

Используемые технологии:

Nodejs, Redis

Регистрация юзера

1. Юзер регистрируется в расширении, с расширения приходит POST реквест на `siteanalyzer.com/api/1.0/register`, в котором содержится информация о браузере юзера
2. Для юзера генерируется уникальный id и он сохраняется в БД — Redis, Mongo или любую другую. Так же для юзера будет создаваться документ `visitedSites`, в котором будут содержаться:
 - URL посещенных сайтов
 - булево значение — результат последней проверки, безопасен ли сайт
 - последнее время посещения
3. В последствии юзер сможет сделать GET реквест на `siteanalyzer.com/api/1.0/analyze/results?key=userKey` что бы получить статистику по посещенным сайтам

Архитектура

1. Приходит POST реквест с расширения в браузере на `siteanalyzer.com/api/1.0/analyze`
2. В теле реквеста содержится:
 - уникальный `userKey`, который генерируется при регистрации пользователя
 - source code странички
3. Из source code странички вытаскивается\скачивается все, что содержится в тегах script — фрагменты
4. Из каждого фрагмента создается хэш-строка `stringHash`, из хэш-строки создается индекс списка `listIndex` по такому принципу: `hash(str) % numBuckets`
Число `numBuckets` может быть выставлен в конфигурационном файле
Поскольку речь идет про 1млн запросов, для начала `numBuckets = 200000`
5. Следует проверка в Redis, существует ли записи по ключу `listIndex` для каждого фрагмента
6. Если записей по ключу `listIndex` не существует:
 - a) производится анализ фрагмента, создается переменная `approved`
 - если количество чисел четное — `approved = true`
 - если количество символов нечетное — `approved = false`
 - b) создается значение для вставки в Redis:
`HMSET listIndex script_1.soruceCode "soruce code ..." script_1.approved "true\false"`
что соответствует объекту:

```
script_1 = {
  soruceCode: "soruce code ...",
  approved: true // или false
}
```
7. Если запись по ключу `listIndex` существует — из нее достаются все содержимое, например:
> `HGETALL listIndex`
 - 1) `"script_1.soruceCode"`
 - 2) `"soruce code ..."`

- 3) "script_1.approved"
- 4) "true"
- 5) "script_2.soruceCode"
- 6) "soruce code ..."
- 7) "script_2.approved"
- 8) "false"

где

soruceCode — строка со скриптом

approved — булево значение, результат предыдущего анализа

Далее текущий sourceCode сравнивается со всеми вытащенными из ячейки sourceCode

Если строки идентичны — достается соответствующее фрагменту значение approved

Если строки отличаются — проводится анализ soruceCode как указано в пункте 6, проанализированному фрагменту и его булевому значению результата анализа присваивается следующий порядковый номер, затем они добавляются в список к существующим значениям и под ключом listIndex возвращается в Redis

8. Таким образом анализируются все фрагменты из поступившего на анализ source code после окончания проверки всего документа, если хотя бы один approved = false, URL признается malware.

9. Производится запись в БД с юзерами, где указывается url, время посещения и результат полного анализа

Масштабируемость

1. $listIndex = hash(str) \% 5000000$

число 5кк взято для с запасом на будущее для 1кк запросов, в последствии это число можно будет увеличить

2. Можно масштабировать количество серверов с Redis разбивая по секторам по listIndex по мере нарастания нагрузки

Версионность API

Приставка 1.0 в версии апи позволяет развивать API, при необходимости оставляя возможность сделать реквест на старую версию