

Программа для ЭВМ
«UBIC CORE»
Архитектура системы

Оглавление

1. ОБЩЕЕ ОПИСАНИЕ АРХИТЕКТУРЫ.....	3
2. КОМПОНЕНТЫ СИСТЕМЫ	3
2.1 pfetcher2 – ядро DPI	3
2.2 dns-tracer – перенаправление DNS	3
2.3 radius-tracer – учет абонентов по RADIUS	4
2.4 identification – загрузчик карты абонентов.....	4
2.5 catalog2 – загрузчик URL-каталога	4
2.6 pstats – статистика.....	4
2.7 0cat – CLI-клиент управляющего RPC.....	4
2.8 Библиотеки	4
3. ПОТОК ДАННЫХ.....	5
4. ТЕХНОЛОГИЧЕСКИЙ СТЕК.....	5
5. СИСТЕМНЫЕ ТРЕБОВАНИЯ.....	5

1. ОБЩЕЕ ОПИСАНИЕ АРХИТЕКТУРЫ

Программа для ЭВМ «UBIC CORE» построена как набор отдельных взаимодействующих процессов (компонентов), обменивающихся данными через разделяемую память и очереди сообщений. Такой подход обеспечивает высокую производительность обработки пакетов, изоляцию сбоев отдельных компонентов и возможность независимого масштабирования подсистем.

Все компоненты Программы написаны на языке C++11 и собираются через систему CMake. Основу решения составляет ядро анализа сетевых пакетов pfetcher2, вокруг которого сгруппированы вспомогательные компоненты идентификации абонентов, перенаправления DNS, учета по RADIUS, загрузки каталогов и служебного управления.

Взаимодействие компонентов организовано двумя основными механизмами. Разделяемая память на базе Boost.Interprocess используется для совместного доступа нескольких процессов к таблицам идентификации абонентов, счетчикам статистики и URL-каталогу. Очереди сообщений ZeroMQ используются для выгрузки результатов во внешний конвейер (PUSH/PULL) и для служебного управляющего RPC-интерфейса (REQ/REP).

Исходные данные (карты идентификации абонентов и URL-каталог) загружаются из СУБД MySQL (база safeinet) в разделяемую память специализированными загрузчиками. Обновление таблиц в разделяемой памяти выполняется атомарной подменой указателя на таблицу, что позволяет не перенаправлять процессы-читатели на время загрузки.

2. КОМПОНЕНТЫ СИСТЕМЫ

2.1 pfetcher2 – ядро DPI

Компонент реализует конвейер «производитель–потребитель» на нескольких потоках: снифер читает пакеты через libpcap и раскладывает их по набору очередей, рабочие потоки разбирают TCP-сессии, HTTP-запросы и TLS-рукопожатия, выполняют классификацию по каталогам и принимают решение о перенаправлении пакетов. Основная логика сосредоточена в классе TCPBlocker; чтение выполняет Sniffer, распределение по очередям – QueueSet, реассемблинг – TCPBuff, сериализацию и выгрузку результата – Dumper.

Решение о перенаправлении (маршрутизации) пакетов принимается по иерархии поисков в каталогах: локальные переопределения, персональные категории, категории по расширению файла, основной каталог, а также специальные механизмы управления интернет-потреблением и разбора поисковых запросов. Итоговое решение (pass, block, clarify, familized) определяется сравнением черных и белых масок пользователя и статьи каталога.

2.2 dns-tracer – перенаправление DNS

Читает DNS-запросы (UDP, порт 53) и для абонентов с установленными параметрами интернет-потребления перенаправляет ответы на адреса из локального CSV-каталога (например, безопасные IP-адреса поисковых систем). Компонент читает таблицу

идентификации __SHID__, фильтрует запросы только в соответствии с политиками интернет-потребления абонентов, разбирает секцию вопроса DNS, ищет домен в потокобезопасном словаре и собирает ответный пакет с пересчетом контрольных сумм IPv4 и UDP.

2.3 radius-tracer – учет абонентов по RADIUS

Слушает RADIUS-аккаунтинг (UDP, порт 1813), извлекает логин абонента и выданный ему IP-адрес, нормализует логин, вычисляет MD5-хеш логина и записывает соответствие «IP ↔ пользователь» в разделяемую память __SHID__. Таким образом таблица идентификации наполняется динамически в дополнение к статической загрузке из MySQL. Поддерживается разбор vendor-специфичных атрибутов (Ericsson/Broadband Forum, Cisco).

2.4 identification – загрузчик карты абонентов

Утилита создает, наполняет и обслуживает сегмент разделяемой памяти __SHID__ с картами идентификации, загружая их из MySQL (база safeinet). В сегменте хранятся хэш-таблицы статической идентификации (IP+VLAN → пользователь), динамической идентификации, соответствия «логин → пользователь» и атрибутов пользователей (маски, группы, параметры интернет-потребления и HTTPS-фильтрации).

2.5 catalog2 – загрузчик URL-каталога

Утилита строит в разделяемой памяти дерево URL-каталога из таблицы MySQL для быстрой категоризации URL обходом дерева. Домен разбивается и укладывается в обратном порядке, затем путь – в прямом; резолвинг выполняется от корня по токенам с запоминанием последнего значимого узла. Данный механизм независим от файловых каталогов linnaeus/rosenthal, с которыми работает ядро pfetcher2.

2.6 pstats – статистика

Утилита управления сегментом разделяемой памяти __PSTATS__ и просмотра счетчиков. В сегмент пишет ядро pfetcher2, читает утилита pstats. Хранятся счетчики прочитанных и обработанных пакетов, HTTP-запросов, перенаправлений, выгруженных фактов, добавленных и удаленных сессий, успешных и неуспешных реассемблингов и иных эксплуатационных показателей; поддерживается расчет скорости в Мбит/с.

2.7 0cat – CLI-клиент управляющего RPC

Отправляет JSON-команды в служебный управляющий RPC-интерфейс («cowboy») ядра pfetcher2 или компонента dns-tracer через ZeroMQ (REQ/REP) и печатает ответ. Позволяет перезагружать каталоги, запрашивать их версии, управлять интернет-потреблением и перезагружать CSV-каталог DNS без остановки компонентов.

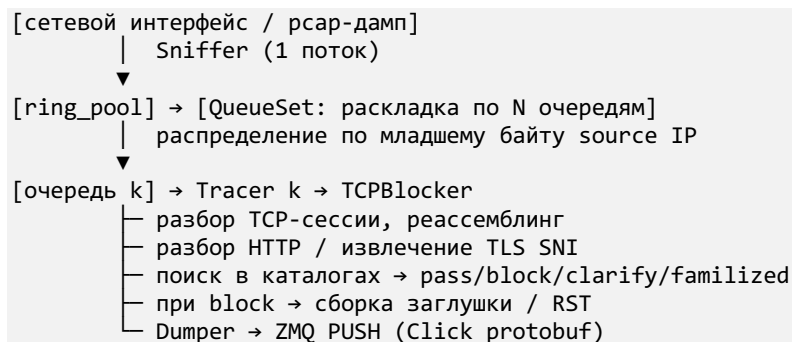
2.8 Библиотеки

- su13lib – базовая библиотека: сетевые утилиты (заголовки, контрольные суммы, packet_emitter), MD5-хэши логинов, потоки, lock-free очереди и пул блоков памяти;

- linnaeus – движок URL-каталогов и управления интернет-потреблением, используется ядром pfetcher2 для файловых каталогов формата .dic;
- rosenthal – движок DACT: разбор поисковых запросов и паттернов для фильтрации по ключевым словам.

3. ПОТОК ДАННЫХ

Обработка пакетов в ядре pfetcher2 организована по следующему конвейеру:



Сопутствующие потоки данных:

- identification/updater и catalog2/updater загружают карты абонентов и URL-каталог из MySQL в разделяемую память;
- radius-tracer динамически наполняет __SHID__ по данным RADIUS-аккаунтинга;
- pfetcher2 и dns-tracer читают __SHID__ для идентификации абонентов; dns-tracer перенаправляет DNS-ответы в соответствии с политиками интернет-потребления абонентов;
- pfetcher2 пишет счетчики в __PSTATS__ и выгружает результаты (Click) во внешний конвейер по ZeroMQ;
- 0cat управляет каталогами и правилами через служебный RPC-интерфейс.

4. ТЕХНОЛОГИЧЕСКИЙ СТЕК

Категория	Технологии
Язык и сборка	C++11, CMake
Чтение пакетов	libpcap
Разделяемая память	Boost.Interprocess
Обмен сообщениями	ZeroMQ (PUSH/PULL, REQ/REP)
Сериализация	Protocol Buffers (click.proto)
Хранилище исходных данных	MySQL (база safeinet)
Служебные библиотеки	su13lib, linnaeus, rosenthal, GFlags
Хэширование	MD5 (нормализованные логины абонентов)

5. СИСТЕМНЫЕ ТРЕБОВАНИЯ

Программные требования: операционная система семейства Linux (рекомендует Debian/Ubuntu/RHEL) с поддержкой libpcap; установленные библиотеки GFlags,

ZeroMQ, libpcap, Threads, Protocol Buffers, Boost.Interprocess и MySQL Connector/C++; СУБД MySQL для хранения исходных карт идентификации и URL-каталога.

Аппаратные требования: минимально для тестирования – 8 ядер ЦПУ, 16 ГБ ОЗУ, 50 ГБ SSD; для производственной эксплуатации – от 16 ядер ЦПУ и от 32 ГБ ОЗУ (с учетом объема сегмента идентификации __SHID__ порядка 8 ГБ), сетевые интерфейсы с пропускной способностью, соответствующей объему обрабатываемых пакетов.