**Номер команды -** 3. Название Bear & Car

**Список участников**:

Белоусов Сергей Викторович

Казаков Иван Павлович

Попов Павел Сергеевич

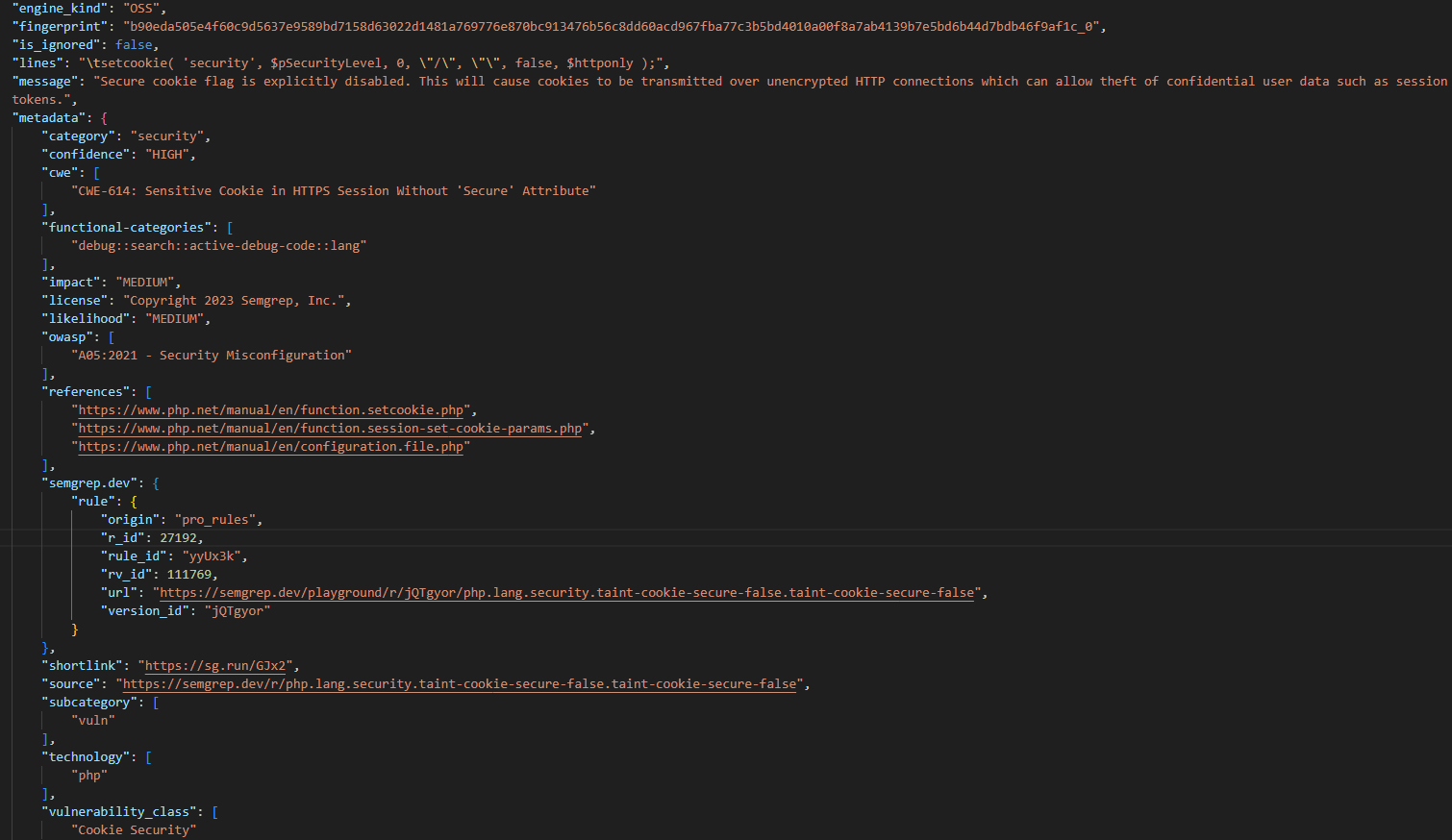
Прокопьев Алексей Алексеевич

Семакин Михаил Олегович

**Название приложения**: DVWA

<https://github.com/digininja/DVWA/>

**Уязвимость:**



**Описание уязвимости:** В строке кода «\tsetcookie( 'security', $pSecurityLevel, 0, \"/\", \"\", false, $httponly );»

Уязвимость, о которой идет речь, связана с отсутствием атрибута Secure у cookies, что делает их уязвимыми для передачи по незащищенным HTTP соединениям. Это потенциально позволяет злоумышленникам перехватывать cookies и использовать их для несанкционированного доступа к сессиям пользователей.

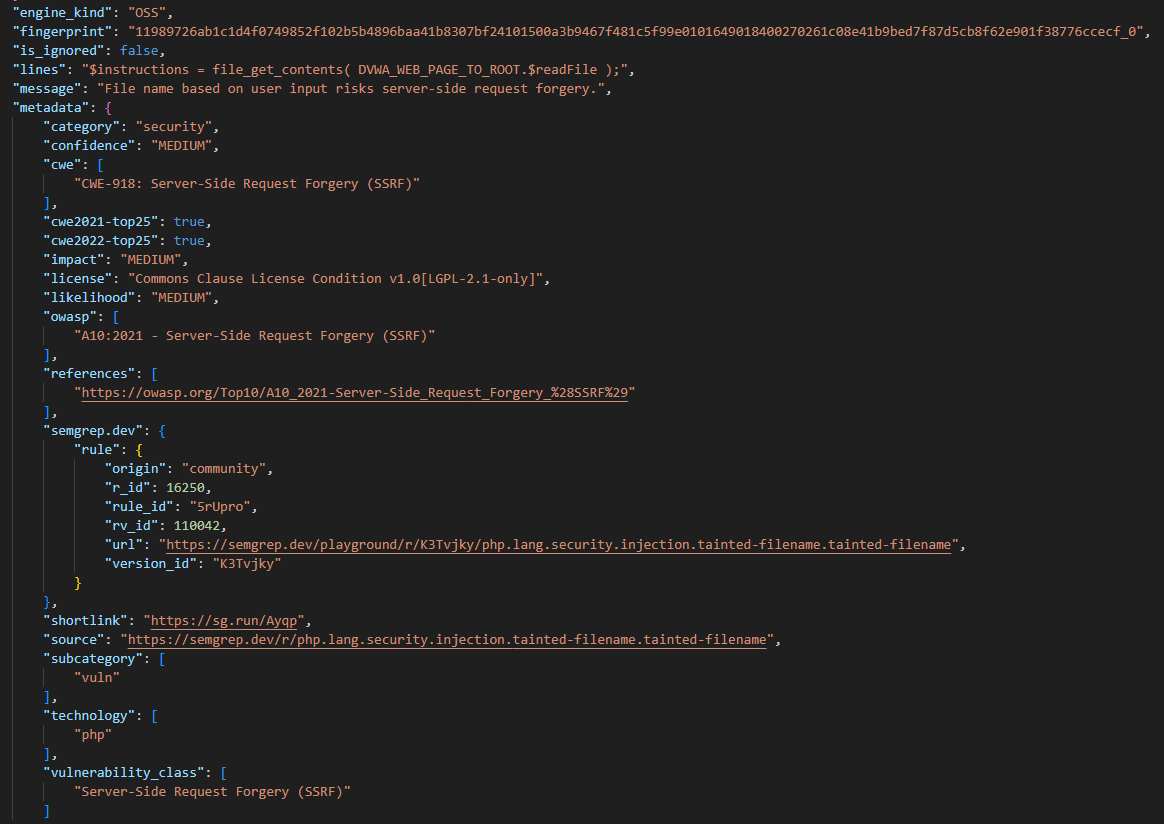
**Способы эксплуатации уязвимости:**

Перехват трафика (Man-in-the-Middle): Без атрибута Secure cookie могут передаваться по незащищенным HTTP соединениям. Злоумышленник, находящийся в той же сети, может перехватить этот трафик и получить доступ к cookies.

**Меры предотвращения уязвимости:**

1. Добавление атрибута Secure: Убедитесь, что все cookies, содержащие конфиденциальные данные, имеют атрибут Secure, чтобы они передавались только по защищенным HTTPS соединениям.
2. Принудительное использование HTTPS: Настройте сервер на принудительное использование HTTPS, чтобы минимизировать вероятность передачи данных по незащищенному HTTP соединению.

**Уязвимость:**



**Описание уязвимости**: В строке кода: «$instructions = file\_get\_contents( DVWA\_WEB\_PAGE\_TO\_ROOT.$readFile );»

Пользователь вводит имя файла, это небезопасно и может привести к SSRF атаке.

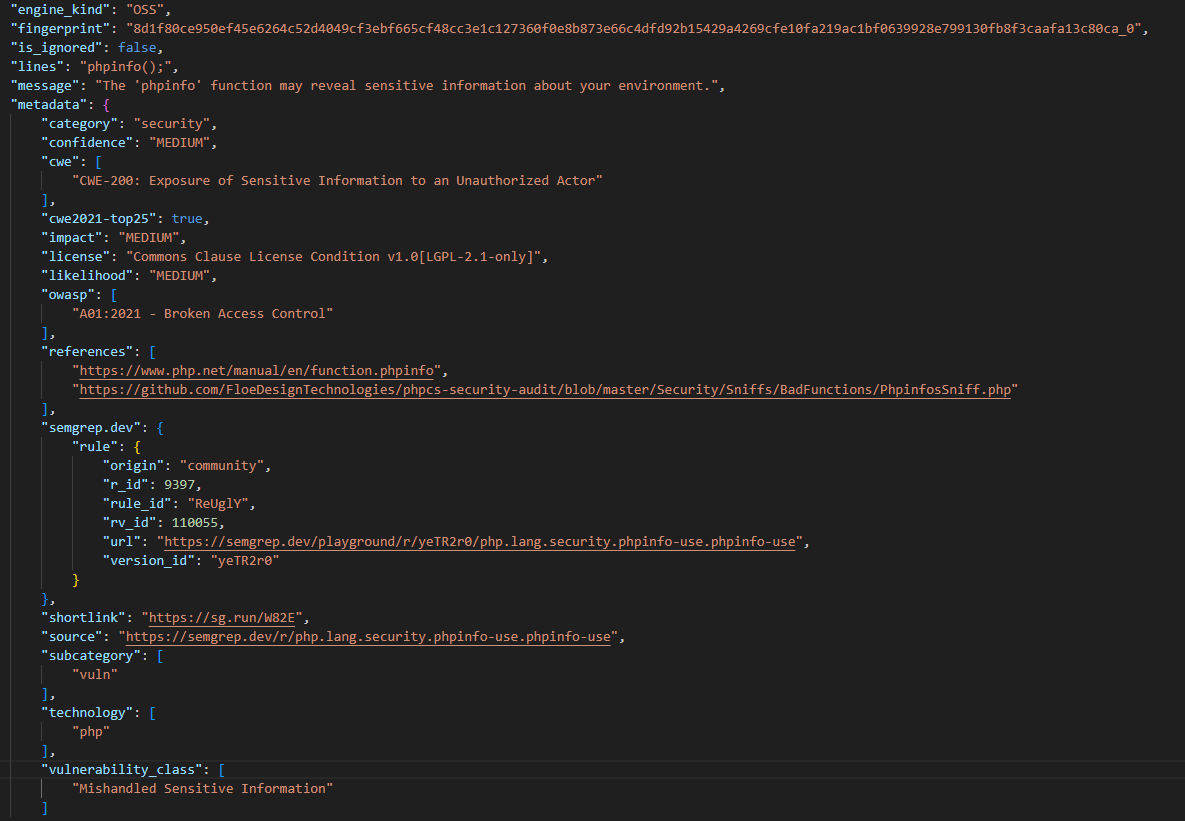
Уязвимость, связана с использованием внешнего ввода для формирования имени файла, который затем считывается с помощью функции «file\_get\_contents».

**Способы эксплуатации уязвимости:**

Манипуляция с пользовательским вводом: Злоумышленник может подставить в пользовательский ввод URL или путь к критически важным файлам на сервере. Это позволит ему получить доступ к данным, к которым он не должен иметь доступ.

В данном примере «$instructions = file\_get\_contents(DVWA\_WEB\_PAGE\_TO\_ROOT.$readFile);», если значение $readFile контролируется пользователем, злоумышленник может передать «/etc/passwd» в качестве значения $readFile. Это приведет к тому, что сервер выполнит запрос к указанному URL или файлу и вернет его содержимое злоумышленнику.

**Уязвимость:**



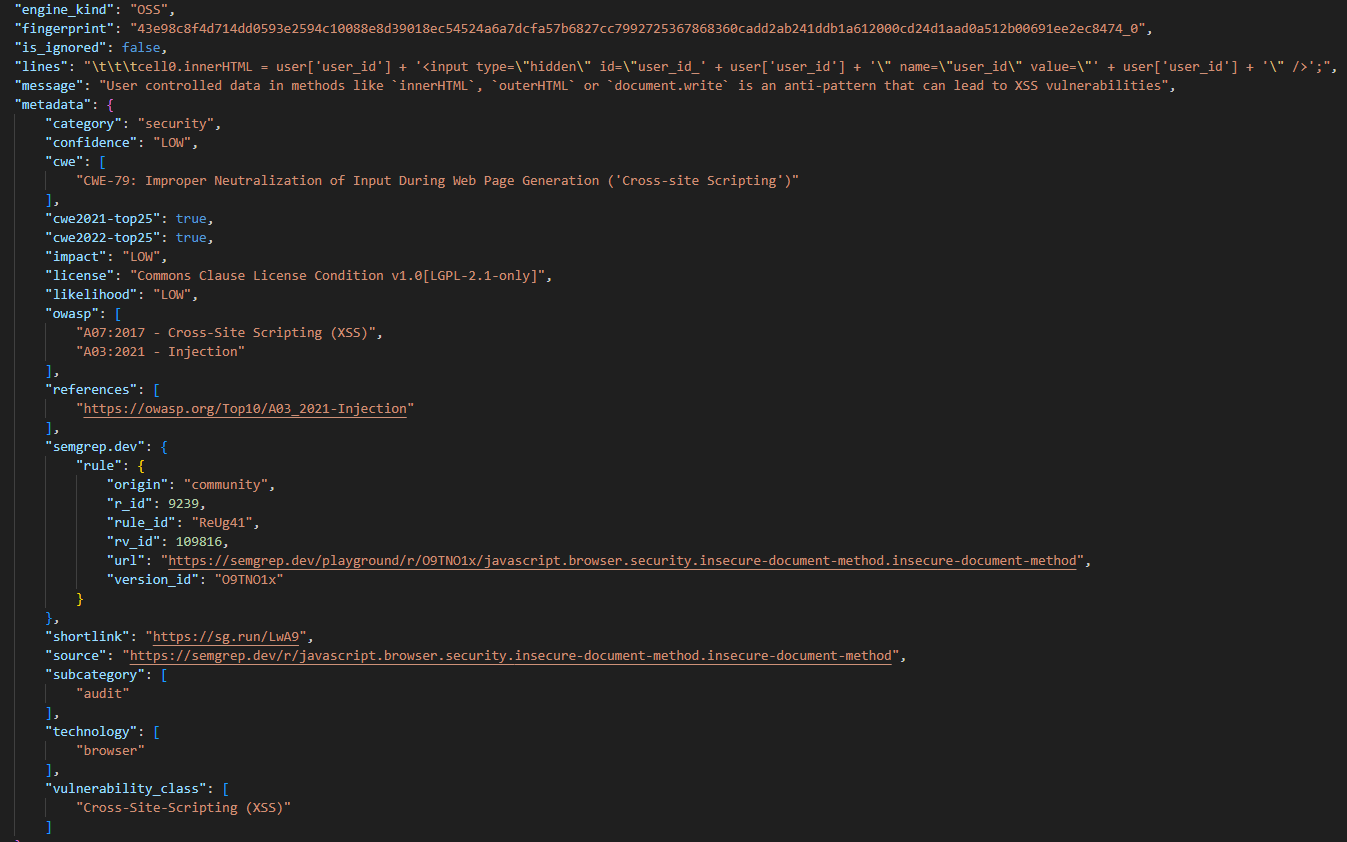
**Описание уязвимости**: В строчке кода «phpinfo();»

Функция «phpinfo()» выводит детальную информацию о текущей конфигурации PHP, включая версии используемых модулей, переменные окружения, установленные расширения и другие конфигурационные параметры.

**Способы эксплуатации уязвимости:**

Сбор информации для атак: Злоумышленники могут использовать информацию, предоставляемую phpinfo(), для сбора данных о сервере, включая установленные версии программного обеспечения и расширений, что может помочь им в выборе и проведении дальнейших атак.

**Уязвимость:**



**Описание уязвимости**: В строчке кода «\t\t\tcell0.innerHTML = user['user\_id'] + '<input type=\"hidden\" id=\"user\_id\_' + user['user\_id'] + '\" name=\"user\_id\" value=\"' + user['user\_id'] + '\" />';»

XSS уязвимость, вызванная вводом от пользователя в функциях `innerHTML`, `outerHTML`, `document.write`

Cross-Site Scripting (XSS) — это уязвимость веб-приложений, при которой злоумышленник может внедрить вредоносный скрипт в веб-страницу, который затем будет выполнен в браузере других пользователей. XSS позволяет атакующему обходить механизмы безопасности веб-приложения и получать доступ к различным данным, а также выполнять различные действия от имени пользователя.

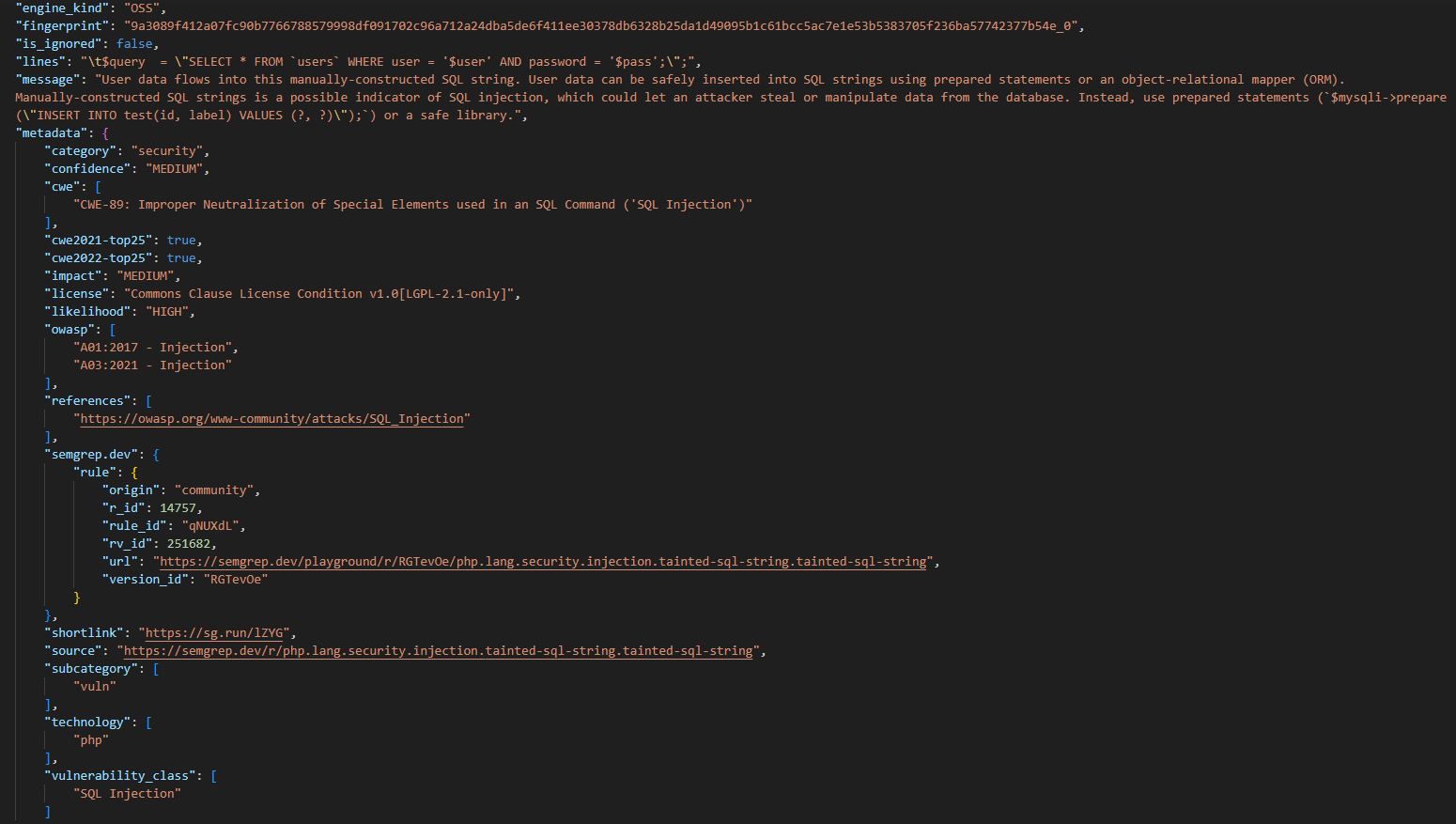
**Способы эксплуатации уязвимости:**

Вставка вредоносного кода: Злоумышленник может ввести вредоносный скрипт в поле ввода, например, в user\_id. Этот скрипт затем будет вставлен в HTML через innerHTML без должной обработки.

**Меры предотвращения уязвимости:**

1. Экранирование данных
2. Использование безопасных методов

**Уязвимость:**



**Описание уязвимости:** «\t$query = \"SELECT \* FROM `users` WHERE user = '$user' AND password = '$pass';\";»

SQL-инъекция позволяет злоумышленнику вставить или изменить SQL-запрос, что может привести к несанкционированному доступу, изменению или удалению данных в базе данных.

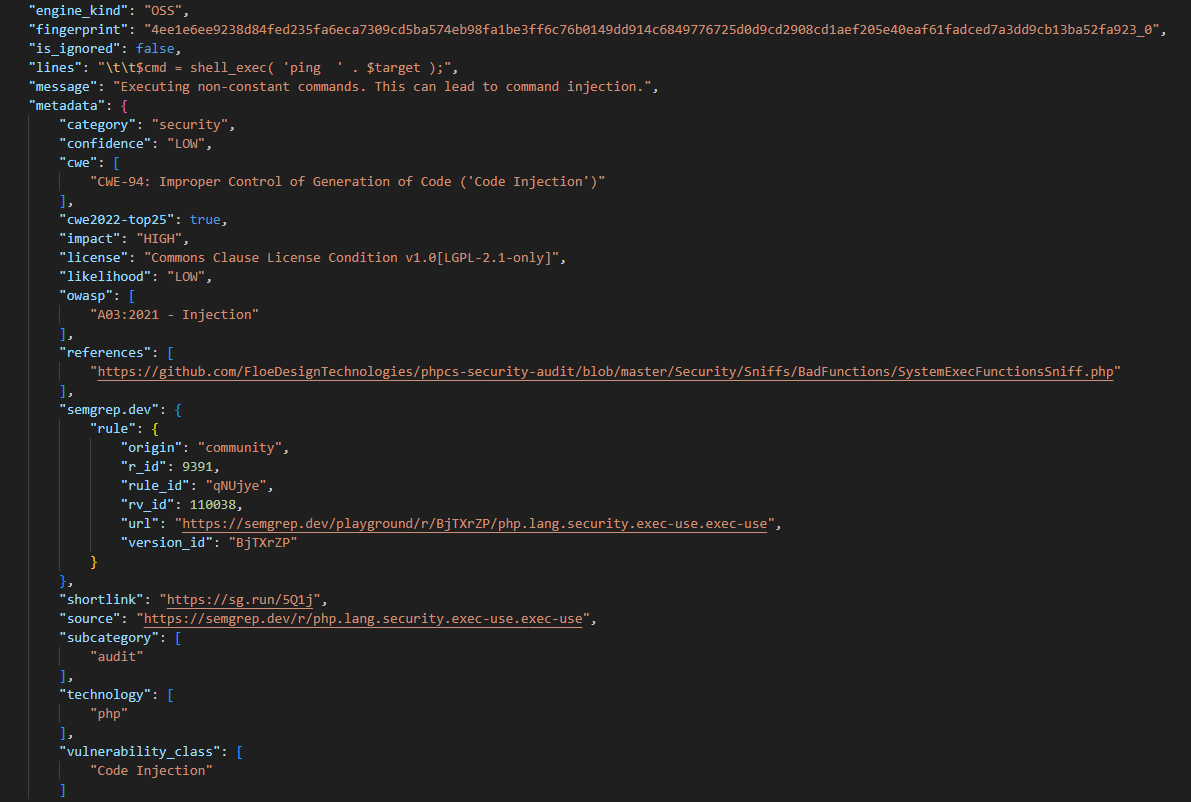
**Способы эксплуатации уязвимости:**

Манипуляция запросом: Эти данные напрямую вставляются в SQL-запрос без предварительной обработки, что позволяет злоумышленнику изменить структуру и логику запроса.

**Меры предотвращения уязвимости:**

1. Использование подготовленных выражений
2. Использование ORM

**Уязвимость:**



**Описание уязвимости**: В строчке «\t\t$cmd = shell\_exec( 'ping ' . $target );»

Code Injection, пользователь может исполнять shell команды.

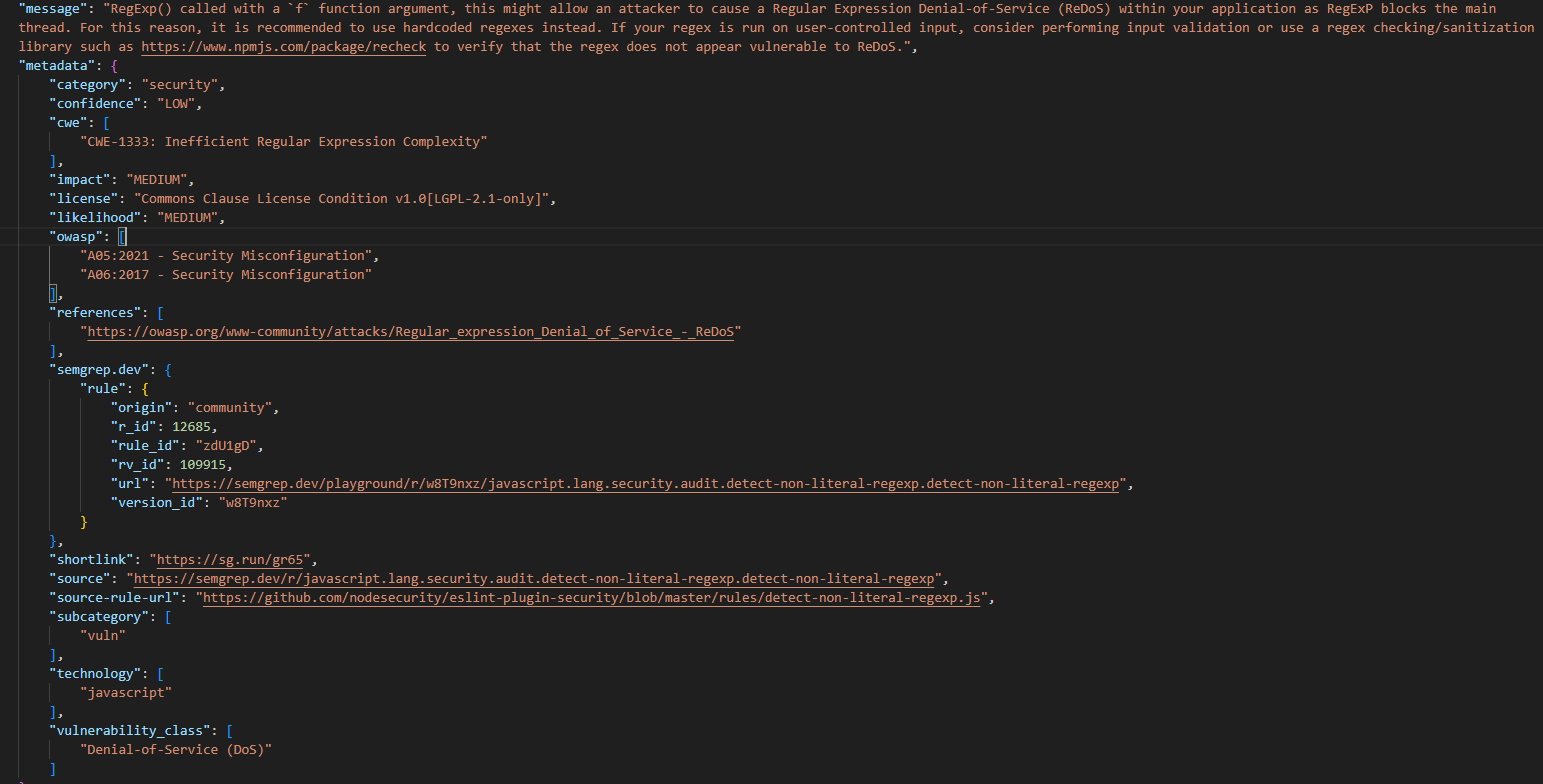
**Способы эксплуатации уязвимости:**

Манипуляция командой: Вредоносные данные напрямую добавляются к команде без предварительной обработки, что позволяет злоумышленнику вставить свои команды.

**Меры предотвращения уязвимости:**

1. Валидация данных
2. Использование escapeshellarg()

**Уязвимость:**



**Описание уязвимости:** DoS с помощью регулярных выражений. Суть заключается в том, что многие регулярные выражения будут очень долго обрабатываться при больших входных данных.

**Способы эксплуатации уязвимости:**

Блокировка основного потока: Поскольку регулярное выражение выполняется в основном потоке, длительная обработка заблокирует выполнение других задач, что приведет к отказу в обслуживании (DoS).

**Меры предотвращения уязвимости:**

1. Использование жестко заданных регулярных выражений:
2. Валидация ввода
3. Лимитированное время выполнения