Ускорение анализатора сетевого трафика Snort при помощи библиотеки Hyperscan

Осенью прошлого года компания Intel открыла исходные тексты библиотеки поиска совпадений для регулярных выражений Hyperscan, обеспечивающей ускорение DPI-приложений за счёт использования SIMD compiler intrinsics.

Использование новых наборов инструкций позволяет добиться значительного прироста производительности в самом узком месте анализаторов трафика: в поиске совпадений по образцу.

Snort - самое известное DPI-приложение с открытым кодом - получил поддержку Hyperscan только в начале этого лета. Представленный патч предназначен для версии Snort 2.9.8.2 но без проблем накладывается на 2.9.8.3 и может быть портирован и на предыдущие релизы.

Сборка

Для сборки Hyperscan сборочное окружение должно содержать Boost и компилятор ragel. В библиотеке используется система сборки cmake. Сложности, если можно их назвать сложностями, вызывает только Boost: данной версии всё ещё нет в большинстве дистрибутивов на момент июня 2016, поэтому скорее всего придётся собираться локально. Собрать буст несложно и весь процесс не очень длителен. Что касается ragel и cmake, то они уже имеются в пакетных репозиториях Gentoo, Ubuntu, CentOS в достаточно свежих версиях.

Boost
Ragel
Hyperscan
Snort

Boost

Нужен Boost выше 1.57. В наиболее ожидаемом случае, если версия буста в используемом дистрибутиве ниже, то надо будет установить более свежую Для этого надо скачать актуальную версию и установить её локально.

```
tar xf ~/tmp/boost_1_61_0.tar.bz2
cd boost_1_61_0
./bootstrap.sh -prefix=$HOME/opt/boost/
./b2 --ignore-site-config  install
```

--ignore-site-config требуется для устранения потенциальных конфликтов с системной версией Boost: игнорирование текущих настроек при сборке локальной версии буста в домашней директории пользователя.

Ragel

Далее нужен ragel, этот компилятор доступен в стандартных репозиториях всех распространённых дистрибутивов.
Поэтому инсталляция проста:

```
emerge ragel
```

или же apt-get install ragel, yum install ragel итп итд

HyperScan

Сборка библиотеки стандартна для проектов использующих cmake:

```
tar xf ~/tmp/hyperscan-4.2.0.tar.gz
cd hyperscan-4.2.0/
mkdir build
cd build
cmake .. -DBOOST_ROOT=$HOME/opt/boost/ -DCMAKE_INSTALL_PREFIX=$HOME/opt/hp
make install
```

Предполагается, что будет использован локальный буст, установленный в $HOME/opt/boost.
Все переменные сборки можно проверить и перезадать при помощи "ccmake ." или же
через аргументы cmake.
В случае использования системного буста указывать -DBOOST_ROOT не следует.

Snort

Подходит кодовая база Snort любой из 2.9.8.* версий: 2.9.8.0, 2.9.8.2.
В статье используется последний стабильный релиз:
2.9.8.3.
Добавление поддержки HyperScan производится патчем версии v1:
https://01.org/sites/default/files/downloads/hyperscan/snort-2982-hyperscan-v1.tar_0.gz

распаковываем исходники Snort:

```
tar xf ~/tmp/snort-2.9.8.3.tar.gz
cd snort-2.9.8.3/
```

накладываем патч:

```
tar xf ~/tmp/snort-2982-hyperscan-v1.tar_0.gz
gunzip -c snort-2982-hyperscan-v1/snort-2982-hyperscan-v1.patch.gz| patch -p2
```

На всех трёх доступных мне машинах с CentOS, Gentoo, Ubuntu этого было
недостаточно для сборки: при линковке snort возникали ошибки вида

```
/opt/hp/lib/libhs.a(mpvcompile.cpp.o): In function 'operator--':
/usr/include/c++/4.8/bits/stl_tree.h:204: undefined reference to 'std::_Rb_tree_decrem
/opt/hp/lib/libhs.a(mpvcompile.cpp.o): In function '__gnu_cxx::new_allocator<ue2::raw_
/usr/include/c++/4.8/ext/new_allocator.h:110: undefined reference to `operator delete(
```

требуется добавление -lstdc++ к списку библиотек в configure.in.
Поэтому открываем любимым редактором файл configure.in, ищем в нём строчку,
где задаётся "-lhs" и заменяем на "-lhs -lstdc++":

```
-    LIBS="${LIBS} -lhs"
+    LIBS="${LIBS} -lhs -lstdc++"
```

Собираем Snort:

```
autoreconf -fi
./configure --enable-intel-hyperscan --with-intel-hyperscan-includes=$HOME/opt/hp/incl
make install
```

Настройка

Собранный таким образом снорт уже будет немного быстрее оригинального, так как
для поиска контента будут использоваться SIMD инструкции. Но прирост будет малозаметен.
Для реального ускорения детектирования необходимо задать движок hyperscan в
опции config detection конфигурационного файла snort.conf:

```
config detection: search-method hyperscan
```

Если использовать актуальный большой набор правил, например открытый набор от [[https://r
Emerging Threats]],
то начальная загрузка снорта значительно замедлится.

Чтобы избежать этого без особых потерь в ускорении можно использовать опцию **split-any-any**
таким образом результирующая строка config detection будет выглядеть так:

```
config detection: search-method hyperscan split-any-any
```

При этом будет наблюдаться выигрыш как в производительности сравнительно со снортом без h
так и по использованию памяти, точные данные зависят от набора правил.
На наборах ET это около 10 процентов (~520 мб и 470 мб в top соответственно).

Без опции **split-any-any** снорт будет анализировать трафик с максимальной скоростью,
но будет гораздо дольше стартовать и будет гораздо более заметен в потреблении памяти сис
Таким образом, для сравнения получаются 4 системы с одним набором правил от ET:

чистый Snort 2.9.8.3 без модификаций
Snort, собранный с Hyperscan, без изменений конфигураци, то есть
использующий библиотеку только для простого поиска контента
Snort с Hyperscan и включённым движком детектирования hyperscan
Snort с включенным движком hyperscan и опцией split-any-any

Результаты

Скорость проще всего оценивается по времени обработки pcap-файлов.
Результирующее ускорение зависит от новизны CPU. Так, на одноядерном Celeron
M560 от старого ноутбука, выигрыш будет в районе считанных процентов:
с гиперсканом дамп будет обрабатываться за 300 секунд, без гиперскана за 310.

Получше результаты на более современном Intel(R) Core(TM) i7-2600:

Время обработки 100+ mb pcap файла без hyperscan:
Run time for packet processing was **39.578298** seconds
Время обработки того же 100+ mb pcap файла с hyperscan:
Run time for packet processing was **35.190397** seconds
Время обработки 100+ mb pcap файла с движком hyperscan + split-any-any:
Run time for packet processing was **24.616165** seconds
Время обработки 100+ mb pcap файла с движком hyperscan:
Run time for packet processing was **22.323232** seconds

Было бы интересно сравнить на Xeon. Видно, что при использовании данной
библиотеки и хорошего железа можно ускорить обработку пакетов, пролетающих
через Snort IDS, практически двукратно.