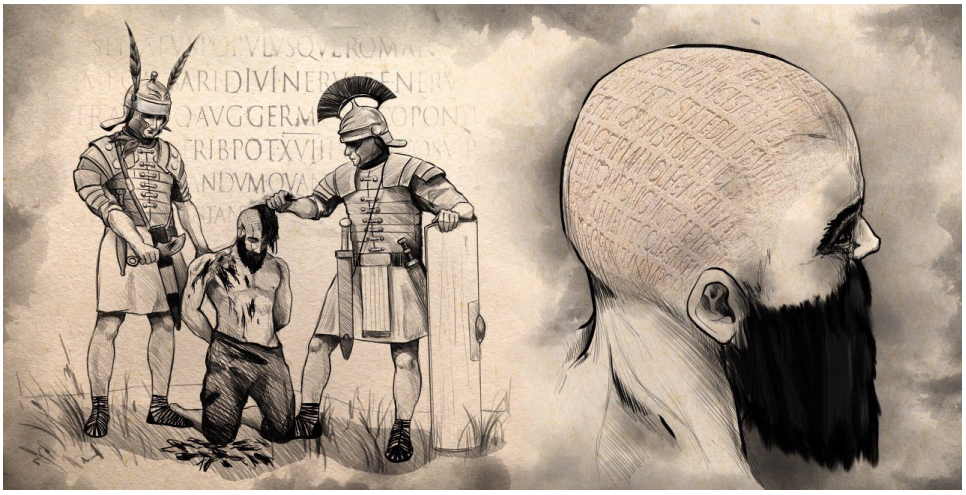
Кибератаках

Алексей Шульмин, Евгения Крылова - Август 3, 2017. 12:00

Стеганография (от греч. στεγανός — скрытый + γράφω — пишу; буквально «тайнопись») — наука, позволяющая спрятать передаваемые данные в некотором контейнере, таким образом скрыв сам факт передачи информации.

В отличие от криптографии, которая скрывает содержимое тайного сообщения, стеганография скрывает сам факт его существования. Впервые понятие стеганографии было введено в 1499, но сам метод существует очень давно. Легенды донесли до нас метод, который использовался в Римской империи: для доставки сообщения выбирали раба, голову которого брили, а затем с помощью татуировки наносили текст. После того, как волосы отрастали, раба отправляли в путь. Получатель сообщения снова обривал голову раба и читал сообщение.



В данной статье мы будем использовать следующие определения:

- Сообщение — внедряемое скрытным образом послание, которое необходимо спрятать;
- Контейнер (стегоконтейнер) — любой объект, используемый для тайного внедрения сообщения;
- Стегосистема — методы и средства, используемые для создания скрытого канала для передачи информации;
- Стегоканал — канал для передачи стегоконтейнера;
- Ключ — ключ для получения скрытого содержания из контейнера (используется не всегда).

В течение всего XX века активно развивалась как стеганография, так и наука об определении факта внедренной информации в контейнер — стегоанализ (по сути — атаки на стегосистему). Но сегодня мы наблюдаем новый и опасный тренд: все больше и больше разработчиков вредоносного ПО и средств кибершпионажа прибегает к использованию стеганографии. Большинство антивирусных решений на сегодняшний день не защищают от стеганографии или защищают слабо, меж тем, нужно понимать, что каждый заполненный контейнер опасен. В нем

могут быть скрыты данные, которые эксфильтруются шпионским ПО, или коммуникация вредоносного ПО с командным центром, или новые модули вредоносного ПО.

На сегодня учеными разработаны и опробованы различные алгоритмы и методы стеганографии, мы отметим следующие:

- **LSB-стеганография** (сообщение скрывается в младших битах (возможно использование одного или нескольких младших бит) контейнера. Чем меньше бит задействовано, тем меньше артефактов получает оригинальный контейнер после внедрения.
- **Метод, основанный на сокрытии данных в коэффициентах дискретного косинусного преобразования** (далее ДКП) — разновидность предыдущего метода, которая активно используется, например, при внедрении сообщения в контейнер формата JPEG. При прочих равных, такой контейнер имеет несколько меньшую емкость чем в предыдущем методе, в том числе за счет того, что коэффициенты «0» и «1» остаются неизменными — внедрение сообщения в них невозможно.
- **Метод сокрытия информации при помощи младших бит палитры**— этот метод по сути является вариантом общего метода LSB, но информация встраивается не в наименее значащие биты контейнера, а в наименее значащие биты палитры, очевидный недостаток такого метода — низкая емкость контейнера.
- **Метод сокрытия информации в служебных полях формата —** довольно простой метод, основанный на использовании служебных полей заголовка контейнера для хранения сообщения. Очевидные минусы — низкая емкость контейнера и возможность обнаружения внедренных данных при помощи обычных программ для просмотра изображения (которые иногда позволяют видеть содержимое служебных полей).
- **Метод встраивания сообщения —** заключается в том, что сообщение встраивается в контейнер, затем при помощи схемы, известной обеим сторонам, извлекается. Можно встроить несколько сообщений в один контейнер, при условии, что способы их внедрения ортогональны.
- **Широкополосные методы**, которые подразделяются на:
    - метод псевдослучайной последовательности; используется секретный сигнал, который моделируется псевдослучайным сигналом.

    - метод прыгающих частот: частота несущего сигнала меняется по определенному псевдослучайному закону.
- **Метод оверлея** — по сути не является настоящей стеганографией, основан на том, что некоторые форматы содержат в заголовке размер данных, или же обработчик этих форматов будет читать файл до маркера конца данных. Примером такого метода является хорошо известный метод «rar-jpeg», который основан на конкатенации графического файла в формате JREG и RAR-архива. ПО для просмотра JPEG будет считывать информацию до границы, указанной в заголовке файла, а RAR-архиватор откинет все, что находится до сигнатуры «RAR!», которая обозначает начало архива. Таким образом, если такой файл открыть в просмотрщике графических файлов — мы увидим картинку, а если в RAR-архиваторе — содержимое RAR-архива. Очевидные минусы такого подхода заключаются в том, что оверлей, добавленный к контейнеру, легко выделяем при визуальном исследовании такого файла.

В этой статье мы рассматриваем только методы сокрытия информации в графических контейнерах и в сетевых пакетах, но область применения стеганографии значительно шире.

За недавнее время мы наблюдали использование стеганографии в следующих вредоносных программах и средствах кибершпионажа:

- **Microcin (AKA six little monkeys);**
- **NetTraveler;**
- **Zberp;**
- **Enfal (its new loader called Zero.T);**
- **Shamoon;**
- **KinS;**
- **ZeusVM;**
- **Triton (Fibbit).**

Почему авторы вредоносного ПО все активней используют стеганографию в своих разработках? Мы видим три главные причины:

- Это позволяет им скрыть сам факт загрузки/выгрузки данных, а не только сами данные;
- Помогает обойти DPI-системы, что актуально в корпоративных сетях;
- Использование стеганографии может позволить обойти проверку в AntiAPT-продуктах, поскольку последние не могут обрабатывать все графические файлы (их слишком много в корпоративных сетях, а алгоритмы анализа довольно дорогие).

Для конечного пользователя детектирование стегоконтейнера может быть нетривиальной задачей. Для примера приведем два контейнера: пустой и заполненный, в качестве которых будем использовать стандартное изображение для графических исследований Lenna.

Lenna.bmp Lenna_stego.bmp

Оба изображения «весят» 786 486 байт, но правое содержит сообщения 10 первых глав «Лолиты» Набокова.

Посмотрите внимательно на эти две картинки. Вы можете различить их? Они одинаковые и по размеру, и по внешнему виду. Тем не менее, одна из них — контейнер с внедренным сообщением.

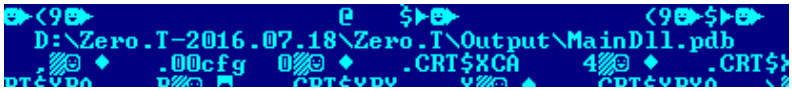
Проблемы налицо:

- Использование стеганографии сегодня — очень популярная идея среди авторов вредоносного и шпионского ПО;
- Антивирусные средства вообще и средства защиты периметра в частности мало что могут сделать с заполненными контейнерами: их очень трудно обнаружить поскольку они выглядят как обычные графические (и не только) файлы;
- Все существующие программы для детектирования стеганографии по сути являются PoC— Proof-of-Concept, и их логика не может быть имплементирована в промышленные средства защиты из-за низкой скорости работы, не слишком высокого уровня детектирования, и иногда даже иногда — из-за ошибок в математике (мы видели и такие случаи).

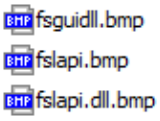
Выше был приведен список вредоносного ПО (впрочем, далеко не полный), которое использует стеганографию для сокрытия своей коммуникации. Рассмотрим конкретный пример из этого списка — вредоносный загрузчик Zero.T.

Этот загрузчик был обнаружен нами в конце 2016 года, но первое описание было опубликовано нашими коллегами из Proofpoint.

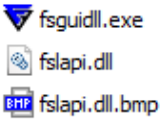
Мы назвали его Zero.T, т.к. такая строка присутствует в его исполняемом коде (в пути к pdb-файлу проекта):



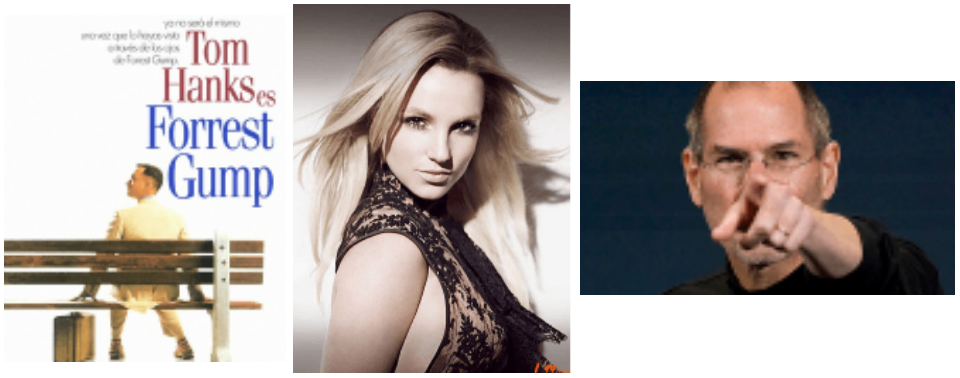
Не останавливаясь здесь на попадании в систему и закреплении в ней, отметим, что Zero.T скачивает полезную нагрузку в виде Bitmap-файлов:



Обрабатывает их особым образом, после чего получает вредоносные модули:



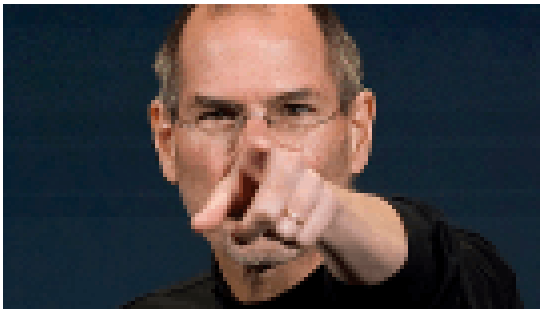
На поверку эти три BMP-файла оказались картинками:



Но картинки эти не совсем обычные. Это — заполненные контейнеры. В каждом из которых несколько (алгоритм допускает вариативность) младших значащих бит заменены на полезную нагрузку.

Так как же определить является ли картинка заполненным контейнером или нет? Существуют разные способы, но самый простой из них — визуальная атака. Суть его заключается в формировании новых изображений на основе исходного, состоящих из наименее значащих бит различных цветовых плоскостей.

Рассмотрим на примере изображения с фотографией Стива Джобса:



Применим к этому изображению визуальную атаку, построим новые изображения из отдельных значащих бит соответствующих разрядов:

На втором и третьем изображении заметны области с высокой энтропией (высокой плотностью данных) — это и есть внедренное сообщение.

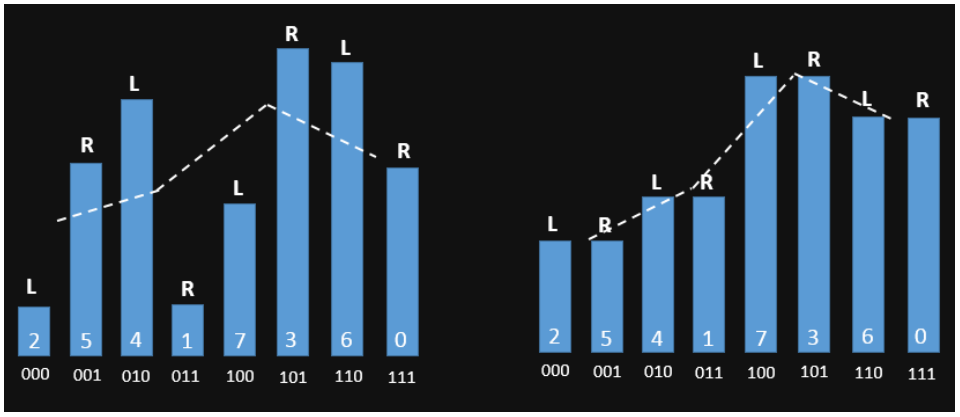
Просто, не так ли? И да, и нет. Это просто, потому что аналитик (и даже простой пользователь!) может с легкостью увидеть внедренные данные). И сложно потому что такой анализ довольно трудно автоматизировать. К счастью, ученые уже давно разработали несколько методов выявления заполненных контейнеров, основанных на статистических характеристиках изображения. Но все они основываются на предположении, что внедренное сообщение имеет высокую энтропию. Чаще всего это действительно так: поскольку емкость контейнера ограничена, сообщение перед внедрением сжимается и/или шифруется, т.е. его энтропия повышается.

Но наш пример из реальной жизни — вредоносный загрузчик Zero.T, — не сжимает свои модули перед внедрением! Вместо этого он увеличивает количество используемых наименее значимых бит: 1,2 или 4. Да, использование большего количества наименее значащих бит приводит к появлению визуальных артефактов в изображении, заметных простому пользователю. Но ведь речь идет об автоматическом анализе. Вопрос, который нам предстоит выяснить: *пригодны ли статистические методы для обнаружения внедренных сообщений в том случае, если их энтропия не высока?*

Статистические методы анализа: гистограммный метод

Описываемый метод, предложенный в 2000 году Андресом Вестфелдом и Андреасом Пфитцманом, также известен как «хи-квадрат»-метод. Попытаемся изложить его суть.

Весь растр анализируется, для каждого цвета считается количество точек такого цвета в растре (для простоты здесь говорим про изображение, имеющее одну цветовую плоскость). Метод исходит из предположения, что количество точек двух соседних цветов («соседние» цвета — цвета, которые отличаются только наименее значимым битом) различается существенно для нормального, обычного изображения (пустого контейнера) (рис. а ниже). И количество пикселей таких цветов является примерно одинаковым для заполненного контейнера (рис. б)



а — пустой контейнер б — заполненный контейнер

Визуально можно представлять себе алгоритм таким образом, это довольно просто для понимания.

Говоря строго, алгоритм заключается в последовательном выполнении следующих шагов:

- Теоретически ожидаемая частота встречаемости пикселей цвета i после внедрения сообщения рассчитывается следующим образом:

$$n_i^* = \frac{|\{\text{colour}|\text{sortedIndexOf}(\text{colour}) \in \{2i, 2i+1\}\}|}{2}$$

- Измеренная частота вхождения символа определенного цвета определена как:

$$n_i = |\{\text{colour}|\text{sortedIndexOf}(\text{colour}) = 2i\}|$$

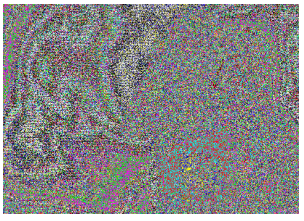
- Хи-квадрат критерий для количества степеней свободы k-1 рассчитывается следующим образом:

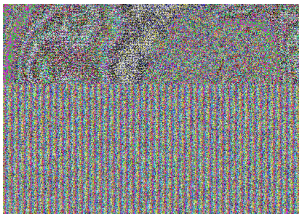
$$\chi^2_{k-1} = \sum_{i=1}^{k} \frac{(n_i - n_i^*)^2}{n_i^*}$$

- P— это вероятность того, что распределения n_i и n_i^* при этих условиях равны. Она рассчитывается при помощи интегрирования функции гладкости:

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_0^{\chi^2_{k-1}} \mathrm{e}^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} \mathrm{d}x$$

Разумеется, мы провели проверку применимости этого метода для детектирования заполненных стегоконтейнеров и вот результаты:

Оригинальное изображение	Визуальная атака	Атака «хи-квадрат», 10 зон
		[analyze_payload] xi_square[0]: 154 [analyze_payload] xi_square[1]: 177 [analyze_payload] xi_square[2]: 250 [analyze_payload] xi_square[3]: 930 [analyze_payload] xi_square[4]: 143 [analyze_payload] xi_square[5]: 194 [analyze_payload] xi_square[6]: 591 [analyze_payload] xi_square[7]: 1552 [analyze_payload] xi_square[8]: 629 [analyze_payload] xi_square[9]: 794

```
[analyze_payload] xi_square[0]: 154
[analyze_payload] xi_square[1]: 177
[analyze_payload] xi_square[2]: 250
[analyze_payload] xi_square[3]: 930
[analyze_payload] xi_square[4]: 143
[analyze_payload] xi_square[5]: 10
[analyze_payload] xi_square[6]: 9
[analyze_payload] xi_square[7]: 11
[analyze_payload] xi_square[8]: 28
[analyze_payload] xi_square[9]: 26
```

Пороговые значения распределения хи-квадрат для p=0,95 и p=0,99 соответственно 101.9705929 и 92.88655838. Таким образом, для зон, у которых рассчитанное значение хи-квадрат меньше порогового, можно принять исходную гипотезу «распределение частот соседних цветов — одинаковое, следовательно, это заполненный стегоконтейнер».

Действительно, если посмотреть на изображения для визуальной атаки, несложно заметить, что эти области содержат внедренное сообщение. Таким образом, для внедренных сообщений с высокой энтропией метод работает.

Статистические методы анализа: RS-метод

Еще один статистический метод обнаружения заполненных стегоконтейнеров был предложен Джессикой Фридрих, Мирославом Гольяном и Андреасом Пфитцманом в 2001 году. Он называется RS-метод, где RS означает «регулярный-сингулярный».

Все изображение разделяется на множество групп пикселей, далее для каждой группы применяется специальная флиппинг-процедура. На основании значения функции-дескриминанта до и после применения флиппинга все группы делятся на регулярные, сингулярные и неиспользуемые.

Алгоритм основывается на предположении, что количество регулярных и сингулярных групп пикселей в оригинальном изображении и в изображении после применения флиппинга должно быть примерно равным. Если количество таких групп существенно меняется в процессе применения флиппинга, это значит, что исследуемое изображение является заполненным контейнером.

По шагам алгоритм работает следующим образом:

- Изображение разделяется на группы из n pixels (x_1, ..., x_n).
- Описывается т.н. дискриминант-функция, которая сопоставляет каждой группе пикселей G = (x_1, ..., x_n). действительное число f(x_1, ..., x_n) ∈ R.
- Мы можем определить дискриминант-функцию для группы пикселей (x_1, ..., x_n) следующим образом:

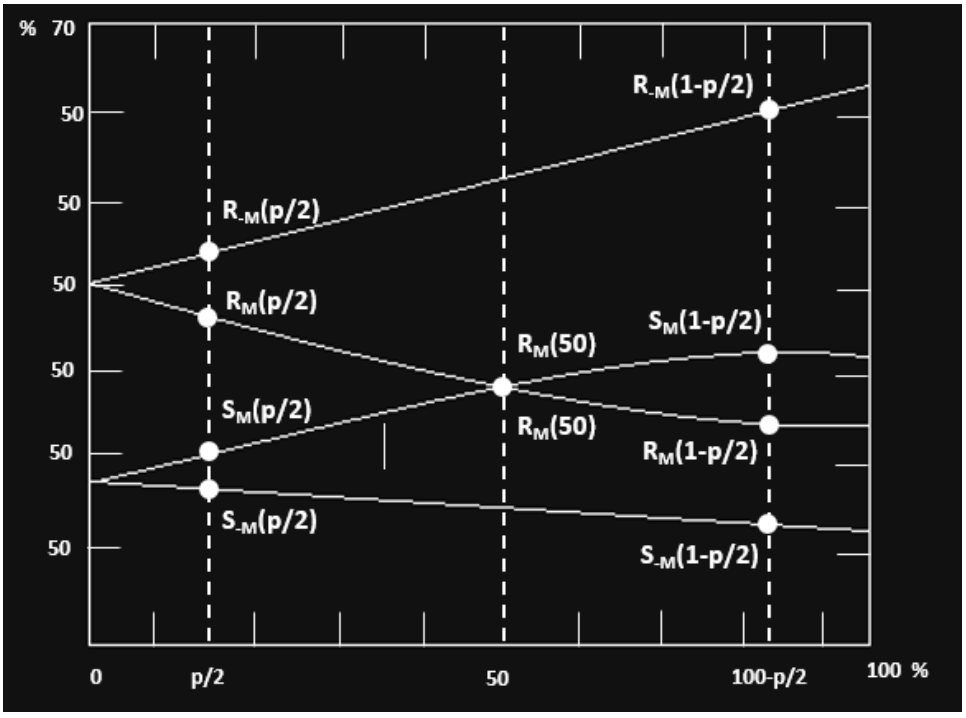
$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n-1} \left| x_{i+1} - x_i \right| .$$

- Также мы определяем функцию флиппинга, имеющую следующие свойства:

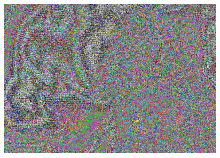
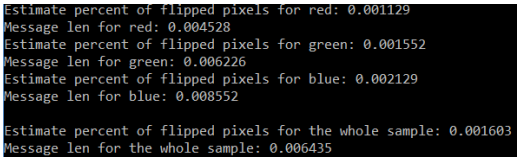
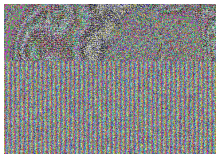
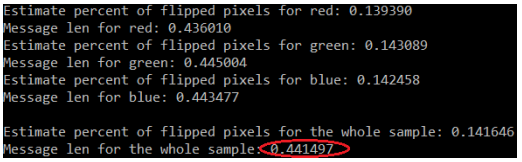
$$F(F(x)) = x$$
$$F_1: 0 \leftrightarrow 1,\ 2 \leftrightarrow 3,\ \ldots,\ 254 \leftrightarrow 255$$
$$F_{-1}(x) = F_1(x+1) - 1$$

На основании значений дискриминант-фукнции до и после флиппинга, все группы пикселей делятся на регулярные, сингулярные и неиспользуемые:

$$\begin{aligned} &\text{Regular groups:} && G \in R && \Leftrightarrow && f(F(G)) > f(G) \\ &\text{Singular groups:} && G \in S && \Leftrightarrow && f(F(G)) < f(G) \\ &\text{Unusable groups:} && G \in U && \Leftrightarrow && f(F(G)) = f(G), \end{aligned}$$



Мы провели исследование и этого метода тоже, получив следующие результаты. Мы использовали те же заполненный и пустой контейнеры, что и в предыдущем опыте.

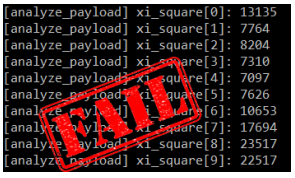
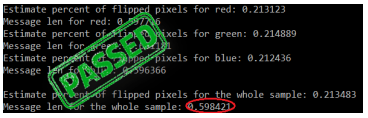
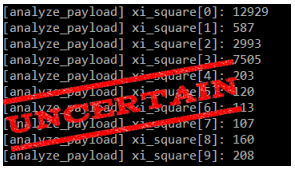
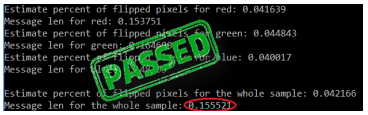
Оригинальное изображение	Визуальная атака	Атака «хи-квадрат», 10 зон
		
		

Обратим внимание, что этот метод атаки не выносит бинарного вердикта «содержит ли этот контейнер внедренное сообщение», вместо этого он определяет примерную длину внедренного сообщения (в процентах).

Из результатов выше понятно, что для пустого сообщения этот метод вынес вердикт о заполнении менее 1% контейнера, а для заполненного — о заполнении примерно 44% контейнера. Очевидно, что результаты слегка неточные. Обратим внимание на заполненный контейнер: из визуальной атаки однозначно следует, что заполнено более 50% контейнера, в то время как RS-атака говорит нам, что заполнено 44% контейнера. Поэтому мы можем применять этот метод если установим некий «порог срабатывания»: наши эксперименты показали, что 10% является достаточным порогом надежности. Если RS-атака утверждает, что более 10% контейнера заполнено — можно доверять этому вердикту и маркировать контейнер как заполненный.

Теперь пришло время выяснить, пригодны ли два рассмотренных способа для тестирования в реальных условиях на контейнерах Zero.T, сообщение в которых имеют обычную энтропию.

Мы провели соответствующие тесты и вот результаты:

Оригинальное изображение	Атака «хи-квадрат»	RS-атака
		
		

Очевидно, что на изображениях с низкой энтропией атака типа «хи-квадрат» не применима — результаты или неудовлетворительные, или не вполне точные, зато RS-атака отработала отлично: в обеих случаях было определено наличие скрытого сообщения. Но что же делать если автоматические методы анализа показали отсутствие внедренного сообщения, а мы все еще подозреваем это?

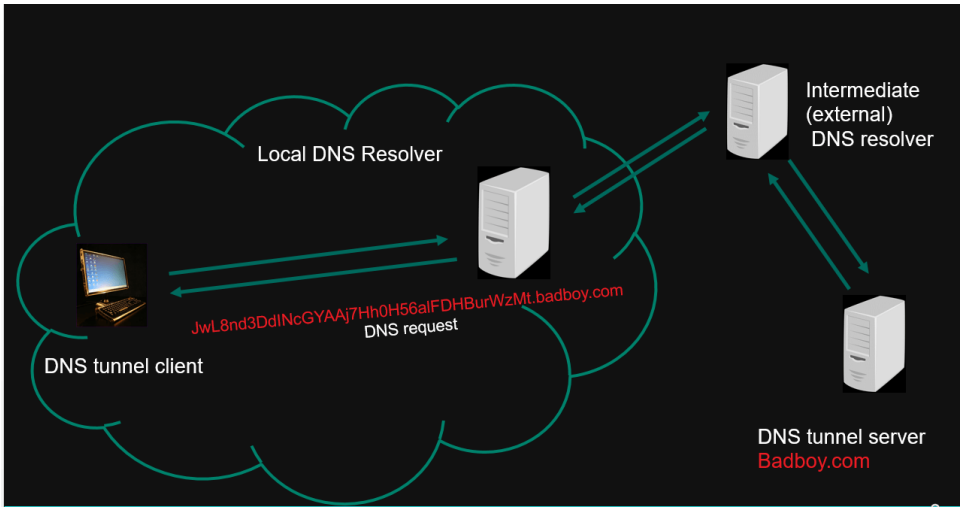
Можно использовать конкретные процедуры для извлечения полезной нагрузки, разработанные для конкретных семейств вредоносного ПО. Так, для рассмотренного в этой статье загрузчика Zero.T мы написали собственную процедуру экстракции внедренного сообщения, которая схематично работает следующим образом:

Очевидно, что, если в итоге мы получили валидный результат (в данном случае — исполняемый файл), исходное изображение было контейнером.

DNS-Tunneling: тоже стеганография?

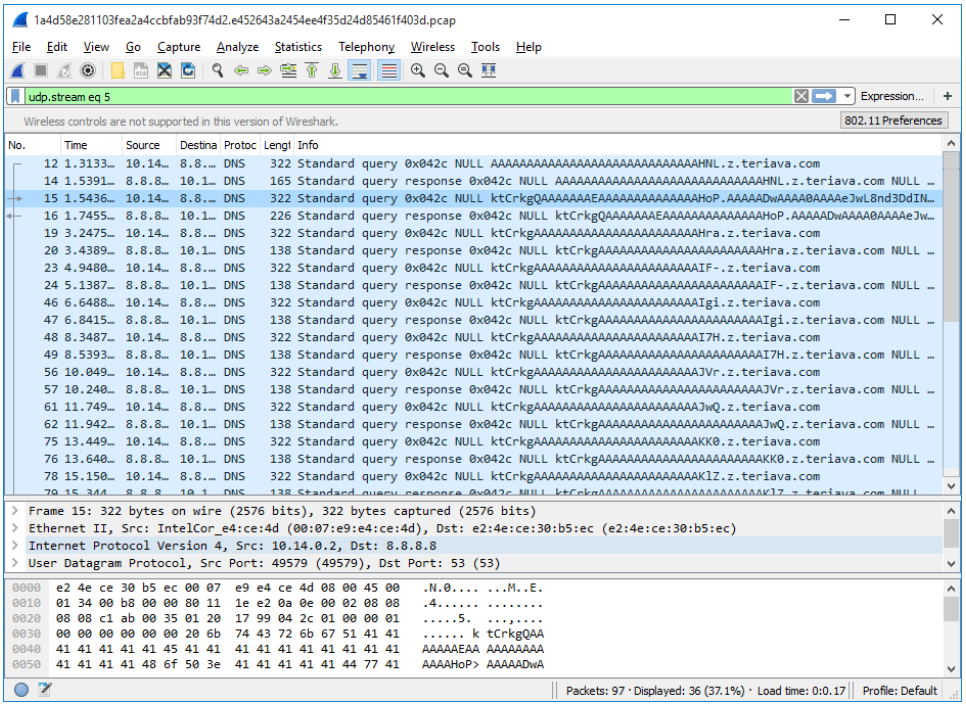
Можем ли мы считать использование DNS-туннеля подвидом стеганографии? Определенно, да. Для начала давайте вспомним, как выглядит схема DNS-туннеля в общих чертах.

В закрытой сети с пользовательской машины посылается запрос на «резолв» домена, например, wL8nd3DdINcGYAAj7Hh0H56a8nd3DdINcGYAlFDHBurWzMt[.]imbadguy[.]com (где доменное имя второго уровня не несет смысловой нагрузки). Локальный DNS-сервер передает запрос внешнему DNS-серверу. Тому, в свою очередь, неизвестно имя 3-го уровня и запрос передается дальше. Таким образом, по цепочке перенаправлений от одного DNS-сервера к другому запрос достигает DNS-сервера домена imbadguy[.]com.



Вместо разрешения запроса на сервере злоумышленник может извлечь из полученного домена нужную ему информацию, расшифровав первую часть его имени. Например, таким способом можно передать информацию о системе пользователя. В ответ DNS-сервер злоумышленника так же посылает некую информацию в зашифрованном виде, передавая ее в доменном имени 3-го уровня или выше.

Таким образом, злоумышленник имеет в запасе для каждого DNS резолва 255 символов, до 63 символов для поддоменов. 63 символа в одну сторону — 63 в ответ, 63 — туда, 63- обратно... Неплохой канал для передачи данных! А главное — скрытый, ведь невооруженным взглядом не видно, что идет обмен какой-то дополнительной информацией.



Для специалистов, знакомых с сетевыми протоколами и в частности с DNS-туннелированием, такой дамп трафика будет выглядеть довольно подозрительно: очень уж много успешных резолвов длинных доменов. В данном случае мы наблюдаем реальный пример трафика зловреда Backdoor.Win32.Denis, который использует DNS туннель как скрытый канал общения со своим командным центром.

Обнаружить DNS-туннель можно с помощью любой известной IDS, например, Snort, Suiricata или BRO Ids. Как именно распознать DNS-туннелирование? Существуют различные способы. Например, очевидно, что доменные имена для резолва при туннелировании значительно длиннее, чем обычно. В сети можно найти достаточное количество вариаций на эту тему:

```
alert udp any any -> any 53 (msg:»Large DNS Query, possible
cover channel»; content:»|01 00 00 01 00 00 00 00 00 00|»;
depth:10; offset:2; dsize:>40; sid:1235467;)
```

Совсем примитивно, но встречается и такое:

```
Alert udp $HOME_NET and -> any 53 (msg: «Large DNS Query»;
dsize: >100; sid:1234567;)
```

Тут можно экспериментировать и стремиться найти свой идеальный баланс между количеством ложных срабатываний и нахождением реального DNS-туннелирования.

На что еще можно обратить внимание помимо подозрительно большой длины доменного имени? На аномальный синтаксис доменных имен. Все мы примерно представляем, как выглядят типовые домены — это в большинстве своем буквы и цифры. Если в доменном имени присутствуют символы, характерные для Base64 кодировки, то это выглядит довольно подозрительно, не так ли? Если при этом еще и длина не маленькая, то явно стоит присмотреться повнимательней.

И таких аномалий можно описать довольно большое количество, регулярные выражения нам в помощь.

Хочется отметить, что даже такой простой по своему смыслу подход к поиску DNS-туннелей дает очень хорошие результаты. Благодаря нескольким таким правилам для IDS мы на входящем потоке вредоносного ПО, поступающем к нам в «Лабораторию», обнаружили несколько новых, неизвестных до этого бэкдоров, использующих скрытый канал общения с командным центром — DNS-туннель.

Выводы

Мы отмечаем сильный положительный тренд: все больше и больше разработчиков вредоносного ПО начинает использовать стеганографию, в том числе — для сокрытия коммуникации с командным центром и для загрузки модулей. Это дает результат, ведь процедуры анализа контейнеров вероятностные и дорогие, следовательно, большинство защитных решений не могут себе позволить обрабатывать все объекты, которые потенциально могут быть заполненными контейнерами.

Однако решения есть, они основаны на комбинировании различных способов анализа, высокоскоростных предетектах, исследовании метаданных потенциально заполненного контейнера и т.п. Такие решения сейчас нашей компанией реализуются в нашей платформе защиты от таргетированных атак — KATA. Ее использование позволяет сотруднику службы информационной безопасности своевременно узнать о возможной таргетированной атаке на защищаемый периметр и/или эксфильтрации данных из него.

Публикации на схожие темы

Пересечение активности GreyEnergy и Zebrocy

Kaspersky Security Bulletin: Угрозы в 2019 году

Kaspersky Security Bulletin 2018. Важные события года

ВСЕГО КОММЕНТАРИЕВ: 2

Алекс

Опубликовано 06/08/2017. 13:11

а что если для защиты просто «слегка портить» любую «входящую» картинку лёгкой модификацией нижних битов некоей случайной последовательностью. Это может «сбить» данные, а обычто достаточно несколько байт для «сбоя» в коде. Пользователь мог бы принять такой риск «порчи» файлов при работе с интернетом (а может и везде).

ОТВЕТИТЬ

Алексей Шульмин

Опубликовано 08/08/2017. 12:39

Да, иногда это может сработать. Но, если пользователь и примет возможную модификацию потенциальных контейнеров во входящем трафике, то в исходящем это может быть неприемлемо. А именно в исходящем трафике обычно передаются эксфильтрованные данные из скомпрометированной системы.

ОТВЕТИТЬ