УДК 004.056.5

***Дмитро ВАСИЛЬКІВ***

*Західноукраїнський національний університет*

## АНАЛІЗ ФАЙЛІВ ЗА ДОПОМОГОЮ SSDEEP

**Вступ**. Враховуючи стрімко зростаючі обсяги цифрових даних та збільшення кількості атак на них, або їх використання у здійснені атак, компаніям та їхнім співробітникам сфери кібербезпеки потрібно все більше автоматизувати процеси фільтрації. Здебільшого ці процеси використовують для відсіювання безпечних файлів від загального потоку серед якого є підозрілі або нецікаві файли [1].

Можливість порівнювати та знаходити схожі файли є одною з ключових для вирішення даної проблеми, проте як вона працює в масштабі. Насправді часто ми стикаємось з неможливістю обробити та перевірити усі потрібні файли, тому що нам довелося б провести нездійсненну кількість порівнянь. Ще гірше, якщо ці порівняння мали б відбуватися поза базою даних. Проте для полегшення даних процесів спеціалісти розробили оптимізацію ssDeep порівнянь, фактично завдяки їй ми можемо зменшити час на дані операції [2].

**Мета:** аналіз файлів з метою виявлення шкідливого програмного забезпечення за допомогою SSDEEP.

### 1. Алгоритм аналізу файлів

SSDeep Hash - це нечіткий хеш-алгоритм, розроблений для швидкого визначення та співставлення файлів. MD5 або SHA-1 (і наступні версії) це традиційні криптографічні хеш-функцій, вони ефективні, якщо потрібно точно перевірити збігу файлах, проте коли нам потрібно перевірити навіть незначні зміни ми можемо використовувати саме SSDeep. Він враховує ці варіації у файлах і фактично це дозволяє його використовувати для аналізу шкідливого програмного забезпечення, цифрової експертизи та інших безпекових завдань. SSDeep - це інструмент з відкритим кодом, який використовується для створення та порівняння нечітких хешів. В його основу покладений алгоритм частково керованого хешування (CTPH), який надає можливість ділити дані на сегменти змінних розмірів. Це гарантує що невеликі зміни не призводять до суттєвої різниці у хеш-значенні, що полегшує визначення схожості між файлами. Алгоритм роботи цього є хешу є досить маштабним, та широко висвітленим у наукових працях, проте в даній роботі, я б хотів висвітлити алгоритм дій під час та перед застосуванням SSDEEP [3, 4].

Алгоритм виконання тестування файлів за допомогою SSDEEP, розпочинається з встановлення інструменту SSDEEP, після цього йде генерація хешів для файлів, далі збереження хешів для бази порівняння, після нього вже відбувається саме порівняння і аналіз результатів. Варто ще зазначити що доцільно доповнити алгоритм автоматизацією процесу та інтерпретація результатів. Алгоритм складається з шести етапів. Розглянемо кожен з етапів детальніше.

1. Встановлення інструменту SSDEEP. Для початку роботи потрібно

встановити цей інструмент. Процес встановлення залежить від операційної системи. Це передбачає завантаження, встановлення та перевірка:

- для ОС Windows потрібно перейти на офіційний сайт SSDEEP або GitHub-репозиторій: SSDEEP на GitHub, завантажити потрібні файли, далі можливо буде необхідність розпакувати архів додати каталог з утилітою до змінної середовища PATH, щоб мати доступ до команди з будь-якого місця треба у розділі Змінні середовища додати шлях до SSDEEP. Для перевірки нам треба буде відкрити командний рядок (Cmd) та ввести команду ssdeep, якщо з'являється довідка або інформація про використання, встановлення завершено успішно;
- для більшості дистрибутивів Linux SSDEEP доступний у стандартних репозиторіях. Для Debian/Ubuntu виконується команда sudo apt update sudo apt install ssdeep, для Fedora/RHEL sudo dnf install ssdeep, а для Arch Linux sudo pacman -S ssdeep;
- в macOS встановлення проходить через Homebrew. Якщо він ще не встановлений потрібно ввести команду:

*/bin/bash -c $(curl -fsSL*

*https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh),*

після цього команда *brew install ssdeep* .встановлює SSDEEP, для перевірки можна ввести *ssdeep* у терміналі. У разі успіху з'явиться список команд або інформація про програму.

Варто зазначити що у користувачів можуть виникати помилки, незрозумілості або інші перешкоди, але головне не зупинятись на цьому технічному етапі. Якщо виникли проблеми, потрібно перевірити документацію в репозиторії SSDEEP. Також важливо завжди використовувати останню версію репозиторію для отримання актуальних функцій і виправлень помилок.

2. Генерація хешів для файлів. Після встановлення SSDEEP, можна створити розмиті хеші для файлів. Це ключовий етап, який дозволяє отримати унікальні рядки для подальшого порівняння.

Для генерація хешів поотрібно відкрити термінал (або командний рядок). Написати команду *ssdeep MediaCreationTool_22H2.exe*, важливо що *MediaCreationTool_22H2.exe* - файл, для якого потрібно згенерувати хеш. Результат може виглядати приблизно так: 196608:MmtHa+5hH1km/Sf7byFXKEBmih9S5rQ5FNFl001p4Ki:Y+5RB/SDbyFBH9 eQD/l00/4. Це і є розмитий хеш MediaCreationTool_22H2 (завантажувача windows 10). також є можливість генерації хешів для декількох файлів. Для цього використовується рекурсивне створення хешів у каталозі завдяки команді *ssdeep -r directory/.* Результат: у терміналі буде виведено хеші для кожного файлу. *Directory/* - шлях до каталогу з файлами [3].

На цьому етапі ми фактично отримуємо потрібну інформацію для створення основи.

3. Створення бази хешів Щоб здійснювати порівняння нових файлів з існуючими, необхідно створити базу хешів. Ця база містить хеші всіх файлів, які будуть використовуватись як еталон.

Для початку нам потрібно згенерувати хеші для всіх файлів у каталозі, найпростіше це зробити за допомогою команди *ssdeep -r directory/ > hash_database.txt.* Фактично вона складається з двох частин, перша це directory/ -

шлях до каталогу з файлами, яка нам знайома з попереднього етапу. А друга hash_database.txt - файл, у який зберігаються результати. Структура цього файлу складається з 3 частин. Перша - розмитий хеш, друга (після символу :) - зменшений підхеш для оптимізації та третя - ім'я файлу. На практиці це може виглядати так:

*384:4iVhXoM/3kuTazPOr0Zp5e...:oM/kuTazPOr0Z... file1.txt*
*192:Wr12kk8RGmN3rzEF4AG0jN...:2k8RGmN3rzEF4... file2.txt*

Важливо, що у користувачів є можливість оновлювати базу хешів. Якщо потрібно додати нові файли до бази то це можна робити двома способами, а саме згенерувати хеш для окремого файлу або для кількох файлів у новому каталозі. Це робиться завдяки команді *ssdeep new_file.txt >> hash_database.txt* або *ssdeep -r new_directory/ >> hash_database.txt* відповідно до потреби. Важливо що обов'язково потрібно ставити символ *>>*, тому що він додає хеш до існуючої бази без її перезапису.

Перевірка бази хешів відбувається досить просто, варто тільки відкрити файл *hash_database.txt*, щоб переконатись, що всі записи є. Також варто перевіряти правильність шляхів до файлів, особливо якщо база використовуватиметься на іншій машині.

Завдяки даному етапу ми закладаєм певну основу для подальшого порівняння. Для комфортнішого та безпечнішого комікування з базою нам краще використовувати добре організовані каталоги, щоб уникати плутанини, зберігати копію бази на випадок її пошкодження та якщо буде потрібно автоматизувати обробку, як варіант можна форматувати базу в JSON або CSV.

4. Порівняння файлів. Після того як база хешів створена, наступний етап - порівняння нових файлів із базою, щоб визначити схожість.

Якщо потрібно перевірити окремий файл на схожість із файлами в баз, то ми просто порівнюємо файл, який потрібно перевірити з файлом із хешами бази, а саме *target_file.txt* та *hash_database.txt.* Потрібна команда виглядає так: *ssdeep -k hash_database.txt target_file.txt.*

У результаті буде виведено список схожих хешів та відсоток схожості (число перед ім'ям файлу ) наприклад:

*ssdeep,1.1--blocksize:hash:hash,filename*
*35:Wr12kk8RGmN3rzEF4AG0jN...:2k8RGmN3rzEF4... file1.txt*
*50:Hd72mx9YGq4DszAB8IJ0kM...:mx9YGq4DszAB8... file2.txt*

Для порівняння кількох файлів з базою використовують рекурсивну перевірку, вона проводиться командою *ssdeep -r new_directory/ -k hash_database.txt* [3].

На цьому етапі ми завершили отримання даних для проведення аналіз, після цього залишається тільки підвести підсумки та як варіант покращення автоматизувати дану роботу. Також потрібно не забувати зберігати результати у файл для подальшого аналізу, це робиться командою ssdeep -k hash_database.txt target_file.txt > comparison_results.txt. І ще для систем моніторингу можна налаштувати скрипти, які аналізуватимуть результати порівняння та генеруватимуть сповіщення, якщо буде знайдено високий рівень схожості.

5. Аналіз результатів. Після порівняння файлів за допомогою SSDEEP необхідно інтерпретувати результати, щоб зробити висновки про схожість і

потенційні загрози. Розтлумачити відсотки схожості можна за такою шкалою:

- 100% - файли ідентичні;
- 80 - 99% - файли майже однакові, можливі невеликі зміни (наприклад, метадані чи зміни коду без зміни структури);
- 50 - 79% - часткова схожість, яка може свідчити про наявність змін у структурі або частині даних, також це можуть бути версії того самого файлу;
- 30 - 49% - низька схожість. Може вказувати на використання однієї бібліотеки або фрагментів коду, нечастий збіг через збіг частини даних;
- менше 30% - ймовірно, файли не пов'язані між собою.

Аналіз результатів SSDEEP дозволяє ідентифікувати схожі файли та оцінити рівень змін. Використовувати порогові значення можна для автоматизації класифікації та зберігання результатів для подальшого моніторингу. Наступний етап - інтеграція в робочий процес.

6. Автоматизація процесу. Щоб ефективно використовувати SSDEEP для аналізу файлів, можна автоматизувати весь процес - від генерації хешів до аналізу результатів. Це зменшить ручну роботу, скоротить час обробки та підвищить точність. Автоматизація процесу з використанням Bash або Python дозволяє ефективно створювати базу, аналізувати нові файли та отримувати сповіщення про підозрілі збіги. Розклад запуску скриптів забезпечить регулярність перевірок.

**Висновок**. SSDEEP дозволяє виявляти модифіковані або частково змінені файли, що особливо корисно для виявлення зловмисного програмного забезпечення або дублікованих даних. SSDEEP показує високий рівень ефективності в завданнях пошуку частково ідентичних файлів, але потребує правильної інтерпретації результатів для зменшення помилкових спрацьовувань. Автоматизація процесу дозволяє значно спростити інтеграцію цього алгоритму в робочий цикл, забезпечуючи оперативність та масштабованість. Таким чином, SSDEEP є надійним інструментом для забезпечення безпеки та аналізу даних, який можна легко адаптувати до різних сценаріїв використання. Також варто пам'ятати, що файли з високою схожістю можна перевіряти детальніше: наприклад, використати інші методи аналізу (статичний чи динамічний).

**Перелік використаних джерел.**

1. Jakobs C., Lambertz M., Hilgert J.-N. Ssdeeper: evaluating and improving ssdeep. Forensic science international: digital investigation. 2022. Vol. 42. P. 301402. [Електронний ресурс].- Режим доступу: https://doi.org/10.1016/j.fsidi.2022.301402 (date of access: 19.11.2024).

2. Virus Bulletin :: Optimizing ssDeep for use at scale. Virus Bulletin: Home. [Електронний ресурс].- Режим доступу: https://www.virusbulletin.com/virusbulletin/2015/11/optimizing-ssdeep-use-scale (date of access: 19.11.2024).

3. GitHub - ssdeep-project/ssdeep: Fuzzy hashing API and fuzzy hashing tool. GitHub. [Електронний ресурс].- Режим доступу: https://github.com/ssdeep-project/ssdeep (date of access: 19.11.2024).

4. Master ssdeep hash for improved file security identification. Blue Team Resources. [Електронний ресурс].- Режим доступу: https://blueteamresources.in/ssdeep-hash/ (date of access: 19.11.2024).