БЕЗОПАСНОСТЬ ЦИФРОВЫХ ТЕХНОЛОГИЙ. – 2023. – № 3 (110). – 23–39

МЕТОДЫ И СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.422 DOI: 10.17212/2782-2230-2023-3-23-39

РАЗРАБОТКА АНАЛИЗАТОРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ФУНКЦИЕЙ МОНИТОРИНГА*

В.А. МАЗУРЕНКО[1], А.В. ИВАНОВ[2]

[1] *630087, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, магистрант кафедры защиты информации. E-mail: mazurenko.2017@stud.nstu.ru*

[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, кандидат технических наук, заведующий кафедрой защиты информации. E-mail: andrej.ivanov@corp.nstu.ru*

В реалиях современного мира, в том числе и перехода из WEB 2.0 в WEB 3.0, ежедневно появляются новые продукты, которые делают жизнь человека удобнее. Это касается и программного обеспечения (ПО), которое используется в бизнесе или повседневной жизни. Появление нового программного обеспечения от специалистов разного уровня несет в себе как положительные, так и отрицательные моменты. Вслед за новыми технологиями появляются и новые угрозы. За 2022 год 70 % самых эксплуатируемых уязвимостей были связаны именно с уязвимостями в ПО. В рамках настоящей статьи предпринимается попытка разработки эффективного анализатора уязвимостей ПО с функцией мониторинга, способного обнаруживать новые уязвимости без оказания нагрузки на сеть. В статье описаны этапы разработки статьи, принцип работы программы, информация об источниках получения данных о целевой системе и о базе уязвимостей. Описан процесс мониторинга и возможности оператора при получении информации о имеющихся уязвимостях. Для проверки эффективности работы подготовлен тестовый стенд с уязвимостями, на котором проверится эффективность разработанного анализатора и инструмента с похожим функционалом.

Ключевые слова: уязвимости, разработка анализатора, программное обеспечение, контроль уязвимостей, мониторинг, кибербезопасность, сканер уязвимости, CVE, NIST

* Статья получена 05 июня 2023 г.

ВВЕДЕНИЕ

В современном мире технологии играют огромную роль в жизни, включая различные аспекты, такие как коммуникация, работа и развлечения. Однако с ростом использования технологий также растет угроза киберпреступности. Хакеры и другие злоумышленники постоянно ищут уязвимости в программном обеспечении (ПО), чтобы получить несанкционированный доступ к системам и данным. В этой ситуации необходимо разработать эффективный инструмент для обнаружения уязвимостей в ПО и предотвращения их эксплуатации.

Сегодня киберпреступность является одной из самых быстрорастущих проблем в области безопасности информации. С каждым днем появляются новые уязвимости и способы атаки на программное обеспечение (ПО), которые могут привести к серьезным последствиям для пользователей, организаций и даже государств.

Для защиты от кибератак разработчики ПО и эксперты по безопасности информации создают новые инструменты, которые позволяют обнаруживать и устранять уязвимости. Однако существующие инструменты анализа уязвимостей не всегда могут обеспечить полноценную защиту от новых угроз, которые могут возникнуть в будущем.

В настоящей статье будет рассмотрена разработка анализатора уязвимостей ПО с функцией мониторинга, который позволит проактивно обнаруживать и предотвращать новые угрозы безопасности. Основная цель работы заключается в разработке и реализации анализатора уязвимостей, способного находить новые и уже известные уязвимости, а также проводить их анализ и мониторинг.

Таким образом, разработка такого анализатора будет способствовать повышению уровня безопасности программного обеспечения, защите пользователей и организаций от потенциальных киберугроз и обеспечению устойчивости информационной инфраструктуры в целом.

1. АРХИТЕКТУРА АНАЛИЗАТОРА УЯЗВИМОСТЕЙ

Архитектура анализатора уязвимостей ПО может быть представлена как совокупность компонентов и модулей, выполняющих функции сканирования, обнаружения и анализа уязвимостей в ПО.

Главными компонентами анализатора уязвимостей ПО являются:

- сбор информации (информации о целевой системе, включая информацию относительно установленного ПО, а также наполнение базы данных существующими уязвимостями);

- сканирование уязвимостей (сканирование системы на наличие известных уязвимостей в ПО и службах, установленных в системе);
- анализ результатов (анализ результатов сканирования и выдача отчета о выявленных уязвимостях);
- мониторинг уязвимостей (мониторинг системы на наличие новых уязвимостей в ПО). Компонент использует систему оповещения.

Преимущества архитектуры анализатора уязвимостей ПО с функцией мониторинга:

- обнаружение новых уязвимостей. Благодаря мониторингу уязвимостей анализатор ПО может обнаруживать новые уязвимости сразу после их появления, что позволяет быстро реагировать на угрозы безопасности;
- автоматизация процесса. Анализатор ПО может автоматически сканировать систему на наличие уязвимостей и выдавать отчеты, что упрощает процесс обнаружения и устранения уязвимостей.

2. ПРИНЦИП РАБОТЫ АНАЛИЗАТОРА

2.1. СБОР ИНФОРМАЦИИ

Информацию об уязвимостях в ПО можно получить из различных источников:

- официальные сайты производителей ПО. Производители выпускают исправления уязвимостей и публикуют информацию об этом на своих сайтах;
- национальные и международные базы данных уязвимостей. Такие организации, как CERT, NIST, CVE, OWASP и другие, поддерживают базы данных уязвимостей, которые содержат информацию о последних уязвимостях, исправлениях и рекомендациях по их устранению;
- форумы, блоги и сообщества. Пользователи и эксперты могут публиковать информацию об уязвимостях, их исправлениях и методах защиты на форумах, в блогах и в социальных сетях;
- журналы безопасности. Журналы, такие как SecurityFocus, Dark Reading, SC Magazine и другие, публикуют статьи, новости и аналитические материалы о последних уязвимостях и их влиянии на информационную безопасность.

В целом, для получения информации об уязвимостях в ПО следует использовать различные источники и подписываться на обновления, чтобы быть в курсе последних событий.

В качестве главных источников информации о существующих уязвимостях решено взять National Institute of Standards and Technology (NIST).

NIST – это Национальный институт стандартов и технологий, агентство Министерства торговли США, созданное для развития и применения технологий, метрологических и стандартных методов в различных отраслях промышленности, включая информационные технологии и кибербезопасность [1].

Основные задачи NIST в области кибербезопасности включают разработку и публикацию стандартов и руководств, проведение исследований и анализа уязвимостей, а также разработку методов и инструментов для защиты информации и кибербезопасности.

NIST также поддерживает каталог уязвимостей (NVD – National Vulnerability Database), который является базой данных уязвимостей и содержит информацию о новых и известных уязвимостях, а также о связанных с ними угрозах и рекомендациях по их устранению.

Кроме того, NIST также разрабатывает и публикует руководства и стандарты в области кибербезопасности, такие как Стандарты шифрования данных (AES, SHA), Стандарты безопасности информационных систем (SP 800) и многое другое.

CVE – это стандарт идентификации уязвимостей в компьютерных системах и программном обеспечении (Common Vulnerabilities and Exposures), который был создан для облегчения обмена информацией об уязвимостях между различными организациями и производителями ПО [2].

Каждая уязвимость, опубликованная в CVE, имеет уникальный идентификатор, который состоит из префикса CVE-, за которым следует уникальный номер, например, CVE-2022-1234. Этот идентификатор используется для идентификации и отслеживания уязвимостей в различных базах данных уязвимостей.

CVE представляет собой совместный проект, который управляется организацией MITRE в партнерстве с CERT/CC, NIST и др. В рамках этого проекта эксперты по безопасности из различных организаций и компаний работают вместе для идентификации и описания новых уязвимостей, определения уровня угрозы и рекомендаций по устранению уязвимостей.

CVE является важным инструментом для анализа и управления уязвимостями в информационных системах и программном обеспечении. Благодаря стандартной идентификации уязвимостей различные организации и производители ПО могут легко обмениваться информацией об уязвимостях, быстро реагировать на новые угрозы и устранять уязвимости в своих продуктах. Из CVE настроена автоматическая выгрузка базы уязвимостей для последующего сравнения с ПО, установленным в системе, а также возможность ручного добавления уязвимостей. Сбор информации об установленном ПО в системе будем осуществлять с помощью выгрузки данных о серверах из Itop, ручного добавления ПО в программу, а также Ansible.

2.2. СКАНИРОВАНИЕ УЯЗВИМОСТЕЙ

Сканирование уязвимостей будет происходить на сервере, сам процесс сканирования заключается в сравнении выгруженной базы CVE и списка установленного ПО, разработка происходила на языке Python [3]. Из ПО извлекается его версия. Сканирование выполняется с помощью функции (листинг 1). Данная функция принимает на вход две версии ПО и оператор сравнения и возвращает True, если оператор сравнения выполняется для этих версий, и False, если оператор сравнения не выполняется.

Листинг 1 – Код функции сравнения

```
def _if_condition_true(v1, op, v2):
    v1 = _get_str_without_last_dot(v1)
    v2 = _get_str_without_last_dot(v2)

    # в версии могут быть буквы, их не учитываем
    v1, letters1 = _extract_letters_at_the_end(v1)
    v2, letters2 = _extract_letters_at_the_end(v2)

    # если версии равны - то сразу возвращаем True
    if v1 == v2 and (op == "<=" or op == ">="):
        return True

    list_v1 = v1.split(".")
    list_v2 = v2.split(".")

    # добиваем меньшую версию нулями до длины большей
    if len(list_v2) > len(list_v1):
        list_v1 = _fill_versions(list_v1, len(list_v2))
    elif len(list_v1) > len(list_v2):
        list_v2 = _fill_versions(list_v2, len(list_v1))
    # LOGIC
    # сравниваем части версий между точками
    l = len(list_v1)
    # здесь уже равенства быть не может, только > <
    # _get_value_to_compare: возвращает int, а при невозможности - str
    for i in range(0, len(list_v1) - 1):
        if op == ">":
            if _get_value_to_compare(list_v1[i]) < _get_value_to_compare(list_v2[i]):
                return False
        elif op == ">=":

            if _get_value_to_compare(list_v1[i]) < _get_value_to_compare(list_v2[i]):
                return False
        elif op == "<":
            if _get_value_to_compare(list_v1[i]) > _get_value_to_compare(list_v2[i]):
                return False
        elif op == "<=":
            if _get_value_to_compare(list_v1[i]) > _get_value_to_compare(list_v2[i]):
                return False
    return True
```

Сначала входные версии подготавливаются к сравнению. Удаляется последний символ (точка), если он присутствует в каждой версии, чтобы избежать сравнения строк различной длины, которые в конце содержат точки. Затем от версий отделяются возможные буквенные символы в конце, которые не учитываются при сравнении.

Далее происходит заполнение меньшей версии нулями до большей длины. Это делается для удобства сравнения, чтобы разные части версии находились на одинаковых позициях в списке. Если на входе есть разные по длине версии, то короткая версия дополняется нулями до большей длины.

После этого происходит логическое сравнение версий. Для этого входные версии представляются в виде списков чисел, разделенных точками. Затем в цикле сравниваются части версии между точками в списке. Если текущая часть одной версии меньше соответствующей части другой версии, то возвращается False. Если же текущая часть одной версии больше или равна соответствующей части другой версии, то цикл продолжается. Если все части версии одной версии больше или равны соответствующим частям другой версии, то возвращается True.

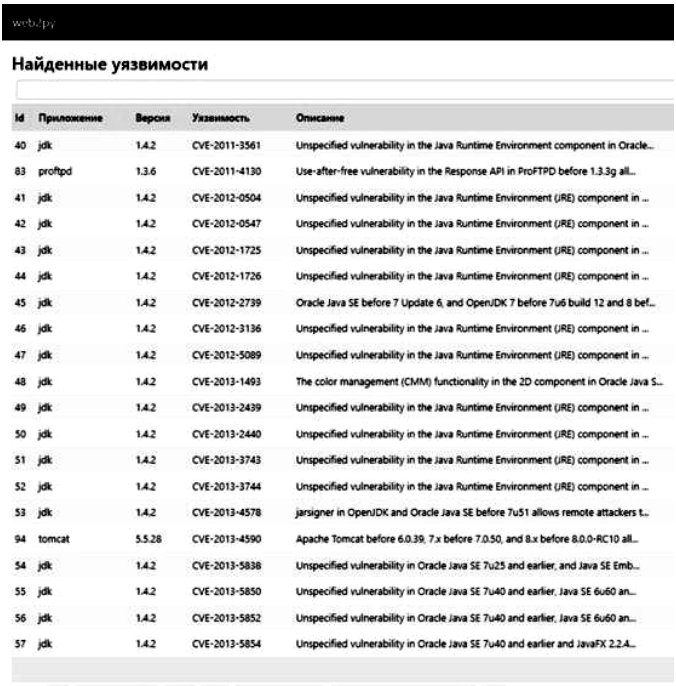
Если входные версии равны и оператор сравнения <= или >=, то функция возвращает True.

2.3. АНАЛИЗ РЕЗУЛЬТАТОВ

В итоге выполнения поиска уязвимостей получена таблица с перечнем уязвимых ПО (рис. 1).

Таблица состоит из следующих столбцов: id, название приложения, его версия, имя уязвимости, ее краткое описание, критичность уязвимости; сервер, на котором найдена уязвимость, и источник, откуда взята информация о наличии данного ПО в системе. Так как система постоянно обновляется, есть вероятность того, что при анализе найденных уязвимостей будет обнаружено, что данного ПО уже нет в системе, но по каким-либо причинам в iTop оно числится, или найденная уязвимость может быть неприменима к системе по ряду причин, в таком случае возможно либо исключить ненужное ПО, либо добавить найденную уязвимость на одном из серверов по id.

Информация о системе хранится в таблице «Таблица ПО с автоматическим добавлением», основные столбцы несут информацию о имени приложения, его версии, вендере, а также об источнике, сообщающем о данном ПО.



web.py

Найденные уязвимости

Id	Приложение	Версия	Уязвимость	Описание
40	jdk	1.4.2	CVE-2011-3561	Unspecified vulnerability in the Java Runtime Environment component in Oracle...
83	proftpd	1.3.6	CVE-2011-4130	Use-after-free vulnerability in the Response API in ProFTPD before 1.3.3g all...
41	jdk	1.4.2	CVE-2012-0504	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
42	jdk	1.4.2	CVE-2012-0547	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
43	jdk	1.4.2	CVE-2012-1725	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
44	jdk	1.4.2	CVE-2012-1726	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
45	jdk	1.4.2	CVE-2012-2739	Oracle Java SE before 7 Update 6, and OpenJDK 7 before 7u6 build 12 and 8 bef...
46	jdk	1.4.2	CVE-2012-3136	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
47	jdk	1.4.2	CVE-2012-5089	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
48	jdk	1.4.2	CVE-2013-1493	The color management (CMM) functionality in the 2D component in Oracle Java S...
49	jdk	1.4.2	CVE-2013-2439	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
50	jdk	1.4.2	CVE-2013-2440	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
51	jdk	1.4.2	CVE-2013-3743	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
52	jdk	1.4.2	CVE-2013-3744	Unspecified vulnerability in the Java Runtime Environment (JRE) component in ...
53	jdk	1.4.2	CVE-2013-4578	jarsigner in OpenJDK and Oracle Java SE before 7u51 allows remote attackers t...
94	tomcat	5.5.28	CVE-2013-4590	Apache Tomcat before 6.0.39, 7.x before 7.0.50, and 8.x before 8.0.0-RC10 all...
54	jdk	1.4.2	CVE-2013-5838	Unspecified vulnerability in Oracle Java SE 7u25 and earlier, and Java SE Emb...
55	jdk	1.4.2	CVE-2013-5850	Unspecified vulnerability in Oracle Java SE 7u40 and earlier, Java SE 6u60 an...
56	jdk	1.4.2	CVE-2013-5852	Unspecified vulnerability in Oracle Java SE 7u40 and earlier, Java SE 6u60 an...
57	jdk	1.4.2	CVE-2013-5854	Unspecified vulnerability in Oracle Java SE 7u40 and earlier and JavaFX 2.2.4...

Рис. 1. Найденные уязвимости в системе

Fig. 1. Vulnerabilities found in our system

Таблица ПО с автоматическим добавлением показана на рис. 2. Прямо из веб-интерфейса пользователь имеет возможность редактировать, к примеру, версию приложения либо удалить информацию о нем. В данной таблице хранится информация о всех приложениях, добавленных как автоматически, так и вручную.

Уязвимости ПО могут быть классифицированы по критичности на основе потенциальных последствий и уровня уязвимости. Обычно уязвимости классифицируются следующим образом.

1. Критические уязвимости – уязвимости, которые могут привести к серьезному нарушению безопасности системы (например, к повышению привилегий на сервере или к получению удаленного доступа к системе без аутентификации).

2. Высокие уязвимости – уязвимости, которые могут привести к существенному нарушению безопасности системы, но не так серьезны, как крити-

ческие уязвимости. Например, это может быть уязвимость, позволяющая злоумышленнику получить доступ к конфиденциальной информации.

3. Средние уязвимости – уязвимости, которые могут привести к некоторому нарушению безопасности системы, но не имеют такого высокого уровня серьезности. Это могут быть, например, уязвимости, которые позволяют провести атаку переполнения буфера или осуществить XSS-атаку.

4. Низкие уязвимости – уязвимости, которые могут привести к небольшому нарушению безопасности системы. Это могут быть, например, уязвимости, связанные с недостаточной обработкой ошибок или с использованием слабых алгоритмов шифрования.

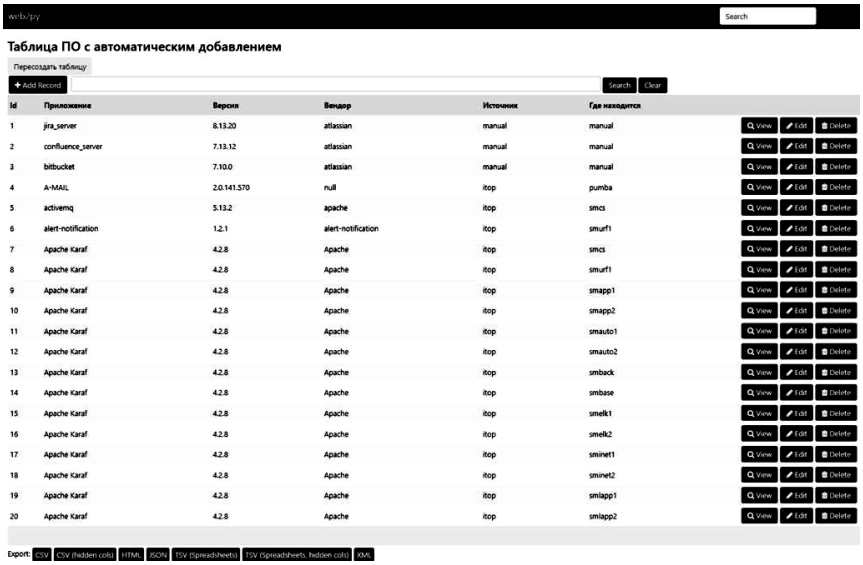


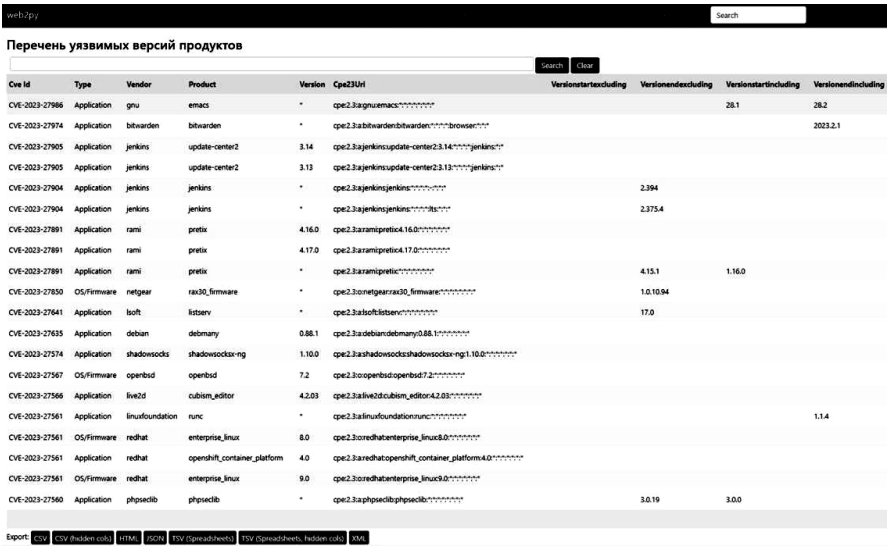
Рис. 2. ПО с автоматическим добавлением

Fig. 2. Software with automatic addition

Классификация уязвимостей по критичности помогает организациям определить, какие уязвимости нужно в первую очередь устранять, а какие можно отложить на более поздний срок. Она также позволяет оценить, насколько важно заботиться о безопасности системы и какие меры безопасности должны быть приняты.

После отсеивания ненужных нам уязвимостей в рамках разработки анализатора уязвимостей ПО будем считать, что необходимые нам уязвимости,

несут информацию об уязвимостях в ПО и ОС. Итоговый вариант таблицы, необходимый для последующего анализа, представлен на рис. 3.



web2py

Search

Перечень уязвимых версий продуктов

Search Clear

Cve Id	Type	Vendor	Product	Version	Cpe23Uri	Versionstartexcluding	Versionendexcluding	Versionstartincluding	Versionendincluding
CVE-2023-27986	Application	gnu	emacs	*	cpe:2.3:a:gnu:emacs:*:*:*:*:*:*:*:*			28.1	28.2
CVE-2023-27974	Application	bitwarden	bitwarden	*	cpe:2.3:a:bitwarden:bitwarden:*:*:*:*:browser:*:*:*				2023.2.1
CVE-2023-27905	Application	jenkins	update-center2	3.14	cpe:2.3:a:jenkins:update-center2:3.14:*:*:*:jenkins:*:*				
CVE-2023-27905	Application	jenkins	update-center2	3.13	cpe:2.3:a:jenkins:update-center2:3.13:*:*:*:jenkins:*:*				
CVE-2023-27904	Application	jenkins	jenkins	*	cpe:2.3:a:jenkins:jenkins:*:*:*:*:*:*		2.394		
CVE-2023-27904	Application	jenkins	jenkins	*	cpe:2.3:a:jenkins:jenkins:*:*:*:*:its:*:*		2.375.4		
CVE-2023-27891	Application	rami	pretix	4.16.0	cpe:2.3:a:rami:pretix:4.16.0:*:*:*:*:*:*:*				
CVE-2023-27891	Application	rami	pretix	4.17.0	cpe:2.3:a:rami:pretix:4.17.0:*:*:*:*:*:*:*				
CVE-2023-27891	Application	rami	pretix	*	cpe:2.3:a:rami:pretix:*:*:*:*:*:*:*:*		4.15.1	1.16.0	
CVE-2023-27850	OS/Firmware	netgear	rax30_firmware	*	cpe:2.3:o:netgear:rax30_firmware:*:*:*:*:*:*:*:*		1.0.10.94		
CVE-2023-27641	Application	lsoft	listserv	*	cpe:2.3:a:lsoft:listserv:*:*:*:*:*:*:*:*		17.0		
CVE-2023-27635	Application	debian	debmany	0.88.1	cpe:2.3:a:debian:debmany:0.88.1:*:*:*:*:*:*:*				
CVE-2023-27574	Application	shadowsocks	shadowsocksx-ng	1.10.0	cpe:2.3:a:shadowsocks:shadowsocksx-ng:1.10.0:*:*:*:*:*:*:*				
CVE-2023-27567	OS/Firmware	openbsd	openbsd	7.2	cpe:2.3:o:openbsd:openbsd:7.2:*:*:*:*:*:*:*				
CVE-2023-27566	Application	live2d	cubism_editor	4.2.03	cpe:2.3:a:live2d:cubism_editor:4.2.03:*:*:*:*:*:*:*				
CVE-2023-27561	Application	linuxfoundation	runc	*	cpe:2.3:a:linuxfoundation:runc:*:*:*:*:*:*:*:*				1.1.4
CVE-2023-27561	OS/Firmware	redhat	enterprise_linux	8.0	cpe:2.3:o:redhat:enterprise_linux:8.0:*:*:*:*:*:*:*				
CVE-2023-27561	Application	redhat	openshift_container_platform	4.0	cpe:2.3:a:redhat:openshift_container_platform:4.0:*:*:*:*:*:*:*				
CVE-2023-27561	OS/Firmware	redhat	enterprise_linux	9.0	cpe:2.3:o:redhat:enterprise_linux:9.0:*:*:*:*:*:*:*				
CVE-2023-27560	Application	phpseclib	phpseclib	*	cpe:2.3:a:phpseclib:phpseclib:*:*:*:*:*:*:*:*		3.0.19	3.0.0	

Export: CSV CSV (hidden cols) HTML JSON TSV (Spreadsheets) TSV (Spreadsheets, hidden cols) XML

Рис. 3. Перечень уязвимых версий продуктов

Fig. 3. A list of vulnerable product versions

2.4. МОНИТОРИНГ УЯЗВИМОСТЕЙ

Функция мониторинга уязвимостей необходима для того, чтобы обнаруживать новые уязвимости в ПО и реагировать на них в максимально короткие сроки. Без такой функции организации могут оставаться уязвимыми для атак на неопределенный период времени, что может привести к краже данных, нарушению конфиденциальности, ущербу репутации и финансовым потерям [6].

Функция мониторинга уязвимостей позволяет непрерывно сканировать системы и приложения на предмет новых уязвимостей, которые могут появиться в результате обновлений ПО или изменений в конфигурации системы [7]. Кроме того, мониторинг уязвимостей также позволяет отслеживать изменения в степени критичности уже известных уязвимостей и немедленно реагировать на угрозы безопасности.

Таким образом, функция мониторинга уязвимостей является необходимой для обеспечения безопасности информации и защиты от внешних угроз [8].

Согласно требованиям Payment Card Industry Data Security Standard (PCI DSS) уязвимости, определенные в рамках процесса сканирования, должны быть исправлены в течение заданного временного периода, известного как «срок устранения уязвимостей» (vulnerability remediation timeframe).

PCI DSS требует, чтобы организации, обрабатывающие платежные данные, устанавливали срок устранения уязвимостей в течение 30 дней [9]. Этот срок отсчитывается от даты обнаружения уязвимости. Однако если уязвимость имеет высокий уровень критичности, PCI DSS требует ее устранения максимум в течение семи дней.

В случае если исправление уязвимости в указанный срок невозможно, организация должна разработать и представить план мер по снижению рисков, связанных с данной уязвимостью [10]. План должен быть одобрен уполномоченным представителем организации, а также должен содержать информацию о дополнительных контролях, принятых для минимизации рисков, связанных с данной уязвимостью.

Согласно PCI DSS проведение сканирования системы на уязвимости требуется в рамках выполнения требования 11.2.2. Оно определяет, что необходимо выполнять квартальное внешнее сканирование сетевых уязвимостей и внутреннее сканирование сетевых уязвимостей, проводимое как минимум ежегодно, а также после любых изменений в сетевой инфраструктуре. Это позволяет выявлять и устранять уязвимости до того, как злоумышленники могут использовать их для атаки на систему.

В России требования по обеспечению безопасности персональных данных регулируются Федеральным законом «О персональных данных» от 27 июля 2006 года № 152-ФЗ [11]. Согласно статье 19 этого закона операторы персональных данных должны принимать меры по защите персональных данных, в том числе обеспечивать их конфиденциальность и недоступность для третьих лиц.

При этом статья 19.2.4. ФЗ «О персональных данных» гласит о необходимости использования средств защиты информации при ее обработке и хранении. В число этих средств входят в том числе системы обнаружения и предотвращения вторжений, антивирусные программы и тесты на проникновение. В контексте данного вопроса тесты на проникновение могут рассматриваться как сканирование на уязвимости.

Кроме того, требования по обеспечению безопасности информации содержатся в законодательстве в области критической информационной инфраструктуры (КИИ). Например, Федеральный закон от 28 июля 2012 года № 187-ФЗ «О безопасности критической информационной инфраструктуры

Российской Федерации» устанавливает обязанность субъектов КИИ разрабатывать и реализовывать меры по защите своей информационной инфраструктуры. В данном случае сканирование на уязвимости может быть одной из таких мер.

Также Федеральный закон от 27 июля 2010 года № 224-ФЗ «О противодействии коррупции» устанавливает требования по обеспечению безопасности информации при ее обработке и передаче. Согласно статье 7.1 этого закона организации должны обеспечивать защиту информации, в том числе от несанкционированного доступа к ней.

3. ОЦЕНКА ЭФФЕКТИВНОСТИ АНАЛИЗАТОРА УЯЗВИМОСТЕЙ ПО

3.1. МЕТОДОЛОГИЯ СРАВНЕНИЯ

Для оценки эффективности работы анализатора ПО с функцией мониторинга было проведено сканирование сторонним сканером, выбор пал на OpenVAS.

OpenVAS работает по следующему алгоритму [12].

1. Подготовка настроек сканирования. Пользователь настраивает параметры сканирования, такие как диапазон IP-адресов для сканирования, порты, которые следует проверить, и типы уязвимостей, которые следует искать [13].

2. Сканирование сети. OpenVAS начинает сканирование сети, отправляя запросы на различные порты и протоколы, чтобы проверить, есть ли на устройстве уязвимости.

3. Сбор информации. OpenVAS анализирует ответы от устройств и определяет, какие уязвимости присутствуют. Он также собирает информацию об устройствах и запущенных на них сервисах [14].

4. Анализ результатов. OpenVAS использует базу данных уязвимостей для сравнения результатов сканирования и определения, какие уязвимости обнаружены. Каждая уязвимость получает уровень критичности, и пользователь получает отчет о найденных уязвимостях.

5. Разрешение уязвимостей. OpenVAS предоставляет советы по разрешению найденных уязвимостей, что помогает улучшить безопасность системы.

6. Повторное сканирование. OpenVAS может использоваться для регулярного повторного сканирования, чтобы убедиться, что уязвимости были решены и система защищена.

В целом, принцип работы сканера уязвимостей OpenVAS заключается в том, чтобы отправлять запросы на порты и протоколы на устройствах [15], чтобы определить наличие уязвимостей и собирать информацию об устройствах. Затем результаты анализируются и сравниваются с базой данных уяз-

вимостей, чтобы определить, какие уязвимости обнаружены, и дать советы по разрешению этих уязвимостей.

Тестирование проводилось на хосте с заведомо известными тремя уязвимостями в программном обеспечении:

1) Apache HTTP Server – необходимость обновить версию;
2) OpenSSL – необходимость обновить версию;
3) PHP – необходимость обновить версию.

Критерии, по которым оценивается тестирование:

1) эффективность тестирования;
2) скорость тестирования;
3) нагрузка на сеть при тестировании;
4) оценка простоты использования;
5) дополнительная информация.

В результате работы OpenVAS сканером были обнаружены все заведомо известные уязвимости.

В результате работы анализатора уязвимостей ПО были определены не все уязвимости, что понизило точность анализатора. Это было связано с тем, что в ITOP нет информации относительно OpenSSL и его версии. Уязвимости, которые выявил анализатор уязвимостей ПО, показаны на рис. 4.

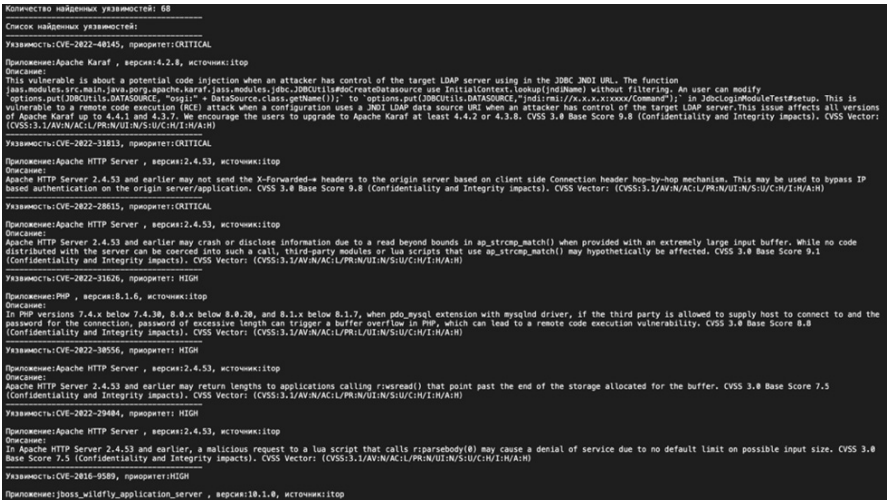


Рис. 4. Результат, который выдал анализатор уязвимостей ПО

Fig. 4. The result of the software vulnerability analyzer

В отличие от OpenVAS анализатор никак не проявляет себя в корпоративной сети и не оказывает никакой положительной нагрузки, что не может не радовать, так как инструмент остается незаметным. Анализатор выдает базовую информацию, необходимую для ознакомления с уязвимостями. Свежий отчет приходит ежедневно в виде сообщения на корпоративную почту, где удобно отслеживать изменения.

Скорость выполнения анализа определяется скоростью выполнения программы, что, в свою очередь, в сотни раз быстрее, чем выполнение сканирования.

3.2. ИТОГИ СРАВНЕНИЯ

По итогам сравнения можно сделать следующие выводы.

1. Анализатор не оказывает нагрузку на сеть, в отличие от OpenVAS.
2. Анализатор работает по принципу сверки полученных данных из двух баз, в то время как OpenVAS сам ищет информацию о системе с помощью сканирования и отправки запросов.
3. Анализатор уязвимостей работает в сотни раз быстрее и предоставляет свежий ежедневный отчет о новых уязвимостях, в то время как деликатное сканирование OpenVAS даже одного хоста занимает значительное время.
4. Анализатор более прост и удобен в использовании; OpenVAS обладает функционалом, понимание которого в первое время может вызывать затруднение.
5. Анализатор и OpenVAS предоставляют описание уязвимости, но OpenVAS это делает в более удобной форме, в отличие от анализатора.

ЗАКЛЮЧЕНИЕ

Разработанный анализатор уязвимостей – удобный инструмент для инженеров информационной безопасности, позволяющий получать информацию о появлении новых уязвимостей или о статусе уже существующих на ежедневной основе. Интерфейс является гибким и дружественным к пользователю: он поддерживает ручное добавление источников информации, CVE, ресурсов для мониторинга, а также имеет удобную систему отчетов.

Анализатор не оказывает нагрузки на корпоративную сеть, не вызывает ложных срабатываний мониторинга, позволяет сократить время поиска уязвимостей до шести минут, уменьшив затрачиваемое время в 7 раз.

Данное решение идеально подойдет для домашнего использования, так как имеет функцию ручного заполнения, которая будет полезна для проведения инвентаризации ПО на компьютере с целью последующего мониторинга ситуации на ПК, а также решение может быть актуально для небольших корпоративных сетей.

СПИСОК ЛИТЕРАТУРЫ

1. Towards deep learning models resistant to adversarial attacks / A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu // 6th International Conference on Learning Representations, ICLR 2018. – Vancouver, BC, Canada, 2018.

2. *Mell P., Scarfone K., Romanosky S.* Common vulnerability scoring system // IEEE Security & Privacy. – 2006. – Vol. 4 (6). – P. 85–89.

3. *Прохоренко Н.А., Дронов В.А.* Python 3 и PyQt5. Разработка приложений. – 2-е., перераб. и доп. изд. – СПб.: БХВ-Петербург, 2018. – 832 с.

4. *Varghese S., Kurian R.* Identifying vulnerabilities in a website using Uniscan and Comparing Uniscan, Grabber, Nikto // Proceedings of the National Conference on Emerging Computer Applications (NCECA). – 2021. – Vol. 3 (1). – P. 225–229.

5. *Воеводин В.А., Ганенков Д.С., Королев С.Д.* Об актуальности применения программного средства MaxPatrol 8 для целей аудита автоматизированных систем // Радиоэлектронные устройства и системы для инфокоммуникационных технологий («РЭУС-2022»). – М., 2022. – С. 240–244.

6. *Sekharan S.S., Kandasamy K.* Profiling SIEM tools and correlation engines for security analytics // 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). – IEEE, 2017. – P. 717–721.

7. О мониторинге угроз и уязвимостей информационной системы / С.Т. Мамбетов, Е.Е. Бегимбаева, С.К. Джолдасбаев, Б.О. Куламбаев, Г.Н. Казбекова // Известия НАН РК. Серия физико-математическая. – 2022. – № 4. – С. 68–80.

8. *Walkowski M., Oko J., Sujecki S.* Vulnerability management models using a common vulnerability scoring system // Applied Sciences. – 2021. – Vol. 11. – P. 1–25. – DOI: 10.3390/app11188735.

9. *Ataya G.* PCI DSS audit and compliance // Information Security Technical Report. – 2010. – Vol. 15 (4). – P. 138–144.

10. *Kotyan S.* A reading survey on adversarial machine learning: Adversarial attacks and their understanding. – URL: https://arxiv.org/pdf/2308.03363.pdf (accessed: 08.09.2023).

11. *Борисенко О.В.* Анализ Федерального закона № 152-ФЗ «О персональных данных» // Электронное приложение к Российскому юридическому журналу. – 2012. – № 2. – С. 26–30.

12. *Рахалкар С.* Краткое руководство по тестированию на проникновение. – Нью-Йорк: Springer Science + Business Media, 2019. – 139 с.

13. *Vimala K., Fugkeaw S.* VAPE-BRIDGE: bridging OpenVAS results for automating metasploit framework // 14th International Conference on Knowledge and Smart Technology (KST). – IEEE, 2022. – P. 69–74.

14. *Кравченко А.С., Шарапова В.О.* О применении сканеров уязвимостей информационных систем в УИС // Техника и безопасность объектов уголовно-исполнительной системы. – Воронеж, 2022. – С. 243–244.

15. *Shahriar H., Zulkernine M.* Taxonomy and classification of automatic monitoring of program security vulnerability exploitations // Journal of Systems and Software. – 2011. – Vol. 84. – P. 250–269.

Мазуренко Виктор Александрович, студент магистратуры Новосибирского государственного технического университета. Основное направление научных исследований – техническая защита информации. E-mail: mazurenko.2017@stud.nstu.ru

Иванов Андрей Валерьевич, заведующий кафедрой защиты информации Новосибирского государственного технического университета. Основное направление научных исследований – информационная безопасность, техническая защита информации. E-mail: andrej.ivanov@corp.nstu.ru

DOI: 10.17212/2782-2230-2023-3-23-39

Development of a software analyzer with monitoring function*

V.A. Mazurenko[1], A.V. Ivanov[2]

[1] *Novosibirsk State Technical University, 20 Karl Marx Prospekt, Novosibirsk, 630087, Russian Federation, master's student of the Information Security Department. E-mail: mazurenko.2017@stud.nstu.ru*

[2] *Novosibirsk State Technical University, 20 Karl Marx Prospekt, Novosibirsk, 630073, Russian Federation, Candidate of Technical Sciences, Head of the Information Security Department. E-mail: andrej.ivanov@corp.nstu.ru*

In the realities of today's world, including the transition from WEB 2.0 to WEB 3.0. Every day there are new products that make human life more convenient. This applies to software that is used in business or everyday life. The emergence of new software from different levels of experts bear in themselves both positive and negative aspects. Following the emergence of new technologies, new threats also appear. Over 2022, 70 % of the most exploited vulnerabilities were due to software vulnerabilities. This article attempts to develop an effective software

* *Received 05 June 2023.*

vulnerability analyzer with monitoring functionality capable of detecting new vulnerabilities without causing a load on the network.
The article will describe the stages of the article's development, the principle of the program's work, information about the sources of obtaining data on the target system and on the vulnerability database. The monitoring process and operator's capabilities in obtaining information about the vulnerabilities will be described. To test the efficiency of the work, a test bench with vulnerabilities will be prepared on which the efficiency of the developed analyzer and the tool with similar functionality will be tested.

Keywords: vulnerabilities, analyzer development, software, vulnerability control, monitoring, cybersecurity, vulnerability scanner, CVE, NIST

REFERENCES

1. Madry A., Makelov A., Schmidt L., Tsipras D., Vladu A. Towards deep learning models resistant to adversarial attacks. *6th International Conference on Learning Representations, ICLR 2018*, Vancouver, BC, Canada, 2018.

2. Mell P., Scarfone K., Romanosky S. Common vulnerability scoring system. *IEEE Security & Privacy*, 2006, vol. 4 (6), pp. 85–89.

3. Prokhorenko N.A., Dronov V.A. *Python 3 i PyQt5. Razrabotka prilozhenii* [Python 3 and PyQt5. Application development]. 2nd ed., rev. St. Petersburg, BHV-Peterburg Publ., 2018. 832 p.

4. Varghese S., Kurian R. Identifying vulnerabilities in a website using Uniscan and Comparing Uniscan, Grabber, Nikto. *Proceedings of the National Conference on Emerging Computer Applications (NCECA)*, 2021, vol. 3 (1), pp. 225–229.

5. Voevodin V.A., Ganenkov D.S., Korolev S.D. [On the relevance of the software tool MaxPatrol 8 for the audit of automated systems]. *Radioelektronnye ustroistva i sistemy dlya infokommunikatsionnykh tekhnologii («REUS-2022»)* [The Radio-electronic devices and systems for the infocommunication technologies ("REDS-2022")]. Moscow, 2022, pp. 240–244. (In Russian).

6. Sekharan S.S., Kandasamy K. Profiling SIEM tools and correlation engines for security analytics. *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2017, pp. 717–721.

7. Mambetov S., Begimbaeva E., Joldasbaev S., Kulambaev B., Kazbekova G. O monitoringe ugroz i uyazvimostei informatsionnoi sisetmy [On monitoring of threats and vulnerabilities of information system]. *Izvestiya NAN RK. Seriya fiziko-matematicheskaya = News of the National Academy of Sciences of the Republic of Kazakhstan. Physico-matematical series*, 2022, no. 4, pp. 68–80.

8. Walkowski M., Oko J., Sujecki S. Vulnerability management models using a common vulnerability scoring system. *Applied Sciences*, 2021, vol. 11, pp. 1–25. DOI: 10.3390/app11188735.

9. Ataya G. PCI DSS audit and compliance. *Information Security Technical Report*, 2010, vol. 15 (4), pp. 138–144.

10. Kotyan S. *A reading survey on adversarial machine learning: Adversarial attacks and their understanding*. Available at: https://arxiv.org/pdf/2308.03363.pdf (accessed 08.09.2023).

11. Borisenko O.V. Analiz Federal'nogo zakona № 152-FZ «O personal'nykh dannykh» [Analysis of Federal Law N 152-FZ "On personal data"]. *Elektronnoe prilozhenie k Rossiĭskomu yuridicheskomu zhurnalu = Electronic supplement to Russian Juridical Journal*, 2012, no. 2, pp. 26–30.

12. Rahalkar S. *Kratkoe rukovodstvo po testirovaniyu na proniknovenie* [Quick start guide to penetration testing]. New York, Springer Science + Business Media, 2019. 139 p. (In Russian).

13. Vimala K., Fugkeaw S. VAPE-BRIDGE: bridging OpenVAS results for automating metasploit framework. *14th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2022, pp. 69–74.

14. Kravchenko A.S., Sharapova V.O. [On the use of vulnerability scanners of information systems in the penal system]. *Tekhnika i bezopasnost' ob"ektov ugolovno-ispolnitel'noi sistemy* [Technics and security of penal system facilities]. Voronezh, 2022, pp. 243–244. (In Russian).

15. Shahriar H., Zulkernine M. Taxonomy and classification of automatic monitoring of program security vulnerability exploitations. *Journal of Systems and Software*, 2011, vol. 84, pp. 250–269.

Для цитирования:

Мазуренко В.А., Иванов А.В. Разработка анализатора программного обеспечения с функцией мониторинга // Безопасность цифровых технологий. – 2023. – № 3 (110). – С. 23–39. – DOI: 10.17212/2782-2230-2023-3-23-39.

For citation:

Mazurenko V.A., Ivanov A.V. Razrabotka analizatora programmnogo obespecheniya s funktsiei monitoringa [Development of a software analyzer with monitoring function]. *Bezopasnost' tsifrovykh tekhnologii = Digital Technology Security*, 2023, no. 3 (110), pp. 23–39. DOI: 10.17212/2782-2230-2023-3-23-39.