М. В. Бойко

ДОСТАТОЧНОСТЬ 10 ПРИНЦИПОВ OWASP ДЛЯ НАПИСАНИЯ БЕЗОПАСНОГО КОДА

OWASP Top 10 - это список, который публикуется открытым проектом обеспечения безопасности веб-приложений (OWASP). Эта статья сравнивает, насколько слабости, описанные в первой десятке списка на самом деле сообщают об уязвимостях, которые перечислены в Национальной базе уязвимостей (NVD). Таким образом, это позволяет эмпирически показать, является ли OWASP Top 10 список достаточно полным, для ориентирования в слабых местах кода, которые были найдены в последнее десятилетие.

Введение

Открытый проект безопасности веб-приложений (OWASP) публикует известный список уязвимостей, ранжированных по порядку от одного до десяти. Они стремятся предоставить этот список в качестве предупреждения разработчикам, чтобы избежать создания уязвимостей в написанных ими приложениях. С другой стороны, национальная база уязвимостей данных является хранилищем данных для всех уязвимостей, которые были официально опубликованы и описаны. Каждая уязвимость обычно имеет назначенный идентификационный номер уязвимости и подверженности (CVE). Идентификаторы CVE обычно имеют система классификации ошибок, приводящих к уязвимостям (CWE), которые описывают в целом типы уязвимостей, наблюдаемых в сообщенных CVE. Между CVE и CWE существуют отношения один ко многим соответственно.

I. Исследовательские вопросы

1. Каковы 10 самых частых CWE за последние десять лет в Национальной базе уязвимостей?

2. Какие идентификаторы CWE используются для уязвимостей, описанных OWASP топ 10?

3. Какой самый частый CWE за последние десять лет

4. Сколько уязвимостей с соответствующими CWE в Национальной базе уязвимостей фактически присутствует в OWASP топ 10 CWE?

II. Подготовка данных

Данные из Национальной базы данных уязвимостей (NVD) можно получить в формате нотации объектов JavaScript (JSON) [1]. Одна из проблем в отношении анализа того, как сообщается об уязвимостях является тот факт, что они сообщаются год за годом, не в одном файле. Кроме того, они не соответствуют структуре таблицы для легкого анализа, так как JSON объектно-ориентированный язык с вложением массивов JSON в JSON объект. Это приводит к необходимости нормализовать данные в виде таблицы для легкого анализы. Скрипт на питоне поможет решить эту проблему, так как все онлайн-конверторы JSON для файлов CSV занимали слишком много времени. Был создан собственный скрипт на Python, чтобы замаппить информацию через фреймы данных Pandas в объекты Series. Общий алгоритм в этого процесса:

1. Нормализация данных

   Преобразуются вложенные объекты JSON в массивы и операция повторяется, пока не доходит до самого внутреннего уровня. Это выполняется функцией из пакета «pandas.io.json».

2. Адаптация данных

   Преобразуется полученный объем данных в Series. Series позволяет нам проверить данные, а затем выбрать конкретный столбец, который нас интересует. Если есть дальнейшее вложение, происходит возврат к шагу 1 еще раз.

3. Извлечение данных

   Теперь извлекается столбец с CWE для связанных идентификаторов по каждой уязвимости. Примечание: может быть более одного CWE номера или идентификатора для любой уязвимости.

4. Преобразование данных

Наконец, данные записываются в файл Excel. Рисунок 1 дает нам точное представление об алгоритме реализация в Python 3.

Приведенная выше реализация алгоритма создает постраничные Excel файлы, которые необходимо объединить вручную в Microsoft Excel.

Далее, мы используем сводную таблицу Microsoft Excel чтобы вычислить количество значений для идентификаторов CWE по всем табли-

цам, полученным из Национальной базы уязвимостей.

```
import pandas as pd
from pandas.io.json import
json_normalize
import json
with open('nvdcve-1.1-2019.json') as
data_file:
    data = json.load(data_file)
df = json_normalize(data,
['CVE_Items'])
cvecol = df['cve']
cvenormalized =
json_normalize(cvecol)
problemtypedata =
cvenormalized['problemtype.problemtyp
e_data']

problemtypelist =
problemtypedata.tolist()
testdf =
pd.DataFrame(problemtypelist)
details = json_normalize(testdf[0],
['description'])
cvevalues = details['value']
cvedf = pd.DataFrame(cvevalues)

cvedf.to_excel(r'cve_counts.xlsx',
sheet_name='2019',
engine='xlsxwriter')
```

Рис. 1 – Адаптация данных и преобразование кода Python

III. Результаты

В целом было несколько интересных наблюдений. Давай попробуем отвечать на них один за другим в соответствии с вопросом исследования предложенные вопросы исследования. Каковы 10 самых частых CWE за последние десять лет в Национальной базе уязвимостей?

В целом, список топ 10 CWE, сгенерированных для статьи:

1. CWE 119 - Переполнение буфера
2. CWE 79 - Межсайтовый скриптинг
3. CWE 20 - Невалидная проверка ввода
4. CWE 200 - Информационная экспозиция
5. CWE 264 - Разрешения, привилегии и доступ управления
6. CWE 84 - Неправильная нейтрализация закодированного URI схемы на веб-странице
7. CWE 310 - Криптографические проблемы
8. CWE 125 - Выход за пределы массива при чтении
9. CWE 399 - Ошибки управления ресурсами
10. CWE 352 - Подделка межсайтовых запросов

Этот список является наиболее значительным вкладом в исследование с полным подсчетом и анализом слабостей, обнаруженных в программном обеспечении С 2010 по 2019 год. Это прежде всего дает нам широкий обзор самых слабых 10 мест безопасности в реальном мире за последнее десятилетие. Какие идентификаторы CWE используются для уязвимостей, описанных OWASP топ 10?

OWASP топ 10 описывает идентификатор CWE под разными категории [2]. Мы не будем вдаваться в подробности относительно того, что они для краткости. Они есть:

- A1 Инъекция
- A2 Сбой аутентификации
- A3 Воздействие на конфиденциальные данные
- A4 Внешнее воздействие на XML
- A5 Взломанный доступ
- A6 Неправильная настройка безопасности
- A7 Межсайтовый скриптинг
- A8 Небезопасная десериализация
- A9 Использование компонентов с известными уязвимостями
- A10 Недостаточное логирование

В приведенном выше списке «A9» особенно важен. Он говорит об использовании программных компонентов, которые, заранее известно, уязвимы. Самый простой способ устранения уязвимости в безопасности заключается в том, чтобы закрыть дыру с уязвимостью и выпустить новую версию.

Все CWE в приведенном выше списке соответствуют различным категории с различной степенью тяжести. Но они в сумме представляют уязвимости, о которых OWASP особенно предупреждает разработчиков и просит их избегать. Какой самый частый CWE за последние десять лет в Национальной базе данных уязвимостей и занимает ли он первую строчку в топ 10 OWASP?

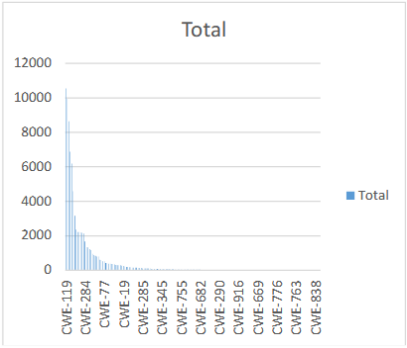

Рис. 2 – Количество идентификаторов CWE в Национальной базе уязвимостей для всех уязвимых мест 2010-2019 гг.

Самый частый из зарегистрированных CWE-119 (переполнение буфера). Это не то же самое, что OWASP Top 10 A1, который говорит

нам об инъекциях. Инъекция сообщается с CWE ID 77,78,88,89,90,81,564,917 и 943 для категории A1 в OWASP топ 10. Это можно визуально наблюдать на рис. 2 приведенном выше.

Еще одно интересное наблюдение из рис. 2 заключается в том, что OWASP действительно фокусируется на безопасности веб-приложений по сравнению с другими архитектурами, такими как настольные и мобильные. Но в зависимости от технологического стека, который используется для манипулированиями данными в фоновом режиме, злоумышленники может обойти меры безопасности, предлагаемые браузером по умолчанию, а затем напрямую атаковать базовое приложение через интерфейс программирования (API).

Например, рассмотрим проблему переполнения буфера во внутреннем сервере, который прослушивает HTTP-запросы. Атакующий может специально создать запрос HTTP GET [3] с параметрами, которые могут превышать пределы буфера и таким образом вызвать переполнение буфера с критическими для безопасности последствиями [7]. Они например могут включать произвольное исполнение кода, что может привести к полному отключению основных системы. Сколько уязвимостей с соответствующими CWE в Национальной базе уязвимостей фактически присутствует в OWASP топ 10 CWE? Ответ здесь, довольно ясен: общее количество слабых мест, перечисленных в Национальной базе уязвимостей, являются частью набором уязвимостей, перечисленных в OWASP топ 10. Все перечисленные слабые места в OWASP топ 10 включены в Национальной базе уязвимостей. Но это также означает, что список топ 10 OWASP недостаточно исчерпывающий, и разработчики должны знать о проблемах, которые могут быть не включены внутри в топ.

На рис. 3 представлен общее количество идентификаторов CWE которые наблюдались в течение 2010-2019 годов в Национальной База уязвимостей, а также присутствовали в OWASP топ 10 [2]. В топ 10 OWASP есть межсайтовый скриптинг "A7". Предполагая, что список является рейтингом уязвимостей, мы можем наблюдать, что для уязвимости, которая позиционируется в самой верхней части списка, полученного в этой статье из Результаты подсчета Национальной базы уязвимостей, не совпадает с топовой уязвимостью, описанной OWASP топ 10.

Это интересное наблюдение можно объяснить тем, что список OWASP топ 10 имеет другие критерии по сравнению с наблюдаемыми результатами за последнее десятилетие. Возможно, OWASP лучше учитывает потребности разработчиков. Таким образом, составители топ 10 более осведомлены об ошибках, которые совершают другие инженеры.

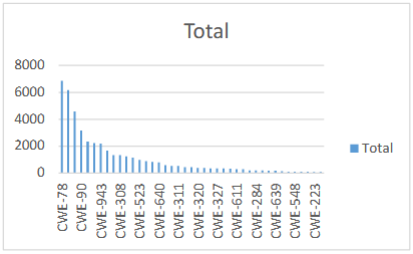

Рис. 3 – Количество идентификаторов CWE в Национальной базе данных уязвимостей для OWASP топ 10 уязвимых мест за 2010-2019 гг.

1. NVD - Data Feeds [Electronic resource] / – Nvd.nist.gov., 2020. – Mode of access: https://nvd.nist.gov/vuln/data-feeds. – Date of access: 06.02.2020.
2. CWE - CWE-1026: Weaknesses in OWASP Top Ten (2017)(3.4.1) [Electronic resource] / – Cwe.mitre.org., 2020. – Mode of access: https://cwe.mitre.org/data/definitions/1026.html. – Date of access: 08.02.2020.
3. Sridhar, M. FLASH IN THE DARK: ILLUMINATING THE LANDSCAPE OF ACTIONSCRIPT WEB SECURITY TRENDS AND THREATS / M. Sridhar, M. Chirva, B. Ferrell, K. Hamlen, D. Karamchandani. – Utdallas.edu. – Mode of access: http://www.utdallas.edu/ hamlen/sridhar17jissec.pdf. – Date of access: 08.02.2020.
4. Garg et al, S. Network-based detection of Android malicious apps / S. Garg // International Journal of Information Security. – 2017. – Vol. 16, № 4. – P. 385–400.
5. Jelen B. Microsoft Excel 2019: Pivot Table Data Crunching / B. Jelen, M. Alexander // Pearson. – 2019. – P.512
6. Heydt, M. Learning Pandas: Get to Grips with Pandas-a Versatile and High-Performance Python Library for Data Manipulation, Analysis, and Discovery / M. Heydt // Packt Publishing. – 2015. –P. 504
7. Black, P. E. Defeating Buffer Overflow: A Trivial but Dangerous Bug / P. E. Black, I. Bojanova // IT Professional. – 2016. – Vol. 18, № 6. – P. 58-61.
8. R. N. Peclat et al, R. N. Semantic Analysis for Identifying Security Concerns in Software Procurement Edicts / R. N. Peclat et al // New Generation Computing. – 2018. – Vol. 36, № 1. – P. 21-40.

Бойко Мария Владимировна, магистрантка каф. ИТАС, boikomary@gmail.com.

Научный руководитель: Навроцкий Анатолий Александрович, заведующий кафедрой информационных технологий автоматизированных систем БГУИР, доцент, navrotsky@bsuir.by.