УДК 519.713

МОДЕЛЬ НЕИСПРАВНОСТЕЙ ПРОГРАММНОГО ПРОДУКТА. КОМПЬЮТЕРНЫЙ ВИРУС

ХАХАНОВ В.И., ФРАДКОВ С.А., ХАНЬКО В.В.

Наука только тогда
достигает совершенства,
когда ей удается
пользоваться математикой

К. Маркс

Представлены результаты исследований, касающиеся проблемы взаимодействия программного продукта (ПП) и неисправностей, которые оказывают негативное действие на ожидаемые результаты, получаемые в ходе исполнения программы. Приведена классификация типов неисправностей по способу возникновения, среде обитания, воздействию на информационные носители. Предложены алгоритмы поиска и устранения вирусов, которые могут быть полезны широкому кругу пользователей программных средств.

1. Актуальность

Компьютерный бум и развитие глобальных сетей послужили катализатором информатизации всех сфер человеческой деятельности. Многообразие и доступность применения программных продуктов породило проблему надежности программного обеспечения компьютерных систем (КС). Как в свое время акцент на проектирование функциональной части цифровой техники привел к несбалансированному отношению между изделиями и средствами диагностического обслуживания, так и сейчас ориентация только на создание функциональных компонентов программных продуктов при дефиците исследований, касающихся их сервисного обслуживания, практически привела к дефициту методов и средств профилактического обслуживания и диагностирования программных продуктов. Если такая наука как техническая диагностика за сорок лет достигла определенных результатов в тестопригодном проектировании, сервисном обслуживании и восстановлении работоспособности цифровой аппаратуры, то в отношении ее программного обеспечения такие задачи в целях их комплексного решения еще не ставились. Но возможно их постановка не актуальна и преждевременна? Однако практика эксплуатации программных продуктов показывает, что:

1. Отсутствие в стране электронной индустрии вынуждает использовать только импортные компьютерные системы, для которых даже диагностирование аппаратуры осуществляется лишь посредством сервисных тест-программ. Поэтому активные исследования, чтобы быть практически направленными, должны ориентироваться на совершенствование сервисного программного обеспечения.

2. Повсеместное несанкционированное пользование лицензионными продуктами (пакетами) предполагает дальнейшее совершенствование методов и средств защиты от "хакеров".

3. Случайное или преднамеренное "заражение" компьютерными "вирусами" (далее без кавычек) обязывает пользователей и разработчиков программ принимать дополнительные усилия в целях защиты КС от вирусов и их последующего устранения. Вирусный модуль может портить информацию, но может выполнять и более сложные целенаправленные функции, например, по поиску и передаче конкретной информации.

4. Разработчик может вставить программный модуль, модифицирующий результаты работы пакета не в интересах основного пользователя.

5. На стадии проектирования могут быть допущены непреднамеренные ошибки, нарушающие выполнение основных функций.

Подтверждением этому служат известные разработчикам и пользователям аксиомы:

1. Программа не может быть без ошибок.

2. Если программа не имеет ошибок, то неполным является тест. (Кратные ошибки могут приводить к правильному результату.)

3. Исправление ошибок приводит к появлению новых.

Упомянутые аксиомы являются весомозначным аргументом в пользу выполнения исследований, направленных на анализ и синтез пары: программа – неисправность. Для этого необходимо взглянуть на проблему прежде всего с позиции математики, чтобы построить модель взаимодействия данной пары. "Модель, с которой оперирует каждая наука, может быть разделена на информационную часть и исчисление. Информационная включает классифицированный фактический материал. Исчисление есть совокупность правил вывода, позволяющих получать дедуктивным путем те или иные следствия из заданных основных научных положений. По мере развития науки растет удельный вес исчисленческой части модели, а значит увеличиваются возможности для математизации науки" [1, с. 43].

2. Модель объекта исследования

Согласно цитате установим основные понятия и определения в целях классификации существующих программных неисправностей и последующего проектирования их модели. Следует заметить, что современная технология изготовления интегральных схем оперирует миллионами вентилей (транзисторов) на кристалле. Поэтому сложность аппаратуры и программных средств становится соизмеримой друг с другом. Отсюда следует методологический вывод об одинаковой размерности задач технической диагностики, подлежащих решению на стадиях проектирования, производства и эксплуатации аппаратурных и программных средств КС.

Другой важный вывод касается применимости стандартизированных понятий и определений, моделей, методов и алгоритмов уже достаточно развитой аппаратурной диагностики к верификации и диагностированию ПП.

Общей концепцией модели программного или аппаратурного изделия (объекта) можно считать конечный абстрактный автомат первого рода, определяемый аналитическим выражением

$$W = \langle X, Y, Z, f, g, \rangle,$$

где X, Y, Z – множества входных, выходных и внутренних состояний автомата; f, g – характеристические функции переходов и выходов, определяю-

щие импликативные отношения на указанных множествах:

$$Z(t)=f[X(t),Z(t-1)];\ Y(t)=g[X(t),Z(t-1)].$$

Объект исследования представлен взаимодействующими компонентами: программой P_i и неисправностью F_i.

Модель – структура компонентов и/или процедур, с определенной степенью адекватности представляющая объект или процесс [2].

Модель неисправностей для объекта P_i представлена конечным множеством дефектов

$$F_i=\{F_{i1}, F_{i2}, \ldots, F_{ij}, \ldots, F_{ik}\}.$$

Программа – совокупность исполняемых команд языков программирования, в пределе ассемблерных; предназначена для ввода, хранения, преобразования, передачи и вывода информации.

Программный продукт – совокупность программ, информационных баз для ввода, хранения, вывода информации; предназначен для минимизации временных и/или материальных затрат при решении конкретных задач.

Программные продукты принято классифицировать по принадлежности к операционным системам и сервисным программам; инструментальным языкам и системам программирования; прикладным системам.

Дефект $F = \{F_p, F_n, F_o\}$ – каждое отдельное несоответствие программного продукта (изделия) требованиям нормативной документации.

Повреждение (F_p) – вид дефекта, определяемый событием, заключающимся в нарушении исправного состояния при сохранении работоспособного.

Несущественный дефект (F_n) – событие, заключающееся в нарушении работоспособного состояния при сохранении состояния правильного функционирования.

Отказ (F_o) – событие, заключающееся в нарушении состояния правильного функционирования.

Рейтинг дефектов по деструктивным возможностям определяется соотношением ($F_p < F_n < F_o$).

Техническое состояние объекта (программного продукта) – совокупность исправного и множества наперед заданных неисправных состояний: $S = \{S_i, S_n, = \{S_r, S_f, S_{nf}, S_p\}\}$.

Исправным (S_i) называется состояние объекта, при котором он соответствует всем требованиям нормативно-технической документации.

Неисправным (S_n) называется состояние объекта при наличии в нем дефекта.

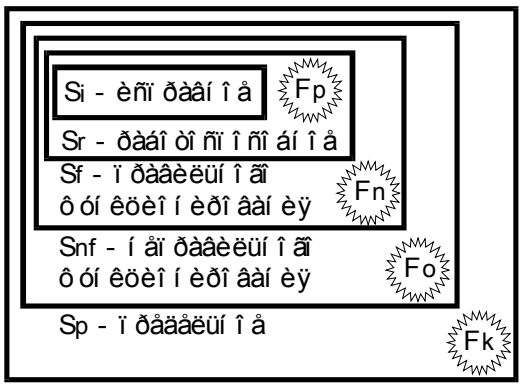
Работоспособным (S_r) называется состояние объекта, при котором он может выполнять свои функции при наличии повреждения.

Состояние объекта, при котором он может выполнять свои функции при наличии несущественного дефекта, называется состоянием правильного функционирования (S_f).

Состояние неправильного функционирования (S_{nf}) определяется объектом, имеющим отказ.

Если объект имеет катастрофические отказы (F_k) или морально устарел на данный момент времени, его техническое состояние определяется как предельное (S_p) или не подлежащее техническому обслуживанию.

Неисправное техническое состояние при наличии дефекта, устранение которого приведет к восстановлению состояния, правильного функционирования, работоспособного или исправного, называется непредельным $S_{np} = \{S_r, S_f, S_{nf}\}$.



Ðèñ. 1. Îòíîøåíèå ñîñòîÿíèé è äåôåêòîâ

Взаимодействие дефектов и состояний объекта представлено на рис. 1.

Тест – вход-выходная последовательность, предназначенная для установления соответствия состояния изделия заданным техническим состояниям:

$$T=||\ T^X_{ij}, T^Y_{rs}\ ||,\ i=1,n;\ j=1,m;\ r=n+1,n';\ s=m+1,m'.$$

Дефект F_r обнаруживается тестом T, если его присутствие искажает хотя бы одно выходное слово фактической реакции объекта R^Y_{ij} по отношению к его исправному состоянию:

$$\exists_{ij}\ (T^Y_{ij} \cap R^Y_{ij}\ |_{Fr}) = \varnothing. \qquad (1)$$

Тип дефекта определяет вид технического состояния [3] программы, которое может быть: исправным – неисправным, работоспособным – неработоспособным, правильного – неправильного функционирования. Далее не рассматриваются первые две пары состояний, поскольку они определяются дефектами, несущественными для правильного выполнения алгоритма функционирования, называемыми повреждениями. Последние, например, могут исказить эргономические свойства окна ввода-вывода информации. Неисправность часто отождествляется с понятием дефекта.

Функционирование – выполнение предписанного программе алгоритма функционирования при ее применении по назначению.

Избыточность – часть объекта (аппаратурного или программного), не имеющая отношения к реализации алгоритма функционирования.

Функциональная часть – компоненты объекта, существенные для реализации алгоритма функционирования.

Существенность – свойство компонента P_i, когда его модификация приводит к изменению алгоритма функционирования объекта, приводящего к результату (1) по выходным состояниям:

$$dP/dP_i=1. \qquad (2)$$

Компонент, обладающий упомянутым свойством, называется существенным.

Выражение (2) задает булеву производную функционала P по элементу P_i, которая в общем случае определяется выражением

$$dP(x)/d(x_i)=P(x_1, \ldots, x_i, \ldots, x_n) \oplus P(x_1, \ldots, \bar{x}_i, \ldots, x_n),$$

где x_i – представлены в алфавите {0,1}.

Если алфавит описания переменных задан теоретико-множественными примитивами, определение производной будет иметь вид

$dP(x)/d(x_i)=P(x_1, ..., x_i, ..., x_n) \cap P(x_1, ..., \bar{x}_i, ..., x_n)$.

Избыточность является несущественной, если возникновение в ней дефекта, не выводящего объект за пределы определения конечного автомата, задаваемого ею, не изменяет алгоритма функционирования объекта. Такой дефект будем называть константным.

Константные дефекты в несущественной избыточности не могут быть обнаружены тестом проверки исправности алгоритма функционирования [2].

В функциональной части все константные дефекты проверяются тестом проверки исправности.

Несущественная избыточность не может быть обнаружена тестом проверки алгоритма функционирования.

Избыточность, внесенная в объект преднамеренно для выполнения алгоритма, модифицирующего функционирование в том числе и других объектов, называется деструктивной.

Деструктивная избыточность, закладываемая в исполняемую программу при ее проектировании с целью съема, преобразования, передачи информации в КС при ее запуске по кодовой комбинации, называется защищенной.

Вирус – функционально избыточный фрагмент исполняемой программы, способный автоматически распространять и/или модифицировать свои копии в компьютерной системе или в сети с целью деструктивного съема, преобразования, передачи информации.

Компьютерный вирус можно рассматривать как незащищенную деструктивную избыточность.

(Первые исследования саморазмножающихся искусственных конструкций упоминаются в работах Д. Неймана, Н. Винера. Фред Коэн (Fred Cohen считается автором термина "компьютерный вирус", который введен им в 1984 году).

По отношению к КС избыточность может выступать как дефект, нарушающий функционирование и/или работоспособность, и/или исправность системы, что служит методологической основой возможности тестирования КС в целях поиска и устранения прежде всего деструктивной избыточности.

Невозможность обнаружения новой деструктивной избыточности ассоциируется с еще не определенными дефектами, для которых принципиально нельзя построить тест, если они не влияют на функционирование компонента КС.

Но выход всегда есть. В данном случае предлагается *комплексное тестирование и анализ воздействия избыточности на КС* в целом, которое за достаточно длительный промежуток времени должно проявиться хотя бы в одном компоненте системы.

Например, тестирование зараженной новым вирусом программы может дать отрицательный результат, если она модифицирует данные в другом файле. Тем не менее можно и нужно определять его наличие, а затем вид и местоположение на основе анализа работы программных средств и сервисных тест-программ. Существующие тестеры, как правило, ориентированы на анализ каждого EXE-модуля в отдельности.

3. Классификация компьютерных вирусов

Проблема верификации ПП заключается в невозможности за приемлемое время качественно решить задачу по установлению факта наличия или отсутствия программных механизмов, нарушающих функционирование компьютерной системы в целом, в соответствии с требованиями нормативно-технической документации.

Практическое решение упомянутой проблемы существует. Оно связано с установлением факта наличия или отсутствия списка наперед заданных дефектов с определенными функциями, которые имеют внешнее проявление в модификации информационных массивов.

Решение проблемы основывается на рассмотрении следующих задач.

1. Информационная классификация существующих вирусов.

2. Создание существенных общих и специфических признаков модели вируса.

3. Проектирование тестов для отдельных существенных компонентов вирусов.

4. Создание общих и специализированных процедур поиска и устранения вирусов, которые могут быть условными, когда принятие окончательного решения принадлежит пользователю, и безусловными – в этом случае алгоритм диагностирования осуществляет выбор стратегии поиска и самостоятельного устранения вируса.

Ниже представлены результаты анализа источников, имеющие целью классификацию программных механизмов-вирусов, искажающих результаты работы ПП.

Дыра (Loophole) - программные упущения или недоработки, позволяющие обойти процессы управления доступом. Одну из дыр содержит подпрограмма ввода и проверки корректности пользовательского, устанавливаемого в Setup, пароля. Его максимальная длина – 8 любых символов. Здравый смысл подсказывает, что если при установленном пароле "aaaa" ввести "auwxao", система запретит доступ. Однако система воспримет как корректные пароли "aaaa", "atyzww", "atyzxs", "atzwts" и, в том числе, "auwxao". Award BIOS, установленный на компьютерах с процессорами i486 и Pentium, хранит в CMOS не пароли, а их двухбайтовые контрольные суммы, алгоритм формирования которых демонстрирует подпрограмма:

```
unsigned PassCRC(char *str)
{unsigned crc=0, i=0;
   while(str[i] && (i<8))
   {crc=_rotl(crc,2)+(unsigned)str[i];i++;}
return crc; }
```

Если злоумышленник может загрузить компьютер и запустить программу, которая считает из CMOS контрольную сумму (ячейки 1Dh-1Ch), то сгенерировать один из возможных паролей – тривиальная задача. Поэтому по степени защищенности упомянутую систему следует считать открытой.

Если компьютер невозможно загрузить без ввода пароля, следует воспользоваться люком.

Люк (Trap door) – скрытый программный механизм, позволяющий обойти защиту системы. В случае с Award BIOS это означает, что наряду с подпрограммой проверки корректности Supervisor password и User password, устанавливаемых в Setup

пользователем, имеется также подпрограмма, обрабатывающая "инженерный" пароль. Для Award BIOS v4.51PG (08/09/96) им является "AWARD_SW". Учитывая, что инженерные пароли хранятся в BIOS в виде контрольных сумм, пароли "j322","j262" и "fddzid", дающие прошитую контрольную сумму (в данном случае 1EAAh), также будут инженерными.

Троянский конь (Trojan horse) – программа или программный механизм, имитирующие выполнение или реально выполняющие некоторую полезную функцию, но в то же время выполняющие и некоторые дополнительные (скрытые) функции (например, позволяющие обойти систему защиты путем скрытого использования законной авторизации вызывающего процесса). Под определение попадает программа, зараженная вирусом. Поэтому поиск троянских программ распадается на поиск собственно троянских коней и вирусов.

Вирусы можно классифицировать [4] по:
- среде обитания;
- способу заражения среды;
- деструктивным возможностям;
- особенностям алгоритма;
- уровню мастерства автора.

По среде обитания вирусы можно разделить на сетевые; файловые, внедряемые в исполняемые файлы; загрузочные, находящиеся в Boot-секторе или в секторе системного загрузчика винчестера MBR; макровирусы, находящиеся в документах и шаблонах Word, Excel. Существуют и сочетания упомянутых вирусов, например, файлово-загрузочные, заражающие как файлы, так и загрузочные секторы дисков, использующие для проникновения "стелс-" и "призрак"-технологии.

По способу заражения вирусы классифицируются на резидентные и нерезидентные. Первые (структура запускающей и резидентной части представлена на рис. 2 и 3 соответственно) при инфицировании компьютера оставляют в оперативной памяти резидентную часть, которая перехватывает обращения операционной системы к объектам заражения и внедряется в них. Резидентные вирусы находятся в памяти и являются активными вплоть до выключения или перезагрузки компьютера. Вторые (рис. 4) не заражают память компьютера и являются активными ограниченное время.

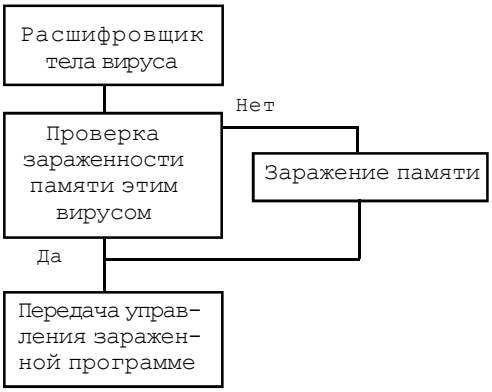


Рис. 2. Структура запускающей части резидентного полиморфного вируса

По деструктивным возможностям вирусы можно разделить на: безвредные, не влияющие на работу компьютера, кроме уменьшения памяти на диске для его хранения; неопасные, влияние которых ограничивается уменьшением свободной памяти на диске и аудио-визуальными эффектами; опасные, которые могут привести к сбоям в работе компьютера; очень опасные, приводящие к потере программ, уничтожению информации и системных данных.

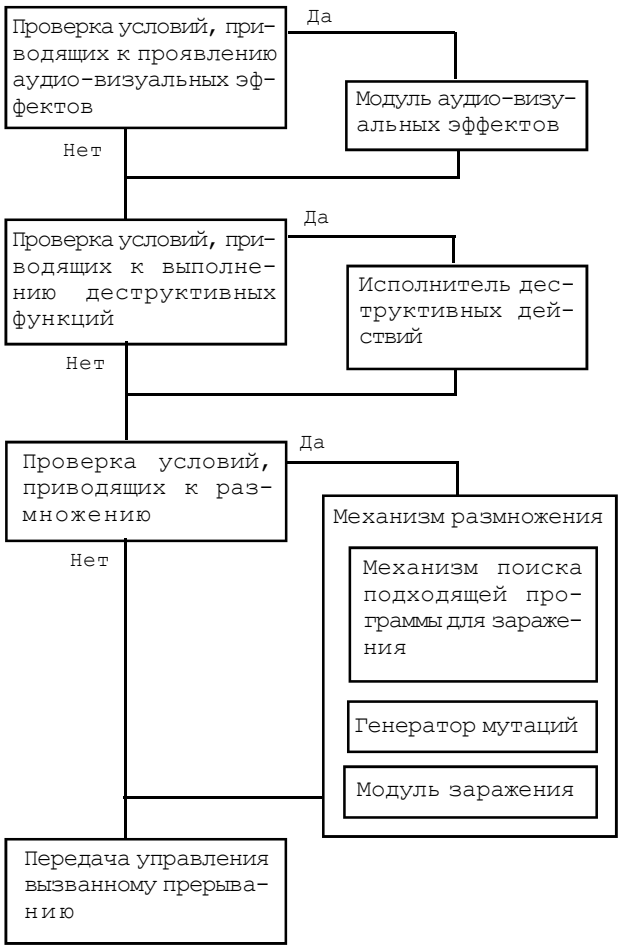


Рис. 3. Структура резидентной части полиморфного вируса (обработчик прерывания)

По особенностям реализации алгоритма можно выделить группы следующих вирусов:

1. "Спутники" (companion), не изменяющие файлы. Алгоритм функционирования заключается в создании для EXE-модулей сопутствующих файлов, имеющих то же самое имя, но с расширением .COM; например, для DISKEDIT.EXE создается файл DISKEDIT.COM. При запуске такого файла MS DOS первым обнаружит и выполнит COM-файл-вирус, который затем запустит и EXE-файл.

2. "Черви" (worm), распространяющиеся в сети и не изменяющие файлы или секторы на дисках. Проникая в память компьютера, они вычисляют сетевые адреса других КС и рассылают по ним свои копии. Такие вирусы иногда создают рабочие файлы на дисках системы, но могут вообще не обращаться к ресурсам компьютера, за исключением оперативной памяти.

3. "Паразитические", изменяющие содержимое дисковых секторов или файлов при распространении

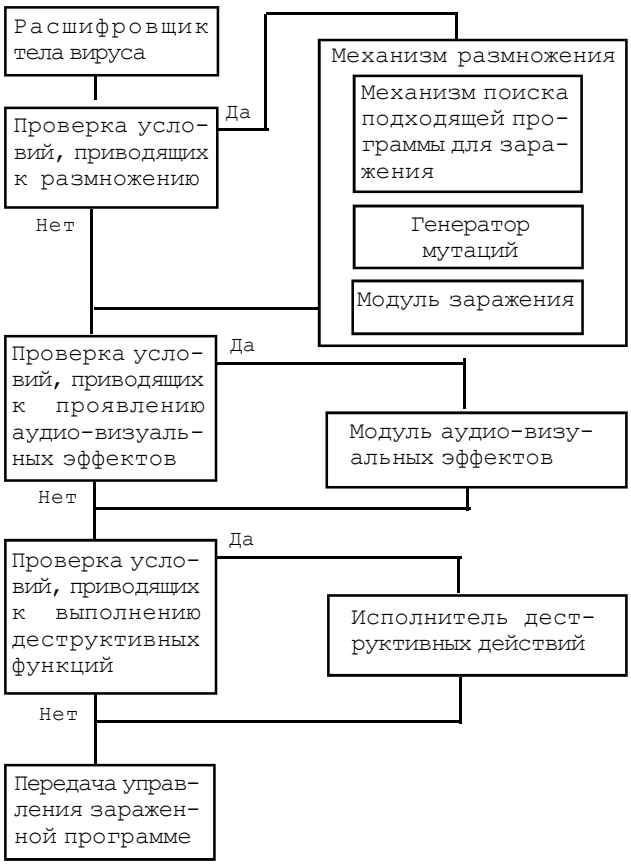


Рис. 4. Структура файлового нерезидентного полиморфного вируса

копий. К ним относятся вирусы, которые не являются "червями" или "спутниками".

4. "Студенческие" — примитивные вирусы, часто нерезидентные и содержащие большое количество ошибок.

5. "Стелс" – вирусы-невидимки, представляющие совершенные программы, перехватывающие обращения DOS к пораженным файлам или секторам дисков и подставляющие вместо себя незараженные участки информации. Такие вирусы при обращении к файлам используют оригинальные алгоритмы, позволяющие обманывать резидентные антивирусные мониторы.

6. Вирусы-"призраки" (полиморфные) — трудно обнаруживаемые вирусы, не имеющие сигнатур для идентификации – не содержащие ни одного постоянного участка программы. Это достигается шифрованием основного тела вируса и модификациями программы-расшифровщика.

4. Алгоритм функционирования файлового вируса

Исполняемые двоичные файлы имеют форматы COM или EXE, отличающиеся заголовками и способом запуска программы на выполнение. Расширение <.COM> или <.EXE> не всегда соответствует действительному формату файла, что не влияет на работу программы. COM- и EXE-файлы заражаются поразному, а значит вирус должен различать упомянутые файлы, делая это по расширению или по имени.

Как правило, вирус инфицирует файл корректно. В этом случае по информации, содержащейся в теле вируса, можно восстановить зараженный файл. При наличии программистских ошибок в теле вируса он может стать непредсказуемым по деструктивным последствиям. Ниже рассматриваются только EXE-файлы, как наиболее распространенные. При заражении такого модуля файл может содержать или не содержать внутренних оверлеев.

Зараженный вирусом не EXE-файл не может быть исполнен, но информация в файле может быть испорчена. Для обеспечения корректной совместной работы вируса и основного модуля зараженной программы вирусу необходимо: дописать свое тело в конец или в середину, в случае внутренних оверлеев, заражаемого EXE-файла; в заголовке (header) программы исправить поля, в том числе точку входа CS:IP, так, чтобы она указывала на адрес первой команды вируса, после исполнения всех команд которого обеспечить возврат в старую точку входа программы.

Ниже представлена структура заголовка EXE-файла на языке C++, необходимая для понимания излагаемого материала:

```
typedef struct {
unsigned Signature; // "MZ"-Марк Збиковски
unsigned LastPage;    // Последняя страница
unsigned ProgramSize; // Загружаемая часть
unsigned RelItems;
unsigned HdrSize; // Длина заголовка
unsigned MinMem;  // Минимум памяти
unsigned MaxMem;
unsigned RSS;     // Отн. сегмент стека
unsigned RSP;     // Размер стека
unsigned CheckSum;
unsigned RIP;     // Смещение точки входа
unsigned RCS; // Отн. сегмент точки входа
unsigned RelTOff;
unsigned OvrNo;
} EXEHDR;
```

Введем обозначения: FileLen – длина заражаемого EXE-файла; VirLen – длина вируса; LoadLen – длина загружаемой части EXE-файла вместе с заголовком; Hdr – заголовок EXE-файла, имеющий структуру EXEHDR.

Признаком отсутствия оверлеев служит выполнение соотношения LoadLen==FileLen, где LoadLen вычисляется по алгоритму:

```
if(Hdr.LastPage)LoadLen=
(Hdr.ProgramSize-1)*512+Hdr.LastPage;
else LoadLen=Hdr.ProgramSize*512;.
```

Для приведенной выше программной структуры в вирусе запоминается старая точка входа (Hdr.RCS: Hdr.RIP), его тело дописывается в конец EXE-файла, а поля заголовка корректируются:

```
Hdr.LastPage=(FileLen+VirLen)%512;
if(Hdr.LastPage)
Hdr.ProgramSize=(FileLen+VirLen)/512+1;
else Hdr.ProgramSize=(FileLen+VirLen)/512;
Hdr.RCS=FileLen/16-Hdr.HdrSize;
Hdr.RIP=FileLen%16;
```

В большинстве программ стек не хранится в EXE-файле. Он создается динамически по адресу, следующему за загруженной в память основной частью программы. Для этого при создании EXE-файла компоновщик устанавливает поле MinMem заголовка как минимум в (StackSize/16+1), где StackSize — размер стека. Таким способом всегда поступает компилятор Turbo Pascal. Для устранения конфликтов между вирусом и областью стека, которые в такой ситуации располагаются по одним и тем же адресам, профессиональные вирусы добавляют к алгоритму

проектирования защитного механизма следующий код (подразумевается, что VirLen кратно 16):

```
if(Hdr.RCS<=Hdr.RSS)Hdr.RSS+=(VirLen/16);.
```

Корректный вирус обязан обеспечить возврат к старой точке входа зараженной программы. Реальный сегмент старой точки входа вычисляется, как (PID+10h +Hdr.RCS), где PID (Program IDentifier) — сегмент, с которого начинается PSP (Program Segment Prefix) программы. PID передается операционной системой запущенной программе в сегментных регистрах DS и ES.

Если программа является оверлейной, проблема ее заражения усложняется, поскольку выполняется соотношение LoadLen<FileLen. При этом возможны варианты: оверлейная программа целиком помещается в оперативную память (FileLen <= FreeMem) или не помещается (FileLen > FreeMem). В первом случае используются те же алгоритмы, что и при заражении неоверлейной программы.

Во втором случае мощные вирусы пробуют "расширить" программу на длину VirLen в месте окончания загружаемой части основного модуля. Дополненная таким образом программа будет работать, если считывание оверлея происходит с конца файла или ее загружаемой части. Иначе результирующий модуль не будет работоспособен. Вирус, некорректно заразивший оверлейную программу, сразу себя выдает.

5. Методы поиска и обнаружения вирусов

Пять лет назад было вполне достаточно двух антивирусных программ для поиска и уничтожения большинства известных в мире вирусов: полидетектора Scan фирмы McAfee Associates и полифага Aidstest Д.Н. Лозинского. С начала 90-х годов стартовала эпидемия советских вирусов, поражающих КС в пределах страны, города, фирмы, отдела. Вследствие этого все большее количество системных программистов подключается к проблеме диагностики компьютерных вирусов. Большинству из них приходится изучать эту проблему с нуля, поскольку в университетах Украины пока не читается курс по компьютерной вирусологии. Тем не менее уже существуют популярные работы [3, 4], дающие начальные знания по проблеме вирусов.

В [7] изложено несколько основных методов поиска вирусов, которые применяются на практике в антивирусных программах. Рассмотрим их в порядке повышения эффективности применения.

1. Резидентные мониторы. Антивирусные модули постоянно находятся в памяти компьютера и отслеживают подозрительные действия других программ. Самый простой резидентный монитор находится в BIOS каждой материнской платы. Если в BIOS Setup установить опцию защиты от вирусов, то при любой попытке записи в Boot-сектор на диске появляется предупреждение. Если такая попытка покажется пользователю неправомочной или нежелательной, он может ее игнорировать. Монитор бесполезен при работе с SCSI-дисководами и несовместим с Windows. Кроме того, он не отслеживает попытку записи со стороны контроллера, находящегося вне ведения BIOS. Эффективно показали себя антивирусные мониторы из комплексов для Windows 95 – VSHWIN32 из McAfee VirusScan.

2. Обнаружение изменений. Процесс заражения компьютера сопровождается модификаций информации на жестком диске: вирус дописывает свой код к файлам, изменяет системные области диска. Антивирусные программы-ревизоры типа ADinf находят такие несоответствия путем сравнения ранее запомненных характеристик всех областей диска с текущими. Но следует учитывать, что не все изменения на диске вызваны воздействием вируса. Большое количество программ сохраняют конфигурационные настройки внутри своего исполняемого файла; например, Turbo Pascal, а Master Boot Record может измениться при обновлении версии операционной системы. Изобретать велосипед, создавая собственный ревизор, нецелесообразно – стандартные программы вполне справляются со своими функциями. Если же обнаружен новый вирус, следует написать для него модуль-антивирус, на что и ориентирована данная работа.

3. Сканирование – наиболее традиционный и наиболее действенный метод поиска и уничтожения вирусов. Он заключается в определении сигнатур, выделенных из ранее обнаруженных вирусов. Сигнатура – уникальная неизменяемая последовательность байтов тела вируса, позволяющая однозначно его идентифицировать. Антивирусные полидетекторы, способные удалять обнаруженные вирусы, называются полифагами. Сканеры могут обнаружить уже известные и предварительно изученные вирусы, для которых была определена сигнатура. Для полиморфных вирусов, изменяющих процедуру расшифровки при создании каждой новой особи вируса, выделить сигнатуру невозможно – в таких случаях на помощь приходит эвристический анализ. Но так как стопроцентные вирусы-мутанты, написанные с помощью Mutation Engine, попадаются редко, а пользователь страдает в основном из-за тривиальных, но достаточно деструктивных студенческих вирусов, было бы неплохо, если пользователи и программисты средней квалификации научились бы писать элементарные антивирусы.

Ниже предлагается сокращенное описание алгоритма обезвреживания вируса ("France-98") в виде основных функций вируса и фага на него.

4. Эвристический анализ используется совместно со сканированием для поиска шифрующихся или полиморфных вирусов. Суть процедуры состоит в выделении из исполняемого файла полиморфного расшифровщика тела вируса. Полное изменение исполняемого кода расшифровщика становится возможным благодаря наличию в командах процессора взаимозаменяемых команд или их последовательностей [6]. В реализациях эвристических анализаторов возможна неоднозначность в виде выдачи сообщений типа "Память, файл возможно заражены вирусом". В данном случае необходимы дополнительные процедуры тестирования для уточнения технического состояния программы.

6. Оценка сложности анализа программы для понимания алгоритма функционирования

Введем следующие обозначения: A_1 – множество спецсредств, позволяющих автоматически находить и устранять определенные дефекты ПП; A_2 – множество спецсредств, облегчающих эксперту анализ ПП; R_{A1} – вероятность наличия у эксперта средства из

множества A_1 на момент времени t; R_{A2} – вероятность наличия у эксперта средства из множества A_2 на момент времени t; L_{A1} – вероятность того, что эксперт опробует имеющиеся у него средства для автоматического поиска и устранения дефектов; L_{A2} – вероятность того, что эксперт опробует имеющиеся у него средства для исследования алгоритма ПП; N – объем программного продукта, число команд и операндов; t – время, число дней.

Тогда вероятность того, что дефект будет найден и устранен каким-либо средством из множества A_1 за время t, определяется по формуле

$$P_1(t) = R_{A1}(t) * L_{A1}. \qquad (1)$$

Для получения численной экспертной оценки стойкости ПП к исследованию введем определение.

Уровнем понимания исследователем программного продукта (УППП) назовем величину U, которая отражает знание и понимание экспертом назначения команд и операндов программы. Единица измерения УППП — число машинных команд, операндов. УППП конкретным исследователем равен N, если эксперт в состоянии откомментировать назначение каждой команды и каждого операнда применительно к функциям, которые решает данный программный продукт. Вероятность, что за время t один эксперт сможет разобраться в алгоритме ПП, будем определять через отношение УППП к объему программы.

Понимание экспертом каждой анализируемой команды или операнда тела программного продукта во многом определяется тем, как он смог освоить уже исследованную часть модуля. Это естественно в силу того, что смысловые нагрузки команд и операторов находятся в тесной взаимосвязи друг с другом.

Отсюда следует

$$P_2(t) = L_{A1} * (U(t)/N), \qquad (2)$$

где N_0 – значение начального УППП ($N_0 <= N$); k – коэффициент сложности анализа.

При этом U(t) имеет смысл только тогда, когда его значение меньше или равно N ($0 <= U(t) <= N$).

Начальный УППП включает знания эксперта об операционной системе и основных принципах функционирования исследуемого пакета: как система осуществляет запуск программы на выполнение; как работает процессор и другие необходимые начальные сведения.

Коэффициент сложности анализа k определяется наличием у эксперта возможности использовать дизассемблеры, стандартные отладчики, эмуляторы, талантом исследователя и сложностью исследуемого кода.

Выражение (2) может быть переписано в виде

$$U(t) = N_0 * \exp(t/k). \qquad (3)$$

Из формулы

$$P_2(t) = L_{A2} * (N_0/N) * \exp(t/k) \qquad (4)$$

следует, что до тех пор, пока знания эксперта об операционной системе равны 0 ($N_0=0$), значение вероятности, что он доберется до понимания алгоритма функционирования, будет всегда стабильным. Ценность (4) в том, что в ней взаимоувязаны важнейшие для практической деятельности характеристики: уровень подготовки исследователя (N_0); объем исследуемого программного продукта (N); сложность программного продукта (k).

Таким образом, при длительном исследовании ПП (t стремится к бесконечности) эксперт рано или поздно доберется до всех слабостей программы. Единственное, что может помешать, это механизмы, позволяющие рассматривать коэффициент сложности анализа k не как константу, а как зависящую от времени переменную, или увеличивать до бесконечности объем программы N. Практические шаги для решения подобной задачи, т. е. как сделать и можно ли это сделать вообще, на сегодняшний день не совсем ясны.

7. Реализация антивирусной программы для вируса "France-98"

Предлагаются принципы создания и обезвреживания вирусов на примере файлового нерезидентного вируса France-98. Вирус относится к классу особо опасных. Проявляется аудио-визуальными эффектами, начиная с четвертого поколения – в первые 5 минут каждого часа при получении управления (при запуске на выполнение зараженной программы) рисует на экране государственный флаг Франции и проигрывает первый куплет Марсельезы. Уничтожает содержимое текущего каталога 14 июля, в день независимости Франции. Не заражает файлы размером больше, чем 262143 байт, и файлы, имя которых начинается на "F". Размер вируса – 1024 байт. Полный исходный текст файлов France.asi и France.asm (200 строк-операторов Ассемблера) ввиду значительного объема не приводится. Упомянутые файлы при ассемблировании дают «запускач» вируса France.exe.

Вначале вирус определяет имя заражаемого файла. Для работы он имеет два каталога – текущий и каталог расположения зараженной программы (иногда они совпадают). Затем вирус ищет возможных кандидатов для своего внедрения и при наличии таковых пытается их заразить. Далее проверяется поколение текущей копии вируса; если она относится к 5 поколению и текущее время принадлежит интервалу между 0 и 5 минутами, вирус проявляется аудио-визуальными эффектами. Если текущая дата – 14 июля, он пытается удалить все файлы текущего каталога.

Характерными модулями подпрограмм, не имеющих прямого отношения к процессу заражения, являются: Marseillaise – рисует флаг Франции и играет Марсельезу; VSound – заставляет таймер генерировать сигнал определенной частоты в Гц и открывает выход на динамик; VDelay – задерживает выполнение программы на X тиков; VNoSound – закрывает выход таймера на динамик; VPlay – проигрывает мелодию, заданную массивом частот и длительностей.

Антивирусная программа, созданная для упомянутого выше вируса, проходит по дереву каталогов при помощи оригинального алгоритма с минимальными стековыми затратами, анализирует EXE-файлы на наличие в них «France-98», и если задан параметр командной строки «/F», пытается вылечить зараженный файл. Формат вызова антивируса: FDScan Drive: [\Directory] [/F]. (Ввиду достаточно большого объема (500 строк Ассемблера) антивирусная программа здесь не приводится.)

Файл "FDScan.asi" содержит макрокоманды, описывающие наиболее часто употребительные функ-

ции файловой системы, наиболее важные структуры и константы.

Процедуры Scan и FindSlash образуют механизм прохода по дереву каталогов, начиная со стартового каталога, заданного путем FileName. Процедура SearchViruses ищет сигнатуру вируса France-98 в последних 8 Кбайтах кода файла. Если сигнатура найдена и был задан ключ командной строки "/F", антивирус пытается корректно удалить вирус и восстановить программу в максимально приближенном к начальному состоянию.

Вспомогательными подпрограммами, не имеющими прямого отношения к поиску вируса, являются: SplitParamStr – разбор командной строки на параметры; StrLen – возвращает длину строки; StrEnd – возвращает указатель на конец строки; StrCopy – создает копию строки; StrUpper – приводит строку к верхнему регистру; StrComp – сравнивает строки; VExpand – расширяет имя файла до полного пути, включая диск и каталог.

8. Заключение

Программный продукт как объект, подверженный различным типам дефектов на стадиях проектирования, эксплуатации, нуждается в современном сервисном диагностическом обеспечении, предназначенном для решения проблем:

– тестопригодного проектирования программ со встроенными тестами и контрольными точками, обеспечивающими наблюдаемость процесса тестирования и исполнения функций;

– верификации и сертификации готовых ПП на предмет проверки функционирования и отсутствия нефункциональных деструктивных модулей;

– поддержания исправного функционирования ПП на стадии эксплуатации и защиты от вирусов и других программных механизмов деструктивного типа.

Задачи тестирования и сертификации ПП следует решать в плане рассмотрения проблемы диагностического обслуживания программно-аппаратурного комплекса методами и средствами технической диагностики цифровой аппаратуры.

Следует пересмотреть существующие стандарты СССР в плане их модернизации с учетом новых условий применимости программно-аппаратурных средств, возникновения, существования, проявления многообразия неисправностей компьютерных систем и сетей.

Литература: 1. *Глушков В.М.* Кибернетика. Вопросы теории и практики. М.: Наука, 1986. 488 с. 2. *Хаханов В.И.* Техническая диагностика элементов и узлов персональных компьютеров. К.: ІЗМН, 1997. 308 с. 3. *Безруков Н.Н.* Компьютерная вирусология: Справочное руководство. К.: УРЕ, 1991. 416 с. 4. *Касперский Е.В.* Компьютерные вирусы в MS-DOS. М.: Эдэль, 1992. 176 с. 5. *Хижняк П.Л.* Пишем вирус и ... антивирус. М.: ИНТО, 1991. 90 с. 6. *Расторгуев С.П.* Программные методы защиты информации в компьютерах и сетях. М.: Яхтсмен, 1993. 188 с. 7. *Фролов А.В.*, *Фролов Г.В.* Умеете ли вы искать вирусы? //Компьютерра. 1996. №33. С.30-33. 8. *Данкан Р.* Профессиональное программирование в MS-DOS. М.: Мир, 1993. 509 с.

Поступила в редколлегию 12.01.98

Хаханов Владимир Иванович, д-р техн. наук, профессор кафедры АПВТ ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей и программных продуктов. Хобби: баскетбол, футбол, горные лыжи. Адрес: 310726, Украина, Харьков, пр. Ленина, 14, тел. 40-93-26.

Фрадков Сергей Александрович, аспирант кафедры АПВТ ХТУРЭ. Научные интересы: вирусология, защита информации, техническая диагностика. Адрес: 310726, Украина, Харьков, пр. Ленина, 14, тел. 40-93-26.

Ханько Вадим Викторович, аспирант кафедры АПВТ ХТУРЭ. Научные интересы: диагностика вычислительных систем и программных продуктов. Адрес: 341000, Украина, Донецкая обл., г. Мариуполь, ул. Новороссийская, 14, кв. 87, тел. (0629) 35-21-33, 37-70-93.