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:

```
"engine_kind": "OSS",
"fingerprint": "b90eda505e4f60c9d5637e9589bd7158d63022d1481a769776e870bc913476b56c8dd60acd967fba77c3b5bd4010a00f8a7ab4139b7e5bd6b44d7bdb46f9af1c_0",
"is_ignored": false,
"lines": "\tsetcookie( 'security', $pSecurityLevel, 0, \"/\", \"\", false, $httponly );",
"message": "Secure cookie flag is explicitly disabled. This will cause cookies to be transmitted over unencrypted HTTP connections which can allow theft of confidential user data such as session tokens.",
"metadata": {
    "category": "security",
    "confidence": "HIGH",
    "cwe": [
        "CWE-614: Sensitive Cookie in HTTPS Session Without 'Secure' Attribute"
    ],
    "functional-categories": [
        "debug::search::active-debug-code::lang"
    ],
    "impact": "MEDIUM",
    "license": "Copyright 2023 Semgrep, Inc.",
    "likelihood": "MEDIUM",
    "owasp": [
        "A05:2021 - Security Misconfiguration"
    ],
    "references": [
        "https://www.php.net/manual/en/function.setcookie.php",
        "https://www.php.net/manual/en/function.session-set-cookie-params.php",
        "https://www.php.net/manual/en/configuration.file.php"
    ],
    "semgrep.dev": {
        "rule": {
            "origin": "pro_rules",
            "r_id": 27192,
            "rule_id": "yyUx3k",
            "rv_id": 111769,
            "url": "https://semgrep.dev/playground/r/jQTgyor/php.lang.security.taint-cookie-secure-false.taint-cookie-secure-false",
            "version_id": "jQTgyor"
        }
    },
    "shortlink": "https://sg.run/GJx2",
    "source": "https://semgrep.dev/r/php.lang.security.taint-cookie-secure-false.taint-cookie-secure-false",
    "subcategory": [
        "vuln"
    ],
    "technology": [
        "php"
    ],
    "vulnerability_class": [
        "Cookie Security"
```


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:


```
"engine_kind": "OSS",
"fingerprint": "11989726ab1c1d4f0749852f102b5b4896baa41b8307bf24101500a3b9467f481c5f99e010164901840027026 1c08e41b9bed7f87d5cb8f62e901f38776ccecf_0",
"is_ignored": false,
"lines": "$instructions = file_get_contents( DVWA_WEB_PAGE_TO_ROOT.$readFile );",
"message": "File name based on user input risks server-side request forgery.",
"metadata": {
    "category": "security",
    "confidence": "MEDIUM",
    "cwe": [
        "CWE-918: Server-Side Request Forgery (SSRF)"
    ],
    "cwe2021-top25": true,
    "cwe2022-top25": true,
    "impact": "MEDIUM",
    "license": "Commons Clause License Condition v1.0[LGPL-2.1-only]",
    "likelihood": "MEDIUM",
    "owasp": [
        "A10:2021 - Server-Side Request Forgery (SSRF)"
    ],
    "references": [
        "https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29"
    ],
    "semgrep.dev": {
        "rule": {
            "origin": "community",
            "r_id": 16250,
            "rule_id": "5rUpro",
            "rv_id": 110042,
            "url": "https://semgrep.dev/playground/r/K3Tvjky/php.lang.security.injection.tainted-filename.tainted-filename",
            "version_id": "K3Tvjky"
        }
    },
    "shortlink": "https://sg.run/Ayqp",
    "source": "https://semgrep.dev/r/php.lang.security.injection.tainted-filename.tainted-filename",
    "subcategory": [
        "vuln"
    ],
    "technology": [
        "php"
    ],
    "vulnerability_class": [
        "Server-Side Request Forgery (SSRF)"
    ]
```


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:


```
"engine_kind": "OSS",
"fingerprint": "8d1f80ce950ef45e6264c52d4049cf3ebf665cf48cc3e1c127360f0e8b873e66c4dfd92b15429a4269cfe10fa219ac1bf0639928e799130fb8f3caafa13c80ca_0",
"is_ignored": false,
"lines": "phpinfo();",
"message": "The 'phpinfo' function may reveal sensitive information about your environment.",
"metadata": {
    "category": "security",
    "confidence": "MEDIUM",
    "cwe": [
        "CWE-200: Exposure of Sensitive Information to an Unauthorized Actor"
    ],
    "cwe2021-top25": true,
    "impact": "MEDIUM",
    "license": "Commons Clause License Condition v1.0[LGPL-2.1-only]",
    "likelihood": "MEDIUM",
    "owasp": [
        "A01:2021 - Broken Access Control"
    ],
    "references": [
        "https://www.php.net/manual/en/function.phpinfo",
        "https://github.com/FloeDesignTechnologies/phpcs-security-audit/blob/master/Security/Sniffs/BadFunctions/PhpinfosSniff.php"
    ],
    "semgrep.dev": {
        "rule": {
            "origin": "community",
            "r_id": 9397,
            "rule_id": "ReUglY",
            "rv_id": 110055,
            "url": "https://semgrep.dev/playground/r/yeTR2r0/php.lang.security.phpinfo-use.phpinfo-use",
            "version_id": "yeTR2r0"
        }
    },
    "shortlink": "https://sg.run/W82E",
    "source": "https://semgrep.dev/r/php.lang.security.phpinfo-use.phpinfo-use",
    "subcategory": [
        "vuln"
    ],
    "technology": [
        "php"
    ],
    "vulnerability_class": [
        "Mishandled Sensitive Information"
    ]
```

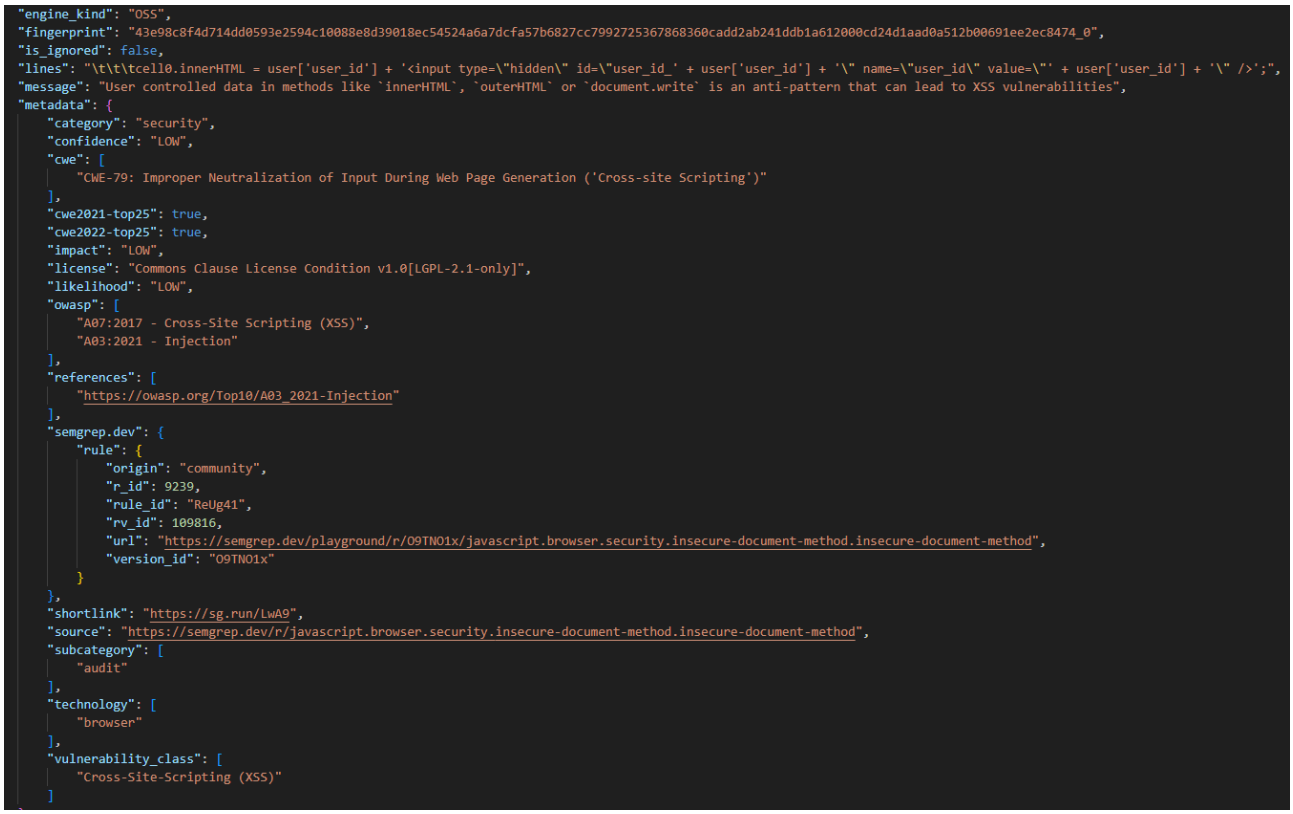

Описание уязвимости: В строчке кода «phpinfo();»

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:



```
"engine_kind": "OSS",
"fingerprint": "43e98c8f4d714dd0593e2594c10088e8d39018ec54524a6a7dcfa57b6827cc7992725367868360cadd2ab241ddb1a612000cd24d1aad0a512b00691ee2ec8474_0",
"is_ignored": false,
"lines": "\t\t\tcell0.innerHTML = user['user_id'] + '<input type=\"hidden\" id=\"user_id_' + user['user_id'] + '\" name=\"user_id\" value=\"' + user['user_id'] + '\" />';",
"message": "User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities",
"metadata": {
    "category": "security",
    "confidence": "LOW",
    "cwe": [
        "CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')"
    ],
    "cwe2021-top25": true,
    "cwe2022-top25": true,
    "impact": "LOW",
    "license": "Commons Clause License Condition v1.0[LGPL-2.1-only]",
    "likelihood": "LOW",
    "owasp": [
        "A07:2017 - Cross-Site Scripting (XSS)",
        "A03:2021 - Injection"
    ],
    "references": [
        "https://owasp.org/Top10/A03_2021-Injection"
    ],
    "semgrep.dev": {
        "rule": {
            "origin": "community",
            "r_id": 9239,
            "rule_id": "ReUg41",
            "rv_id": 109816,
            "url": "https://semgrep.dev/playground/r/O9TNO1x/javascript.browser.security.insecure-document-method.insecure-document-method",
            "version_id": "O9TNO1x"
        }
    },
    "shortlink": "https://sg.run/LwA9",
    "source": "https://semgrep.dev/r/javascript.browser.security.insecure-document-method.insecure-document-method",
    "subcategory": [
        "audit"
    ],
    "technology": [
        "browser"
    ],
    "vulnerability_class": [
        "Cross-Site-Scripting (XSS)"
    ]
```


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:


```
"engine_kind": "OSS",
"fingerprint": "9a3089f412a07fc90b7766788579998df091702c96a712a24dba5de6f411ee30378db6328b25da1d49095b1c61bcc5ac7e1e53b5383705f236ba57742377b54e_0",
"is_ignored": false,
"lines": "\t$query  = \"SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';\";",
"message": "User data flows into this manually-constructed SQL string. User data can be safely inserted into SQL strings using prepared statements or an object-relational mapper (ORM).
Manually-constructed SQL strings is a possible indicator of SQL injection, which could let an attacker steal or manipulate data from the database. Instead, use prepared statements (`$mysqli->prepare
(\"INSERT INTO test(id, label) VALUES (?, ?)\");`) or a safe library.",
"metadata": {
    "category": "security",
    "confidence": "MEDIUM",
    "cwe": [
        "CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')"
    ],
    "cwe2021-top25": true,
    "cwe2022-top25": true,
    "impact": "MEDIUM",
    "license": "Commons Clause License Condition v1.0[LGPL-2.1-only]",
    "likelihood": "HIGH",
    "owasp": [
        "A01:2017 - Injection",
        "A03:2021 - Injection"
    ],
    "references": [
        "https://owasp.org/www-community/attacks/SQL_Injection"
    ],
    "semgrep.dev": {
        "rule": {
            "origin": "community",
            "r_id": 14757,
            "rule_id": "qNUXdL",
            "rv_id": 251682,
            "url": "https://semgrep.dev/playground/r/RGTevOe/php.lang.security.injection.tainted-sql-string.tainted-sql-string",
            "version_id": "RGTevOe"
        }
    },
    "shortlink": "https://sg.run/lZYG",
    "source": "https://semgrep.dev/r/php.lang.security.injection.tainted-sql-string.tainted-sql-string",
    "subcategory": [
        "vuln"
    ],
    "technology": [
        "php"
    ],
    "vulnerability_class": [
        "SQL Injection"
    ]
```


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:


```
"engine_kind": "OSS",
"fingerprint": "4ee1e6ee9238d84fed235fa6eca7309cd5ba574eb98fa1be3ff6c76b0149dd914c6849776725d0d9cd2908cd1aef205e40eaf61fadced7a3dd9cb13ba52fa923_0",
"is_ignored": false,
"lines": "\t\t$cmd = shell_exec( 'ping  ' . $target );",
"message": "Executing non-constant commands. This can lead to command injection.",
"metadata": {
    "category": "security",
    "confidence": "LOW",
    "cwe": [
        "CWE-94: Improper Control of Generation of Code ('Code Injection')"
    ],
    "cwe2022-top25": true,
    "impact": "HIGH",
    "license": "Commons Clause License Condition v1.0[LGPL-2.1-only]",
    "likelihood": "LOW",
    "owasp": [
        "A03:2021 - Injection"
    ],
    "references": [
        "https://github.com/FloeDesignTechnologies/phpcs-security-audit/blob/master/Security/Sniffs/BadFunctions/SystemExecFunctionsSniff.php"
    ],
    "semgrep.dev": {
        "rule": {
            "origin": "community",
            "r_id": 9391,
            "rule_id": "qNUjye",
            "rv_id": 110038,
            "url": "https://semgrep.dev/playground/r/BjTXrZP/php.lang.security.exec-use.exec-use",
            "version_id": "BjTXrZP"
        }
    },
    "shortlink": "https://sg.run/5Q1j",
    "source": "https://semgrep.dev/r/php.lang.security.exec-use.exec-use",
    "subcategory": [
        "audit"
    ],
    "technology": [
        "php"
    ],
    "vulnerability_class": [
        "Code Injection"
    ]
```


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:

```
"message": "RegExp() called with a `f` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main
thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization
library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.",
"metadata": {
    "category": "security",
    "confidence": "LOW",
    "cwe": [
        "CWE-1333: Inefficient Regular Expression Complexity"
    ],
    "impact": "MEDIUM",
    "license": "Commons Clause License Condition v1.0[LGPL-2.1-only]",
    "likelihood": "MEDIUM",
    "owasp": [
        "A05:2021 - Security Misconfiguration",
        "A06:2017 - Security Misconfiguration"
    ],
    "references": [
        "https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS"
    ],
    "semgrep.dev": {
        "rule": {
            "origin": "community",
            "r_id": 12685,
            "rule_id": "zdU1gD",
            "rv_id": 109915,
            "url": "https://semgrep.dev/playground/r/w8T9nxz/javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp",
            "version_id": "w8T9nxz"
        }
    },
    "shortlink": "https://sg.run/gr65",
    "source": "https://semgrep.dev/r/javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp",
    "source-rule-url": "https://github.com/nodesecurity/eslint-plugin-security/blob/master/rules/detect-non-literal-regexp.js",
    "subcategory": [
        "vuln"
    ],
    "technology": [
        "javascript"
    ],
    "vulnerability_class": [
        "Denial-of-Service (DoS)"
    ]
```


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