Труды XXII научной конференции по радиофизике

СЕКЦИЯ
«ИНФОРМАЦИОННЫЕ СИСТЕМЫ. СРЕДСТВА, ТЕХНОЛОГИИ, БЕЗОПАСНОСТЬ»

Председатель – Л.Ю. Ротков, секретарь – А.А. Рябов.
Нижегородский государственный университет им. Н.И. Лобачевского.

ЖУРНАЛИРОВАНИЕ DNS-ЗАПРОСОВ ПУТЁМ ПЕРЕХВАТА API-ФУНКЦИЙ ОС WINDOWS

А.В. Бугров, Д.В. Демьяненко

ННГУ им. Н.И. Лобачевского

Windows API (англ. *application programming interfaces*) – общее наименование целого набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows корпорации «Майкрософт». Является самым прямым способом взаимодействия приложений с Windows. Для создания программ, использующих Windows API, «Майкрософт» выпускает комплект разработчика программного обеспечения, который называется Platform SDK, и содержит документацию, набор библиотек, утилит и других инструментальных средств для разработки. Windows API спроектирован для использования в языке Си для написания прикладных программ, предназначенных для работы под управлением операционной системы MS Windows. Работа через Windows API – это наиболее близкий к операционной системе способ взаимодействия с ней из прикладных программ.

Очень часто в системном программировании возникает задача изменения стандартного поведения системных API-функций. Это нужно, к примеру, если программист хочет организовать собственную обработку какого-либо сообщения и только потом передать стандартной процедуре. Технологии перехвата нужны не только в этом случае, но и, например, для предварительной обработки результатов системных функций поиска файлов FindFirst и FindNext, EnumProcess, которая перечисляет процессы в Windows и т. д. Причем в этих целях такие технологии применяют как антивирусные средства, так и различного рода вредоносное программное обеспечение. Очень часто перехват бывает важен в процессе отладки программ и является одной из основных технологий, применяемых в отладчиках. Перехват API функций позволяет обойти многие ограничения системы. В рамках работы предлагается использовать перехват API-функций для журналирования DNS-запросов процессов. Это позволяет узнать, к каким сайтам обращается тот или иной процесс в ходе своей работы, что может быть полезно в том числе и при анализе подозрительной либо вредоносной программы. Также перехват дает возможности осуществить подмену и выдать вместо IP-адреса запрашиваемого сайта какой-либо другой в зависимости от целей.

DNS (англ. Domain Name System – система доменных имён) – компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене (SRV-запись). Распределённая база данных DNS поддерживается с помощью иерархии DNS-серверов, взаимодействующих по определённому протоколу.

Разрешением имен называется процесс, благодаря которому текстовые имена, например www.microsoft.com или Mycomputer, преобразуются в числовой адрес, например 192.168.1.1, распознаваемый стеком сетевого протокола. Существуют 3, связанных с TCP/IP протоколов разрешения имен, предоставляемых: система имен домена – Domain Name System (DNS), Windows-служба имен Интернета – Windows Internet Name Service (WINS), и протокол разрешения имен одноранговой сети – Peer Name Resolution Protocol (PNRP).

Сетевое приложение, которому требуется получить разрешение DNS-имени в IP-адрес, отправляет поисковый DNS-запрос, используя протокол UDP/IP (TCP/IP используется для запросов, у которых размер ответа превышает 512 байт) на DNS-сервер. Серверы DNS реализуют распределенную базу данных, состоящую из пар имя–IP-адрес, используемых для осуществления преобразования, и каждый сервер обслуживает преобразование для конкретной зоны.

Ключевыми функциями разрешения имен являются: DnsQuery (и ее вариации: DnsQuery_A (ANSI), DnsQuery_W (Unicode), DnsQuery_UTF8 (UTF-8)), DnsQueryEx, которая является более новым вариантом предыдущей, функции getaddrinfo и GetHostByName.

В разрезе API-функций порядок разрешения имен в Windows 7 и Windows Server 2008 R2 следующий:

1) приложение применяет API-функцию DnsQuery() или одну из функций API сокетов Windows – GetAddrInfo () или GetHostByName () – для разрешения имени. Если имя плоское, служба «DNS-клиент» создает полное доменное имя (FQDN), используя заданные DNS-суффиксы;
2) служба «DNS-клиент» ищет полученное полное доменное имя в кеше распознавателя DNS; кеш содержит содержимое файла Hosts и результаты последних удачных или неудачных запросов на разрешение имен. В случае успеха используется обнаруженный результат и процесс на этом завершается;
3) служба «DNS-клиент» проверяет полное доменное имя не предмет применимости к нему каких-либо правил NRPT;
4) если полное доменное имя не подпадает ни под какое правило или к нему применимо только одно правило исключения, служба «DNS-клиент» пытается разрешить полное доменное имя средствами заданных на интерфейсах DNS-серверов;
5) если полное доменное имя подпадает под единственное, не являющееся исключением правило, служба «DNS-клиент» обрабатывает его в соответствии с этим правилом;
6) если полное доменное имя подпадает под несколько правил, служба «DNS-клиент» сортирует правила по приоритету в следующем порядке: полное доменное имя, самый длинный совпадающий префикс (включая IPv4 и IPv6 подсети) или все адреса (Any) и определяет правило, наиболее близко соответствующее полному доменному имени;
7) определив самое приоритетное правило, служба «DNS-клиент» обрабатывает его в соответствии с этим правилом;

В случае Windows 8, 8.1, 10 отличие лишь в том, что вместо DnsQuery() будет использоваться функция DnsQueryEx(). Также стоит отметить, что в ходе отладки моей перехватывающей программы было выявлено, что при разрешении имени через GetHostByName эта функция делает это посредством DnsQueryEx(), то есть перехват и GetHostByName в данном случае смысла не имеет.

Перехват API-функций Windows может быть осуществлен следующими способами:

1) модификация машинного кода прикладной программы. В этом случае модифицируется машинный код, отвечающий в прикладной программе за вызов той или иной функции API;

2) модификация таблицы импорта. Перехватывающая программа находит в памяти таблицу импорта пользовательской программы и исправляет адреса интересующих ее функций на адреса своих перехватчиков. В момент вызова API-функции программа считывает ее адрес из таблицы импорта и передает по этому адресу управление;
3) перехват функций LoadLibrary и GetProcAddress. Идея методики проста: если перехватить GetProcAddress, то при запросе адреса можно выдавать программе не реальные адреса интересующих ее функций, а адреса своих перехватчиков;
4) метод, сочетающий методику 2 и 3. В данной методике модифицируется таблица импорта, причем в обязательном порядке перехватываются функции LoadLibrary и GetProcAddress библиотеки kernel32.dll. В результате у программы не остается шансов узнать правильный адрес функции;
5) модификация программного кода API функции. Методика состоит в том, что перехватчик находит в памяти машинный код интересующих его функций API и модифицирует его. При таком методе перехвата функции уже нет надобности в модификации таблицы импорта запущенных программ и передаче программам искаженных адресов при вызове GetProcAddress;
6) модификация библиотек DLL на диске. Данная методика состоит в том, что системная библиотека модифицируется на диске. Методы модификации аналогичны описанным выше, только изменения производятся не в памяти, а на диске;
7) перехват в режиме ядра. Для перехвата функций необходимо написать драйвер, который произведет модификацию таблицы SDT. Перед модификацией драйверу необходимо сохранить адреса перехватываемых функций и записать в таблицу SDT адреса своих обработчиков;

В рамках данной работы реализован перехват методом сплайсинга (методика 5 из списка выше). Сплайсинг (от англ. Splice – «сращивать или склеивать концы чего-либо») – метод перехвата API функций путём изменения кода целевой функции. Обычно изменяются первые 5 байт функции. Вместо них вставляется переход на функцию, которую определяет программист. Чтобы обеспечить корректность выполнения операции, приложение, которое перехватывает функцию, обязано дать возможность выполниться коду, который был изменён в результате сплайсинга. Для этого приложение сохраняет заменяемый участок памяти у себя, а после отработки функции перехвата восстанавливает изменённый участок функции и дает полностью выполниться настоящей функции.

Все функции стандартных dll Windows поддерживают hot-patch point. При использовании этой технологии перед началом функции располагаются пять неиспользуемых однобайтовых операций nop, сама же функция начинается с двубайтовой инструкции mov edi, edi. Места занимаемого пятью nop достаточно чтобы разместить команду перехода на функцию-перехватчик. Два байта, занимаемых mov edi, edi, предоставляют достаточно места для команды перехода на код размещенный на месте пяти nop. При этом, так как инструкция mov edi, edi не выполняет никаких осмысленных действий, её затирание никак не влияет на работоспособность исходной функции.

В реализации перехвата была использована библиотека mhook, с использованием которой был осуществлен перехват функций DnsQueryEx и getaddrinfo. Посредством

первой была осуществлена попытка разрешения имени «yandex.ru», которая была перехвачена; путем использования второй – «microsoft.com», которая тоже была перехвачена.

В рамках данной работы был реализован перехват API-функций windows, используемых для разрешения DNS-имен, создано программное средство, позволяющее организовать журналирование DNS-запросов процессов.

СОЗДАНИЕ СТЕНДА ДЛЯ ТЕСТИРОВАНИЯ РАСПРЕДЕЛЕННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ ДИСТРИБУТИВА DOCKER НА БАЗЕ ОС ASTRA LINUX

В.В. Котлякова[1)], И.В. Кузьмина[2)]

[1)] ННГУ им. Н.И. Лобачевского
[2)] НИФТИ ННГУ им. Н.И. Лобачевского

Сложность распределенных информационных систем, в том числе и применяемых в научных исследованиях, обусловлена многими факторами, такими как необходимость поддержки протоколов взаимодействия между компонентами этой системы, периодическая недоступность сервисов, сложность управления распределенными транзакциями. При разработке информационных систем важной задачей также является обеспечение информационной безопасности, реализующей доступность, целостность и конфиденциальность данных. В связи с этим в последние годы растёт потребность в защищённых решениях для структур, работающих с конфиденциальной информацией. Одним из них является Astra Linux Special Edition – операционная система (ОС) специального назначения на базе Linux-ядра, созданная для нужд организаций, которые работают с информацией ограниченного доступа.

Кроме того при тестировании сложных информационных систем часто возникают следующие проблемы:

- необходимость испытаний программного обеспечения (ПО) в различных пользовательских конфигурациях, количество которых превышает количество физических компьютеров, выделенных для тестирования;
- большие временные затраты на развертывание и настройку тестовых стендов, содержащих множество различных компонентов, между которыми обеспечивается сетевое взаимодействие;
- большие временные затраты на создание резервных копий систем и их конфигураций, а также восстановление состояния этих систем после запуска тестов;
- невозможность воспроизведения дефекта, найденного специалистом по тестированию, на машине разработчика, потеря времени на его поиск и исправление;
- необходимость в испытаниях программы в условиях аппаратной среды, которой нет в распоряжении команды тестирования;
- необходимость тестирования программного продукта в условиях, требующих быстрого переключения между пользовательскими конфигурациями.

Для решения описанных проблем на всех этапах разработки, тестирования и эксплуатации программных комплексов в распределенных информационных системах широко используется виртуализация вычислительных ресурсов [1]. Одним из наиболее распространенных инструментов для автоматизации процесса тестирования ПО является программный инструмент с открытым исходным кодом Docker, который позволяет быстро создавать изолированное окружение для разработки и тестирования с гибкой настройкой отдельных компонентов. Однако использование ОС Astra Linux Special Edition в качестве базовой платформы для разработки сложных программно-аппаратных комплексов накладывает ограничения на использование программных продуктов,

не прошедших процедуру сертификации по требованиям информационной безопасности. Этот факт делает невозможным использование ПО с открытым исходным кодом при разработке и тестировании систем.

В данной работе решена задача создания дистрибутива Docker 18.06 из открытого исходного кода для использования при разработке и тестирования ПО на базе ОС Astra Linux Special Edition 1.6. Полученный инструмент виртуализации использован для автоматизации процесса тестирования распределенной информационной системы. Все этапы развертывания стенда для тестирования представлены на рисунке. Полученные результаты могут быть использованы при выполнении научно-исследовательских работ [2] в процессе разработки и тестирования сложных программно-аппаратных комплексов.

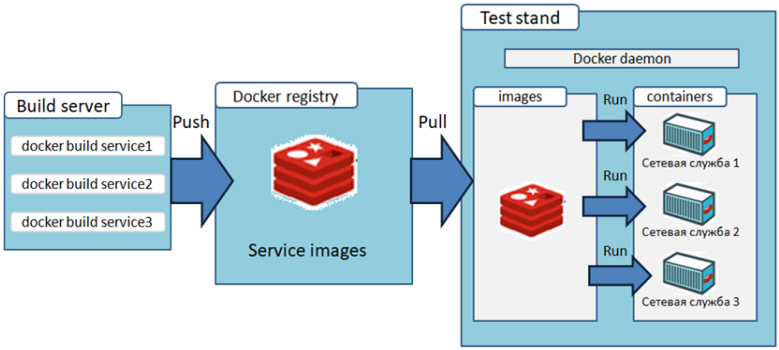

Рис.

[1] Leung F., Neiger G., Rodgers D. et al. Intel Virtualization Technology // Intel Technology Journal. 2006. Vol. 10.

[2] Кузьмина И.В., Фидельман В.Р. Разработка программного обеспечения сложных аппаратно-программных комплексов с использованием принципов непрерывной интеграции // Известия высших учебных заведений. Поволжский регион. Технические науки. 2012. № 2. С. 44.

ПРИМЕНЕНИЕ КОНТЕЙНЕРОВ С НЕТРИВИАЛЬНОЙ СТАТИСТИКОЙ В СТЕГАНОГРАФИЧЕСКИХ АЛГОРИТМАХ

А.А. Горбунов, А.Г. Леонова

ННГУ им. Н.И. Лобачевского

При описании методов сокрытия одного набора данных внутри другого широко используется понятие стеганографического контейнера, являющегося сообщением или файлом, в которое встраивается скрываемая информация [1]. Важным условием качества стеганографического метода является неотличимость по внешним признакам пустого, заполненного и незначительно искажённого контейнера, что определяет специфику областей данных контейнера, куда помещаются биты скрываемого сообщения.

Одним из хорошо изученных на сегодняшний день методов применения стеганографии является LSB (англ. less significant bit – наименее значимые биты), основанный на нечуткости человека к малым изменениям цвета и/или звука. Младшие биты контейнера, в качестве которого может выступать файл изображения, аудио или видеофайл, заменяются битами встраиваемого сообщения. Вносимые при этом искажения в аудиовизуальные данные слабо воспринимаются органами чувств человека, однако компьютерный анализ позволяет обнаружить факт сокрытия.

В настоящей работе проводилось исследование над контейнерами-изображениями (файлами BMP), однако всё нижесказанное можно легко обобщить на случай контейнера-аудиофайла. Были сделаны следующие предположения:

- младшие биты контейнера коррелированны;
- сообщение, которое необходимо скрыть, представляет собой последовательность некоррелированных бит;
- контейнер при передаче по каналу может быть искажён из-за шума или вследствие атаки [2].

При размещении сообщения в каждом из младших бит контейнера замена коррелированных бит псевдослучайной последовательностью может быть легко обнаружена. Данное обстоятельство обуславливает необходимость отыскивать точки, для которых допустимо искажение. Их тем больше, чем больше зашумлённость контейнера. Предположим, что адресат ожидает получить BMP файл заданного размера и примерно знает, что должно быть изображено на картинке. Нарушитель, пытающийся помешать передаче возможного скрытого сообщения путём искажений контейнера, лишён возможности обрезать или подменить картинку, иначе он себя выдаст.

Бит можно считать лежащим в однородной области, если в таблицах разности ему соответствует величина меньше некоторого порога: $|a_i - a_{i-1}| < p$, где a_i, a_{i-1} – значения компонент RGB двух соседних точек изображения, p – пороговое значение, которое можно обосновать различными способами в зависимости от требований, предъявляемых к конкретной стеганографической системе.

Как правило, младшие биты контейнера коррелированны по простому закону. Обычно, глядя на младшие биты, можно узнать, что было изображено на картинке в целом. Такие контейнеры назовём контейнерами с тривиальной статистикой.

Однако существуют контейнеры, в которых корреляция младших бит подчиняется сложному закону, и, глядя на коррелированные младшие биты, мы не можем с уверенностью сказать, как ведут себя старшие биты и что именно изображено на картинке. Такие контейнеры назовём нетривиальными.

Удобно получать нетривиальные контейнеры, различным образом совмещая несколько тривиальных. При этом выгоднее совмещать растровые изображения, младшие биты которых коррелированны меньше, чем младшие биты изображений, созданных средствами векторной графики. Все компоненты, нарисованные в графических редакторах, довольно легко восстановить, иногда даже не имея представления об их первоначальном виде. Поэтому эти области бит менее пригодны для встраивания. Наложение одних изображений на другие, при котором создаются границы, увеличивает неоднородность контейнера, то есть его ёмкость. Однако если задействовать эффект полупрозрачности, при котором часть бит изображения принадлежит одному тривиальному контейнеру, а часть – другому, неоднородность возрастает в разы.

В рамках данной работы был проведён компьютерный эксперимент по созданию контейнера-изображения путём наложения одного файла BMP на другой равного размера с 50% прозрачности. По результатам эксперимента показано появление на итоговом контейнере-изображении упорядоченной структуры, в которой меньше 10% точек взяты из одного исходного изображения, меньше 10% – из другого, а остальные отличаются от соответствующих точек обоих исходных изображений. При этом неоднородность младших бит изображения очень высока, но может быть описана простым математическим соотношением в виде двух ансамблей точек, внутренние статистики которых не коррелированны между собой.

В случае увеличения количества использованных исходных файлов для создания контейнера с нетривиальной статистикой до трёх, таких ансамблей точек будет уже три и т.д. При этом изображения, которым в сумме досталось больше 50% прозрачности, восстанавливались примерно так же, как и изображения с 50% прозрачности. Очевидно, что чем меньше прозрачность исходного изображения, тем больше его точек не претерпят изменения и тем легче нарушителю восстановить составляющую сложного контейнера.

Преобразование сжатия с потерями и последующей декомпрессией не нарушало упорядоченной структуры полученных контейнеров с нетривиальной статистикой. Однако в общем случае их нельзя назвать более робастными, чем контейнеры с тривиальной статистикой, по отношению к подобным преобразованиям изображений. Из общих соображений понятно, что чем сильнее сглаживается растр с десятком наложений высокой прозрачности, тем менее чётким становится изображение.

[1] Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография теория и практика – Киев: МК-пресс, 2006.
[2] Грибунин В.Г., Оков И.Н., Туринцев И.В. Цифровая стеганография. – Москва: Солон-пресс, 2009.

ПРИНЦИПЫ ПОСТРОЕНИЯ DPI-СИСТЕМ ДЛЯ ЭФФЕКТИВНОГО АНАЛИЗА И КЛАССИФИКАЦИИ СЕТЕВОГО ТРАФИКА

Р.Г. Нужный, Л.Ю. Ротков, В.А. Мокляков

ННГУ им. Н.И. Лобачевского

Общие положения

Традиционно задача классификации сетевого трафика может быть сформулирована как получение некоторых характеристик сетевого трафика с определением класса, к которому данный вид трафика относится. В качестве входных характеристик могут выступать как данные пакетов, так и различные статистические характеристики, а в качестве выходных идентификатор приложения, ответственного за генерацию трафика, или идентификатор вида трафика, например VoIP- или BitTorrent-трафик. Задача классификации является важной для организации сетевого взаимодействия. Быстрый рост количества передаваемого трафика и пропускной способности каналов связи приводит к необходимости поиска алгоритмов с пониженной вычислительной сложностью [1].

Значительное увеличение доли шифрованного трафика приводит к ограничению подходов на основе анализа содержимого. В условиях распространения средств анализа и фильтрации многие разработчики сетевых приложений развивают механизмы, противодействующие идентификации используемых протоколов, что также усложняет анализ. В настоящее время активно продвигается технология классификации сетевого трафика на основе глубокого анализа пакетов (Deep Packet Inspection, DPI) (рис. 1).

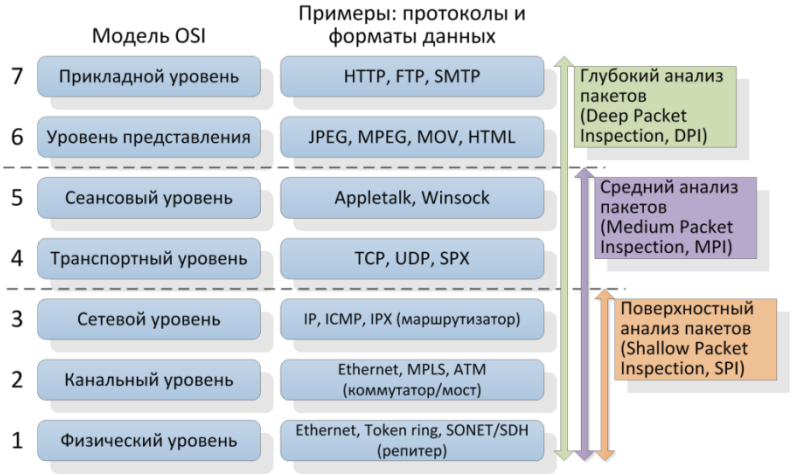

Рис. 1

Технология классификации сетевого трафика на основе глубокого анализа пакетов

В рамках данного подхода анализатор сетевого трафика просматривает содержимое каждого пакета полностью. Одним из важных достоинств подхода является то, что

системы на базе DPI могут принимать решение не только по содержимому пакетов, но и по косвенным признакам, присущим каким-то определённым сетевым сервисам и протоколам. Для этого может использоваться статистический анализ. Например, анализ частоты встречи определённых символов, длин пакетов, расстояние между метками времени последовательных пакетов и т.д. Общий список применения обширен: классификация, ограничение полосы, приоритезация, маркировка, кэширование и т. д. Технология DPI получила развитие, прежде всего, из-за стремительного роста вычислительных способностей процессоров, их быстродействия и, соответственно, возможностей для более полного и точного анализа сетевых данных. В настоящее время она используется для высокоскоростной обработки и идентификации большого числа приложений в реальном времени. Решения на основе DPI хорошо масштабируются как по ширине сетевого канала (известны решения, работающие на каналах порядка 100 Гбит/сек и выше), так и по числу идентифицируемых приложений (в существующих решениях — порядка нескольких тысяч). С точки зрения реализации, основной компонент любого решения DPI – модуль классификации, отвечающий за классификацию сетевых потоков. При этом в зависимости от целей применения DPI, классификация может выполняться с различной точностью:

- тип протокола или приложения (например, Web, P2P, VoIP);
- конкретный протокол уровня приложения (HTTP, BitTorrent, SIP);
- приложение, использующее протокол (Google Chrome, µTorrent, Skype).

Технология DPI на данный момент является текущим стандартом де-факто для средств анализа и классификации сетевого трафика и относится к области критически важных технологий необходимых для обеспечения, как сетевой безопасности, так и требований законодательства. Вследствие этого в последнее время на международном уровне был принят ряд стандартов, требований и рекомендаций по особенностям реализации, внутреннему устройству и набору функций соответствующих средств [2, 3]. Эта технология редко применяется в межсетевых экранах — это скорее область IDS/IPS систем, в качестве исключений можно указать экраны Hogwash и Shield. Однако межсетевые экраны, относящиеся к четвёртому поколению [4] могут учитывать данные IDS/IPS систем в процессе анализа.

Рис. 2

Типичная схема использования системы DPI [5] приведена на рис. 2, где «Внешний интерфейс» — решение типа «DPI как сервис», PCRF - Policy and Charging Rules

Function — устройство, хранящее политики и правила, применяемые к трафику, «Внутренний интерфейс» — устройство хранящее статистику, журналы, результаты применения правил к трафику, и т.д. Концепция «DPI как сервис» может также рассматриваться как отделение инфраструктурной части анализа сетевого трафика от бизнес-логики в рамках отдельных прикладных задач (сбор статистики, межсетевой экран, IDS/IPS системы и др.).

Технология классификации шифрованного сетевого трафика на основе глубокого анализа пакетов

Мировая статистика показывает, что доля шифрованного трафика сети Интернет уже превысила 50 %, и эта тенденция будет сохраняться (рис. 3).

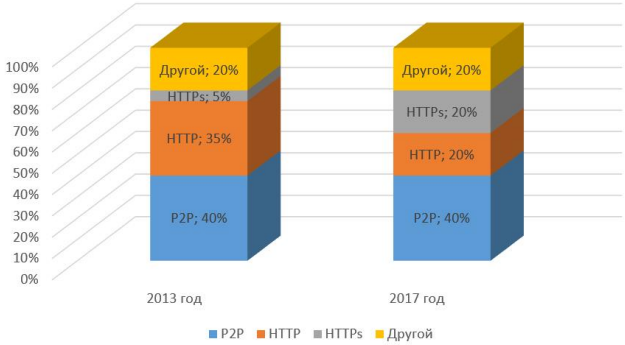

Рис. 3

Один из наиболее распространённых протоколов, использующихся для шифрования передаваемых данных – SSL, в частности обеспечивающий слой шифрования в HTTPS. Получить SSL-сертификат для своего сайта или сервиса становится проще, и каждый владелец ресурса выбирает протокол HTTPS, защищая данные своих пользователей. Для операторов связи и интернет-провайдеров остро встает вопрос выбора способов классификации шифрованного трафика. Особенностью шифрованных данных является то, что к ним ограниченно применимы алгоритмы классификации, использующие данные пакетов, уровня приложения (DPI на рис. 1). Таким образом, шифрование снижает применимость этого, достаточно обширного, класса алгоритмов. Кроме того, попытка применять данные подходы на шифрованном трафике, не отделяя его предварительно от остального потока, существенно снижает пропускную способность компонента классификации, так как приводит к частому проявлению «худшего» случая – просмотру всех данных пакета при отсутствии решения по его принадлежности к некоторому классу [6]. Для преодоления этого недостатка могут использоваться подходы, определяющие наличие факта сжатия или шифрования на основе измерения энтропии [7, 8]. Для классификации шифрованного трафика разрабатываются специализированные подходы. Некоторые из них основаны на анализе первых нескольких пакетов соединения – т.н. «рукопожатие», при котором стороны договариваются об ис-

пользуемом алгоритме шифрования, его параметрах и т.д. [9]. Классификация зашифрованного трафика не предполагает его дешифрацию, информация, содержащаяся внутри пакетов, остается конфиденциальной и видна только пользователю и удаленному узлу. Эти методы предназначены для интернет-провайдеров и операторов связи, которым классификация помогает гибко управлять трафиком и обеспечивать более высокое качество предоставления услуг (QoS и QoE).

Несмотря на то, что напрямую DPI подход к шифрованному трафику не применим, классификация такого трафика всё же осуществляется. Это возможно, если сервис, с которым осуществляется обмен использует расширение Server Name Identification (SNI) протокола TLS для явного указания имени хоста, с которым будет осуществляться обмен. Соответственно классификатор может поддерживать базу имён хостов для наиболее часто используемых сервисов и на основании этого имени идентифицировать трафик, например к хосту google.com.

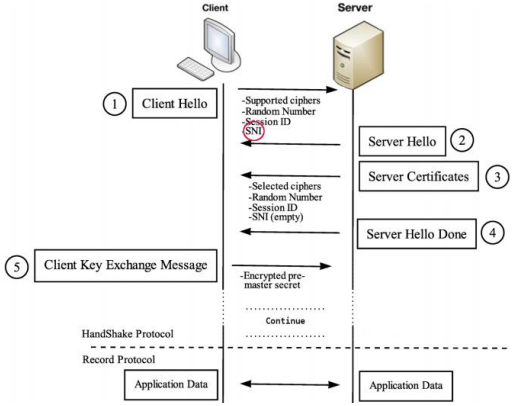

Рис. 4

Такая база, по сути, является аналогом базы портов IANA с тем отличием, что идентифицирующим протокол признаком является не номер порта, извлекаемый из заголовка транспортного уровня, а строковое имя хоста, извлекаемое из заголовка TLS (рис. 4). Изначально это расширение было реализовано для того, чтобы была возможность предоставлять различные сертификаты для сайтов, использующих один и тот же IP-адрес и TCP-порт, что в настоящее время достаточно распространено.

Классификация зашифрованного P2P-трафика основана на методе определения IP-адресов известных P2P-пиров. Так как во время сессии P2P фаза инициализации соединения не шифруется, то на этом этапе возможна идентификация IP-адресов узлов (пиров). Весь трафик, идущий с этих адресов, классифицируется как P2P (например, bittorrent) и может быть приоритезирован в соответствии с текущей пропускной способностью канала связи. Для идентификации различных сервисов передачи голосовой, видео и текстовых сообщений используется метод поиска бинарных шаблонов (сигнатур) в потоке трафика. Как правило, эти шаблоны находятся в первых 2-3 пакетах. Этот метод использует комбинацию различных исследований трафика: джиттер, задержка, длина пакета, расстояние между пакетами и другие. [10, 11]

Благодаря использованию современных методов классификации трафика операторы связи и интернет-провайдеры могут использовать функции приоритизации и оптимизации даже для зашифрованного пользовательского трафика. Это значит, что возможность повышать качество обслуживания клиентов не связана с нарушением конфиденциальности их деятельности в сети Интернет.

[1] Гетьман А.И., Маркин Ю.В., Евстропов Е.Ф., Обыденков Д.О. Обзор задач и методов их решения в области классификации сетевого трафика. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 117-150. DOI: 10.15514/ISPRAS-2017-29(3)-8.
[2] David L. Cannon. CISA Certified Information Systems Auditor Study Guide, 2nd Edition, 2008, ISBN: 978-0-470-23152-4.
[3] Рекомендация МСЭ-Т Y.2770, Требования к углубленной проверке пакетов в сетях последующих поколений, издание 1.0, 20.11.2012.
[4] Рекомендация МСЭ-Т Y.2771, Структура углубленной проверки пакетов, 01.07.2014.
[5] White A., Krishnan S., Bailey M., Monrose F., Porras P. Clear and Present Data: Opaque Traffic and its Security Implications for the Future. NDSS, 2013.
[6] Olivain J., Goubault-Larrecq J. Detecting subverted cryptographicprotocols by entropy checking. Technical report, Laboratoire Spcificationet Verification, June 2006.
[7] Bernaille L., Teixeira R. Early recognition of encrypted applications. In Proceedings of the 8th international conference on Passive and active network measurement (PAM'07), 2007, Springer-Verlag, Berlin, Heidelberg, 165-175.
[8] Global Internet Phenomena Spotlight: Encrypted Internet Traffic. https://www.sandvine.com/downloads/general/global-internetphenomena/ 2015/encrypted-internet-traffic.pdf, дата обращения 01.12.2015.
[9] IP Fragmentation Attacks on Checkpoint Firewalls. https://www.giac.org/paper/gsec/589/ip-fragmentation-attacks-checkpointfirewalls/ 101350, дата обращения 01.12.2015.
[10] Encrypted traffic can be classified https://vasexperts.ru/blog/shifrovannyj-trafik-mozhet-byt-klassificirovan/, дата обращения 01.05.2019.
[11] DPI. http://nag.ru/articles/article/22432/dpi.html, дата обращения 01.12.2015.

ПОДХОД К УСТАНОВЛЕНИЮ СОЕДИНЕНИЙ В РАСПРЕДЕЛЕННОЙ VPN

А.А. Рябов, М.А. Тетеркин

ННГУ им. Н.И. Лобачевского

Для решения задач с большим объемом вычислений используется технология peer-to-peer(P2P), которая создает соединения межу несколькими компьютерами и объединяет их в виртуальную сеть без участия посредника взаимодействий. Каждый из участников сети одновременно являться и сервером и клиентом. Выбор такой топологии сети при использовании протоколов защиты, например, технологий туннелирования: IPsec, L2TP и OpenVPN, позволяет производить распределённые вычисления и обмениваться данными по сети Internet. На данный момент существует несколько решений децентрализованного VPN сервиса: NeoRouter[1], Gbridge[2], Privatix Network[3].

Все рассмотренные системы создают защищённые P2P сети, но серверная часть присутствует во всех реализациях. В основном используется сервер для аутентификации. Кроме того все системы используют протокол L2TP, который не обеспечивает конфиденциальности передаваемых данных, а так же протокол SSL не являющийся в должной степени безопасным. Если углубляться подробнее, то решение Privatix Network поддерживает большую часть протоколов туннелирования, такие как SSTP,L2TP,OpenVPN , некоторые из которых обеспечивают высокий уровень защиты передаваемых данных. Но протоколы OpenVPN требуют дополнительной программной релизации, а остальные используемые протоколы не обеспечивают конфиденциальности пользователя. В решении Gbridge, возможно используется 2-х факторная аутентификация, возможно унаследованная от сервисов Google, но использование стороннего сервиса делает систему зависимой от третьей стороны, а это понижает её надёжность. В системе Neo Router создаются туннели только типа SSLv3, который уязвимость архитектуры под кодовым названием [4] POODLE («пудель», Padding Oracle On Downgraded Legacy Encryption, CVE-2014-3566), которая позволяет расшифровать содержимое защищённого канала коммуникации. Уязвимости, выявленные в реализациях, дают повод усомниться в их надёжности. В связи с этим возникла идея проектирования альтернативной реализации на базе протокола IPsec, который поддерживается большинством устройств пользователей.

В сети должны быть основная группа компьютеров заказчиков вычисления (customer stations(CS)), которые распределяют и выполняют вычисления, и группа наёмных компьютеров (hired station(HS)), которые выполняют вычисления. Количество HS может динамически меняться, а сами станции могут являться мобильными. С владельцами станций необходимо заключить договор о пользовании ресурсами HS и о защите вычислений от «утечек». Но так как данные вычислений будут передаваться по отрытым каналам, их безопасность будет обеспечиваться протоколом IPsec.

В предлагаемой системе для передачи основных пакетов данных и установления соединения будет задействован IPsec, с использованием для аутентификации Revised Mode режима шифрования открытым ключом. А также модифицированный канал для IKE.

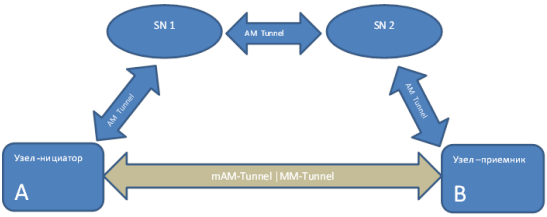

Рис.

Между передатчиком и приёмником существует 2 промежуточных узла, через которые пройдут инкапсулированные пакеты IKE. То-есть, между 4 точками сети попарно создаются служебные туннели с IKE в Aggressive Mode режиме, по которым в виде шифрованных данных передаются пакеты IKE основного туннеля для связи между оконечными узлами, инициатором и приёмником. После поднятия основного туннеля и завершения аутентификации оконечных узлов служебные туннели удаляются. Схематично это представлено на рис.

Сложная конструкция первой фазы позволяет увеличить защищённость передачи. Также возможно увеличение скорости установления соединения за счёт уменьшения длины ключей, так как актуальность данных которые они шифруют становится минимальной после установления основного туннеля. Кроме того даже при перехвате данных на этапе формирования внешней 3-х этапной оболочки для получения важных сведений необходимо произвести атаку на внутренний поток данных.

Ещё одним важным фактором защищённости такой системы является случайность установления внешней оболочки, что усложняет выбор потока для совершения атаки. Защиту данной системы можно повысить за счёт увеличения длины ключа для основного потока и периодической заменой ключей после установления соединения. Фаза 2 протекает как и в стандартном IPsec. После установления основного туннеля система туннелей созданная в Фазе 1 удаляется и стирается из памяти всех устройств участников. Для поддержания защиты передаваемых данных по установленному туннелю ключи Диффи-Хэлмана и SA будут обновляться согласно политике безопасности. При длительном отсутствии передаваемых данных по туннелю он закрывается, и при повторном подключении процедура повторяется заново с самого начала, с созданием служебного туннеля первой фазы. Кроме того, защищённость системы зависит от алгоритмов шифрования, поэтому в системе предполагается использование алгоритмов AES и ГОСТ 28147-89AES, и хэш-функций SHA-256, SHA-512, ГОСТ Р 34.11-2012 и, возможно, MD5 для менее критичной информации. Такая система позволяет создать защищённое соединение между точками, но безопасность узлов передачи тоже требуется повысить. Для решения этой проблемы будет использоваться аутентификация с помощью Pre-shared key и(или) сертификата пользователя, в зависимости от требуемого уровня безопасности сети. Регистрация в сети будет проходить при помощи самоподписанных сертификатов.

[1] http://www.neorouter.com/documents
[2] https://enacademic.com/dic.nsf/enwiki/10968231
[3] https://privatix.io/ru
[4] https://xakep.ru/2014/10/15/poodle

ВНЕДРЕНИЕ ЦИФРОВЫХ ВОДЯНЫХ ЗНАКОВ В АУДИОФАЙЛЫ МЕТОДОМ ИЗМЕНЕНИЯ ВРЕМЕНИ ЗАДЕРЖКИ ЭХО-СИГНАЛА

А.А. Горбунов, Е.В. Тюленева

ННГУ им. Н.И. Лобачевского

Рассмотрение вопросов защиты авторских прав в глобальной информационной сети становится достаточно значимой темой в наше время. Личные права на произведение в сети Internet могут быть нарушены путем следующих действий: присвоения авторства на произведение, неверного указания имени автора при размещении работы, внесения в нее правок без согласия создателя. Имущественные авторские права чаще всего нарушаются путем извлечения прибыли от использования творческой работы лицом, которое не имеет прав зарабатывать на произведении. Это обусловлено тем, что в условиях ожесточенной конкуренции злоумышленникам быстрее и проще украсть чужую интеллектуальную собственность и сделать себе имя, нежели создавать свою [1].

В данной работе рассматриваются вопросы защиты прав на интеллектуальную собственность в виде аудиоданных в сети Internet. Целью работы является изучение способа решения вышеуказанной проблемы с помощью внедрения в аудиофайлы цифровых водяных знаков (ЦВЗ). Приводятся основные методы встраивания ЦВЗ в аудиосигналы и программная реализация одного из них.

Цифровой водяной знак – это данные о владельце или авторе цифровой информации, встроенные в цифровые аудиоданные, видеоизображение или текст, которые могут быть обнаружены и извлечены специальными способами для выявления своей подлинности либо для предъявления претензий владельцу самой информации [2]. ЦВЗ встраивается в несущие данные таким образом, что он неотделим от них. Сами же данные со встроенным в них ЦВЗ оказываются общедоступными, но перманентно помеченными.

Водяные знаки могут быть скрытыми или публичными. Скрытые водяные знаки могут служить механизмами аутентификации или проверки целостности содержания, что предполагает использование этих водяных знаков только определенным кругом лиц, обладающим знаниями о секрете. Публичные водяные знаки служат в качестве переносчиков информации, при этом сам водяной знак может считываться кем угодно. Эти водяные знаки не должны быть обнаружены или удалены третьей стороной.

Несмотря на прозрачность для восприятия, существование водяного знака устанавливается, когда информация, отмеченная водяным знаком, проходит через подходящий детектор водяных знаков, схема которого представлена на следующем рисунке.

Рис.

В связи с тем, что область исследований цифровых знаков тесно связана с сокрытием информации и стеганографией, для внедрения скрываемой информации в аудиосигналы можно использовать методы, применимые в других видах стеганографии.

Для того, чтобы перейти к обсуждению вопросов внедрения информации в аудиосигналы, необходимо определить требования, которые могут быть предъявлены к стеганосистемам, применяемым для встраивания информации в аудиосигналы [2]:

- скрываемая информация должна быть стойкой к наличию различных окрашенных шумов, сжатию с потерями, фильтрованию, аналогово-цифровому и цифро-аналоговому преобразованиям;
- скрываемая информация не должна вносить в сигнал искажения, воспринимаемые системой слуха человека;
- попытка удаления скрываемой информации должна приводить к заметному повреждению контейнера;
- скрываемая информация не должна вносить заметных изменений в статистику контейнера.

Существует пять основных методов встраивания информации в аудиосигналы: методы кодирования наименее значащих бит, методы фазового кодирования, методы расширения спектра, методы маскирования цифровых водяных знаков, методы встраивания информации с использованием эхо-сигнала [3].

Суть метода изменения времени задержки эхо-сигнала заключается в следующем. К параметрам эхо, несущим внедряемую информацию, относятся: начальная амплитуда, время спада, сдвиг и дельта (времена задержки между исходным сигналом и его эхо). При уменьшении сдвига в определенной точке человеческое ухо перестает различать два сигнала. Эту точку трудно определить точно, так как она зависит от исходной записи, типа звука и слушателя. Для большинства типов сигналов и для большинства слушателей слияние двух сигналов происходит при расстоянии между ними около 0,001 секунды. Рекомендованные же значения амплитуды накладываемого эхо-сигнала – 80% для сигнала с меньшей задержкой и 30% для сигнала с большей задержкой. Кодер использует два времени задержки: одно для кодирования нуля, другое для кодирования единицы. И то, и другое время задержки меньше того, на котором человеческое ухо может распознать эхо. Кроме уменьшения времени задержки необходимо правильным образом установить начальную амплитуду и время спада. Для того, чтобы закодировать более одного бита, исходный сигнал разделяется на малые участки; каждый участок рассматривается как отдельный сигнал, и в него внедряется один бит информации. Результирующий закодированный сигнал (содержащий несколько бит внедренной информации) представляет собой комбинацию отдельных участков.

Декодирование представляет собой определение промежутка времени между сигналом и эхо. Для этого необходимо рассмотреть автокорреляционную функцию кепстра.

$$Cs(q) = \frac{1}{2\pi} \int_{-\infty}^{\infty} ln\ |S(\omega)|^2 e^{-i\omega q} d\omega$$

Правило декодирования основано на том, что всплеск автокорреляционной функции будет иметь место через δ_1 или δ_0 секунд после исходного сигнала [2].

В рамках работы программно реализован данный алгоритм. Реализованное приложение встраивания ЦВЗ имеет удобный графический пользовательский интерфейс и позволяет выбирать оцифрованные звуковые файлы без внедренного водяного знака. Пользователю необходимо указать путь к исходному аудио файлу в формате .wav, сам водяной знак и путь, по которому будет сохранено помеченное аудио. На выходе пользователь получает также файл .wav, но уже с внедренным ЦВЗ. Функционал программы тажке позволяет вычленять водяной знак, т.е. включает детектор водяных знаков. Для детектирования также необходимо указать путь к файлу и водяной знак, наличие которого нужно проверить.

С помощью приложения проведены исследования зависимости возникновения ошибок при детектировании водяного знака от выбранных задержек и амплитуд эхосигналов. Установлено, что использование определенных значений позволяет безошибочно детектировать наличие водяных знаков, отклонение же от этих параметров приводит к возникновению ошибок, вплоть до вероятности битовой ошибки 50%.

Хотелось бы заметить, что круг задач защиты авторского права, решаемых с помощью ЦВЗ, постоянно расширяется и не ограничивается использованем в аудиоданных, видеоизображениях или тексте. Так, появились алгоритмы встраивания ЦВЗ в интернет-радиовещании. Также крупнейшие производители программного обеспечения начинают предоставлять свои продукты с возможностями встраивания в них ЦВЗ для решения проблем, связанных с пиратством. На сегодняшний день становится всё более очевидным, что при наличии достаточной правовой поддержки цифровые водяные знаки могут стать наиболее эффективным инструментом для защиты авторских прав на интеллектуальную собственность в сети Internet.

[1] Ларичев В.Д., Трунцевский Ю.В. Защита авторского права в аудиовизуальной сфере. Уголовно-правовой и криминалистический аспекты. – М.: Дело, 2004, 352 с.
[2] Грибунин В.Г., Оков И.Н., Туринцев И.В. Цифровая стеганография. – М.: Солон-Пресс, 2009, 272 с.
[3] Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика. – Киев: МК-Пресс, 2006, 288 с.

ПРИМЕНЕНИЕ МЕТОДОВ ФОНЕТИЧЕСКОГО АНАЛИЗА РЕЧИ ДЛЯ ВЫЯВЛЕНИЯ ЛИДЕРА В КОМПАНИИ

Р.А. Васильев

ННГУ им. Н.И. Лобачевского

Не секрет, что в любом коллективе есть свои лидеры и аутсайдеры. Узнать, какова позиция сотрудника в организации, насколько она благоприятна для его дальнейшего развития можно с помощью проективных методик. Одной из самых информативных является методика выявления лидера на основе фонетического анализа речи и идентификации по голосу.

Информационная система идентификации дикторов по голосу («ИС ИДГ») – это зарегистрированная в Роспатенте программа для ЭВМ [1], предназначенная для идентификации дикторов и тестирования эмоционального состояния личности по голосу. В ее названии отражены особенности принципа действия ИС ИДГ, а именно: анализ фонетического строя речи диктора в зависимости от текущего эмоционального состояния последнего.

Принцип действия ИС ИДГ основан на автоматической оценке качества речи диктора на базовом, фонетическом уровне по общесистемному шенноновскому критерию минимума требуемой избыточности (МТИ) речевого сигнала. По существу, это первая попытка в мире – в теории и на практике – преодолеть острейшую проблему многокритериальности устной речи с позиций строгого, теоретико-информационного подхода [2]. Тем больший интерес для специалистов широкого профиля представляют публикуемые далее результаты теоретического и экспериментального исследования ИС ИДГ в задаче тестирования эмоционального состояния личности по голосу. Для проверки эффективности ИС ИДГ проведены экспериментальные исследования [3].

По результатам проведенных исследований можно сделать следующие выводы:

- подтверждена устойчивость предложенного информационного показателя качества речи дикторов на разных текстах и в разное время ее записи;
- установлена высокая чувствительность относительной величины требуемой избыточности (ОВТИ) по отношению к эмоциональным нагрузкам на диктора в процессе его монолога;
- экспериментально подтверждена прямо пропорциональная зависимость ОВТИ от интенсивности физической нагрузки на диктора.

Таким образом, в результате проведенного исследования дано экспериментальное обоснование принципа минимума требуемой избыточности в роли информационного показателя качества речи диктора, который нацелен не на сравнение речи разных дикторов между собой, а на исследование влияния разного рода факторов на качество речи конкретного диктора [4].

Анализируя колебания ОВТИ в процессе речеобразования относительно ее значения в заведомо комфортных условиях, мы можем установить как факт отклонения психологического состояния диктора от средних показателей, что позволяет выявить потенциального лидера в организации.

[1] Свид. о гос. регистрации программы для ЭВМ №2015663306 Программа идентификации дикторов по голосу / Васильев Р.А. Зарег. 15.12.2015г. – М.: Роспатент, 2015.
[2] Савченко В.В. Информационная теория качества речи // Изв. вузов России. Радиоэлектроника. 2011. Вып. 1. С. 17.
[3] Савченко В.В., Васильев Р.А. Анализ эмоционального состояния дикторов по голосу на основе фонетического детектора лжи // Научные ведомости Белгородского государственного университета. 2014. Вып. № 21(192)32\1. С. 186.
[4] Васильев Р.А. Исследование особенностей идентификации дикторов по голосу при различиях в произношении дикторов // Безопасность информационных технологий. 2013. № 1. С. 85.

Секция «Информационные системы.
Средства, технологии, безопасность»

Заседание секции проводилось 21 мая 2019 г.
Председатель – Л.Ю. Ротков, секретарь – А.А. Рябов.
Нижегородский государственный университет им. Н.И. Лобачевского.