www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

SSDeep и все-все-все

Введение

Для систем класса анализа внешней поверхности атак (External Attack Surface Management, EASM), непрерывно перелопачивающих обширное пространство Интернета, способность эффективно выявлять повторяющийся или тесно связанный контент имеет важное значение. Для решения этой задачи мы в СайберОК интегрировали в СКИПА различные механизмы от тривиальных регэкспов до больших языковых моделей (LLM, Large Language Model). В этой статье мы рассмотрим один из подобных трюков – использование нечеткого хеширования (fuzzy-hashing aka similarity hashing).
Кстати, LLM используем для генерации регэкспов, чтобы Уроборос. Но это уже другая история.
Сама балалайка доступна на GitHub, наслаждайтесь: https://github.com/cyberok-org/similarity.
Видео об этом:
https://www.youtube.com/watch?v=4wSxp7t6huA&t=1634

Задачи

Традиционные методы, такие, как точное сопоставление, имеют проблемы с вариациями в тексте или структурными различиями. Рассмотрим несколько ситуаций, когда четкое сопоставление работает плохо или просто не работает:

- выявление схожих ответов серверов (баннеров) для их кластеризации и определения ПО (application fingerprint);
- выявление типовых интерфейсов, например «заглушек» от провайдеров и хостеров, интерфейсов управления;
- выявление фишинговых страниц, вредоносных js-вставок и других индикаторов компрометации;
- определение сходства или различия веб-страниц, например приложений, опубликованных на разных доменных именах одного узла для задач оптимизации анализа и дедупликации результатов.

Для решения этой задачи мы используем нечеткое хеширование – мощный инструмент, предлагающий детальный подход к сравнению текстовых и бинарных данных. В этой статье мы углубимся в сферу нечеткого хеширования, уделив особое внимание их использованию в задачах EASM.

Нечеткое хеширование

Нечеткое хеширование — это метод, который генерирует хеши, представляющие содержимое файлов или данных таким образом, чтобы можно было сравнивать их даже если содержимое слегка изменено. Многим коллегам в кибербезе, особенно работающим с вредоносным ПО, известен ssdeep — широко используемый алгоритм нечеткого хеширования, который превосходно обнаруживает сходства между текстовыми блоками или файлами, создавая контекстно-зависимые хеши, которые фиксируют закономерности в данных.
Если интересны подробности, добро пожаловать в табличку в следующем подразделе.

Принципы нечеткого хеширования

www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

Нечеткое хеширование учитывает контекст и структуру данных. Это позволяет алгоритму хеширования идентифицировать схожие шаблоны или блоки контента в этих данных, даже если присутствуют небольшие вариации или модификации.

Часто алгоритмы разбивают содержимое файлов на мелкие куски, вычисляя по ним значения хешей для дальнейшего использования, что позволяет обнаруживать сходство на более детальном уровне. Это делается просто, или используются специальные функции (например, rolling hash). Некоторые алгоритмы поверх этого используют другие структуры (например, lsh). Вариантов масса.

Все алгоритмы выдают некоторое значение — меру похожести (threshold). Оно имеет разные границы и смысл, но суть одна — чем более похожи сравниваемые файлы, тем оно больше (или наоборот, меньше). Регулируя threshold, можно установить нужную чувствительность для каждого из алгоритмов.

Существует множество различных реализаций нечеткого хеширования. В своей работе мы используем ssdeep, tlsh, mrsh, nilsimsa, simhash. Все они выдают весьма приличный, но разный результат (из-за чего их сложно сравнивать, да и метрика достаточно абстрактная). Важно выбирать правильный хеш в зависимости от задач (посмотреть на результаты каждого и найти лучший или лучших для себя) и (очень важно!!!) требований производительности (чтобы не ждать 3 недели, когда оно, наконец, досчитает). Время работы, как правило, растёт квадратично в зависимости от количества данных. Так происходит потому, что алгоритмы, в основном, каждый документ сравнивают с каждым. Асимптотика верна в общем случае без использования модификаций, но, например tlsh, в отличие от остальных перечисленных, сам по себе работает за nLog(n) из-за специфики своих хешей, а некоторые другие можно ускорить иначе, но об этом не здесь.

Также все эти инструменты не любят, когда данных мало — хеш получается нерепрезентативный. Алгоритмы ведут себя по-разному — некоторые игнорируют такие файлы, другие оставляют. Для некоторых из наших задач (например, анализа заголовков прикладных протоколов или ответов UDP-серверов) очень важен анализ схожести небольших файлов, поэтому алгоритмы mrsh и tlsh не всегда применимы.

По количеству данных, генерируемых хеш-функцей, если брать ssdeep за ориентир:

- mrsh — получает меньше за счёт того, что игнорирует небольшие файлы;
- tlsh — сравним с ssdeep, но сильно зависит от ситуации, может выдать в несколько раз меньше данных. В том числе из-за игнорирования некоторых файлов;
- simhash — больше примерно в 1.3 раза;
- nilsimsa — больше примерно в 1.5-2 раза;

Все эти значения весьма условны, так как зависят от данных и от трешхолдов (были использованы ssdeep - 70, tlsh - 55, mrsh - 30, nilsimsa - 110, simhash - 4)

ssdeep Очень популярный инструмент, написан на C. Работает достаточно долго по сравнению с другими (не критично, если данных не очень много), эту проблему можно решить при использовании определённых оптимизаций, завязанных на уменьшение числа сравниваемых файлов. По сравнению с другими score более значимый (медленнее уменьшается в зависимости от похожести — score=30 и score=80 могут означать разное и нет резкой границы, после которой нет дубликатов, вместо этого есть большой промежуток, в котором уменьшается похожесть).

www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

Трешхолд — % схожести (0 - 100), можно указывать 70-80, но находятся дубликаты и при меньших значениях. 20 — около минимально значимой границы, 70 — более-менее оптимальный, чтобы не потерять нужное.

tlsh

Очень популярный инструмент, написан на C++.
Работает быстро (в принципе и асимптотически).
Из проблем — игнорирует небольшие файлы. Есть порты на другие языки.
Трешхолд — расстояние (0 - 9999, меньше — лучше), относительно разумные значения при 55, но может зависеть от данных. Совсем мягкая граница — 300, больше неё смысла ставить нет, 55 — разумная граница, чтобы взять необходимое.

mrsh

Написано на C, работает медленнее ssdeep (если смотреть на том же количестве данных, без уменьшения).
Из проблем - игнорирует небольшие файлы. В целом тоже работает.
Трешхолд — % схожести (0 - 100), хорошие результаты дает 30.

nilsimsa

Есть реализация на Python.
Часть со сравнением может работать долго, но алгоритм сравнения быстрый и не сложный, так что при переписывании на другой язык проблем со скоростью не возникает (хотя n^2 остаётся).
Трешхолд — от -127 до 128. 128 - одинаковые. Можно ставить 110. В целом и с 0 находятся дубликаты, но данных, если ставить меньше 110, может получиться много.

simhash

С реализацией ситуация аналогична nilsimsa.
Трешхолд — расстояние (меньше - лучше), можно ставить 4, мягкая граница — 20.

Как это работает

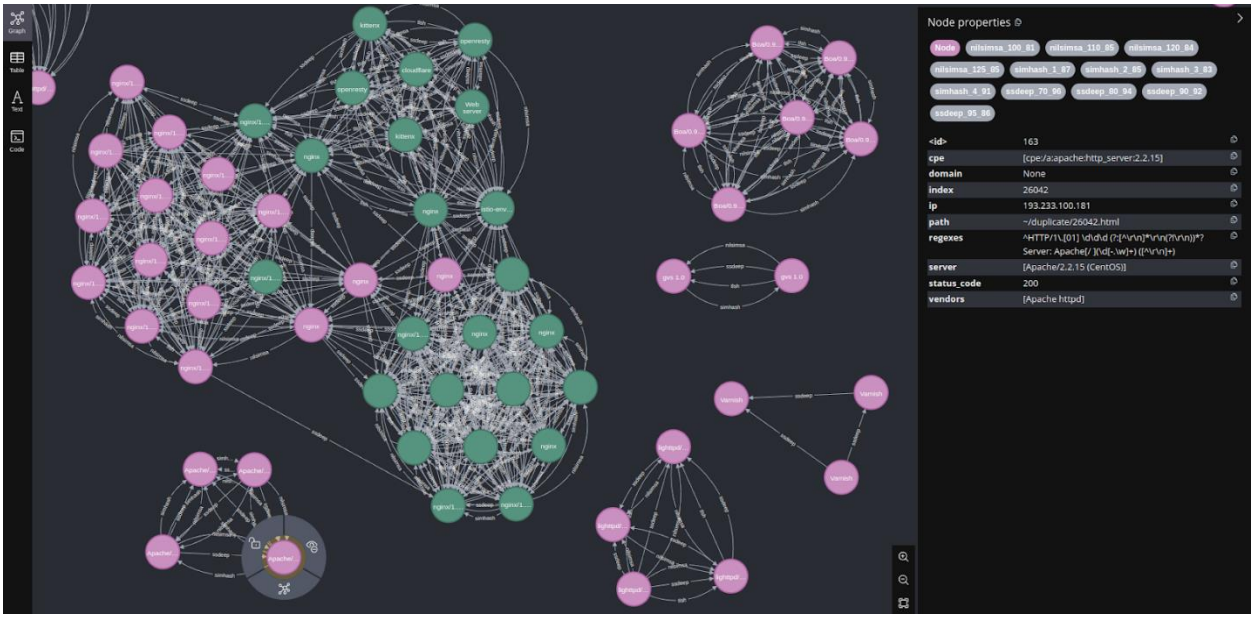
Для представления графовых данных мы использовали neo4j.
В нашем случае алгоритм работы таков:

1. Берем ответы серверов (баннеры, тело), информацию об источнике и прочее.
2. Получаем из этих данных нужную информацию, структурируем и записываем в ноды графа (одна нода — один ответ сервера).
3. Считаем хеши по телу, ищем похожие файлы. Результат — список пар документов и score алгоритма по ним.
4. Считаем кластеры по рёбрам для каждого алгоритма с разными трешхолдами (две ноды в одном кластере есть неотфильтрованное ребро, выданное алгоритмом).
5. Модифицируем ноды, полученные в п. 2, добавляя в каждую информацию о том, в каких кластерах она есть (в neo4j можно использовать label для этого).
6. Создаём ноды, проходимся по списку рёбер и проводим их между нодами, помечая score и название алгоритма.
7. Наслаждаемся исследованием того, что получилось (используя всевозможные запросы).
8. Profit!

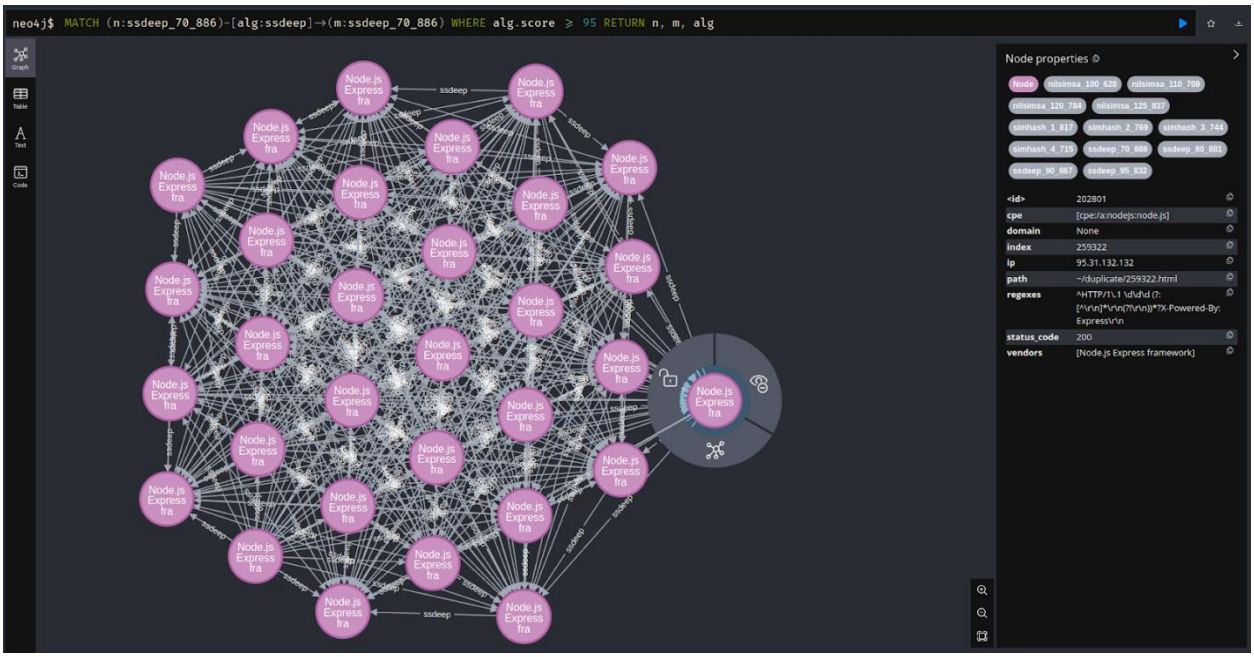
(4 и 5 пункт в принципе опциональны, но с ними удобнее + это позволяет искать интересные случаи)

Хватит теории!

Посмотрим, как это работает на практике.



На скрине случайно взятые кластеры. Тут видно, что есть рёбра разных алгоритмов между нодами, а в нодах есть информация о том, к чему они относятся и с какими другими связаны.



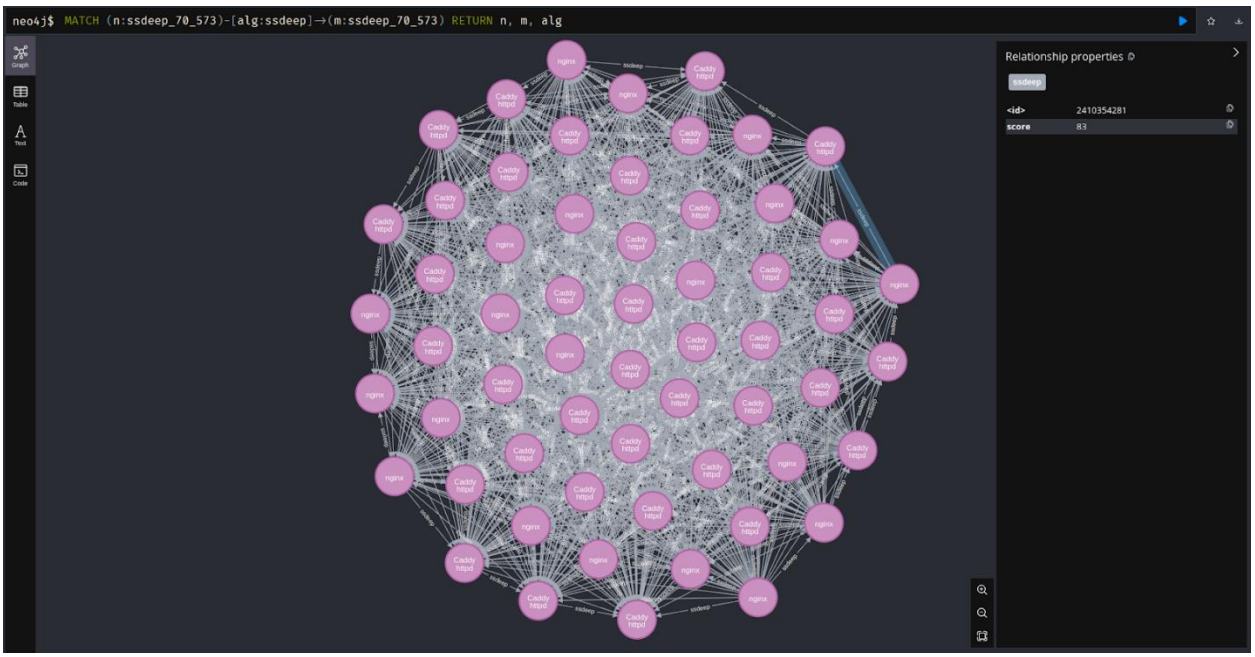
А тут мы взяли ноды конкретного кластера, изобразив только связи, полученные ssdeep, причём только те, у которых score ≥ 95

На этом можно не ограничиваться и строить гипотезы о том, как всё устроено, проверять их, находить примеры, которые им не соответствуют. И, таким образом, лучше понимать что-то про данные, выбрать те, на которые стоит смотреть и, поскольку есть визуализация, исследовать детальнее.

Для такого поиска удобно писать скрипты, которые проходятся по кластерам с разными трешхолдами по разным алгоритмам (для этого у нас есть специальная структура, однако опустим это) и, связывая их с данными по нодам, считают метрики, и фильтруют по ним.

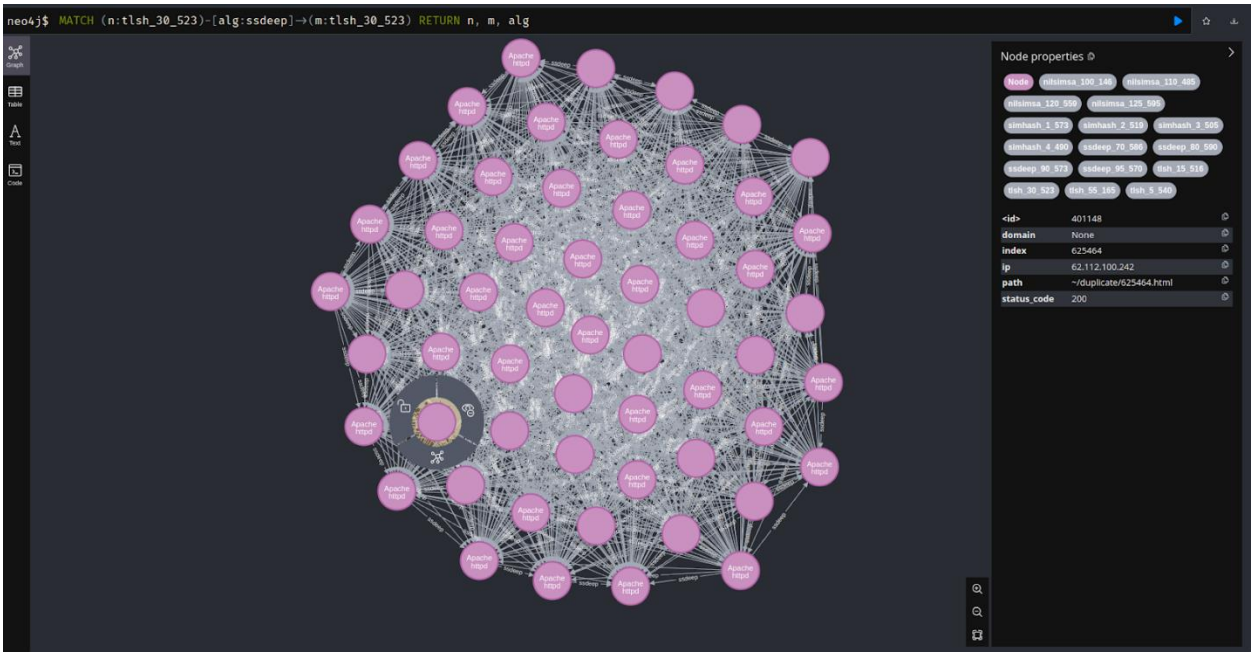
Пример 1

Например, мы разметили данные и хотим проверить, что ничего не забыли (наша разметка как-то коррелирует с похожестью, например, связана с product или status code). Но что это значит для кластера? Можем найти те узлы, в которых есть несколько разных многочисленных значений для выбранной метрики или те, что вовсе не размечены.

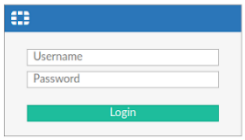


Так, например, при поиске кластера, у которого есть 2 разных часто встречающихся вендора, нашлось такое. Тут много Caddy httpd и много nginx, и они связаны. Если посмотреть на данные, то видно, что страницы и правда очень похожи, хоть вендор и разный.

Продолжим с тем же, поищем что-нибудь аналогичное с неразмеченным.



Нашёлся такой кластер, тут много нод с Apache httpd, про остальное не известно. Если смотреть на данные, то везде такая форма, разве что цвет меняется.

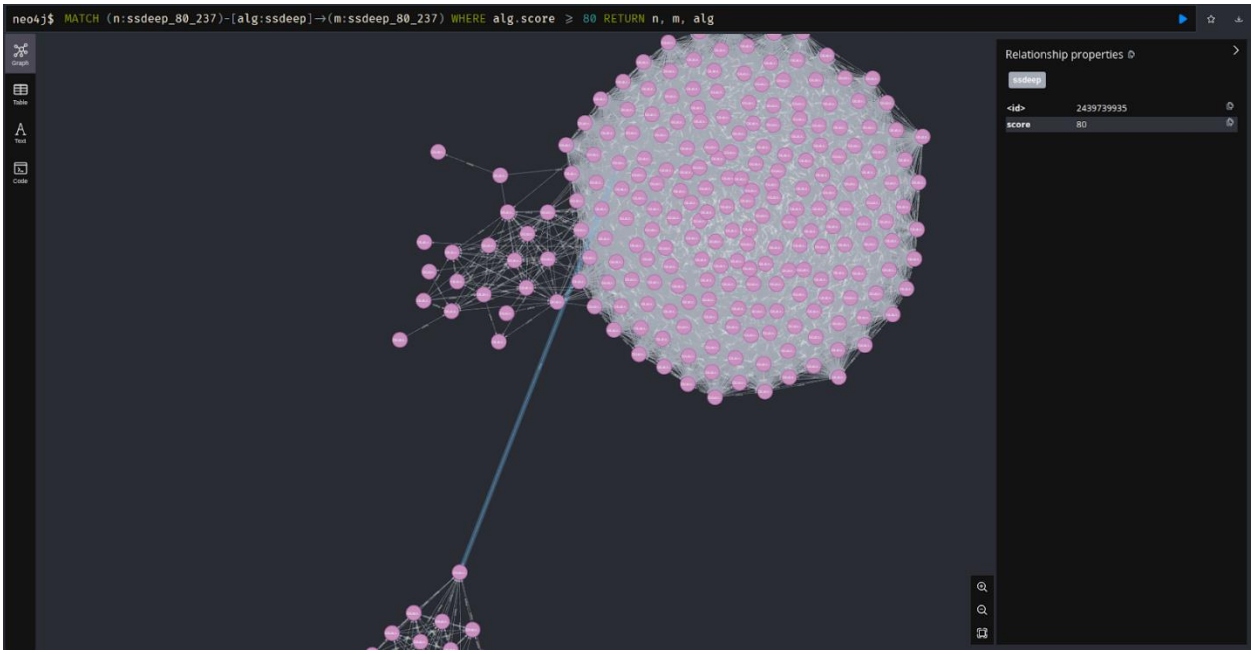


При ближайшем рассмотрении очевидно, что это Fortinet, который нельзя не детектировать, поскольку эта “железка” всегда просит, чтобы ее обновили, чтобы кто-то случайно не сломал Рунет.

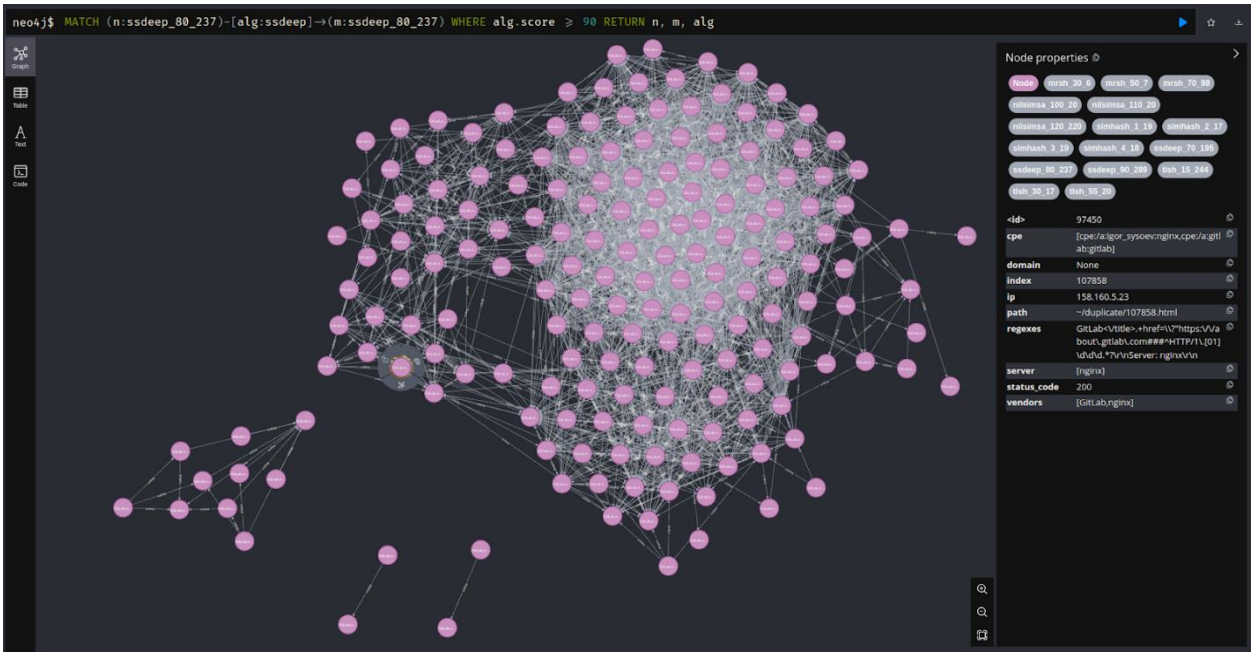
Дальше можно детальнее разбираться со случаями, понимать почему оно не определилось в качестве продукта или продолжать искать более интересные случаи.

Пример 2

Другой пример. У нас есть кластер и часто бывает интересно посмотреть, как он распадается (в данном контексте это значит, что часть рёбер, связывающих его, перестанет существовать, если увеличить трешхолд — он разобьется на более мелкие). Тут нас интересуют достаточно большие кластеры, которые разделяются на несколько других, тоже немаленьких.



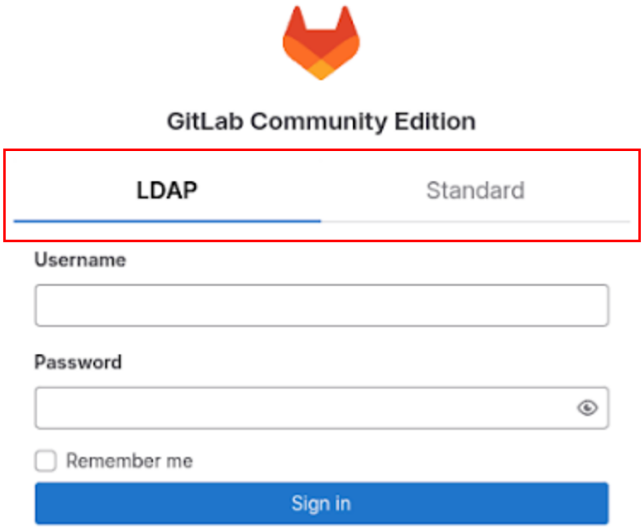
Посмотрим на кластер, найденный таким методом. Тут есть ребро со score 80, соединяющее две части и ещё группа нод, связанная не так сильно с остальными. А теперь увеличим threshold.



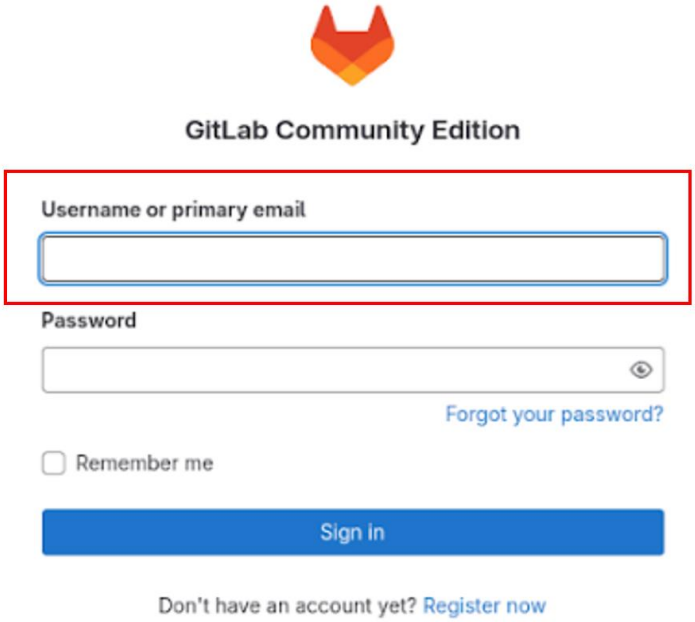
Теперь у нас есть несколько отдельных более мелких кластеров (тут ещё не отображаются единичные). Посмотрим на то, что есть в каждом.

www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

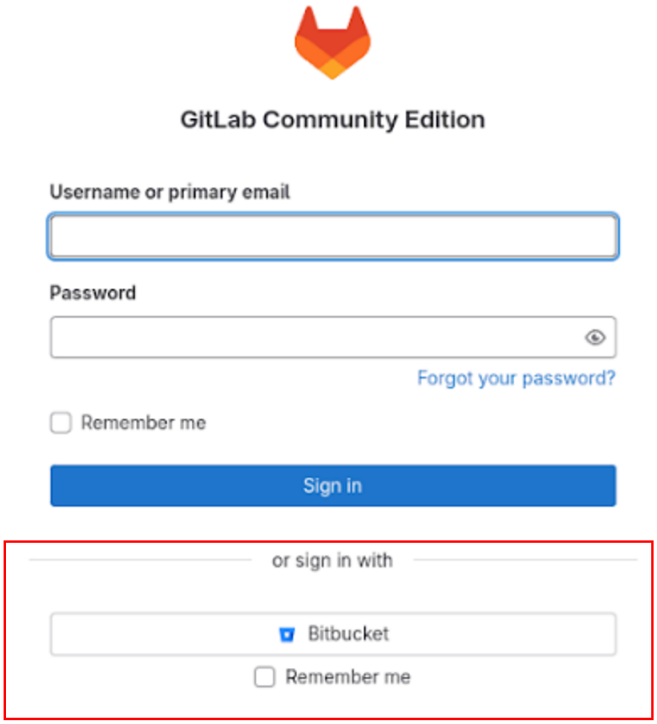
Пример 1



Пример 2



Пример 3



Пример 4

Это 4 примера, каждый отображает, как выглядят страницы в полученных кластерах. Видно, что везде gitlab, но с различиями, и именно по ним мы разделяем группы, увеличивая score.

Также может быть интересно посмотреть на похожесть доменных имён дубликатов или исследовать кластеры, отобранные по менее тривиальной метрике, полученной из всего, что есть в ноде. В общем, можно ещё много чего интересного достать, главное придумать.

Выводы

Нечеткое хеширование предлагает сложный метод сравнения веб-страниц и выявления похожего контента в Интернете в задачах кибербезопасности, облегчая повседневную работу аналитиков и, в конечном итоге, делая наш кибермир более безопасным. Использование нечеткого хеширования для сравнения данных EASM является ценным активом в распутывании сложной паутины онлайн-контента и обеспечении его безопасности. Спектр применения очень широк, от мониторинга уязвимостей и индикаторов компрометации до рекона при пентестах или расследовании инцидентов. Кстати, ssdeep можно очень сильно оптимизировать, но это уже совсем другая история.

Авторы:

Андрей Погребной, CyberOK
Антон Шмаков, CyberOK
Сергей Гордейчик, CyberOK