Network Working Group
Request for Comments: 2308
Updates: 1034, 1035
Category: Standards Track

M. Andrews
CSIRO

March 1998

Негативное кэширование запросов DNS (DNS NCACHE)

Negative Caching of DNS Queries (DNS NCACHE)

Статус документа

Этот документ содержит спецификацию стандартного протокола, предложенного сообществу Internet, и служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола можно узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться свободно.

Авторские права

Аннотация

В [RFC1034] приведено описание кэширования негативных откликов. Однако в этом описании допущена существенная ошибка в том, что серверам имён не разрешено передавать такие кэшированные отклики другим распознавателям (resolver), что существенно снижает эффект кэширования. В этом документе решаются вопросы повышения этой эффективности с учётом реального опыта. Документ служит заменой параграфа 4.3.4 в [RFC1034].

Негативное кэширование является необязательной частью спецификации DNS и имеет дело с кэшированием данных об отсутствии RRset [RFC2181] или доменных имён.

Негативное кэширование полезно в плане снижения времени отклика для ответов с отрицательными результатами. Это кэширование также снижает число сообщений, которые передаются между распознавателями и серверами имён, что снижает общий трафик в сети. От значительной части трафика DNS в сети Internet можно избавиться, если все распознаватели будут поддерживать негативное кэширование. С учётом этого обстоятельства такое кэширование уже не может рассматриваться, как необязательная часть распознавателя DNS.

1. Терминология

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с [RFC2119].

Negative caching - негативное кэширование
: Хранение информации о том, что чего-то не существует. Можно сохранять информацию о конкретных значениях записей, а можно обратить ситуацию и сохранять сведения о том, что записи не существует. Хранение информации о том, что чего-либо не существует, ответ не может быть получен или его не дают называется негативным кэшированием.

QNAME
: Имя в запросном разделе (query section) ответа или место, где оно распознаётся как CNAME или цепочка CNAME, поле данных последней записи CNAME. Последней CNAME в данном случае считается запись, содержащая значение, которое не распознаётся как другая запись CNAME. В реализациях следует принимать во внимание порядок включения записей CNAME в отклик. Первой должна указываться запись с меткой из раздела запроса (query section) и далее с соблюдением порядка записи с метками из раздела данных предыдущей (записи), если требуется более одной записи CNAME. Это позволяет обработать последовательность записей в один проход и снижает нагрузку на получателя информации. Среди записей CNAME могут размещаться другие имеющие отношение к делу записи (такие, как SIG RR [RFC2065]).

NXDOMAIN
: Другой вариант выражения для Name Error RCODE, как описано в параграфе 4.1.1 [RFC1035]. В документе используются оба термина.

NODATA
: Псевдо-RCODE, указывающий, что имя корректно для данного класса, но записей этого типа нет. Код NODATA выводится из ответа.

FORWARDER - пересылающий сервер
: Сервер имён, используемый для распознавания (resolve) запросов взамен прямого использования цепочки полномочных серверов имён. Обычно пересылающий сервер имеет более качественный доступ в Internet или поддерживает кэш большего размера, совместно используемый множеством распознавателей имён. Идентификация серверов в качестве FORWARDER и получение информации о пересылающих серверах выходит за рамки этого документа. Однако если вы планируете использовать свой сервер в качестве пересылающего, в запросе будет установлен флаг желательности пересылки.

От читателей этого документа ожидается понимание [RFC1034], [RFC1035] и [RFC2065].

2. Негативные отклики

Большинство негативных откликов общего назначение показывает отсутствие в DNS некого конкретного RRset. Данный случай рассматривается в первых разделах этого документа. Другие негативные отклики могут показывать особенности серверов имён. Этим вопросам посвящён раздел 7 (другие негативные отклики).

Негативные отклики указываются одним из рассмотренных в следующих параграфах способов

2.1. Ошибка в имени

Ошибки в имени (NXDOMAIN) указываются текстом Name Error в поле RCODE. В этом случае домена, указанного QNAME, не существует. Отметим, что в разделе ответов (answer) могут присутствовать записи SIG и CNAME, а в разделе полномочий (authority) могут быть RRset SOA, NXT [RFC2065] и SIG.

Можно различить отсылки (referral) и отклики NXDOMAIN по присутствию NXDOMAIN в RCODE, независимо от наличия записей NS или SOA в разделе полномочий (authority).

Отклики NXDOMAIN можно разделить на 4 типа в зависимости от содержимого раздела полномочий. Эти типы показаны ниже вместе с отсылками для сравнения. Не имеющие значения поля в примерах опущены.

Отклик NXDOMAIN типа 1

Заголовок

```
RDCODE=NXDOMAIN
```

Запрос

```
AN.EXAMPLE. A
```

Ответ

```
AN.EXAMPLE. CNAME TRIPPLE.XX.
```

Полномочия

```
XX. SOA NS1.XX. HOSTMASTER.NS1.XX. ....
XX. NS NS1.XX.
XX. NS NS2.XX.
```

Дополнительно

```
NS1.XX. A 127.0.0.2
NS2.XX. A 127.0.0.3
```

Отклик NXDOMAIN типа 2

Заголовок

```
RDCODE=NXDOMAIN
```

Запрос

```
AN.EXAMPLE. A
```

Ответ

```
AN.EXAMPLE. CNAME TRIPPLE.XX.
```

Полномочия

```
XX. SOA NS1.XX. HOSTMASTER.NS1.XX. ....
```

Дополнительно

```
<пусто>
```

Отклик NXDOMAIN типа 3

Заголовок

```
RDCODE=NXDOMAIN
```

Запрос

```
AN.EXAMPLE. A
```

Ответ

```
AN.EXAMPLE. CNAME TRIPPLE.XX.
```

Полномочия

```
<пусто>
```

Дополнительно

```
<пусто>
```

Отклик NXDOMAIN типа 4

Заголовок

```
RDCODE=NXDOMAIN
```

Запрос

```
AN.EXAMPLE. A
```

Ответ

```
AN.EXAMPLE. CNAME TRIPPLE.XX.
```

Полномочия

```
XX. NS NS1.XX.
XX. NS NS2.XX.
```

Дополнительно

```
NS1.XX. A 127.0.0.2
```

```
NS2.XX. A 127.0.0.3
```

Отклик-отсылка

Заголовок

```
RDCODE=NOERROR
```

Запрос

```
AN.EXAMPLE. A
```

Ответ

```
AN.EXAMPLE. CNAME TRIPPLE.XX.
```

Полномочия

```
XX. NS NS1.XX.
XX. NS NS2.XX.
```

Дополнительно

```
NS1.XX. A 127.0.0.2
NS2.XX. A 127.0.0.3
```

Отметим, что в 4 примерах откликов NXDOMAIN известно, что имя AN.EXAMPLE. существует и имеет своим значением запись CNAME. NXDOMAIN указывает на TRIPPLE.XX, о котором известно, что его не существует. С другой стороны, в примере отсылки показано, что AN.EXAMPLE существует и имеет в качестве значения CNAME RR, но ничего не известно о существовании TRIPPLE.XX, кроме того, что серверами NS1.XX и NS2.XX можно воспользоваться в качестве следующего этапа поиска информации.

При отсутствии записей CNAME отклик NXDOMAIN указывает на имя в метке RR из раздела вопросов (question).

2.1.1 Специальная обработка Name Error

В этом параграфе рассматриваются ошибки, встречающиеся при негативном кэшировании откликов NXDOMAIN.

Многие из имеющихся распознавателей (resolver) не способны корректно детектировать и обрабатывать все формы откликов NXDOMAIN. Некоторые трактуют отклики NXDOMAIN типа 1, как отсылки. Для решения этой проблемы рекомендуется, чтобы серверы, являющиеся полномочными для отклика NXDOMAIN, передавали только отклики TYPE 2 NXDOMAIN (т. е., в разделе authority содержится запись SOA, но нет записей NS). Не являющиеся полномочными серверы передают отклик NXDOMAIN типа 1 одному из таких устаревших распознавателей и в результате запрос к полномочному серверу становится необязательным. Нежелательным но не критичным исключением является случай, когда запрашивающий сервер используется в качестве пересылающего (FORWARDER). Если распознаватель использует сервер в качестве FORWARDER, такому распознавателю необходимо запретить передачу ему откликов TYPE 1 NXDOMAIN, используя взамен TYPE 2 NXDOMAIN.

На некоторых распознавателях обработка продолжается некорректно, если флаг полномочий не установлен и возникает цикл, завершающийся по достижению порогового числа повторов с возвратом SERVFAIL. Это вызывает проблемы, если ваш сервер имён указан на таком распознавателе, как FORWARDER. Если сервер имён используется таким распознавателем в качестве пересылающего (FORWARDER), нужно устанавливать флаг полномочий в откликах NXDOMAIN для таких распознавателей. На практике это не вызывает проблем даже при постоянном включении и BIND, начиная с версии 4.9.3, по умолчанию ведёт себя именно так.

2.2. Нет данных

NODATA указывается ответом с RCODE = NOERROR и отсутствием имеющей отношение к делу информации в разделе ответов. Раздел полномочий будет содержать запись SOA или в нем совсем не будет записей NS.

Отклики NODATA алгоритмически определяются из содержимого откликов, поскольку для их индикации отсутствует значение RCODE. В некоторых случаях для того, чтобы убедиться в том, что NODATA является корректным откликом, может потребоваться отправка другого запроса.

Раздел полномочий может содержать наборы записей NXT и SIG в дополнение к записям NS и SOA. Записи CNAME и SIG могут присутствовать в разделе ответов.

Можно различить отклик NODATA и отсылку (referral response) по наличию записи SOA в разделе полномочий или отсутствию там же записей NS.

Отклики NODATA можно разделить на три типа по содержимому раздела полномочий (authority). Эти типы, а также отклик с отсылкой приведены ниже. Некоторые поля, не имеющие значения, в примерах не указаны.

Отклик NODATA типа 1

Заголовок

```
RDCODE=NOERROR
```

Запрос

```
ANOTHER.EXAMPLE. A
```

Ответ

```
<пусто>
```

Полномочия

```
EXAMPLE. SOA NS1.XX. HOSTMASTER.NS1.XX. ....
EXAMPLE. NS NS1.XX.
EXAMPLE. NS NS2.XX.
```

Дополнительно

```
NS1.XX. A 127.0.0.2
NS2.XX. A 127.0.0.3
```

Отклик NODATA типа 2

Заголовок

```
RDCODE=NOERROR
```

Запрос

```
ANOTHER.EXAMPLE. A
```

Ответ

```
<пусто>
```

Полномочия

```
EXAMPLE. SOA NS1.XX. HOSTMASTER.NS1.XX. ....
```

Дополнительно

```
<пусто>
```

Отклик NODATA типа 3

Заголовок

```
RDCODE=NOERROR
```

Запрос

```
ANOTHER.EXAMPLE. A
```

Ответ

```
<пусто>
```

Полномочия

```
<пусто>
```

Дополнительно

```
<пусто>
```

Отсылка

Заголовок

```
RDCODE=NOERROR
```

Запрос

```
ANOTHER.EXAMPLE. A
```

Ответ

```
<пусто>
```

Полномочия

```
EXAMPLE. NS NS1.XX.
EXAMPLE. NS NS2.XX.
```

Дополнительно

```
NS1.XX. A 127.0.0.2
NS2.XX. A 127.0.0.3
```

Эти примеры, в отличие от приведённых выше примеров NXDOMAIN, не включают записей CNAME, однако это может быть сделано, как в примерах NXDOMAIN, и в этом случае будет включаться последнее значение CNAME (QNAME), для которого сделано заключение NODATA.

2.2.1. Специальная обработка NODATA

В настоящее время имеется множество распознавателей, которые не способны корректно детектировать и обработать все формы откликов NODATA. Некоторые распознаватели трактуют отклики TYPE 1 NODATA, как отсылки. Для решения этой проблемы серверам, которые полномочны отправлять отклики NODATA, использовать только отклики TYPE 2 NODATA, в которых полномочный раздел содержит запись SOA, но не включает записей NS. Отправка отклика TYPE 1 NODATA такому распознавателю неполномочным сервером будет лишь вызывать избыточные запросы. Если сервер указан, как FORWARDER для другого распознавателя, на нем также может потребоваться запрет использования TYPE 1 NODATA для неполномочных откликов NODATA.

Некоторые серверы не могут установить для RCODE значение NXDOMAIN при наличии записей CNAME в разделе ответов. Если требуется определённый ответ NXDOMAIN/NODATA, в таком случае распознаватель должен повторить запрос, используя QNAME в качестве его метки.

3. Негативные отклики от полномочных серверов

Серверы имён, имеющие полномочия для зоны, **должны** включать запись SOA этой зоны в раздел ответов отклика с NXDOMAIN или отклика, показывающего отсутствие данных запрошенного типа. Это требуется потому, что отклик может кэшироваться. В качестве значение TTL для этой записи берётся меньшее из значений поля MINIMUM из SOA и поля TTL самой SOA - это значение показывает, как долго распознаватель может сохранять в кэше негативный отклик. Запись TTL SIG, связанная с записью SOA, также следует обрезать в соответствии с TTL из записи SOA.

Если зона подписана [RFC2065], **должна** быть добавлена запись SOA и соответствующие записи NXT и SIG.

4. Поле SOA Minimum

В прошлом поле SOA Minimum было перегружено и использовалось с тремя разными целями - минимальное значение TTL для всех записей RR в зоне, используемое по умолчанию значение TTL для записей RR без своего поля TTL, а также TTL для негативных откликов.

Несмотря на изначальное определение, в качестве минимального значения TTL для всех RR в зоне этот параметр никогда не применялся и в настоящее время такое использование не рассматривается.

Второй вариант - использование в качестве принятого по умолчанию значения TTL для RR без явного указания TTL в первичном файле зоны применим только к основным (primary) серверам. После переноса зоны все RR получают явные значения TTL и нет возможности определить, было значение TTL для записи задано явно или получено из принятого по умолчанию при переносе зоны. Если сервер не требует явно включать в записи RR значения TTL, ему следует обеспечивать механизм получения отсутствующих значений TTL без использования поля MINIMUM в записи SOA.

Формат первичного файла (Master File), определённый в разделе 5 [RFC 1035], расширен для включения директивы:

```
$TTL <TTL> [comment]
```

Для всех записей, указанных после этой директивы и не включающих явного значения TTL, устанавливается TTL из директивы $TTL. Записи SIG без явного указания TTL получают время жизни из «исходного TTL» записи SIG [RFC 2065] (параграф 4.5).

Оставшаяся трактовка SOA Minimum - TTL для негативных откликов - служит единственным назначением этого поля.

5. Кэширование отрицательных ответов

Подобно обычным откликам, отрицательные ответы также имеют ограниченный срок действия (TTL). Поскольку в разделе ответов нет записи, к которой может быть применено это значение TTL, для его передачи нужен иной метод. Решением является включение записи SOA для зоны в раздел полномочий. При создании полномочным сервером такой записи в качестве TTL для неё берётся меньшее из значений SOA.MINIMUM и SOA TTL. Это значение TTL уменьшается подобно тому, как это происходит для обычных кэшированных откликов и нулевое значение (0) показывает, что кэшированный негативный отклик больше **недопустимо** использовать.

Негативный отклик, связанный с ошибкой имени (NXDOMAIN), следует кэшировать для того, чтобы его можно было найти и возвратить в ответ на другой запрос с теми же <QNAME, QCLASS>, для которых был получен кэшированный отрицательный ответ.

Негативный отклик, связанный с отсутствием данных (NODATA), следует кэшировать для того, чтобы его можно было найти и возвратить в ответ на другой запрос с теми же <QNAME, QTYPE, QCLASS>, для которых был получен кэшированный негативный отклик.

Запись NXT, если она имеется в разделе полномочий полученного негативного отклика, **должна** сохраняться для того, чтобы её можно было найти и возвратить с записью SOA в разделе authority; так же следует поступать со всеми записями SIG в разделе полномочий. Для ответов NXDOMAIN есть не вполне очевидная связь между записями NXT и QNAME. Запись NXT **должна** иметь имя владельца, совпадающее с именем запроса для откликов NODATA.

Негативные отклики без записей SOA **не следует** кэшировать, поскольку в этом случае нет возможности предотвратить зацикливание отрицательных ответов между парой серверов даже при малых значениях TTL.

Несмотря на то, что DNS формирует дерево серверов, ошибки в настройке могут приводить к возникновению петель в графах запросов (например, два сервера могут указать один другого в качестве пересылающего (forwarder)). Без снижения TTL для кэшированного негативного отклика при его попадании на следующий сервер отсчёт TTL будет сбрасываться. Это может приводить к бесконечной циркуляции отклика между парой таких серверов.

Как и при кэшировании обычных откликов распознавателям важно ограничивать время нахождения отрицательного ответа в кэше, поскольку протокол разрешает кэширование на срок до 68 лет. Предел времени хранения в кэше следует делать не больше аналогичного параметра для позитивных ответов и предпочтительно обеспечивать возможность его настройки. Значение от 1 до 3 часов будут работать достаточно хорошо и подойдут для использования по умолчанию. Значения более суток могут вызывать проблемы.

6. Негативные ответы из кэша

Когда сервер при ответе на запрос встречает кэшированный негативный отклик, он **должен** добавить кэшированную запись SOA в раздел authority своего отклика со значением TTL, декрементированным на время хранения записи в кэше. Это позволяет своевременно и корректно избавляться от устаревших откликов NXDOMAIN / NODATA.

Если запись NXT была кэширована вместе в записью SOA, она **должна** добавляться в раздел authority. Если запись SIG была кэширована вместе с записью NXT, её **следует** добавлять в раздел authority.

Как и всем ответам из кэша, негативным ответам **следует** иметь неявную отсылку, встроенную в ответ. Это даёт распознавателю возможность обнаруживать полномочный источник. Неявная отсылка характеризуется записями NS в разделе authority, направляющими распознаватель к полномочному источнику. Отклики NXDOMAIN типов 1 и 4 содержат неявные отсылки, как и отклики NODATA типа 1.

7. Другие негативные отклики

Кэширование других негативных откликов не рассматривается в имеющихся RFC. Не существует способа указать для таких откликов желаемое значение TTL. Нужно соблюдать осторожность, чтобы не возникло петель пересылки.

7.1 Отказ сервера (опция)

Отказы серверов делятся на два основных класса. К первому классу относятся ситуации, когда сервер может определить некорректность своей конфигурации для зоны. Это может быть в результате включения в список серверов для зоны сервера, который реально таковым не является, или в случаев сервера зоны, который по тем или иным причинам не получил данных зоны. Причиной отсутствия данных зоны может служить отсутствие файла зоны или ошибки в нем, а также в результате того, что другой сервер, от которого должны быть получены такие данные, утратил доступность или по какой-либо причине не может или не хочет «отдать» зону.

Ко второму классу относятся случаи, когда серверу требуется получить ответ откуда-нибудь, но он не способен это сделать по причине отказов в сети, отсутствия ответов от других серверов, возврата серверами отказов из-за ошибок и т. п.

В обоих случаях распознаватель **может** кэшировать отклик с отказом сервера. При этом **недопустимо** кэшировать такой отклик более, чем на пять (5) минут и кэшироваться **должна** комбинация <query name, type, class, server IP address>.

7.2 «Умерший»/недоступный сервер (опция)

«Мёртвыми»/недоступными серверами считаются те, от которых нет никакого ответа на запрос или транспортный уровень указывает недоступность или отсутствие сервера. Сервер может считаться «мёртвым» или недоступным, если он не ответил на запрос в течение 120 секунд.

Примеры индикации транспортного уровня приведены ниже:

сообщения ICMP об ошибках, показывающие недоступность хоста, сети или порта;

сброс TCP;

сообщения стека IP с индикацией, подобной перечисленной выше.

Распознаватель[1] **может** кэшировать индикацию «мёртвого» сервера. При этом **недопустимо** сохранять такие записи более пяти (5) минут. Индикация **должна** сохраняться для комбинации <query name, type, class, server IP address>, если транспортный уровень не сообщил об отсутствии сервера (в этом случае запись относится ко всем запросам для данного IP-адреса).

8. Отличия от RFC 1034

Негативное кэширование в распознавателях больше не является опциональным - если распознаватель выполняет кэширование, он должен кэшировать и негативные отклики.

Неполномочные отрицательные ответы **могут** кэшироваться.

Запись SOA из раздела authority **должна** кэшироваться. Индикация ошибок в именах должна кэшироваться для пар <query name, QCLASS>, индикация отсутствия данных - для комбинаций <query name, QTYPE, QCLASS>.

Кэшированная запись SOA должна добавляться в отклик. Ранее это явно не разрешалось, поскольку не делалось различия между записями SOA, кэшированными в обычном режиме и в результате негативного отклика, что вызывало проблемы при добавлении кэшированной обычным образом записи SOA к кэшированному негативному отклику.

В формат первичных файлов добавлена директива $TTL.

9. История негативного кэширования

В этом разделе кратко представлена история негативного кэширования в DNS без спецификаций такого кэширования.

Интересно отметить, что в серверах CHIVES и BIND были применены одинаковые концепции.

История ранней работы CHIVES (параграф 9.1) была представлена Rob Austein <sra@epilogue.com> и воспроизводится ниже в том же виде, как в [MPA].

9.1. CHIVES[2]

Как-то весной 1985 я сказал Paul Mockapetris, что наш опыт работы с его распознавателем JEEVES DNS показывает необходимость той или иной схемы негативного кэширования. Paul предложил просто кэшировать полномочные ошибки, используя значение SOA MINIMUM для зоны, содержащей искомые записи RR. Я вполне уверен, что этот разговор состоялся до публикации RFC-973, но для меня осталось непонятным, была ли эта идея спонтанным ответом на моё замечание или Paul уже планировал включить её в документ, который стал позднее RFC-973. В любом случае ни один из нас не был уверен в том, что значение SOA MINIMUM обеспечит правильную метрику для использования, но оно доступно и находится под контролем администратора искомой зоны. Оба эти обстоятельства в тот момент представлялись нам важными.

Позднее, в 1987 году я выпустил beta-версию CHIVES - распознавателя DNS, написанного для замены JEEVES от Paul. CHIVES включал механизм поиска пути, который интенсивно использовался на нескольких сайтах (включая мой собственный), поэтому в CHIVES был включён также механизм негативного кэширования на основе значений SOA MINIMUM. Основная стратегия заключалась в кэшировании полномочных кодов ошибок с ключами в виде точных параметров запроса (QNAME, QCLASS, QTYPE) со значением TTL для записи в кэше, равным SOA MINIMUM. CHIVES не пытался отслеживать записи SOA RR, если они не были представлены в полномочном отклике, поэтому он полностью не устранял трафик сообщений об ошибках DNS, но значительно его сокращал. Следует принимать во внимание, что это происходило примерно в одно время с коллапсом ARPANET, вызванным перегрузкой, связанной с экспоненциальным ростом и «старым» (до VJ) алгоритмом повтора TCP, поэтому негативное кэширование обеспечивало существенное снижение времени отклика DNS для пользователей, почтовых демонов и т. п.

Насколько я знаю, CHIVES был первым распознавателем с поддержкой негативного кэширования. Разработка CHIVES пришлась на сумрачные годы TOPS-20, поэтому система не получила широкого распространения, однако те немногие машины, где она работала, имели слишком важное значение, чтобы быстро их отключить, независимо от эксплуатационных расходов. Незначительное число пользователей CHIVES не позволило должным образом развить систему. Однако были получены некоторые интересные для технологии DNS результаты, из которых к рассматриваемым здесь вопросам имеет отношение лишь конфигурационный параметр MAXTTL.

[1] В оригинале ошибочно сказано «сервер». См. https://www.rfc-editor.org/errata_search.php?eid=461. *Прим. перев.*
[2] В оригинале этот заголовок пропущен. См. http://www.rfc-editor.org/errata_search.php?rfc=2308&eid=4627. Прим. перев.

Опыт работы с JEEVES уже показал, что записи RR часто использовали неразумно большие значения TTL (значение 99999999 было популярно в течение многих лет в результате ошибок в коде и документации нескольких ранних версий BIND), а надёжные сами по себе программы, которые слепо полагались на эти значения TTL, могли порождать столько странностей, что зачастую приходилось перезапускать распознаватели для очистки кэшей от мусора. Так в CHIVES появился параметр MAXTTL, который задавал максимальное «разумное» значение TTL в принимаемых записях RR. Для записей RR с TTL > MAXTTL устанавливалось значение TTL = MAXTTL или эти записи просто отбрасывались, в зависимости от конфигурационных параметров.

Когда началось применение негативного кэширования CHIVES в реальных условиях, стало ясно, что значение SOA MINIMUM зачастую слишком велико и может вызывать для негативного кэширования те же проблемы, которые были отмечены для RR при чрезмерных TTL для обычного кэширования (здесь снова причиной послужили ошибки в нескольких ранних версиях BIND, когда вторичный сервер обоснованно отвергал всю информацию о своих зонах, если при (пере)запуске ему не удавалось связаться с первичными серверами). С этого началось ограничение TTL при негативном кэшировании значениям MAXTTL и эксперименты были продолжены.

В конфигурации, которая представлялась хорошо работающей на WSMR-SIMTEL20.ARMY.MIL (последняя из основных машин TOPS-20, которая послужила основным пользователем CHIVES и обеспечила наиболее продолжительную базу для эксперимента), было установлено значение MAXTTL около трёх дней. Большая часть трафика SIMTEL20 в последние годы использования была связана с электронной почтой и для почтовой очереди был установлен тайм-аут в одну неделю, что предоставляло «застрявшим» сообщениям несколько попыток полного распознавания DNS без перегрузки системы бесполезными запросами. Поскольку мы использовали только один параметр MAXTTL (причин этого я сейчас не помню), а не раздельные для позитивного и негативного кэширования, влияние установки MAXTTL на негативное кэширование не очевидно.

CHIVES включает ещё один, в какой-то мере спорный, механизм, который в некоторых случаях использовался вместо негативного кэширования. Демон распознавания CHIVES можно настроить на загрузку первичных файлов DNS, что позволяет ему действовать в качестве «скрытого вторичного сервера» (в современных терминах). При такой настройке распознаватель имеет прямой доступ к полномочной информации для интенсивно используемых зон. Это отражено в механизмах поиска путей CHIVES - обычно используется два раздельных пути поиска, один из которых работает с локальными полномочными данными зон, а второй может генерировать обычные итеративные запросы. Это избавляет от необходимости негативного кэширования в случаях заведомо высокой нагрузки (например, распознаватель на XX.LCS.MIT.EDU всегда загружает файлы зон для LCS.MIT.EDU и AI.MIT.EDU, помещая оба этих суффикса в «локальный» путь поиска, поскольку с хостами в этих двух зонах связан основной объем трафика DNS). Эту функцию могут использовать не все сайты, использующие CHIVES, - например в C.CS.CMU.EDU сделан выбор в пользу «удалённого» пути поиска, поскольку в CMU имеется слишком много разных субзон, для которых использование «теневых» фалов слишком хлопотно и удобней работать с негативным кэшированием даже для локального трафика.

В целом я продолжаю считать, что выбранная модель негативного кэширования была достаточно разумной - администратор зоны задаёт время хранения записей в кэше, а в настройках распознавателя определяется реальное время кэширования, которое может составлять интервал от 0 до заданного администратором времени хранения записей. Сейчас я бы много сделал по-другому (например, использовал новое поле SOA вместо MINIMUM), но меня бы больше взволновало, если бы за прошедшие годы не было придумано хотя бы несколько улучшений.

9.2. BIND

Хотя это не было первой попыткой негативного кэширования в BIND, в июле 1993 года в BIND 4.9.2 ALPHA был реализован код Anant Kumar из ISI для проверки применимости и негативного кэширования (NCACHE). Этот код использовал 10-минутное значение TTL для негативного кэширования и кэшировались только индикации, которые были негативными откликами, NXDOMAIN и NOERROR_NODATA. Это послужило причиной появления кода NODATA, отмеченного выше.

Mark Andrews из CSIRO добавил код (RETURNSOA), сохраняющий запись SOA так, чтобы её можно было получить с помощью аналогичного запроса. Из UUNET жаловались на то, что они получают старые ответы после загрузки новой зоны и опция была отключена в BIND 4.9.3-alpha5 (апрель 1994). В действительности это показывало необходимость очистки пространства, которое зона будет занимать. Функциональность для этого была добавлена в BIND 4.9.3 BETA11 patch2 (декабрь 1994).

RETURNSOA была снова разрешена по умолчанию в BIND 4.9.5-T1A (август 1996).

10. Пример

Приведённый ниже пример использует подписанную зону, в которой присутствуют только серверы имён. Будем делать запрос для имени WWW.XX.EXAMPLE и рассматривать немедленный отклик и отклик по прошествии 10 минут. Отметим, что 1) в течение 10 минут ожидания срок действия записей NS для XX.EXAMPLE истекает, 2) значение TTL для записей SIG не задаётся явно в файле зоны и в качестве времени действия подписи будет использоваться TTL для RRset.

Файл зоны:

```
$TTL 86400
$ORIGIN XX.EXAMPLE.
@       IN      SOA     NS1.XX.EXAMPLE. HOSTMATER.XX.EXAMPLE. (
                        1997102000      ; порядковый номер
                        1800    ; обновление (30 минут)
                        900     ; повтор (15 минут)
                        604800  ; срок действия (7 дней)
                        1200 ) ; минимум (20 минут)
        IN      SIG     SOA ...
  1200  IN      NXT     NS1.XX.EXAMPLE. A NXT SIG SOA NS KEY
        IN      SIG     NXT ... XX.EXAMPLE. ...
   300  IN      NS      NS1.XX.EXAMPLE.
   300  IN      NS      NS2.XX.EXAMPLE.
```

```
            IN      SIG     NS ... XX.EXAMPLE. ...
            IN      KEY     0x4100 1 1 ...
            IN      SIG     KEY ... XX.EXAMPLE. ...
            IN      SIG     KEY ... EXAMPLE. ...
    NS1     IN      A       10.0.0.1
            IN      SIG     A ... XX.EXAMPLE. ...
      1200  IN      NXT     NS2.XX.EXAMPLE. A NXT SIG
            IN      SIG     NXT ...
    NS2     IN      A       10.0.0.2
            IN      SIG     A ... XX.EXAMPLE. ...
      1200  IN      NXT     XX.EXAMPLE. A NXT SIG
            IN      SIG     NXT ... XX.EXAMPLE. ...
```

Первичный отклик

Заголовок

```
RDCODE=NXDOMAIN, AA=1, QR=1, TC=0
```

Запрос

```
WWW.XX.EXAMPLE. IN A
```

Ответ

```
<пусто>
```

Полномочия

```
XX.EXAMPLE.       1200 IN SOA NS1.XX.EXAMPLE. ...
XX.EXAMPLE.       1200 IN SIG SOA ... XX.EXAMPLE. ...
NS2.XX.EXAMPLE.   1200 IN NXT XX.EXAMPLE. NXT A NXT SIG
NS2.XX.EXAMPLE.   1200 IN SIG NXT ... XX.EXAMPLE. ...
XX.EXAMPLE.      86400 IN NS  NS1.XX.EXAMPLE.
XX.EXAMPLE.      86400 IN NS  NS2.XX.EXAMPLE.
XX.EXAMPLE.      86400 IN SIG NS ... XX.EXAMPLE. ...
```

Дополнительно

```
XX.EXAMPLE.      86400 IN KEY 0x4100 1 1 ...
XX.EXAMPLE.      86400 IN SIG KEY ... EXAMPLE. ...
NS1.XX.EXAMPLE. 86400 IN A   10.0.0.1
NS1.XX.EXAMPLE. 86400 IN SIG A ... XX.EXAMPLE. ...
NS2.XX.EXAMPLE. 86400 IN A   10.0.0.2
NS3.XX.EXAMPLE. 86400 IN SIG A ... XX.EXAMPLE. ...
```

Отклик по истечении 10 минут

Заголовок

```
RDCODE=NXDOMAIN, AA=0, QR=1, TC=0
```

Запрос

```
WWW.XX.EXAMPLE. IN A
```

Ответ

```
<пусто>
```

Полномочия

```
XX.EXAMPLE.        600 IN SOA NS1.XX.EXAMPLE. ...
XX.EXAMPLE.        600 IN SIG SOA ... XX.EXAMPLE. ...
NS2.XX.EXAMPLE.    600 IN NXT XX.EXAMPLE. NXT A NXT SIG
NS2.XX.EXAMPLE.    600 IN SIG NXT ... XX.EXAMPLE. ...
EXAMPLE.         65799 IN NS  NS1.YY.EXAMPLE.
EXAMPLE.         65799 IN NS  NS2.YY.EXAMPLE.
EXAMPLE.         65799 IN SIG NS ... XX.EXAMPLE. ...
```

Дополнительно

```
XX.EXAMPLE.      65800 IN KEY 0x4100 1 1 ...
XX.EXAMPLE.      65800 IN SIG KEY ... EXAMPLE. ...
NS1.YY.EXAMPLE. 65799 IN A   10.100.0.1
NS1.YY.EXAMPLE. 65799 IN SIG A ... EXAMPLE. ...
NS2.YY.EXAMPLE. 65799 IN A   10.100.0.2
NS3.YY.EXAMPLE. 65799 IN SIG A ... EXAMPLE. ...
EXAMPLE.         65799 IN KEY 0x4100 1 1 ...
EXAMPLE.         65799 IN SIG KEY ... . ...
```

11. Вопросы безопасности

Предполагается, что этот документ не добавляет каких-либо значимых угроз безопасности к тем, что уже имеются при использовании данных из DNS.

При использовании негативного кэширования может оказаться возможным распространение атак на службы через сообщения NXDOMAIN с очень большим значением TTL. Без негативного кэширования сделать это будет гораздо сложнее. Похожего эффекта можно было раньше добиться за счёт распространения некорректных записей A, которые могли сделать сервер недоступным. Эффект у этих угроз похожий, но психологическое воздействие существенно различается. При некорректной записи A вы скажете: «Чертова сеть, опять не работает» и повторите попытку завтра. В случае NXDOMAIN вы скажете: «Они отключили сервер и перестали быть доступными», и возможно больше не будете обращаться к этому серверу.

Практическим примером сказанного выше может служить сервер SMTP, поведение которого вполне предсказуемо. При атаке NXDOMAIN почтовые сообщения будут «заворачиваться» незамедлительно, в при атаке с некорректными записями A почта будет помещаться в очередь и может быть доставлена после подавления или завершения атаки.

Для успеха такой атаки индикация NXDOMAIN должна быть внесена в родительский сервер (или загруженный кэширующий распознаватель). Одним из способов реализации является использование записи CNAME, которая приведёт к запросам со стороны родительского сервера к серверу атакующих. Распознаватели, которые хотят предотвратить такие атаки, могут сделать новый запрос для окончательной записи QNAME, игнорируя данные NS в ответах на первоначальные запросы.

Реализация проверки допустимости значений TTL будет снижать эффективность таких атак, поскольку для успехи атаки потребуется более частая вставка подложных данных.

DNS Security [RFC2065] обеспечивает механизм проверки корректности негативных откликов с помощью записей NXT и SIG. Данный документ поддерживает использование этого механизма за счёт поддержки передачи защитных записей даже неосведомленными о защите серверами.

Благодарности

Автор благодарит Rob Austein за историю сервера имён CHIVES, а также рабочую группу DNSIND и, в частности, Robert Elz за полезный технический и редакторский вклад в подготовку этого документа.

Литература

[RFC1034] Mockapetris, P., «DOMAIN NAMES - CONCEPTS AND FACILITIES,» STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., «DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION,» STD 13, RFC 1035, November 1987.

[RFC2065] Eastlake, O., and C. Kaufman, «Domain Name System Security Extensions,» RFC 2065, January 1997.

[RFC2119] Bradner, S., «Key words for use in RFCs to Indicate Requirement Levels,» BCP 14, RFC 2119, March 1997.

[RFC2136] P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound, «Dynamic Updates in the Domain Name System (DNS UPDATE)», RFC 2136, April 1997[1].

[RFC2181] Elz, R., and R. Bush, «Clarifications to the DNS Specification,» RFC 2181, July 1997.

Адрес автора

Mark Andrews

CSIRO - Mathematical and Information Sciences

Locked Bag 17

North Ryde NSW 2113

AUSTRALIA

Phone: +61 2 9325 3148

EMail: Mark.Andrews@cmis.csiro.au

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

[1]В оригинале этот документ не включён в список. См. https://www.rfc-editor.org/errata_search.php?eid=4489. *Прим. перев.*