БЕЗОПАСНОСТЬ ЦИФРОВЫХ ТЕХНОЛОГИЙ. – 2025. – № 4 (119). – 43–58

МЕТОДЫ И СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004 DOI: 10.17212/2782-2230-2025-4-43-58

АНАЛИЗ ЭКСПЛУАТАЦИИ SQL-ИНЪЕКЦИЙ С ИСПОЛЬЗОВАНИЕМ КИБЕРПОЛИГОНА AMPIRE*

Т.С. РЕЙН[1], А.К. БОЛОТОВ[2], А.Р. УСТИМЕЦ[3]

[1] *РФ, 650000, г. Кемерово, ул. Красная, 6, Кемеровский государственный университет, кандидат физико-математических наук, доцент кафедры информационной безопасности. E-mail: tsrein@mail.ru*
[2] *РФ, 650000, г. Кемерово, ул. Красная, 6, Кемеровский государственный университет, лаборант кафедры информационной безопасности. E-mail: bolotov160600@gmail.com*
[3] *РФ, 650000, г. Кемерово, ул. Красная, 6, Кемеровский государственный университет, лаборант кафедры информационной безопасности. E-mail: artem03yst@mail.ru*

В настоящее время обучение практическим навыкам кибербезопасности, таким как поиск уязвимостей ПО, анализ сетевого трафика и администрирование ОС, является важным этапом подготовки специалистов по информационной безопасности. Информационная инфраструктура современных организаций зачастую сложна, и различные компоненты могут быть неочевидно взаимосвязаны [1]. Поэтому именно практические навыки могут сыграть решающую роль в предотвращении и быстром устранении инцидентов в области ИБ.

Одной из самых распространенных уязвимостей является SQL-инъекция – уязвимость, которая позволяет злоумышленнику выполнять произвольный запрос к базе данных, манипулируя самим запросом, с целью завладения необходимой информацией. Таким образом, злоумышленник может получить доступ к конфиденциальной информации и даже закрепиться в информационной системе, получив контроль над сервером.

Статья посвящена анализу механизмов эксплуатации SQL-инъекций и методов устранения их последствий на примере киберполигона Ampire.

Рассмотрены сценарии использования различных типов SQL-инъекций (классическая, UNION-based, Blind SQL), методы автоматизированных атак с использованием инструментов, таких как sqlmap, и практические подходы к устранению уязвимостей. Цель работы – выработать базовые рекомендации по написанию защищенного кода с использованием параметризованных запросов и современных фреймворков.

Киберполигон Ampire – платформа от компании «Перспективный мониторинг», которая служит для проведения киберучений с целью отработки навыков кибербезопасности. КемГУ является одним из пользователей этой платформы, обучая студентов навыкам защиты информации от внешних и внутренних угроз.

* Статья получена 05 ноября 2025 г.

Особое внимание уделено образовательной ценности киберполигона для подготовки специалистов в условиях, приближенных к реальным, что способствует снижению рисков, связанных с уязвимостями веб-приложений.

Ключевые слова: SQL-инъекция, кибербезопасность, киберполигон Ampire, Sqlmap, уязвимость, параметризованные запросы, обучение специалистов, веб-приложения

ВВЕДЕНИЕ

Сегодня обучение практическим навыкам кибербезопасности является важным этапом обучения будущего специалиста. Хакеры владеют широким арсеналом уязвимостей, используя любую возможность для проникновения в систему и получения контроля над ней. Вследствие этого становится актуальной проблемой получения навыков выявления и устранения уязвимостей и их последствий в условиях, наиболее приближенных к реальным.

Для решения этой проблемы можно использовать киберполигон Ampire – среду, эмулирующую сеть реального предприятия с компьютерами персонала, вебсайтом организации, файловым сервером и даже промышленными SCADA-системами [2]. Для киберполигона разрабатываются специальные уязвимые узлы. Это могут быть как системы со слабыми паролями, так и версии приложений с уязвимостями в программном коде.

Важной частью киберполигона является виртуальная машина нарушителя, которая в автоматическом режиме при помощи специальных скриптов проводит автоматизированную атаку, используя популярные эксплойты и уязвимости и опираясь на известные утилиты для атак и тестирования, например Sqlmap или Metasploit Framework.

Еще одним компонентом, с которым непосредственно взаимодействуют обучающиеся, является система мониторинга и реагирования. При помощи IDS (Intrusion Detection System – система обнаружения вторжений) отслеживаются подозрительные события в системе, показывается точное время, сетевой пакет, адрес получателя и отправителя, а также возможная используемая уязвимость и ее идентификационный номер CVE (Common Vulnerabilities and Exposures – распространенные уязвимости). Команда мониторинга проводит анализ событий в системе, определяет возможные случаи взлома и перенаправляет их команде реагирования с указанием возможного способа их устранения. Реагирование, в свою очередь, взаимодействует с системой предприятия, закрывает найденные уязвимости и устраняет последствия атаки – от дефейса сайта компании до добавленных хакером пользователей и открытого удаленного доступа.

Кемеровский государственный университет активно использует киберполигон Ampire для обучения студентов практическим навыкам кибербезопасности.

Киберполигон развернут на базе Министерства цифрового развития и связи Кузбасса, где регулярно проходят тренировки с отработкой разных сценариев взлома виртуальной инфраструктуры и разной комбинацией уязвимых узлов.

SQL-инъекции остаются одними из самых распространенных уязвимостей [3]. Злоумышленники используют ошибки в программном обеспечении, чтобы внедрить свой собственный SQL-код и получить доступ к базе данных или даже загрузить на сервер собственный файл. Киберполигон Ampire содержит множество сценариев, где эксплуатируется эта уязвимость. Например, в работе [4] рассматривается один сценарий с UNION SQL-инъекцией.

Для обучения специалистов по информационной безопасности практическим навыкам предлагается несколько подходов, например CTF или заранее подготовленная виртуальная инфраструктура и набор лабораторных работ к ним.

В работе [5] производится сравнительный анализ нескольких крупных CTF-платформ с точки зрения обучения. CTF (Capture the flag – захват флага) – формат соревнований, в своей классической форме представляет состязание между Red team (атакующей стороной) и Blue team (стороной защиты). Имеется уязвимая виртуальная инфраструктура, называемая *vulnerable box* или сокращенно *vulnbox*. В этой инфраструктуре хранится флаг, последовательность символов, которую должна заполучить красная команда. Красная команда занимается взломом, выискивая уязвимости и недочеты в системе безопасности, чтобы получить флаг. Синяя команда выстраивает систему защиты, мониторит события, фильтрует трафик в инфраструктуре и исправляет обнаруженные ошибки в программах. Иногда участники выступают одновременно и в синей, и в красной команде, атакуя команды друг друга и защищая свои флаги.

Также существует формат King of the Hill – «Царь горы», где несколько команд должны проникнуть на один сервер, а потом выбивать из него друг друга, пока не останется только одна. Авторы сравнивают несколько CTF-платформ с открытым исходным кодом с точки зрения простоты установки, кроссплатформенности, легкости конфигурирования и расширяемости. В итоге самой удобной по всем критериям оказалась платформа CTFd. Однако ни одна платформа не позволяла организовать взаимодействие между участниками, и фокус был в основном на точку зрения атакующих, а не защищающихся. В частности, идет обучение эксплуатации SQL-инъекции на уязвимых веб-приложениях, но нет заданий на устранение уязвимостей в коде.

В работе «SQL Injection Teaching Based on SQLi-labs» [6] рассматривается использование специального веб-приложения SQLi-labs с открытым исходным кодом. Это приложение основано на архитектуре Apache + PHP + MySQL и содержит 22 урока, которые рассматривают всевозможные типы SQL-

инъекций – от простых, двойных до слепых инъекций. Участники могут взломать уязвимое приложение, познакомиться с его исходным кодом и найти уязвимые места в программном коде. Для каждой лабораторной работы имеется подробное методическое описание, описывающее принцип атаки и методы ее устранения. Недостатком такого метода обучения можно назвать ограниченность архитектуры и необходимость каждый раз заново настраивать виртуальный веб-сервер.

Для обучения защиты от SQL-инъекций требуется большое количество примеров запросов с различными типами уязвимости. Соммерволь и соавторы показали, что агенты обучения с подкреплением могут успешно создавать различные типы SQL-инъекций [7]. Была создана симуляция CTF (Capture the flag). В среде существует веб-сервер либо с определенной уязвимостью, либо без нее. Среда получает запрос от агента, обрабатывает его и возвращает ответ. Агент взаимодействует с сервером, определяет тип уязвимости и затем начинает автоматизированный эксплойт. Предварительная обработка ответа сервера идет через наличие в ответе версии SQL и ошибки в ответе или через определение длины ответа. Если ответ от сервера длинный, то скорее всего эксплойт удачный. В качестве возможных действий выбрано 65 дискретных значений для определения типа уязвимости, разных типов уязвимости и обнаружения, что уязвимости нет.

Агент не понимает их значения, действуя только методом проб и ошибок. В качестве метода обучения использовалось *Q*-обучение, в частности, из-за простоты и хорошей интерпретируемости. Было проведено три серии симуляции: первые две с миллионом эпизодов, а последняя уже с десятью миллионами эпизодов. Первая симуляция содержала три типа уязвимостей, вторая – четыре, третья – шесть (с трафиком и без трафика). Было выделено три фазы обучения: разведка, доработка и эксплуатация. На первом этапе от начала до $1{,}3 \times 105$ эпизодов модель только знакомится с окружением, и большинство запросов неуспешны; на втором этапе с $1{,}3 \times 105$ до 9×105 эпизодов модель уже находит решения, но ищет более оптимальные подходы. На третьем этапе модели искусственно выставляют приоритет на выполнение уязвимостей.

Было выявлено, что такие типы инъекций, как Union-based и Boolean-based, сильно зависят от времени обучения, в то время как модель научилась быстро находить сервера без уязвимостей. Особенно рост времени обучения был замечен с добавлением новых уязвимостей. В итоге первый агент проэксплуатировал все 1000 уязвимостей, второй агент не справился только со слепой Boolean-based-инъекцией. Третий агент справился с 9998 уязвимостями без трафика, также не справившись с двумя слепыми Boolean-based-инъекциями. Добавление трафика уменьшило до 95,83 % вероятность успеха, особенно в Time-based-инъекциях. Таким образом, было показано, что модели

машинного обучения с подкреплением могут находить и эксплуатировать уязвимости в подготовленной среде, таким образом снабжая примерами уязвимостей исследователей, которые могут сделать их частью обучения.

Киберполигон представляет собой специализированную тестовую среду, предназначенную для моделирования, анализа и отработки киберугроз. Это контролируемая виртуальная инфраструктура с сетевыми, аппаратными и программными компонентами, в которой можно безопасно реализовывать сценарии кибератак, тем самым тренируя практические навыки обучающихся. В настоящей статье был выбран киберполигон Ampire, так как он содержит большую базу подготовленных сценариев атак, содержащих SQL-инъекции различных типов, а IDS VipNet следит за действиями виртуального нарушителя, что позволяет провести полный анализ атаки, выявить уязвимые узлы и устранить уязвимость. Киберполигон отслеживает действия пользователя и оповещает, когда уязвимость закрыта и устранены последствия ее эксплуатации.

Таким образом, в работе исследуется анализ механизмов эксплуатации и методы устранения последствий SQL-инъекций в различных сценариях, которые предлагает киберполигон Ampire.

1. АНАЛИЗ СЦЕНАРИЕВ КИБЕРПОЛИГОНА

1.1. ОПИСАНИЕ СЦЕНАРИЕВ

Киберполигон Ampire предлагает несколько сценариев с SQL-инъекциями, базирующихся на Union based SQL и Blind SQL.

По сценарием понимается имитация реальной атаки на инфраструктуру с помощью скриптов, которая осуществляется в автоматическом режиме с помощью заранее подготовленных скриптов на Python.

В каждом сценарии злоумышленник производит атаку на компании, используя уязвимости в программном коде. Производится инъекция SQL-кода, которая позволяет злоумышленнику загрузить свой вредоносный файл на сервер или эскалировать свои полномочия и получить полный доступ к базе данных. Атаки могут проводиться как с использованием специализированных средств (sqlmap), так и вручную.

1.2. СЦЕНАРИИ С UNION BASED-ИНЪЕКЦИЕЙ

В одном сценарии внешний злоумышленник атакует демилитаризированную зону DMZ-компании, где располагается ее веб-портал. Для автоматического сканирования и атаки использовалась утилита sqlmap – популярный

инструмент для поиска и эксплуатации SQL-инъекции с открытым исходным кодом. Это можно понять в ходе анализа IDS, в частности, в описании произошедшего подозрительного события, что можно увидеть на рис. 1. Здесь в качестве столбцов таблицы указаны время события с точностью до миллисекунды, класс правила, название правила, ip-адреса и порты отправителя и получателя пакета.

14:32:43.3...	web-applica...	ET WEB_SERVER PHP System Command i...	185.88.181.55	44928	10.10.1.20	80
14:32:43.3...	web-applica...	AM EXPLOIT Generic Command Injection: '...	185.88.181.55	44928	10.10.1.20	80
14:32:43.3...	web-applica...	AM EXPLOIT Generic Possible PHP Injectio...	185.88.181.55	44928	10.10.1.20	80
14:32:43.3...	web-applica...	ET WEB_SERVER Exploit Suspected PHP In...	185.88.181.55	44942	10.10.1.20	80
14:32:43.3...	web-applica...	AM EXPLOIT Generic Command Injection i...	185.88.181.55	44942	10.10.1.20	80
14:32:43.3...	web-applica...	AM EXPLOIT Generic XSS in URI var 1	185.88.181.55	44958	10.10.1.20	80
14:32:43.3...	web-applica...	ET WEB_SERVER Exploit Suspected PHP In...	185.88.181.55	44958	10.10.1.20	80
14:32:43.3...	attempted-a...	ET WEB_SERVER Suspicious Chmod Usage...	185.88.181.55	44958	10.10.1.20	80
14:32:43.3...	web-applica...	AM EXPLOIT Generic Command Injection i...	185.88.181.55	44958	10.10.1.20	80
14:32:43.4...	policy-violat...	ET POLICY Executable and linking format (...	185.88.181.55	4444	10.10.1.20	52172
14:33:42.6...	attempted-r...	ET SCAN Sqlmap SQL Injection Scan	185.88.181.55		10.10.1.20	80

Рис. 1. Фрагмент отчета VipNet IDS NS о многочисленных попытках SQL-инъекций

В ходе работы была запущена атака на виртуальную инфраструктуру вымышленной компании, а именно на ее веб-сайт, который некорректно обращается к базе данных новостей. Киберполигон выполняет скрипты атаки, и мы можем наблюдать следы в IDS и в самой инфраструктуре. Затем производится анализ пакетов из IDS и собственно пострадавших узлов инфраструктуры на предмет взлома.

На рис. 1 представлены многочисленные попытки SQL-инъекций, отраженные в виде событий в IDS. Система IDS анализирует содержимое сетевых пакетов и классифицирует их согласно внутренним правилам, указывает на SQL-инъекции, об этом говорит как user-agent пакетов (sqlmap), так и непосредственно содержимое запроса.

В ходе анализа атаки на веб-сайт в IDS был обнаружен подозрительный GET-запрос, представленный на рис. 2.

Инъекция использует оператор UNION, чтобы объединить результат существующего SQL-запроса с новым набором данных.

По части GET-запроса news/view?id= можно понять, что уязвимым является запрос получения новостей на сайте (рис. 3). Здесь параметр id обрабаты-

вается некорректно, что дает возможность злоумышленнику внедрить свой SQL-код. Кроме того, в запросе загружается файл, закодированный шестнадцатеричным числом, расположение которого указано на рис. 4.

На рис. 5 приведен PHP-скрипт, который открывает reverse shell для злоумышленника, что позволяет злоумышленнику закрепиться и продолжить атаку на сеть компании.

```
GET/news/view/?id=-6150 UNION ALL SELECT NULL,NULL,CONCAT(0x71626a7671,IFNULL(CAST
  (LENGTH(LOAD_FILE(0x2f7661722f7777772f68746d6c2f6874646f63732f6e6577732f766965772
  f746d7075746c6c722e706870)) AS.
```

Рис. 2. GET-запрос, содержащий SQL-инъекцию

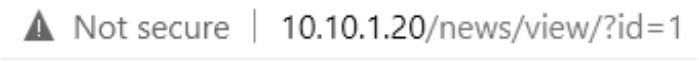

Рис. 3. Уязвимый URL

Рис. 4. Путь вредоносного скрипта

Чтобы исправить эту уязвимость, необходимо найти функцию, отвечающую за обработку запросов к новостям компании. Эта php-функция findByID() располагается в файле CModel.php (рис. 5).

```
public function findById($id)
    $db = CDb::getConnection();
    $table = this->tableName();
    $result = $db->prepare("SELECT * FROM $table WHERE id = $id");
$result->execute();
return $result->fetch();
```

Рис. 5. Уязвимая функция

Здесь параметр id принимается как есть, не производятся какие-либо проверки на его корректность. Чтобы это исправить, нужно внедрить условие, чтобы параметр id содержал исключительно числовые значения, что, в свою очередь, исключит наличие букв и служебных символов, необходимых для эксплуатации SQL-инъекции. Исправленная функция приведена на рис. 6.

```
public function findById($id)
    $db = CDb::getConnection();
    $table = this->tableName();
    if (is_numeric($id))
{
    $result = $db->prepare("SELECT * FROM $table WHERE id = $id");
}
else
{
    $result = $db->prepare("SELECT * FROM $table WHERE id=1");

}
$result->execute();
return $result->fetch();
```

Рис. 6. Исправленная функция

Далее была запущена следующая атака, также основанная на Union based. Нарушитель также использует sqlmap для атаки на веб-сервер, где хранится сайт с каталогом во внешнем контуре компании. В IDS замечаем большое количество сканирований нашей инфраструктуры в DMZ-зоне (рис. 7). В столбцах указаны время получения пакета; код, соответствующий идентификатору сработавшего правила; количество сработавших правил в пакете; название правила; класс правила.

13:11:32.509 10.12.2024	2008538	1	ET SCAN Sqlmap SQL Injection Scan	attempted-recon
13:11:32.503 10.12.2024	2008538	1	ET SCAN Sqlmap SQL Injection Scan	attempted-recon
13:11:32.481 10.12.2024	2008538	1	ET SCAN Sqlmap SQL Injection Scan	attempted-recon
13:11:32.461 10.12.2024	2008538	1	ET SCAN Sqlmap SQL Injection Scan	attempted-recon
13:11:32.438 10.12.2024	2008538	1	ET SCAN Sqlmap SQL Injection Scan	attempted-recon
13:11:32.363 10.12.2024	2016672	1	ET WEB_SERVER SQL Errors in HTTP 200 Response (error i...	bad-unknown
13:11:32.357 10.12.2024	2008538	1	ET SCAN Sqlmap SQL Injection Scan	attempted-recon
13:11:32.323 10.12.2024	2016672	1	ET WEB_SERVER SQL Errors in HTTP 200 Response (error i...	bad-unknown
13:11:32.288 10.12.2024	2008538	1	ET SCAN Sqlmap SQL Injection Scan	attempted-recon
13:11:32.053 10.12.2024	3001075	1	AM SQL Generic SQLi in HTTP URI: 'SELECT FROM' query	client-side-exploit

Рис. 7. Сканирование на предмет уязвимости

Затем последовала лавина попыток виртуального хакера воспользоваться уязвимостью. В итоге успешным для злоумышленника оказался POST-запрос, представленный на рис. 8.

```
POST /index.php?product='%20UNION%20ALL%20SELECT%20%22%22,%22Result:%3Cbr%3E%3C
  ?php%20if(isset($_POST%5B'cmd'%5D))%7B$op=system($_POST%5B'cmd'%5D);die;%7D
  ?%3E%22,%22%22,%22%22,%22%22%20INTO%20OUTFILE%20'/var/www/html/webshell.php.
```

Рис. 8. POST-запрос с UNION SELECT-инъекцией

На рисунке видно, что при помощи SQL-команды UNION SELECT пересылается PHP-скрипт, с помощью которого злоумышленник открывает удаленную оболочку на зараженной машине для возможного управления в дальнейшем.

Здесь используется аналогичный подход, как и в предыдущей атаке. Злоумышленник записывает вредоносный php-код, используя system() для выполнения системных команд на сервере. Это работает, так как СУБД MySQL обладает правом записи в данной директории. Используя shell, хакер запускает meterpreter-сессию для дальнейшей атаки на систему. Злоумышленник воспользовался Metasploit Framework – популярной платформой для создания и загрузки эксплойтов. Следы этого можно заметить на сервере (рис. 9).

```
root@SQL-Server:/home/user# cd /var/www/html/
root@SQL-Server:/var/www/html# ls
caidao.php  css  index.php  meterpreter.php  robots.txt  site-map.txt  webshell.php
```

Рис. 9. Следы взлома на веб-сервере

Злоумышленник использует модуль China Chopper Caidao PHP Backdoor Code Execution – это webshell, которым пользуются обычно китайские хакеры.

В результате атакующий получает возможность:

– выполнять произвольные команды на сервере;
– просматривать, изменять или удалять файлы;
– эскалировать привилегии до уровня администратора сервера;
– использовать сервер для атак на другие цели.

Для устранения SQL-инъекции типа UNION SELECT необходимо исправить код для получения параметра products. Находим необходимую функцию в файле index.php, который отвечает за обработку запросов на веб-сервере. Здесь $product также никак не обрабатывается (рис. 10).

```
$q = "Select * from products where product_name like '%product%'";
```

Рис. 10. Уязвимый запрос

Для устранения SQL-инъекции можно выполнить аналогичные действия, но более разумно будет использовать параметризованные запросы с подго-

товленным выражением. Вместо запроса $q используется подготовленный запрос, показанный на рис. 11.

```
$stmt = $com->prepare("SELECT * FROM products WHERE product_name
    LIKE ?");
$product = $product . '%'; //Добавляем подстановочный знак
$stmt->bind_param("s", $product);
$stmt->execute();
$result = $stmt->get_result();
```

Рис.11. Подготовленный запрос для получения параметра product

1.3. СЦЕНАРИЙ СО СЛЕПОЙ ИНЪЕКЦИЕЙ

Следующим рассмотренным типом уязвимости является слепая SQL-инъекция. Особенностью такого типа инъекций является то, что злоумышленник не видит напрямую результатов своих запросов, а может замечать изменения лишь по косвенным признакам (например, по времени отклика сервера).

Была запущена атака по сценарию с виртуальным нарушителем, работающим внутри инфраструктуры компании, что усложняет его обнаружение. Следы взлома также показываются в IDS и остаются на затронутых в ходе атаке узлах.

В таком сценарии имеется злоумышленник, действующий изнутри компании, в которой использовались слабые пароли и неактуальные версии ПО с выявленными уязвимостями. Здесь хакер воспользовался Redmine – веб-приложением для управления проектами с открытым исходным кодом, написанным на Ruby. Злоумышленник воспользовался атакой XSS для включения REST API на сервере. В свою очередь, запросы REST API уязвимы для инъекции SQL-кода.

В IDS находится подозрительный запрос, представленный на рис. 12.

```
GET /issues.xml?project_id=1&subproject_id=2,3-IF%20
  ((select%20description%20from%20issues%20where%20subject%20like%20%22SECRET%25%22
  )%20like%20%22LET_IT_B0%25%22,%20SLEEP(5),%20null);
```

Рис. 12. Запрос со слепой SQL-инъекцией

Для проверки сервера на уязвимость написан похожий запрос и проверена задержка (рис. 13). Status 200 – это код состояния HTTP «OK», означающий успешный запрос; параметр type document запрашивается основным HTML-документом; initiator-other показывает, что запрос был инициирован неявным способом; size 3.7 кб показывает размер загруженного ответа; time 20.11 указывает на то, что обработка этого запроса заняла долгое время. Значит, сервер реагирует на инъекцию в запросе.

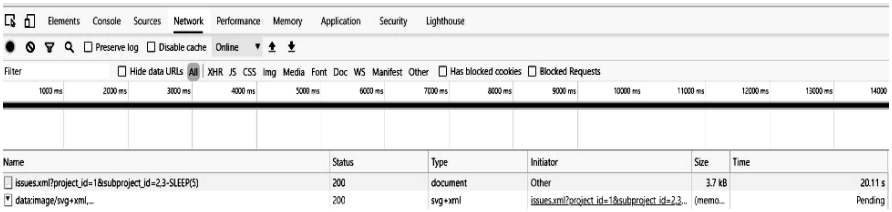

Рис. 13. Проверка на уязвимость

Злоумышленник перебирает символы в описании проекта с помощью GET-запросов, ориентируясь на задержку как знак того, что символ подобран верно. Это необходимо для получения контроля над сервером Redmine.

Таким образом, методом перебора злоумышленник получил верное описание проекта LET_IT_B, которое он применил в дальнейшем для эксплуатации уязвимости Redmine.

За обработку подобных запросов в Redmine отвечает файл query.rb, расположенный в директории /var/www/redmine/app/models. Находим уязвимую функцию project_statement, где для получения subproject_id используется each, который не обеспечивает фильтрацию символов. Заменив each на map, мы будем принимать в качестве идентификационного номера только числовые значения (рис. 14).

```
def project_ statement
    project_clauses = []
    if project && !project.descendants.active.empty ?
        if has_filter?("subproject_id")
            case operator_for("subproject_id")
            when '='
                # include the selected subprojects
                # ids = [project.id] + values_for("subproject_id").each(a: to i)
                ids = [project.id] + values_for("subproject_id").map(&:to_i)
                project_clauses « "#{Project.table_name}.id IN (%s)" % ids.join(',')
            when '!*'
                # main project only
                project_clauses « "#{Project.table_name}.id = %d" % project.id
            else
                #all subprojects
                project_clauses « "#{Project.table_name}.lft >= #{project.lft} AND #{Project.table_name}.rgt <=
                    #{project.rgt}"
            end
        elsif Setting.display_subprojects_issues?
            project_clauses « "#{Project.table_name}.lft >= #{project.lft} AND #(Project.table_name}.rgt <=
                #{project.rgt}"
        else
            project_clauses « "#{Project.table_name}.id = %d" % project.id
        end
    elsif project
        project_clauses « "#{Project.table_name}.id = %d" % project.id
    end
    project clauses.any? ? project_clauses.join(' AND ') : nil
end
```

Рис. 14. Исправленная уязвимость

Такой уязвимости подвержены версии Redmine вплоть до 3.3.9, которая и стояла на этом сервере. В результате эксплуатации уязвимости злоумышленник получил доступ к конфиденциальным частям базы данных, обходя проверку прав пользователя.

1.4. РЕКОМЕНДАЦИИ

На основе проведенного анализа были разработаны следующие рекомендации по недопущению и профилактике sql-инъекций. Наиболее эффективным подходом является продумывание безопасности на этапе разработки. Так, необходимо использовать параметризованные запросы, в которых используются плейсхолдеры, и вводимые данные воспринимаются уже как строки, а не команды SQL, что не позволяет ломать структуру запроса.

Еще одним действенным методом является применение ORM (Object-Relational Mapping – объектно-реляционное отображение). Здесь программист работает не напрямую с SQL-кодом, а с классами и объектами на языке программирования, которые уже потом преобразуются в запросы к БД. Здесь уже по умолчанию выполняется параметризация запросов и валидация типов. Популярными ORM являются SQLAlchemy, Doctrine и др. Однако этой метод может не подходить для систем, требующих максимального быстродействия, так как ORM может генерировать неоптимальные запросы.

Также хорошей практикой будет обязательная валидация и нормализация вводимых пользователем данных, а также проверка на уровне кода, что данные введены в правильном формате. В качестве вспомогательного метода можно проверять запрос на наличие специфических символов – кавычек или команд, характерных для SQL, например UNION или ORDER BY.

Для защиты веб-приложений уже в ходе эксплуатации необходимо использовать WAF (Web Application Firewall – межсетевой экран веб-приложений). Используя заложенные в них правила, они могут блокировать SQL-инъекции и предупреждать защитников о подозрительных запросах и атаках.

Кроме того, необходимо поддерживать ПО в актуальном состоянии, следить за уязвимостями используемого ПО и своевременно применять патчи от разработчиков.

ЗАКЛЮЧЕНИЕ

Был изучен теоретический материал, посвященный SQL-инъекциям и обучению противодействия им. Существующие платформы обучения нацелены на демонстрацию самой уязвимости и опыта на стороне атакующих, не давая навыков непосредственно анализа уязвимого кода. Такой навык

вырабатывается тренировками на виртуальной инфраструктуре киберполигона Ampire.

Были рассмотрены варианты воздействия SQL-инъекций на инфраструктуру компании на примере сценариев киберполигона Ampire. Эти сценарии позволяют обучающимся получить практический опыт нахождения и устранения последствий различных уязвимостей. Было показано, что злоумышленники часто применяют такое автоматизированное средство поиска и эксплуатации уязвимостей, как sqlmap. В более сложных случаях злоумышленник использует знакомые CVE, если видит уязвимые версии программ. Чаще всего атакам такого типа подвергаются веб-серверы, а именно запросы к базам данным, например БД новостей или БД товаров.

Разработчикам следует уделять особое внимание при написании таких запросов, внедряя проверки ввода или параметризуя запросы. Кроме того, некоторые фреймворки, например Laravel, не позволят написать функцию в таком явном уязвимом виде.

В случае слепой инъекции злоумышленник скорее всего знал о такой уязвимости данной версии Redmine. Поэтому следует поддерживать используемое ПО в актуальном состоянии, следить за новыми обнаруженными уязвимостями в используемых программах. Кроме того, хорошей практикой был бы аудит открытого исходного кода программ на предмет уязвимостей, но это требует много времени и квалифицированных специалистов. Также, современные ORM, такие как SQLAlchemy или Doctrine, позволяют писать код на более высоком уровне абстракции, не задавая SQL-запросы в коде вручную.

Преимуществом киберполигона можно назвать то, что он моделирует целую инфраструктуру, похожую на настоящее предприятие, и показывает многофазные сценарии атак, где SQL-инъекция может быть лишь одним из многих элементов Cyber-Kill Chain. Кроме того, может использоваться разная архитектура веб-приложений, например Apache или Nginx, и разные базы данных, работающие с языком SQL, например MySQL или MariaDB.

В дальнейшем планируется создание собственных уязвимых узлов и сценариев для киберполигона, которые охватывали бы и другие типы SQL-инъекций, например Error based, или инъекции второго порядка, которые реализуются при модификации данных в системе (например, при смене пароля).

Таким образом, SQL-инъекция до сих пор остается актуальной уязвимостью, и обучение на киберполигоне Ampire может помочь специалистам приобрести практические навыки, чтобы обнаруживать и исправлять ее.

СПИСОК ЛИТЕРАТУРЫ

1. *Рейн Т.С., Торгулькин В.В.* Основы информационной безопасности: учебное пособие. – Кемерово: КемГУ, 2024. – 117 с. – ISBN 978-5-8353-3270-0.

2. Киберполигон Ampire: сайт. – URL: https://amonitoring.ru/ampire-po/ (дата обращения: 30.11.2025).

3. *Болотов А.К.* Варианты реализации SQL-инъекций в веб-приложениях // Фундаментальные и прикладные исследования в информатике и цифровизации: материалы симпозиума XVIII (L) Международной научной конференции студентов, аспирантов и молодых ученых, приуроченной к 50-летию КемГУ, Кемерово, 26 апреля 2023 г. Вып. 24. – Кемерово, 2023. – С. 122–124. – EDN DROXCD.

4. *Болотов А.К.* Методика выполнения и противодействия SQL-инъекции на WEB-портал на киберполигоне Ampire // Фундаментальные и прикладные исследования в информатике и цифровизации: материалы симпозиума XIX (LI) Международной научной конференции студентов, аспирантов и молодых ученых, Кемерово, 25 апреля 2024 г. Вып. 25. – Кемерово, 2024. – С. 85–88. – EDN EUJDGA.

5. *Лапонина О.Р., Матошенко В.А*. Сравнительный анализ CTF-платформ для обучения кибербезопасности // International Journal of Open Information Technologies. – 2022. – Vol. 10 (4). – URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-ctf-platform-dlya-obucheniya-kiberbezopasnosti (дата обращения: 30.11.2025).

6. SQL injection teaching based on SQLi-labs / C. Ping, W. Jinshuang, Y. Lanjuan, P. Lin // 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, – IEEE, 2020. – P. 191–195. – DOI: 10.1109/ICISCAE51034.2020.9236904.

7. *Sommervoll Å.Å., Erdődi L., Zennaro F.M.* Simulating all archetypes of SQL injection vulnerability exploitation using reinforcement learning agents // International Journal of Information Security. – 2024. – Vol. 23 (1). – P. 225–246.

Рейн Татьяна Сергеевна, доцент Кемеровского государственного университета, кафедра информационной безопасности, кандидат физико-математических наук. Область научных интересов – методы социальной инженерии в компьютерной безопасности, поиск и анализ уязвимостей. E-mail: tsrein@mail.ru

Болотов Артём Константинович, лаборант кафедры информационной безопасности Кемеровского государственного университета. Область научных интересов – уязвимости информационных компьютерных систем. E-mail: bolotov160600@gmail.com

Устимец Артём Русланович, лаборант кафедры информационной безопасности Кемеровского государственного университета. Область научных интересов – уязвимости информационных компьютерных систем. E-mail: artem03yst@mail.ru

DOI: 10.17212/2782-2230-2025-4-43-58

Analysis of SQL-injection exploitation using the Ampire cyber range*

T.S. Rein[1], A.K. Bolotov[2], A.R. Ustymec[3]

[1] *Kemerovo State University, 6 Krasnaya Street, Kemerovo, 65000, Russian Federation, Candidate of Physical and Mathematical Sciences, Associate Professor of Department of Information Security. E-mail: tsrain@mail.ru*
[2] *Kemerovo State University, 6 Krasnaya Street, Kemerovo, 65000, Russian Federation, laboratory assistant of Department of Information Security. E-mail: bolotov160600@gmail.com*
[3] *Kemerovo State University, 6 Krasnaya Street, Kemerovo, 65000, Russian Federation, laboratory assistant of Department of Information Security. E-mail: artem03yst@mail.ru*

At present, the acquisition of practical skills in cybersecurity – such as software vulnerability detection, network traffic analysis, and operating system administration –represents an essential stage in the training of information security specialists. The information infrastructure of modern organizations is often highly complex, and its various components may be interconnected in ways that are not immediately apparent. Consequently, practical competencies play a decisive role in both preventing and promptly mitigating information security incidents.
One of the most widespread vulnerabilities is the SQL injection – an exploit that enables an attacker to execute arbitrary queries to a database by manipulating the query itself, thereby obtaining access to sensitive information. In some cases, the attacker may also secure persistence within the information system, gaining control over the server.
This study is devoted to the analysis of SQL injection exploitation mechanisms and approaches to mitigating their consequences, using the Ampire cyber range as a case study. The paper examines scenarios involving different types of SQL injections (classical, UNION-based, and blind SQL), automated attack techniques employing tools such as sqlmap, and practical methods for eliminating vulnerabilities. The objective was to formulate basic recommendations for writing secure code through the use of parameterized queries and modern frameworks.
The Ampire cyber range, developed by the company Perspektivny Monitoring, serves as a platform for conducting cyber exercises aimed at developing applied cybersecurity skills. Kemerovo State University is among its users, employing the platform to train students in protecting information from both external and internal threats.
Special emphasis is placed on the educational value of the cyber range in preparing specialists under conditions approximating real-world environments, thereby reducing risks associated with web application vulnerabilities.

Keywords: SQL injection, cybersecurity, Ampire cyber range, sqlmap, vulnerability, parameterized queries, specialist training, web applications

* *Received 05 November 2025.*

REFERENCES

1. Rein T.S., Torgul'kin V.V. *Osnovy informatsionnoi bezopasnosti* [Fundamentals of information security]. Kemerovo, KemSU Publ., 2024. 117 p. ISBN 978-5-8353-3270-0.

2. *Kiberpoligon Ampire* [Cyberpolygon Ampire]. Website. Available at: https://amonitoring.ru/ampire-po/ (accessed 30.11.2025).

3. Bolotov A.K. [Options for implementing SQL injections in web applications]. *Fundamental'nye i prikladnye issledovaniya v informatike i tsifrovizatsii* [Fundamental and applied research in computer science and digitalization]. Proceedings of the symposium of the XVIII (L) International scientific conference of students, graduate students and young scientists, dedicated to the 50th anniversary of KemSU, Kemerovo, April 26, 2023. Iss. 24. Kemerovo, 2023, pp. 122–124. (In Russian).

4. Bolotov A.K. [Methodology for executing and countering SQL injection on a WEB portal at the Ampire cyber testing ground *Fundamental'nye i prikladnye issledovaniya v informatike i tsifrovizatsii* [Fundamental and applied research in computer science and digitalization]. Proceedings of the symposium of the XIX (LI) International scientific conference of students, graduate students and young scientists, Kemerovo, April 25, 2024. Iss. 25. Kemerovo, 2024, pp. 85–88. (In Russian).

5. Laponina O.R., Matoshenko V.A. Sravnitel'nyi analiz CTF-platform dlya obucheniya kiberbezopasnosti [Comparative analysis of CTF platforms for cybersecurity training]. *International Journal of Open Information Technologies*, 2022, no. 4. Available at: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-ctf-platform-dlya-obucheniya-kiberbezopasnosti (accessed 30.11.2025).

6. Ping C., Jinshuang W., Lanjuan Y., Lin P. SQL injection teaching based on SQLi-labs. *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, Dalian, China, 2020, pp. 191–195. DOI: 10.1109/ICISCAE51034.2020.9236904.

7. Sommervoll Å.Å., Erdődi L., Zennaro F.M. Simulating all archetypes of SQL injection vulnerability exploitation using reinforcement learning agents. *International Journal of Information Security*, 2024, vol. 23 (1), pp. 225–246.

Для цитирования:

Рейн Т.С., *Болотов А.К.*, *Устимец А.Р.* Анализ эксплуатации SQL-инъекций с использованием киберполигона Ampire // Безопасность цифровых технологий. – 2025. – № 4 (119). – С. 43–58. – DOI: 10.17212/2782-2230-2025-4-43-58.

For citation:

Rein T.S., Bolotov A.K., Ustimec A.R. Analiz ekspluatatsii SQL-in"ektsii s ispol'zovaniem kiberpoligona Ampire [Analysis of SQL injection exploitation using the Ampire cyber range]. *Bezopasnost' tsifrovykh tekhnologii = Digital Technology Security*, 2025, no. 4 (119), pp. 43–58. DOI: 10.17212/2782-2230-2025-4-43-58.