблемные участки на маршруте к заданному узлу.

Влияние сетевых проблем:

- Потеря или задержка сетевых пакетов, что может привести к снижению производительности приложений и служб.
- Недоступность сетевых ресурсов, таких как серверы, базы данных или веб-сайты, что может привести к простою приложений работающих через сеть.
- Неправильное маршрутизирование или нарушение безопасности, что может повлиять на интеграцию и связность сети.

MTR

`mtr` выводит информацию о промежуточных хопах, аналогично `traceroute`, но отличается тем, что продолжает слать пакеты на каждый хоп в режиме реального времени. Это позволяет получать актуальные данные о времени отклика и потерях пакетов.

Преимущества использования `mtr`:

- Он помогает определить точное место возникновения проблемы на маршруте к узлу.
- Предоставляет информацию о времени отклика и потерях пакетов на каждом хопе, что полезно для анализа и диагностики.
- Непрерывный режим мониторинга помогает отслеживать изменения в сети и выявлять временные проблемы.

Информация о потере пакетов (`Loss%`) позволяет определить наличие проблем на определенных хопах, а столбцы `Avg`, `Best`, `Worst` и `StDev` предоставляют информацию о времени отклика и его статистике.

Пример использования `mtr`:

```
tr google.com
```

команда запустит `mtr` для мониторинга маршрута к `google.com` и будет выводить тистику времени отклика и потерь пакетов для каждого хопа в режиме реального мени.

имер вывода команды:

```
OST: имя_хоста        Loss%   Snt   Last   Avg  Best  Wrst StDev
 1.|-- прыжок1          0.0%     5    0.5   1.2   0.5   2.3   0.8
 2.|-- прыжок2          0.0%     5    1.5   2.0   1.5   2.8   0.4
 3.|-- прыжок3          0.0%     5    2.8   3.1   2.8   3.7   0.3
 4.|-- прыжок4          0.0%     5    5.1   5.2   4.9   5.6   0.2
 5.|-- прыжок5          0.0%     5    4.3   4.5   4.3   4.9   0.2
 6.|-- google.com       0.0%     5    6.1   5.8   5.4   6.6   0.4
```

нный пример показывает что проблем сетевого характера нет. Значение `Loss%` равно %, что означает отсутствие потери пакетов на всех хопах. Столбец `Avg` показывает днее время отклика (в миллисекундах) на каждом хопе. Время отклика на каждом хопе яется стабильным и достаточно низкими.

имер, когда на маршруте наблюдаются проблемы:

```
HOST: имя_хоста        Loss%    Snt   Last   Avg  Best  Wrst StDev
  1.|-- прыжок1          0.0%     5    0.5   1.2   0.5   2.3   0.8
  2.|-- прыжок2          5.0%     5    1.5   2.0   1.5   2.8   0.4
  3.|-- прыжок3         10.0%     5    2.8   3.1   2.8   3.7   0.3
  4.|-- прыжок4         50.0%     5    5.1   5.2   4.9   5.6   0.2
  5.|-- прыжок5         80.0%     5    4.3   4.5   4.3   4.9   0.2
  6.|-- google.com     100.0%     5    6.1   5.8   5.4   6.6   0.4
```

В этом примере наблюдается потеря пакетов на нескольких хопах. Значение `Loss%` показывает процент потери пакетов на каждом хопе. Как видно, потери пакетов возникают на хопах 2, 3 и 4, а также на хопе 5 уже наблюдается потеря 80% пакетов. На последнем хопе (целевом сервере) потеря пакетов достигает 100%. Это указывает на проблемы в сети на пути к целевому серверу.

Пример, когда проблемы возникают у пользователя на его устройстве:

```
OST: имя_хоста        Loss%    Snt   Last   Avg  Best  Wrst StDev
 1.|-- прыжок1          0.0%     5    0.5   1.2   0.5   2.3   0.8
 2.|-- прыжок2          0.0%     5    5.1   5.2   4.9   5.6   0.2
 3.|-- имя_устройства   100.0%     5    0.0   0.0   0.0   0.0   0.0
 4.|-- google.com       100.0%     5    6.1   5.8   5.4   6.6   0.4
```

анном примере видно, что на хопе 5 (`имя_устройства`) наблюдается потеря всех пакетов 0% потерь). Это указывает на проблему, возникающую на устройстве пользователя, рее всего на его компьютере или домашнем роутере. Проблема может быть связана с фигурацией сетевых настроек, проблемами с подключением или неисправностью ройства.

имер, когда у целевого сервера есть проблемы:

```
OST: имя_хоста        Loss%    Snt   Last   Avg  Best  Wrst StDev
 1.|-- прыжок1          0.0%     5    0.5   1.2   0.5   2.3   0.8
 2.|-- прыжок2          0.0%     5    1.5   2.0   1.5   2.8   0.4
 3.|-- прыжок3          0.0%     5    2.8   3.1   2.8   3.7   0.3
 4.|-- прыжок4          0.0%     5    5.1   5.2   4.9   5.6   0.2
 5.|-- прыжок5          0.0%     5    4.3   4.5   4.3   4.9   0.2
 6.|-- google.com       50.0%     5    6.1   5.8   5.4   6.6   0.4
```

анном примере на хопе 6 (`google.com`) наблюдается 50% потери пакетов. Это может зывать на проблемы, возникающие непосредственно на целевом сервере. Возможные чины могут включать перегрузку сервера, неполадки в сети у хостинг-провайдера или облемы с самим сервером.