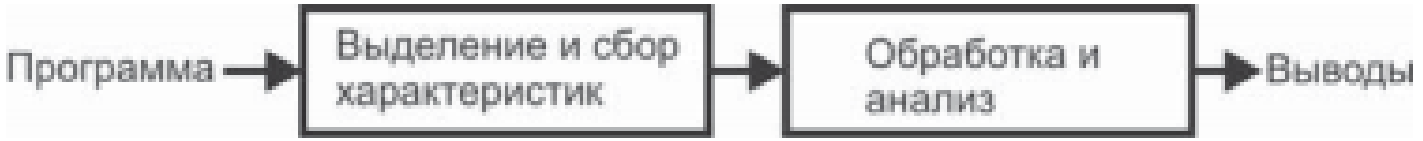
Обнаружение вирусов

Процедуры детектирования вирусов

Существуют различные типы антивирусных программ: сканеры, фаги, инспекторы, мониторы, вакцинаторы и прочие. Принцип действия большинства из них основан на детектировании (обнаружении) вирусов. Целью детектирования является разбиение всех программ, попавших в поле зрения антивируса, на два класса: «здоровые»; «больные», то есть либо зараженные вирусом, либо представляющие собой вирус per se. В общем случае процедуру детектирования вирусов можно разделить на следующие этапы:



Методы сбора характеристик о программе

Методы этапа, предназначенного для выделения и сбора характеристик «подозрительной» программы, целесообразно разделить на:

1) статические – оперирующие с двоичным образом программы на носителе информации или в памяти;

2) динамические – рассматривающие программу как процесс выполнения алгоритма, то есть как последовательную смену состояний.

В свою очередь, методы этапа, посвященного обработке данных и анализу характеристик, принято разделять на:

1) формальные – оперирующие фиксированной моделью вируса и использующие заранее определенные алгоритмы;

2) эвристические – пытающиеся воспроизвести процесс познания, присущий человеку.

Анализ косвенных признаков

Анализ косвенных признаков Наиболее примитивные методы обнаружения вирусов основаны на изучении косвенных признаков («слабых сигнатур»), характеризующих зараженность. Как правило, проверка наличия или отсутствия этих признаков возможна либо «на глазок», либо с применением штатных утилит операционной системы.

Кроме того, большое количество косвенных признаков можно обнаружить, сканируя код программы на характерные фрагменты. Например:

1) байт E9h в начале COM-файлов соответствует команде «JMP» и может свидетельствовать о зараженности ее «стандартным» методом;

2) цепочка байтов E80000h или E8000000h в начале программы может означать попытку вычислить «дельта-смещение»;

3) команда «MOV AH, 4Ch» (константа 4CB4h) помогает вирусу искать файлы в каталоге, а «CMP AX, ‘MZ’» (цепочка байтов 3Dh 4D4 5Ah) – различать COM- и EXE-программы;

4) константа 0EDB88320h часто используется при расчете CRC-32, а константы типа 553B5C78h или 0AE17EBEFh встречаются в вирусах, ищущих в таблицах экспорта адреса функций CreateFileA и FindFirstFileA по контрольным суммам их имен;

5) и т. п

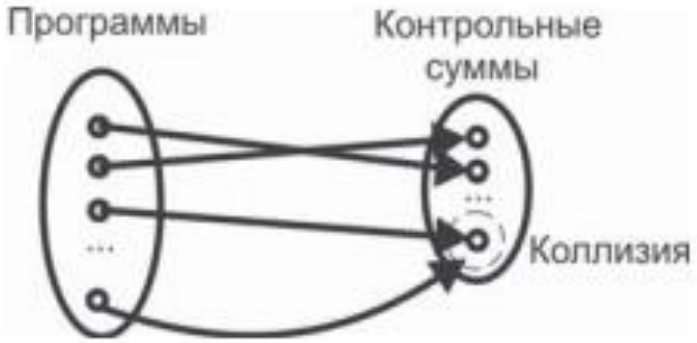
Простые сигнатуры

Наиболее популярным методом детектирования вредоносных программ является проверка сигнатур. Под сигнатурой понимается: фрагмент (или набор фрагментов), который всегда встречается в конкретном вирусе и никогда – в иных программах (в том числе и в других вирусах). Сигнатуры используются для детектирования вредоносных программ, сохраняющих свой код постоянным от копии к копии. Кроме того, сигнатурный поиск можно применить и для поиска некоторых разновидностей полиморфных вирусов. Единственная разновидность вирусов, к которой совсем не применимо сигнатурное детектирование, – это «метаморфы», то есть вирусы, использующие идеи замены и пермутации (перемешивания) команд для всего своего тела.

В идеале сигнатура должна включать в себя всю постоянную часть вируса, но на практике – для минимизации требуемой памяти и ускорения поиска в файле – используются сравнительно короткие цепочки байтов, состоящие из нескольких отрезков. В общем случае сигнатура S может быть представлена в виде множества троек: S = {Ci , Pi , Ti }, где i – номер отрезка сигнатуры (если она состоит из нескольких частей); Ci – значение отрезка; Pi – позиция отрезка; Ti – шаг трассировки, на котором необходимо контролировать отрезок. То есть три различных компонента сигнатуры отвечают на вопросы «что», «где» и «когда». Разумеется, в составе сигнатуры могут встречаться и другие компоненты вспомогательного назначения. В частности, это может быть признак: откуда отсчитывать смещение фрагмента – от начала файла, конца файла или точки входа в программу. Впрочем, этот признак тоже отвечает на вопрос «где».

Контрольные суммы

Существенно уменьшить объемы требуемой памяти позволяет отказ от сигнатур в пользу контрольных сумм. Контрольная сумма – результат применения к произвольному набору данных некой хешфункции, рассчитывающей короткий «дайджест» постоянной длины (например, всего 4 байта). В базе данных антивируса можно хранить не сами сигнатуры, а их короткие контрольные суммы. Такие же суммы рассчитываются «на лету» по содержимому тестируемых файлов. Несовпадение хранимого образца с результатом расчета означает отсутствие соответствующего вируса. А совпадение – лишь очень высокую (но не единичную!) вероятность заражения. Причина кроется в сути контрольных сумм: они, как и любые другие хеш-функции, представляют собой отображение большого множества «объектов»-прообразов на малое множество «дайджестов»-образов. Соответственно, всегда возможна коллизия: несколько различных «объектов» могут иметь одинаковые «дайджесты», в частности «плохие» контрольные суммы могут оказаться не только у зараженных, но и у вполне «здоровых» файлов. Невысокая вероятность коллизий и трудность их целенаправленного генерирования – суть критерии качества того или иного метода расчета контрольных сумм. С этой точки зрения хороши так называемые «криптографические» хеш-функции: MD5, SHA-1, RIPEMD, ГОСТ 34.11-94 и др.1 К сожалению, эти алгоритмы довольно сложны с вычислительной точки зрения и не способны обеспечить высокую скорость расчета контрольных сумм. Поэтому на практике в антивирусах нашли применение несколько менее стойкие к коллизиям, зато гораздо быстрее вычислимые «технические» хеш-функции.



Контрольные суммы

Чаще всего используются циклические избыточные коды (CRC – cyclic redundancy code). Метод использования основан на представлении блока данных в виде непрерывного полинома с битовыми коэффициентами. В качестве контрольного кода используется остаток от деления этого полинома на более короткий «порождающий» полином, имеющий длину N битов. Техника деления такова: 1) в конец блока данных добавляется N–1 нулевых битов; 2) вместо арифметического деления используется операция «сложение по модулю 2»; 3) сложение с «левыми» нулями промежуточных остатков не производится. Существуют как программные реализации, основанные непосредственно на делении «уголком» (пример на языке Ассемблера можно найти в главе, посвященной Win32-вирусам), так и оптимизированные по скорости – благодаря сдвигу сразу на 8 битов и использованию таблицы корректирующих коэффициентов. Вот пример реализации для «быстрого» вычисления CRC-32

```
1010000 | 1001
1001
  1100
  1001
   1010
   1001
     11  <- CRC
```

Вопросы эффективности

Считается, что одним из важнейших критериев качества современного антивируса является эффективность использования вычислительных ресурсов. Речь идет о быстродействии, требованиям к дисковой и оперативной памяти и т. п. На момент написания этих строк известны несколько миллионов разновидностей вредоносных программ, следовательно, антивирус в общем случае вынужден выполнять такое же количество детектирующих операций (например, сравнений сигнатур) по отношению к каждому файлу. Неудивительно, что антивирусный монитор способен вызывать многосекундные задержки при контроле доступа к файлам, а антивирусный сканер – затрачивать десятки часов на проверку диска. Дошло до того, что типичный пользователь при покупке выбирает не тот антивирус, который «знает больше» или «лечит лучше», а тот, который «работает быстрее». Надо сразу оговориться, что в современных условиях удовлетворительного решения проблемы, по-видимому, не существует. Современная антивирусная реализация – это клубок компромиссов: одни производители жертвуют надежностью детектирования, другие – быст родействием, третьи – количеством распознаваемых вредоносных программ, четвертые – возможностью лечения и т. п.

Выбор файловых позиций

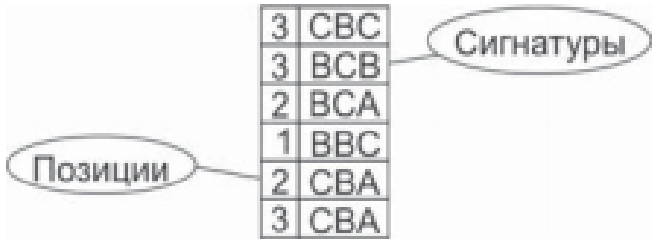
счет уменьшения размера первой, «файловой» базы данных. Речь идет о том, чтобы искать «сигнатуры» в файле не везде, а только в избранных позициях. К сожалению, применение «метода n-грамм» при выборе «хороших» сигнатур, по-видимому, приводит к необходимости поиска сигнатур по самым разнообразным смещениям в файле. В худшем случае все эти смещения окажутся различны, и антивирусу придется считывать данные из стольких разных позиций в файле, сколько записей в его справочной базе. Удачным компромиссом было бы использование для выбора (и поиска) сигнатур конечного числа файловых позиций, например всего трех: 1) начало файла (или тела вируса, или кодовой секции программы, или потока в документе); 2) конец; 3) центральная позиция между началом и концом. Другой вариант – введение дискретного ряда смещений: 0, , 2, …, где – некий интервал, который зависит от длины вируса V. Можно, например, положить = V/100, и тогда для любого вируса всегда иметь ровно Q = 100 различных позиций, из которых могут считываться цепочки байтов – потенциальные сигнатуры. Это означает, что при помощи «метода n-грамм» будет осуществляться выбор из небольшого количества сигнатур, расположенных в заранее определенных файловых позициях. Впрочем, слишком маленьким это число тоже не должно быть, иначе многие «похожие» вирусы (например, принадлежащие одному семейству) могут оказаться неразличимыми.

Выбор файловых позиций

Дальнейшие рассуждения проиллюстрируем на примере маленьких таблиц, моделирующих N = 6 различных вирусов. «Коды» этих вирусов формируются всего из трех различных байтов (символов): ‘A’, ‘B’ и ‘C’ . Записи, внесенные в антивирусную базу, выделены

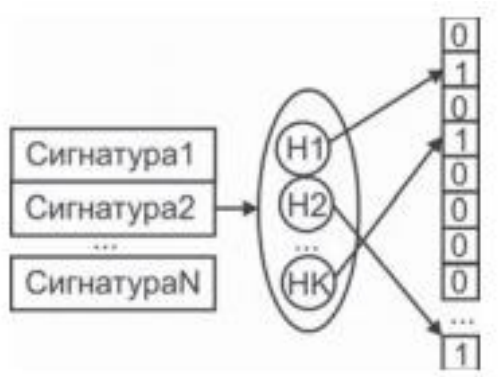
Текст слайда

Разумеется, проще всего сформировать «антивирусную» базу, разместив в ней записи в том порядке, в каком изначально были перечислены вирусы т слайда



Фильтр Блума

Это метод хеширования, позволяющий сразу отбрасывать записи, отсутствующие в вирусной базе данных. Идея заключается в том, чтобы вместе с базой данных, содержащей N записей о вирусах, хранить «битовую карту» – массив из M первоначально обнуленных битов. При добавлении к базе новой сигнатуры для нее рассчитываются K различных хеш-функций, и в «карте» устанавливаются в «1» K битов1 . Индексами (адресами в карте) для этих битов являются значения рассчитанных хешей. В итоге после завершения создания базы «карта» оказывается заполнена перемешанными значениями «0» и «1».



Метод половинного деления

Существенно ускорить поиск позволяет «дихотомия» («половинное деление») лексикографически упорядоченной таблицы с пронумерованными по порядку записями (. Если известно, что аргумент поиска меньше среднего ключа в таблице, то можно не проверять элементы в диапазоне от этого среднего и до конца таблицы. Исходная таблица окажется разделена на две примерно равные части, в одной из которых заведомо находится искомая запись. Эту «половину» можно вновь разделить пополам по средней записи и т. д., пока либо искомая запись не будет обнаружена, либо не будет доказано ее отсутствие.

Разбиение на страницы

Это концепция, предусматривающая разделение таблицы по какому-либо признаку на независимые части («страницы») и поиск только в некоторых из них. Например, наиболее естественно разделить таблицу вирусной базы данных на отдельные страницы в зависимости от значения поля «позиция» или комбинации «позиция+начало_сигнатуры». Внутри страницы записи упорядочены лексикографически по значению поля «сигнатура». Такой подход позволит считывать целиком всю страницу в память и искать в ней нужную запись методом половинного деления, избежав многочисленных обращений к носителю. Кроме того, на компьютере с несколькими процессорами (или ядрами в процессоре) можно будет считывать в память сразу несколько страниц и выполнять поиск параллельно. Для ускорения доступа к базам данных, разбитым на страницы, обычно используют «индексные таблицы», содержащие номера страниц и их адреса на носителе