**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра технологий программирования**

**АНАЛИЗ ВРЕДНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ DENSE PIXEL DISPLAY**

Курсовой проект

|  |  |
| --- | --- |
|  | Сечко Марины Юрьевны  студентки 3 курса,  специальность «компьютерная безопасность»  Научный руководитель:  доцент кафедры технологий программирования,  Мушко Вилена Владимировна |

Минск, 2017

**Аннотация**

Сечко М.Ю. Анализ вредоносного программного обеспечения с использованием dense pixel display: Курсовой проект / Минск: БГУ, 2017.

В работе рассматриваются техника визуализации большого набора данных при помощи пиксельного метода и её применимость к анализу вредоносного программного обеспечения. Реализовано построение пиксельных графиков по входным данным с использованием языка программирования Python.

**Анатацыя**

Сячко М.Ю. Аналіз шкоднаснага праграмнага забеспячэння з выкарыстаннем dense pixel display: Курсавы праект / Мінск: БДУ, 2017.

У працы разглядаюцца тэхніка візуалізацыі вялікага набору дадзеных пры дапамозе піксельнага метаду і яе дастасавальнасць да аналізу шкоднаснага праграмнага забеспячэння. Рэалізавана пабудова піксельных графікаў па уваходным дадзеных з выкарыстаннем мовы праграмавання Python.

**Annotation**

Sechko M.Y. Malware analysis using dense pixel display: Course project / Minsk: BSU, 2017.

The work deals with the technique of visualizing a large data set using a pixel method and its applicability to the analysis of malicious software. Drawing pixel plots on the input data using the Python programming language has been developed.

**РЕФЕРАТ**

Курсовой проект, 28 стр., 18 рис., 4 источника.

**Анализ вредоносного программного обеспечения с использование dense pixel display**

**Ключевые слова**: визуализация данных, анализ вредоносного ПО, пиксельные методы визуализации, dense pixel display.

**Объект исследования** – пиксельные методы визуализации и анализ вредоносных программ.

**Цели работы** – облегчение процесса анализа больших наборов данных при помощи визуализации пиксельными методами; реализация метода визуализации.

**Результат работы** – инструмент, позволяющий по входных данным построить пиксельный график.

**Область применения** – анализ данных о вредоносном программном обеспечении, анализ больших наборов данных.

СОДЕРЖАНИЕ

[Введение 5](#_Toc501368803)

[1. Анализ вредоносного программного обеспечения 6](#_Toc501368804)

[1.1. Типы анализа вредоносного по 6](#_Toc501368805)

[1.2. Анализ кибер-атак 8](#_Toc501368806)

[1.3. Цели анализа данных 9](#_Toc501368807)

[2. Пиксельные методы визуализации 10](#_Toc501368808)

[2.1. Общие сведения 10](#_Toc501368809)

[2.3.1. Расположение пикселей 12](#_Toc501368810)

[3. Задача проекта 17](#_Toc501368811)

[4. Реализация визуализации пиксельного графика 18](#_Toc501368812)

[4.1. Пример визуализации спам-атак 19](#_Toc501368813)

[4.2. Пример визуализации файлов вредоносных программ 24](#_Toc501368814)

[Заключение 26](#_Toc501368815)

[Список литературы 27](#_Toc501368816)

ВВЕДЕНИЕ

Визуализация данных – один из методов, используемых для передачи данных или информации путем их представления в виде визуальных объектов (например, точек или линий), содержащихся в графике.

Первичной целью визуализации данных является четкая и эффективная передача информации с помощью статистической графики, графиков и информационной графики (инфографики).

Визуализация данных – это и искусство, и наука. Как эстетичность и красота, так и функциональность должны идти рука об руку, обеспечивая понимание довольно разреженного и сложного набора данных, передавая его ключевые аспекты более интуитивно понятным способом.

Отличным примером важности визуализации данных является так называемый квартет Энскомба — четыре набора числовых данных, у которых простые статистические свойства (среднее значение, дисперсия, корреляция) идентичны, но их графики существенно отличаются.

Уже из этого можно сделать вывод, что при анализе данных визуализация данных не только желательна, а даже необходима.

Область применения визуализации достаточно велика – она используется в научных и статистических исследованиях (в частности, в прогнозировании, интеллектуальном анализе данных, бизнес-анализе), в педагогическом дизайне для обучения и тестирования, в новостных сводках и аналитических обзорах. Сфера анализа вредоносного программного обеспечения не стала исключением в потребности визуализации.

1. АНАЛИЗ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Из-за растущей угрозы со стороны вредоносного программного обеспечения исследование уязвимых систем становится все более важным. Необходимость регистрации и анализа активности охватывает сети, отдельные компьютеры, а также мобильные устройства. Хотя существуют различные автоматические подходы и методы для обнаружения и идентификации вредоносных программ, фактический анализ постоянно растущего числа подозрительных образцов является трудоемким процессом для аналитиков вредоносного ПО. Использование визуализации может помочь поддержать этот процесс анализа в отношении исследования, сравнения и обобщения образцов вредоносных программ.

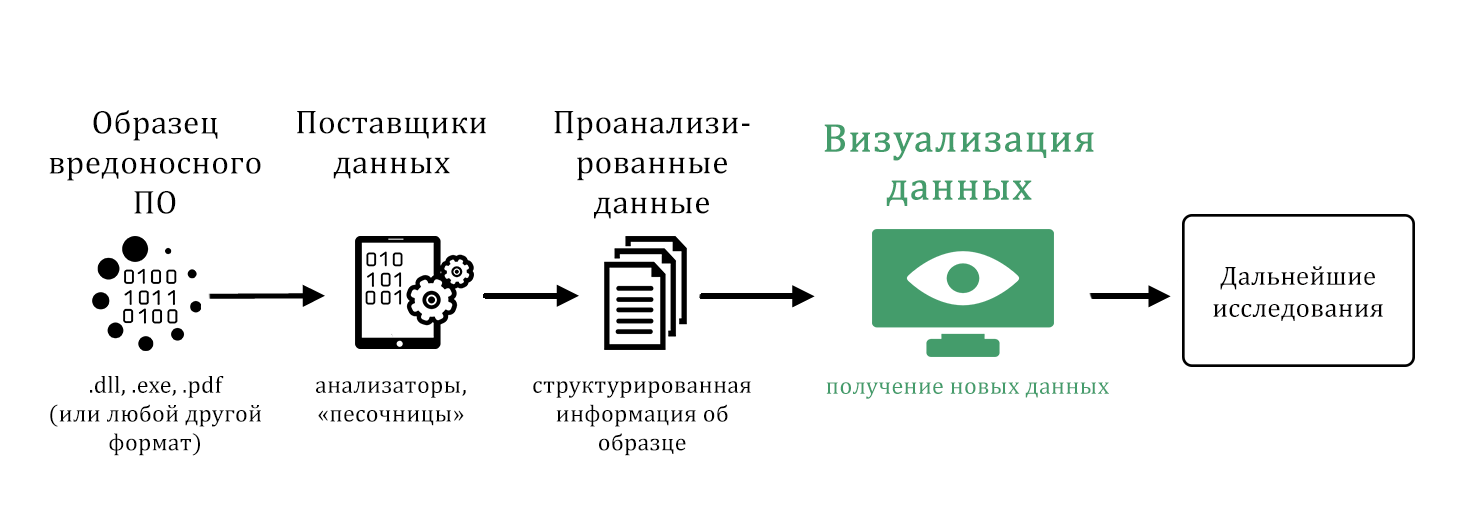


Рисунок 1.1. Схема анализа вредоносного программного обеспечения

В общем случае, схему анализа любой вредоносной программы можно представить в виде, представленном на рисунке 1. В качестве входных данных может выступать сам исполняемый файл или какая-либо информация о нём – отчёт о его функциях, обращениях к библиотекам, время выполнение и так далее. Далее этот файл проходит через один из так называемых «поставщиков данных» – любой инструмент, который статически или динамически анализирует входной файл и возвращает собранную информацию для дальнейшей обработки. После этого наступает этап визуализации полученных данных с целью выявлений новых интересных свойств или закономерностей. Полученные результаты могут использоваться для дальнейших исследований и/или получения выводов о входном файле.

****1.1 Типы анализа вредоносного ПО****

Существует два основных подхода к анализу вредоносного ПО: статический и динамический.

* *Статический анализ* описывает методы, которые не требуют фактического выполнения проверяемого образца. В зависимости от глубины анализа файл может быть проверен на его основные свойства (например, тип файла, контрольная сумма), легко извлекаемая информация (например, строки, информация об импортах DLL библиотек) или полностью разобран.
* *Динамический анализ* выполняет файл в главной системе, при этом различные инструменты контролируют выполнение проверяемого образца и записывают соответствующую информацию в лог выполнения. В зависимости от глубины анализа, это могут быть как простые операции файловой системы, так и целые инструкции, захваченные через отладчик.

Заметим, что несмотря на то, что динамический анализ может показать больше при базисном (имеющем небольшую глубину) анализе, при более продвинутом анализе вредоносного ПО используют именно статический анализ – при помощи дизассемблеров, методов обратной разработки и так далее. Причиной этому является зависимость динамического анализа от среды анализа: данные, полученные при этом типе анализа, могут являться неполными в силу текущих свойств соответствующей системы. Однако статический анализ проводится медленнее, так как аналитику приходится вручную разбираться в структуре исполняемого файла.

Выделим некоторые свойства, которые могут быть интересны для анализа вредоносного программного обеспечения.

1. Хэш-код (hash code) файла. Двоичный код образца проходит через процесс хеширования и полученный хэш сопоставляется с другими, хранящимися в базе поставщиков данных, чтобы определить, является ли весь файл (контрольная сумма) или части кода злонамеренными. Некоторые инструменты позволяют по полученному хэшу сразу вычислять и семейство вредоносной программы.
2. Информация о заголовках в файле описывает фактический тип файла (несмотря на возможное изменение типа файла) и секции его кода.
3. Импорты библиотек и функций могут указывать на функциональность самого файла. Как правило, библиотеки содержат набор функций, относящихся к определенной области работы. Таким образом, можно будет предположить, какую цель преследует исполняемый файл.
4. Инструкции ЦПУ (центрального процессорного устройства) несут в себе информацию об основной функциональности программы. Они могут быть получены при помощи процесса обратной разработки и дизассемблирования кода либо при динамическом отслеживании значений регистров.
5. Информация об API-вызовах и системных вызовах может помочь распознать, к примеру, создание новых файлов или модификации в регистре.
6. Операции с файловой системой, регистрами, операции с сетью и так далее.

1.2. Анализ кибер-атак

Вредоносные программы не представляли бы никакой угрозы, если бы всегда оставались только на компьютерных системах, где они были разработаны. Однако существуют методы для внедрения их на другие системы.

Кибератака – это любой тип наступательного маневра, который совершается отдельными лицами, группами или организациями и который атакует компьютерные информационные системы, компьютерные сети или персональные компьютеры с целью внедрения вредоносного ПО.

Выделяют следующие виды атак в интернете:

* Вандализм и пропаганда — порча интернет-страниц, замена содержания другими элементами; рассылка обращений пропагандистского характера или вставка пропаганды в содержание других интернет-страниц.
* Сбор информации — взлом частных страниц или серверов для сбора информации и/или её замены на фальшивую.
* Отказ сервиса — атаки с разных компьютеров для нарушения функционирования сайтов или компьютерных систем.
* Вмешательства в работу оборудования — атаки на компьютеры, которые занимаются контролем над работой какого-либо оборудования, что приводит к его отключению или поломке.
* Атаки на пункты инфраструктуры — атаки на компьютеры, обеспечивающие жизнедеятельность городов, их инфраструктуры, таких как телефонные системы, водоснабжения, электроэнергии, пожарной охраны, транспорта и так далее.

Выделим «полезные» свойства кибер-атак, благодаря которым можно их анализировать:

1. IP-адреса могут дать информацию, откуда или куда была направлена атака.
2. Время атаки или атак может указывать на часовой пояс, в котором находится злоумышленник.
3. Протоколы, которые использовались при передаче данных.

Обеспечивать безопасность систем с каждым днём становится всё сложнее из-за появления новых типов вредоносных программ. Понимание моделей атаки дает более глубокое понимание уязвимости сети, которое, в свою очередь, может использоваться для защиты сети от будущих атак.

1.3. Цели анализа данных

Выделяют три цели, для которых используется анализ вредоносных элементов.

* *Индивидуальный анализ* используется для анализа отдельного образца вредоносного ПО, выявления целей его выполнения и пути их достижения.
* *Классификация* вредоносного ПО. Здесь анализ используется для определения семейства вредоносных программ, к которому относится заданный образец.
* *Анализ выборки* объектов. В этом случае анализ используется для выявления схожести в поведении большого количества образцов вредоносного ПО с целью получения какой-либо статистики.

2. ПИКСЕЛЬНЫЕ МЕТОДЫ ВИЗУАЛИЗАЦИИ

2.1 Общие сведения

Методы визуализации приобретают все большее значение при изучении и анализе больших объемов многомерной информации. Одной из важных техник визуализации, которая особенно интересна для визуализации больших наборов многомерных данных, является класс пиксельных методов – так называемые плотностные пиксельные графики (dense pixel display). Основная идея пиксельных методов визуализации состоит в том, чтобы одновременно представлять как можно больше объектов данных на экране путем сопоставления каждого значения данных с пикселем экрана, учитывая его цвет и расположение.

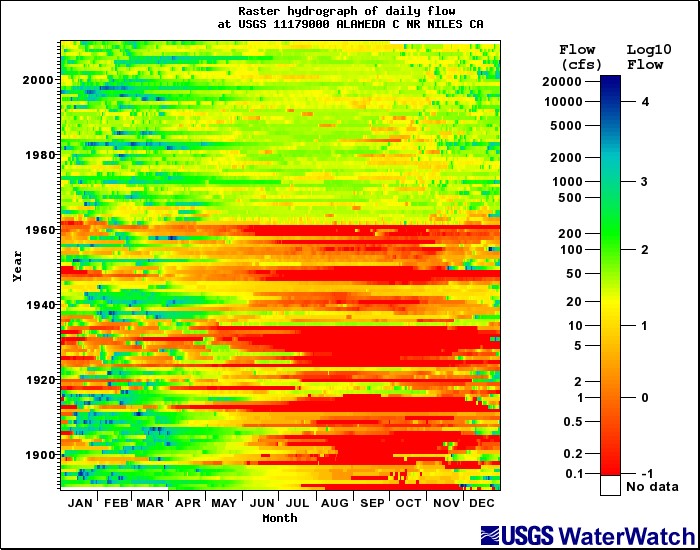


Рисунок.2.1. Пример использования пиксельных графиков

Первые очевидные плюсы такого метода визуализации заключаются в следующем:

* Отсутствие пустого места. Каждый объект представляется пикселем, так что заполнение графика происходит максимальным образом.
* Отсутствие перекрытий. Так как каждый объект занимает определенное место на графике, возможность перекрытия одним образцом другого пропадает – визуализация более понятна для восприятия.
* Цветовая интерпретация понятна и легко читаема человеком.
* Можно визуализировать действительно огромные наборы данных.

Минусы метода:

* Необходимо найти функцию расположения пикселей такую, чтобы между ними не было пересечений, но в то же время чтобы сохранялась понятность графика.
* В одном графике, как правило, представляется одно свойство набора данных. Возникает необходимость строить несколько графиков для охвата всех интересующих атрибутов.

Пример использования графика представлен на рисунке 2.1, на котором отображены данные о суточном водном потоке в городе Аламида за промежуток времени длиной в сто лет.

2.2. Свойства пиксельного графика

Все пиксельные методы визуализации разделяют экран на несколько областей – окон. Для набора данных размерности (то есть каждый образец из данных имеет измерений (атрибутов, свойств)) экран разделяется на окон – по одному на каждое измерение. В некоторых случаях происходит разделение на окно – дополнительное -ое окно используется для отображения какой-либо результирующей функции по всем остальным свойствам. Внутри окон значения данных упорядочены в соответствии с заданной общей сортировкой (рис.2.2). Корреляции, функциональные зависимости и другие интересующие отношения между измерениями могут быть обнаружены путем связывания соответствующих областей в нескольких окнах.

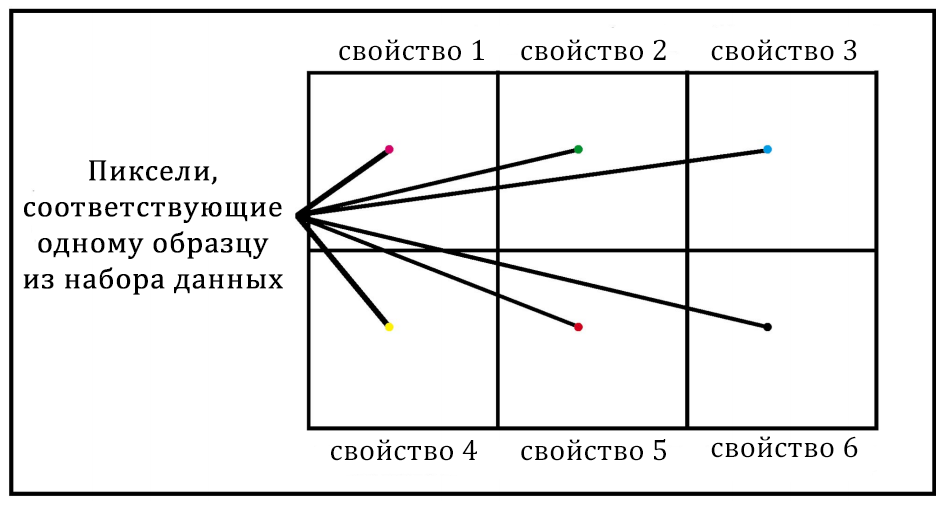


Рисунок 2.2. Представление образца данных на пиксельном графике

При этом возникает несколько вопросов в построении графика:

1. Сопоставление значений данных с цветами: нужно тщательно спроектировать отображение значений на графике, чтобы оно стало интуитивно понятным.
2. Расположение (компоновка) пикселей внутри окон. Решение этого вопроса зависит от входных данных и задачи визуализации. В большинстве случаев, проблема расположения может быть формально описана как проблема оптимизации, и различные техники визуализации подходят для различных задач оптимизации.
3. Форма окон. Например, при прямоугольной форме окон, как показано на рисунке 2.1, для наборов данных с большим количеством атрибутов окна для разных измерений иногда расположены далеко друг от друга, из-за чего становится трудно найти какие-либо связи между этими атрибутами. Как и в предыдущем пункте, проблема формы окон может быть представлена как проблема оптимизации.
4. Упорядочивание окон для измерений. В большинстве случаев не существует определенного порядка расположения окон измерений на графике. Чтобы обнаружить зависимости и корреляции между атрибутами, представленными в под окнах, нужно разместить связанные измерения рядом друг с другом. Из этого снова следует проблема оптимизации.

В этой работе мы остановимся на более подробном рассмотрении только второй проблемы.

2.3.1 Расположение пикселей

В этом пункте рассмотрим проблему расположения пикселей более подробно. Данный вопрос является одним из наиболее важных, поскольку только хорошее расположение позволит обнаружить кластеры или корреляции между измерениями. Для рассмотрения проблемы компоновки нужно различать наборы данных, которые имеют естественный порядок объектов (например, зависящие от времени данные) и наборы данных без естественного порядка.

* Определим проблему расположения пикселей в пиксельном графике для набора данных с естественным порядком объектов.

*Проблема компоновки* пикселей для множества объектов , расположенных в соответствии с их естественным порядком, заключается в нахождении отображения этого множества на окно размера , т.е. биективного отображения , такого, чтобы минимизировать функцию

, где

- -расстояние между пикселями, соответствующими элементам и .

Таким образом, мы должны найти функцию, при которой минимизируется суммарное расстояние между получившимися пикселями относительно минимального расстояния двух пикселей в прямоугольном окне размера .

Задача отображения упорядоченных одномерных наборов данных в двумерное измерение рассматривалась математиками еще в прошлом веке. Так называемые *заполняющие пространство кривые* решают в точности выше указанную задачу оптимизации и обеспечивают минимизацию заданной функции.

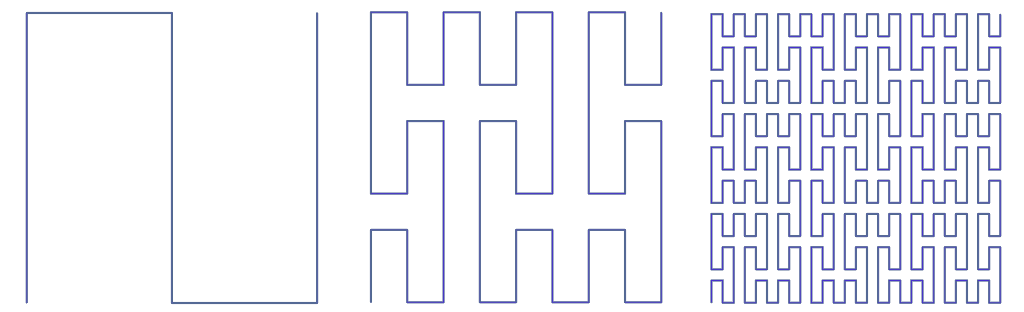
**

Рисунок 2.3. Три итерации кривой Пеано

*Семейство заполняющих пространство кривых*, или *кривые Пеано* – семейство параметрических кривых, образ которых содержит квадрат. Другими словами, кривая из этого семейства представляют собой непрерывное отображение из единичного отрезка в единичный квадрат, то есть проходит через любую точку этого квадрата.

Это значит, что если обозначить через координаты точки в единичном квадрате, а через расстояние вдоль кривой, на котором эта точка достигается, то точки, имеющие близкие к значения, будут иметь также близкие значения и к. Обратное не всегда верно — некоторые точки, имеющие близкие координаты имеют достаточно большую разницу в расстоянии .

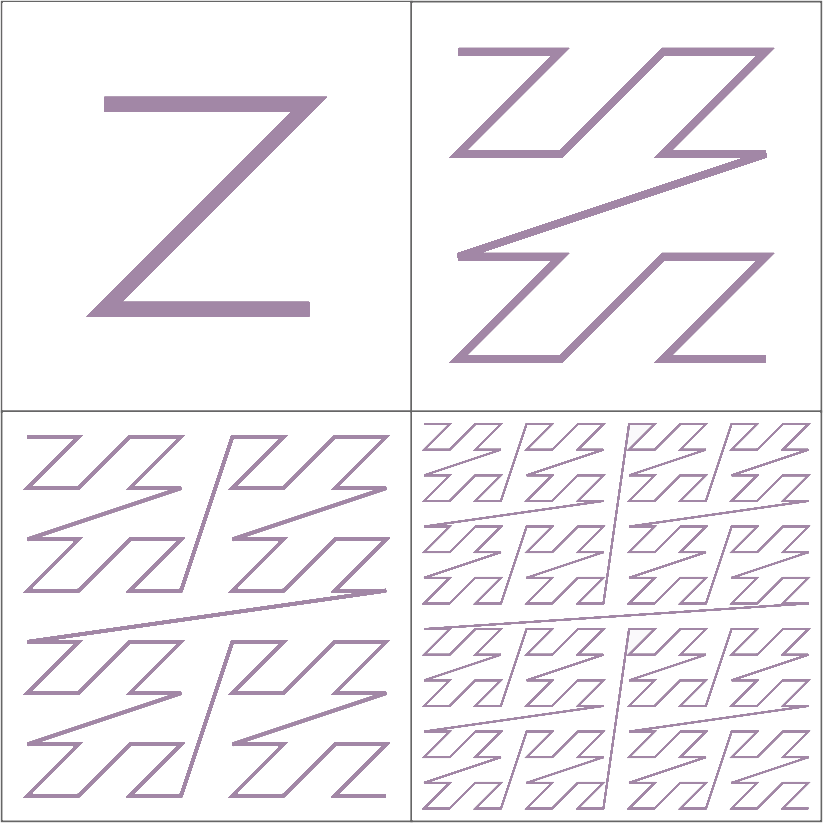
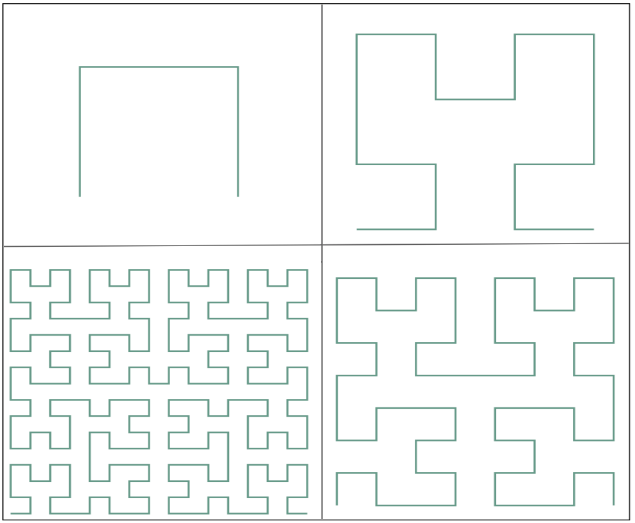


Рисунок 2.4. Четыре итерации кривой Гильберта (слева) и кривой Мортона (справа)

Примерами заполняющих пространство кривых являются кривые Пеано, Мортона и Гильберта (рис 2.4). Все они являются фрактальными кривыми, что означает, что для их реализации можно использовать рекурсию.

* Для данных, которые не имеют естественного порядка, предлагается следующий выход. В этом случае главной целью визуализации является представление на графике только тех значений, которые удовлетворяют некоторому входящему *запросу*. Простой запрос для -размерного набора данных может соответствовать точке в -мерном пространстве; если задан диапазон запроса, то ему будет соответствовать область в -мерном пространстве. Результатом запроса являются объекты из набора данных, которые находятся в области запроса.

Результат запроса может оказаться очень большим или пустым – в большинстве случаев количество подходящих объектов предсказать невозможно. Чтобы получать более информативные результаты от запроса, на графике можно изображать не только объекты, которые входят в область запроса, но и объекты, которые расположены «близко» к этому запросу.

Понятие «близко» может быть определено дополнительной функцией, которая и будет определять расстояние от объекта до запроса. Если объект принадлежит области запроса, расстояние равняется 0. Для числовых и временных типов данных можно использовать обычные функции расстояния: модуль и, например, количество секунд от одной даты до другой.

Таким образом, мы можем добавить -ый атрибут к объектам – расстояние до входящего запроса. Теперь мы имеем набор данных с естественным порядком – в соответствии с полученным расстоянием. А значит, к нему можно применить техники из предыдущего пункта – кривые Пеано.

Проблема заключается в том, что интуитивно предполагается, что объекты, которые удовлетворяют запросу, должны находиться в середине графика, в то время как кривые Пеано располагают наиболее подходящие элементы в левом нижнем углу. Однако эту проблему можно решить, к примеру, следующим образом, который представлен на рисунке 2.5.

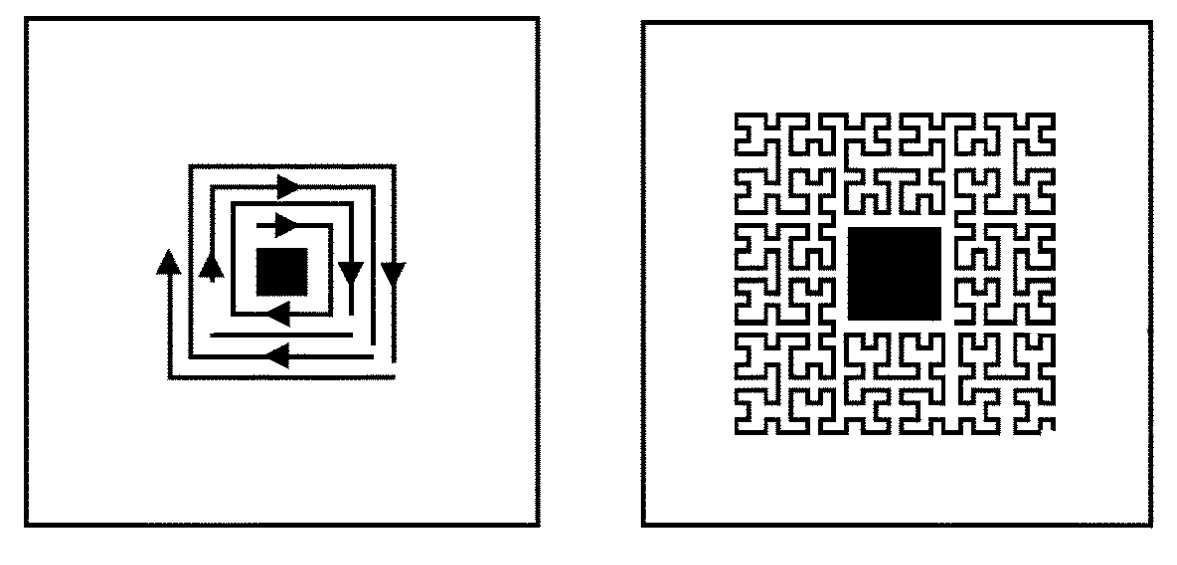
****

Рисунок 2.5. Варианты расположения объектов, удовлетворяющих запросу – спиральный обход и обход по кривой Гильберта

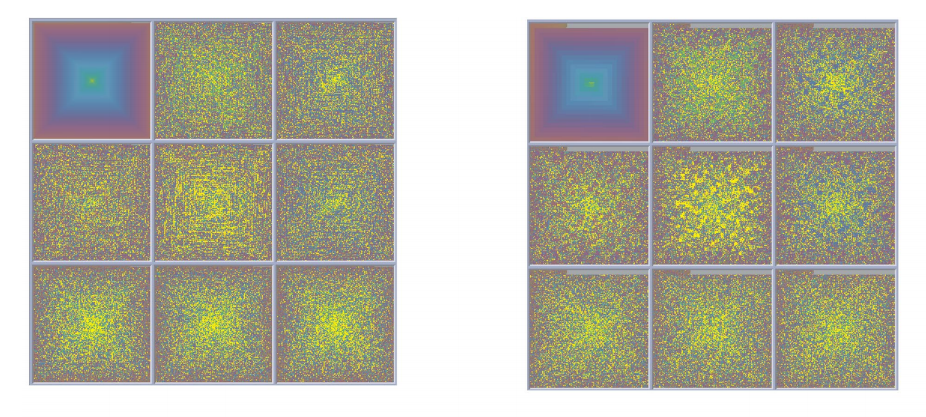
Стоит отметить, что вариант с простой спиральной компоновкой (на рис. Х слева) недостаточно хорошо выполняет требование, сформулированное в предыдущем пункте – а именно, локальность объектов. Данные, которые в одномерном пространстве находились рядом, на графике могут располагаться далеко друг от друга. Поэтому решением может являться опять же использование кривых Пеано. 

Рисунок 2.6. Сравнение обходов по спиральной кривой (слева) и по кривой Гильберта (справа)

Как видно на рисунке Х, кластеризация заметна более хорошо в случае использования заполняющих пространство кривых.

Стоит отметить, что в использовании кривых Пеано заключается следующий минус: для того, чтобы полностью заполнить квадрат, длина его стороны должна являться степенью числа 2. Это значит, что в плохих случаях (когда количество данных немного больше степени числа 4) размеры окна становятся большими, но график в окне не заполняется полностью; либо не все объекты из набора будут визуализированы.

3. ЗАДАЧА ПРОЕКТА

* Реализовать рисование пиксельных графиков по входящим данным
* Реализовать отображение атрибутов, которые представляются дискретными или непрерывными величинами
* Предоставить пользователю выбор кривой, по которой будет проходить компоновка пикселей

Анализируя поставленные задачи, был сделан выбор в пользу языка программирования Python с библиотеками PIL и pandas. Также были использованы библиотеки geolite2 и hilbert\_curve.

4. РЕАЛИЗАЦИЯ ВИЗУАЛИЗАЦИИ ПИКСЕЛЬНОГО ГРАФИКА

Определим свойства реализуемой визуализации:

* Цвета для отображения значений будет зависеть от двух цветов, которые задает пользователь, – начального и конечного.
  + В случае, когда атрибут представляет собой дискретную величину, разделим цветовой диапазон от начального до конечного цвета на количество уникальных элементов во входных данных. Далее каждому значению из уникальных сопоставим цвет из получившегося градиента.
  + Для непрерывных величин пользователю будет предлагаться ввести количество шагов, на которые будет разбит диапазон значений. Цветовой диапазон разбивается на такое же количество шагов, и эти два разбиения сопоставляются между собой. Таким образом, если рассматриваемая величина попадает в -ый шаг диапазона значений, она будет отображена -ым цветом в градиенте.
* Прямоугольная форма окна.
* Упорядочивание окон для измерений будем проводить по порядку поступления атрибутов данных.
* Горизонтальное расположение пикселей и расположение по кривой Гильберта. При горизонтальном расположении пользователь может задать размеры графика; при выборе кривой Гильберта – длину данных, которые нужно визуализировать.
* Будем рассматривать наборы данных с естественным порядком.
* Для реализации компоновки по кривой Гильберта используется готовая библиотека.

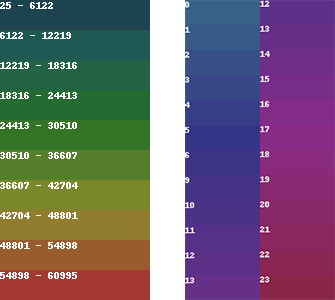


Рисунок 4.1. Пример цветовой легенды для непрерывной (слева) и дискретной (справа) величин

Покажем реализацию общей функции, которая используется для создания графика непрерывной и дискретной величин на рисунке 4.2.

## 

Рисунок 4.2. Общая функция для непрерывной и дискретной величин

Как можно видеть, параметрами в ней являются:

* список данных, отсортированных в нужном порядке;
* размер окна (для варианта горизонтальной компоновки);
* максимальный размер данных, которые будут отображаться;
* функция, которая определяет положение пикселя на графике по порядковому номеру в исходных данных; зависит от типа кривой компоновки;
* функция, которая определяет, каким образом сопоставить текущему элементу цвет из словаря цветов; зависит от типа величины;
* словарь цветов – список из цветов полученной ранее цветовой легенды.

Таким образом, для последующих дополнений функционала нужно будет указать только функцию компоновки пикселей на графике.

4.1 Пример визуализации спам-атак

Для визуализации был найден набор данных о спам-атаках – пакеты трафика, которые поступали от нескольких коллекторов Netflow v9 в сети испанского провайдера за март – июнь 2016 года.

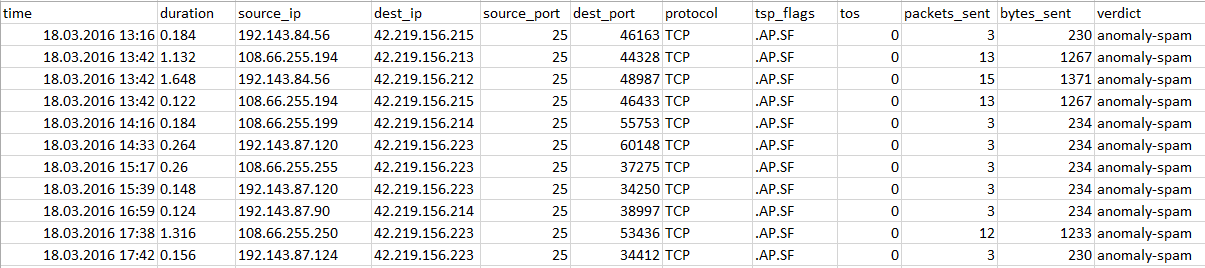


Рисунок 3.3. Часть входных данных о спам-атаках

Пример набора показан на рисунке 4.3, в котором присутствуют поля:

* Время начала потока данных
* Продолжительность потока данных
* IP-адрес отправителя и получателя
* Порт источника для UDP или TCP, 0 для других протоколов
* Порт назначения для UDP или TCP, тип и код для ICMP или 0 для других протоколов
* IP протокол
* Флаги, включенные при TCP протоколе
* Тип обслуживания
* Количество переданных пакетов и байтов
* Вердикт

Была создана выборка размером в элемента, состоящая из случайных записей из каждого месяца из указанного промежутка. Данные считаются упорядоченными по времени начала потока данных. Используется компоновка по кривой Гильберта.

1. Часы в течении дня, когда начинались потоки данных – рисунок 4.2. По графику можно сделать вывод, что в большинстве случаев атаки выполнялись либо в промежутке времени от 20 до 23 часов, либо в промежутке от 11 до 13 часов. Меньше всего атак происходило в районе полуночи. Кластеры в данном случае не несут в себе пользы, так как скорее всего они образовались из-за последовательных отправлений пакетов в одно и то же время. Справа на рисунке 4.4 представлена легенда для графика.

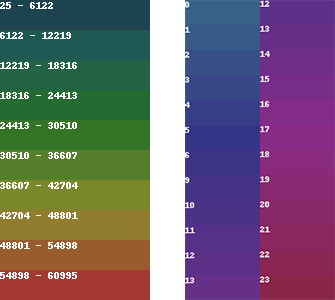
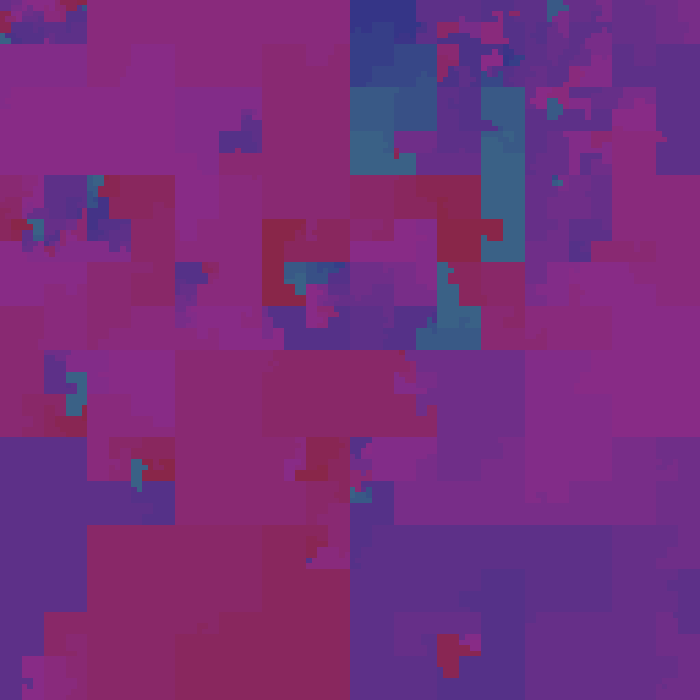


Рисунок 4.4. Визуализация часов начала потока данных

2. Представление стран отправителя и получателя представлена на рисунке 4.5. При помощи пакета ipwhois из колонок «source\_ip» и «dest\_ip» были получены коды стран, из которых проводилась атака. Для этих двух графиков был создана общая цветовая легенда (рис. 4.6) для лучшего восприятия.

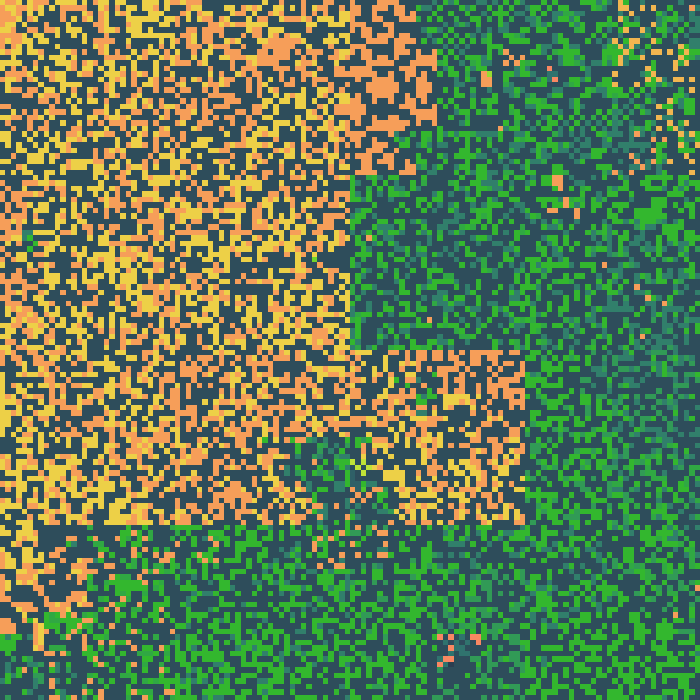
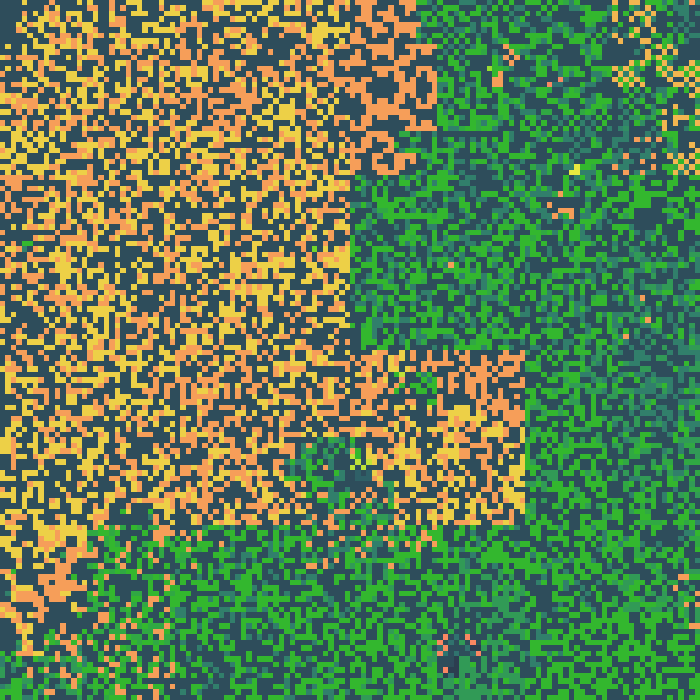
 

Рисунок 4.5. Визуализация стран: отправителя (слева) и получателя (справа)

Легко можно заметить, что графики очень похожи – это значит, что в большинстве случаев атака направлялась в ту же страну, откуда и исходила. Можно также отметить, что за весь промежуток времени стабильно атаковался (и атаковал) Китай, в первой половине набора много атак из Южной Африки и США, а во второй их заменили Корея и Германия.

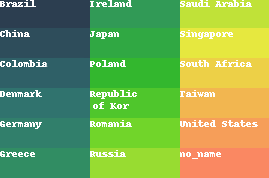


Рисунок 4.6. Цветовая легенда для стран отправителей и получателей

4. Продолжительность потоков данных представлена на рисунке 4.7. В этом пункте данные представляют собой непрерывную величину. Было выбрано число шагов градиента, равное 50. Даже при таком большом количество шагов для относительно маленького диапазона (от 0 до 340 мс) почти весь график заполнен одним цветом, что означает преобладание низкой продолжительности потока. Существует только небольшой участок, где и достигается максимальная продолжительность потока данных.

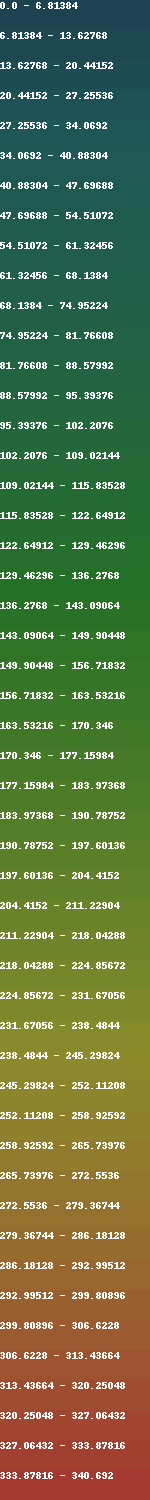
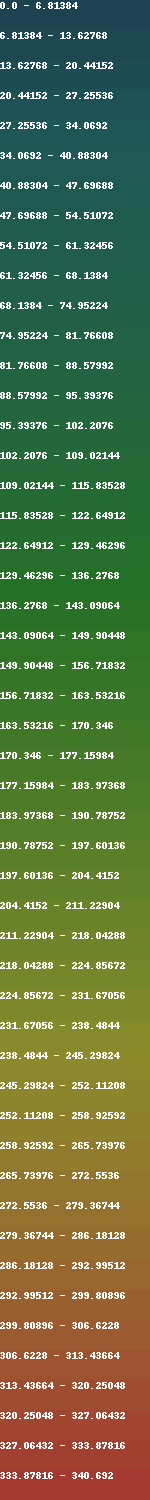
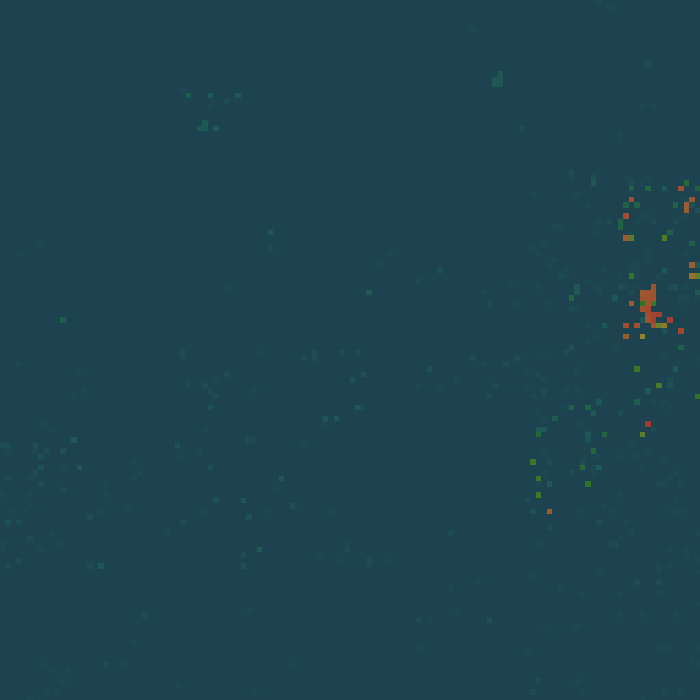


Рисунок 4.7. Визуализация продолжительности потока данных

5. Использование флагов при TСP/IP протоколе. Как было вычислено, все из представленных образцов использовали TСP протокол при передаче данных. Теперь можно посмотреть на флаги, которые выставлялись при этих передачах, используя график, представленный на рисунке 4.8. Но прежде ознакомимся с ними:

* URG — поле «Указатель важности» задействовано;
* ACK — поле «Номер подтверждения» задействовано ;
* PSH —инструктирует получателя протолкнуть данные, накопившиеся в приёмном буфере, в приложение пользователя ;
* RST — оборвать соединения, сбросить буфер (очистка буфера);
* SYN — синхронизация номеров последовательности;
* FIN — флаг, будучи установлен, указывает на завершение соединения.

На графике можем заметить такое же разделение, как и на предыдущих, что значит, что для каждой половины набора данных характерны определенные свойства, отличающиеся от другой половины. Вверху заметен особенный участок, в котором просматривается узор – скорее всего, эти пакеты посылались почти в одно время и этот промежуток времени является одной продолжительной атакой.

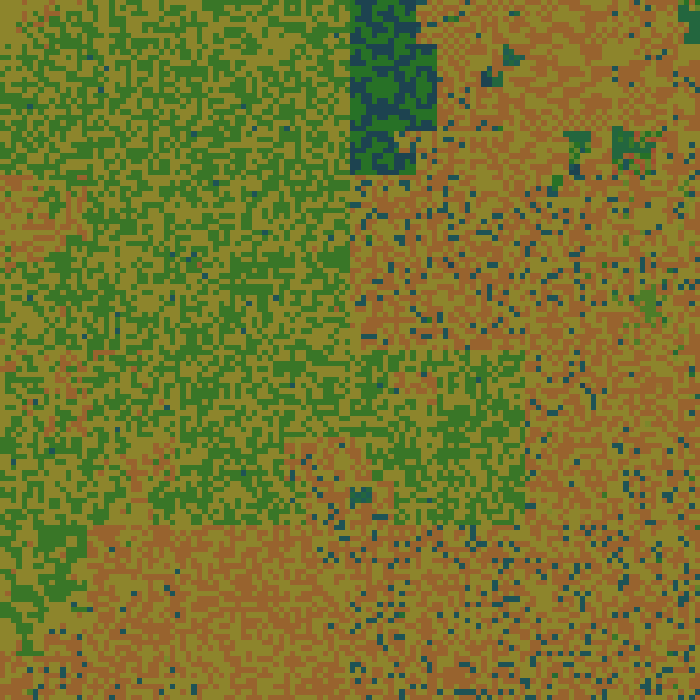


Рисунок 4.8. Визуализация установленных флагов при TСP соединении

Теперь составим пиксельный график в таком виде, каким он является по определению – состоящим из окон, каждое из которых соответствует одному свойству. Пять первых окон отображают выше указанные атрибуты, шестое – количество отправленных пакетов. Получившийся результат можно увидеть на рисунке 4.9.

Сравнивая окна между собой, можно найти некоторые корреляции между, например, первыми тремя, а также между четвертым и шестым.

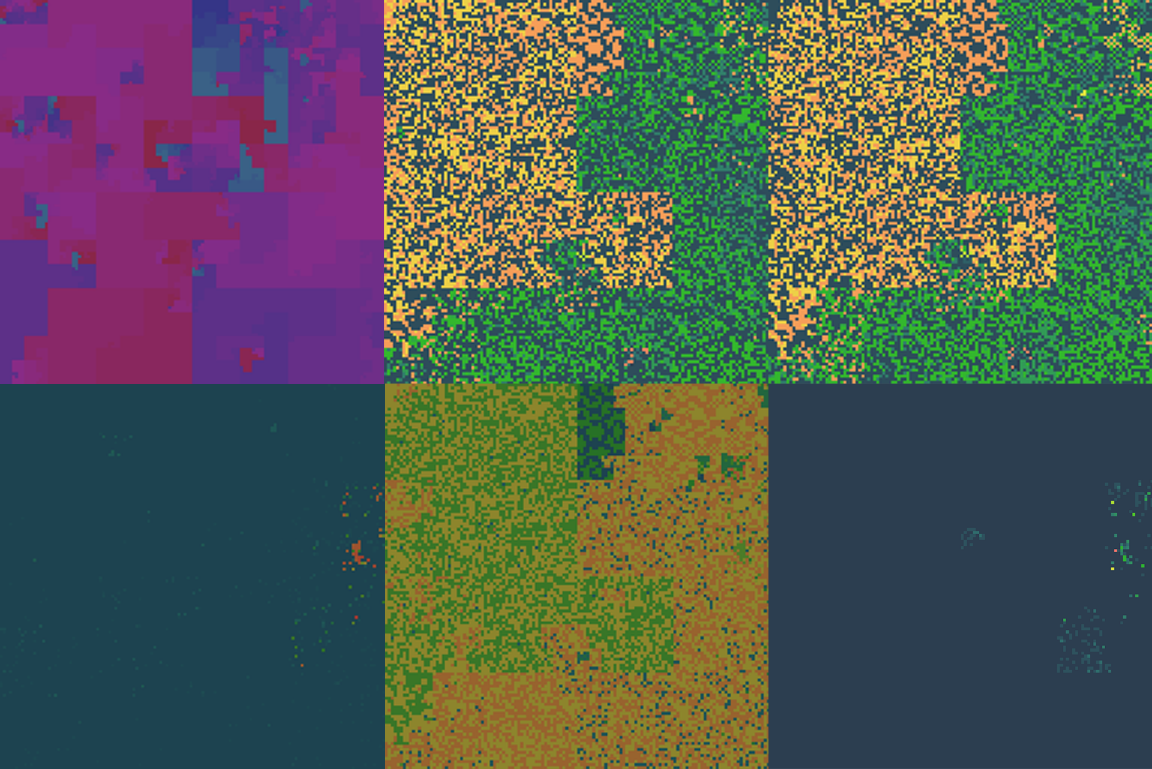


Рисунок 4.9. Пиксельный график, построенный по набору данных о спам-атаках

4.2 Пример визуализации файлов вредоносных программ

Для визуализации исполняемых файлов вредоносного ПО был найден набор данных, содержащий в себе последовательность вызовов функций или API-вызовов при исполнении соответствующей программы (пример показан на рисунке 4.10). В данном наборе содержались как вредоносные программы, так и безвредные.

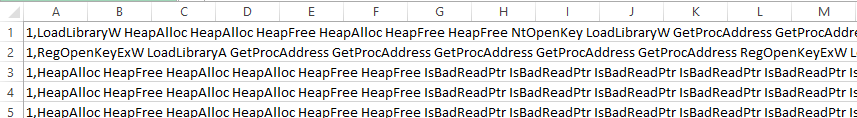


Рисунок 4.10. Образец входных данных

Для всех объектов из файла была построена общая цветовая легенда, чтобы визуализация была информативной. Так как уникальных названий вызовов очень много, градиент получился очень плавный, что мешает четкому разделению вызовов по цветам. Однако даже при этом условии визуализация дала интересные результаты.

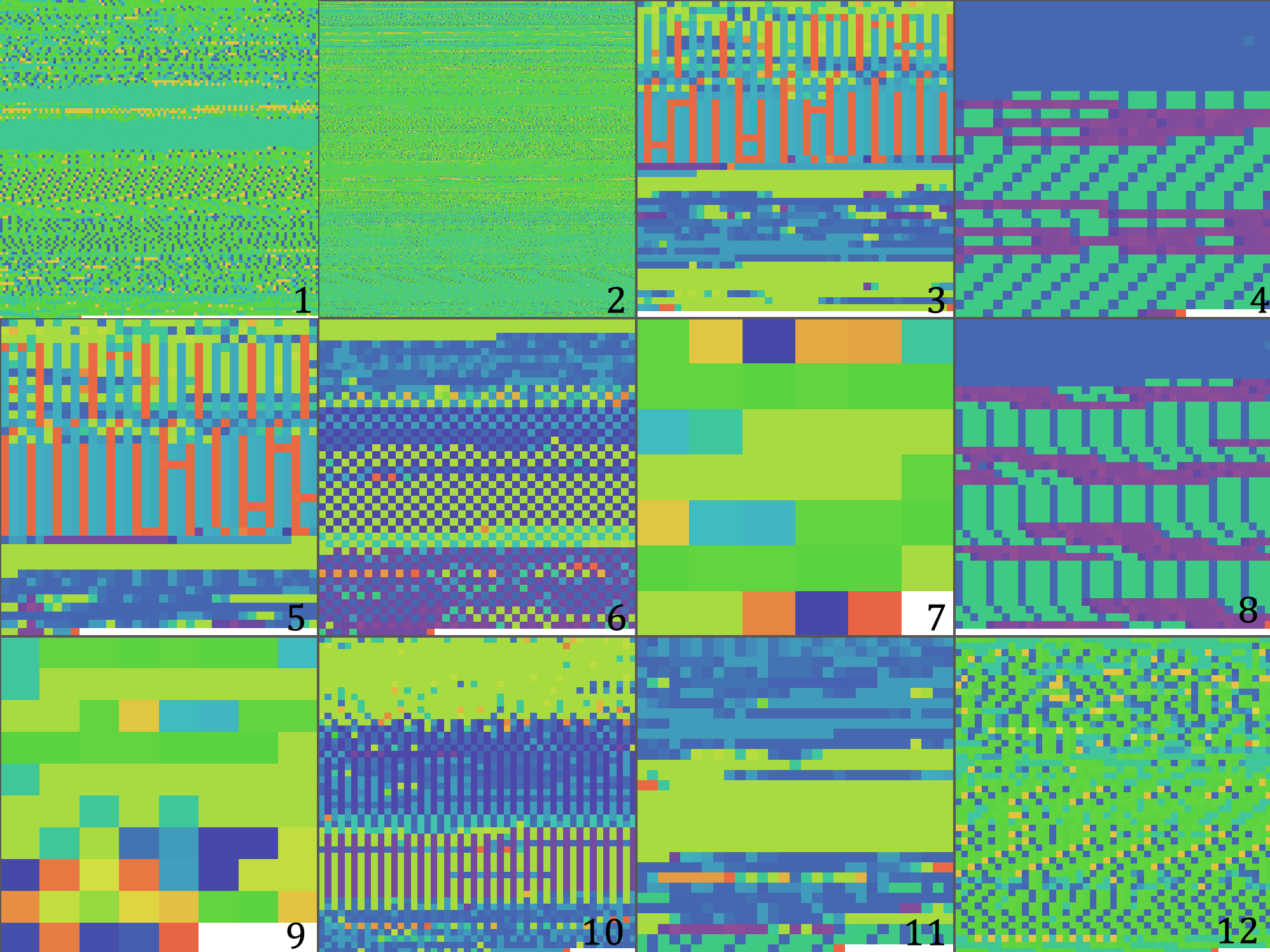


Рисунок 4.11. Результаты визуализации системных вызовов вредоносных программ

Для этого примера была использована компоновка пикселей горизонтальными линиями, чтобы показать и её пользу. К тому же, компоновку по кривой Гильберта использовать было бы неэффективно, потому что количество вызовов в каждом образце сильно отличается друг от друга, соответственно, получались графики значительно разных размеров и не заполненные до конца. Ширина окна для каждого образца выбиралась примерно равной квадратному корню из размера набора его вызовов (чтобы в итоге получался квадрат).

На рисунке 4.11 представлено несколько полученных образцов. Здесь можно увидеть пары очень похожих вредоносных программ – таких, как 3 и 5 или 4 и 8. Таким образом, без особых усилий можно найти семейства вредоносных программ, т.е. провести кластеризацию данных. При выборе фиксированной длины, можно было бы узнать, какие программы начинают своё выполнение одинаково, а какие нет.

Анализируя отдельный объект, можно легко понять, какие функции вызывались чаще всего и в каком порядке. Можно найти закономерности вызовов, а также составить общую картину действий программы.

Представленные два примера показывают, что пиксельный график может быть полезным при любом из типов анализа – индивидуальном, кластеризации или анализа большого набора данных – если правильно подобрать входные данные.

ЗАКЛЮЧЕНИЕ

В процессе реализации курсового проекта были изучены основы анализа данных о вредоносном ПО, различные техники визуализации данных, в частности класс пиксельных методов.

Реализована простейшая визуализация набора данных с компоновкой пикселей по горизонтальным прямым и по кривой Гильберта. Продемонстрированы примеры, созданные при помощи реализованной программы.

В дальнейшем планируется реализовать другие методы компоновки пикселей на графике, другие формы окон, подбор цветов для отображения в реальном времени. Так же планируется реализовать приложение по построению пиксельных графиков, добавить интерактивность графика – отклик на запросы.

СПИСОК ЛИТЕРАТУРЫ

1. M. Wagner, F. Fischer, R. Luh, A Survey of Visualization Systems for Malware Analysis, 2015
2. Daniel A. Keim, Designing Pixel-Oriented Visualization Techniques: Theory and Applications, 2000
3. Набор данных для анализа [Электронный ресурс] – 2017. – Режим доступа: <https://nesg.ugr.es/nesg-ugr16/> - Дата доступа 10.12.2017
4. Набор данных для анализа [Электронный ресурс] – 2017. – Режим доступа: [http://csmining.org/index.php/malicious-software-datasets- – Дата доступа 14.12.2017](http://csmining.org/index.php/malicious-software-datasets-%20–%20Дата%20доступа%2014.12.2017)