5. *Ramkrishna Chatterjee, Barbara G. Ryder.* Relevant Context Inference. Symposium on Principles of Programming Languages, p. 133-146, 1999.

6. *Patrick Cousot, Radhia Cousot.* Abstract interpretation frameworks. *Journal of Logic and Computation,* 2(4):511-547, 1992.

7. *Dor N., Rodeh M., and Sagiv M.* CSSV: Towards a Realistic Tool for Statically Detecting All Buffer Overflows in C. In Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, pages 155–167, San Diego, California, 2003.

8. *Wagner D., Foster J. S., Brewer E. A., and Aiken A.* A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities. In Proceedings of 7th Network and Distributed System Security Symposium, Feb. 2000.

9. *Emami M., Ghiya R., and Hendren L.* Context-Sensitive Interprocedural Points-to Analysis in the Presence of Function Pointers. In Proceedings of the SIGPLAN '94 Conference on Program Language Design and Implementation, Orlando, US, 1994.

А.С. Моляков
Россия, г. Москва, ОАО «ИнфоТеКС»,

ИССЛЕДОВАНИЕ СКРЫТЫХ МЕХАНИЗМОВ УПРАВЛЕНИЯ ЗАДАЧАМИ ЯДРА WINDOWS NT 5.1

В предисловии хочу выразить благодарность научному руководителю д.ф.-м.н., профессору Грушо А.А.В данной статье я наглядно покажу коллизии в сопряжении проектов верхнего и нижнего уровней на платформе Windows NT.

Актуальность. Острота проблемы обеспечения безопасности субъектов информационных отношений, защиты их законных интересов при использовании информационных и управляющих систем, хранящейся и обрабатываемой в них информации все более возрастает. Проблема защиты вычислительных систем становится еще более серьезной и в связи с развитием и распространением вычислительных сетей, территориально распределенных систем и систем с удаленным доступом к совместно используемым ресурсам. Увеличение числа компьютеров в организации приводит к потере контроля над ними.

Объект исследования: программное обеспечение штатных средств защиты на базе платформы Windows NT 5.1(объекты микроядра , модули диспетчера оборудования , подсистема управления вводом-выводом).

Практическая значимость работы

Результаты выполненных исследований могут быть использованы при построении ответственных программных систем , для прогнозирования и оценки надежности ПО и при расчете надежности систем в целом. Метод систематического поиска уязвимостей позволяет обнаруживать скрытые процессы как в третьем кольце защиты , так и в нулевом , используя неявные механизмы в сопряжении проектов верхнего и нижнего уровней.

Научная новизна

- разработана методология анализа надежности ПО как процесс возникновения и устранения ошибок в ПО на всех этапах жизненного цикла ПО, предложена методика тестирования ,реализован новый метод – метод систематического поиска уязвимостей ОС Windows;

- представлены таблицы состояний ЦП(логических переходов), механизмов сопряжения проектов верхнего и нижнего уровней и процессов выявления недекларированных уязвимостей ядра Windows NT 5.1.

Архитектура памяти NT — это система виртуальной памяти, использующая 32-разрядные адреса в линейном адресном пространстве.Каждой прикладной про-

грамме выделяется адресное пространство в границах от 64 кбайт до 2 Гбайт (первые 64 кбайт полностью закрыты). Программы «невидимы» друг для друга, хотя могут общаться через механизмы OLE и DDE. Верхняя область принадлежащего программе 2 Гбайт блока памяти содержит так называемые client-side DLL (клиентские динамические библиотеки) подсистемы Win32. ОС может отслеживать количество описателей, открытых для данного объекта.

После инициализации процессор находится в режиме реального адреса. В процессе страничной трансляции адресов полученный линейный адрес разбивается на три части. Старшие десять бит линейного адреса являются индексом элемента из каталога таблиц. По этому элементу определяется физический адрес таблицы страниц. Биты 21-12 линейного адреса выбирают элемент из этой таблицы страниц. Выбранный элемент определяет физический адрес страницы. Младшие 12 бит линейного адреса определяют смещение от начала страницы. Необходимо корректное сопряжение ЦП и внешнего контроллера при организации и управлении внешними, внутренними циклами обработки информации (внутренней и внешней синхронизации).Архитектура процессора Intel позволяет реализовать программно-аппаратную модель с высокой надежностью и малой латентностью процессов в ядре ОС Windows.Нужно помнить , что критические ошибки при работе с аппаратным стеком будут обработаны ЦП как исключение и приведет к аварийному завершению.

Использование возможностей контроллера внешних прерываний позволяет реализовать предупреждения возникновения исключительных ситуаций [3] в системе. Рассматривая ЦП как конечный автомат с микропрограммным управлением, мы решаем задачу дискретного логарифмирования log2(4)=2 , только 2 значащих разряда используются при кодировании на данном уровне. Триггерные схемы переключаются подачей сигналов SET/RST на входы микросхем.Событие в системе представляет реакцию на случайные двоичные коды , подаваемые на схемы :

Механизм физической защиты процессора Intel 3000 реализует аппаратно поддержку 4 колей защиты

00	Кодировние прав доступа к сегментам кода в ОС Windows
01	
10	
11	

Ранжирование и локализация процессов в системе

Адресное пространство всех процессов в системе разделено на четыре физических сегментов памяти .Оценим поведение при случайных воздействиях на ЦП и нарушение прав доступа на физическом уровне. Системные разработчики не уделяют должного внимания рассмотрению механизмов сопряжения проектов верхнего и нижнего уровней , алгоритмам машинной логики и рассмотрения процессов в микроядре , изучению неявных механизмов взаимодействия объектов ядра , выявлению коллизий в описании технологий Intel Core.В системе rang(0-3) сопоставимы с идентификацией переключения между физическими сегментами памяти. При работе на аппаратном уровне значащими являются 2 бита при кодировании состояний ,так что мы можем закодировать однозначно 4 разных состояния (00, 01, 10,11), которые описывают область адресации при переключении контекста задач процессором .Мы получили поле характеристики 2 , состоящее из базисных взаимнопростых элементов(примитивов). Это говорит о самом низком уровне

кодирования информации и оценки состояний . Процессор генерирует коды на выполнение задач и контролирует выполнение инструкций , поэтому управляющее слово процессора состоит из слова состояние задачи и полей управления контекстом переключения задач, которое представляет собой 64-битовую структуру. Бит P сигнализирует о том, что объект находится в физической памяти Бит S свидетельствует о системности выделенного объекта в ОС Windows (проверка загружен ли модуль ,запущен ли процесс ядра ОС).Бит подчинения реализует модель наследования ООП родитель-дочерний процесс,бит вложенной задачи служит для установки системой NT-флага, признака выполнения(генерации) последующей вложенной системной задачи(процесса) .Биты X1X2X3 кодируют привилегии на доступ к данным пользователя (RWE).Биты X4 X5 кодируют права доступа к сегментам кода файлов исполняемого формата ОС Windows.Биты S1S2 определяют(характеризуют) состояние (статус) активной задачи процессора (00-инициализация, 01 - передача данных пользователей системы,10 - выполнение программного кода , 11- завершение и контроль(опрос)состояния запущенного приложения).Значит, требование [2] оптимальности приводит к реализации программными средствами двух колец защиты. Требование разумности и достаточности приводит к необходимости уменьшение затрат на разработку ядра и быстрой реализации гибкого и удобного прикладного интерфейса .Компания MicroSoft [5] умалчивает о существовании возможности аппаратно - реализованной схемы управления задачами процессора Intel 3000.При разработке ОС необходимо выполнение требований по сопряжению проектов высокого и низкого уровней, адекватность модели описания функционирования на языках высокого уровня и машинной логики процессора как конечного автомата с микропрограммным управлением.При кодировании используются две конфигурации 1X и 0X (два кольца защиты).Существование уровня 0X создает неопределенность на уровне выполнения машинных кодов(00 и 01- качественно разные конфигурации при выполнении инструкций процессором). Совмещенность пространства SYSTEM_EXECUTE и режима ядра, дающего полный доступ к аппаратно-реализованным ресурсам не целесообразно. Уровни 00 и 01 определены процессором как привилегированные, совмещение вносит избыточность в структуру ядра и неопределенность , латентность при организации и управлении системными задачами. Анализ ядра Windows NT 5.1 на целевой платформе Intel 3000, проведенный мною на контрольно-испытательном стенде НИП «Информзащита» показал необходимость корректного обращения системных управляющих программ к супервизорному модулю ядра ОС. Необходимо использовать расширенные средства MASM при программировании процедур низкого уровня и взаимодействии с аппаратным стеком и системными областями памяти для написания специальных макрообращений, для чего необходимо описать расширение языка процессора Intel , описать структуры с использованием метаязыка модели языка IDL процессора Intel 3000 C5X.Модули системного отладчика работают с регистрами Dr0-Dr7 и типами данных type_byte (8-битный код, подаваемый на внутреннюю шину процессора) . Мощность PV - пространства (генерации P – задач и пространство управления V - векторы адресации переходов по Д.Дейкстры) равна 32*8 =256 (100H) , поэтому размер кодовых таблиц в ASCII формате Intel процессора равна 100h. Состояние управления и организации имеет два базисных вектора в заданной системе процессов ,причем размер каждого кодового слова процессора при запросе на выполнение 10h , тогда размерность матрицы будет 10*10=100h , значит , мы определили множество базисных элементов , более того , поскольку данный уровень детализации работает с атомарными операциями (бинарного сложения и вычитания, умножения и деления) , группа P- полноты , состоящая из примитивов ,

потому надежная на уровне машинной логики. Оценим всю полноту конструкций языка P-кода при работе с сегментами кода и данных пользователей , учитывая , что размерность инструкции процессора 64 бита, а размерность бинарных векторов управления контекстом задач 32 бита, тогда размерность полей данных о всех параметрах задачи (генерации и управления имеет размерность Робщ = 64 + 32 = 96 (матрица PV-структур).Для решения задачи распознавания необходимо передавать и идентификаторы (SE-chain – цепочки символов , заголовки ,по которым идет выборка операций из кодовых таблиц).В среде Windows исполнимые файлы имеют заголовок в 3 бита .При передачи запросов на выполнение процессору системные модули преобразуют данные из Unicode в ASCIIZ код (загрузчик не работает с расширенным форматом данных , кодирует последовательности загружаемых сегментов кода идентификаторами, разделяя NULL- символов на конце каждого заголовка(знаки препинания),реализует ассоциативную логику процессора при работе с TLB – буфером, хранящим списки задач и выборку из внутренней кэш-памяти кода .Таким образом, сложность Рсум = 96 +3 +1 = 100h (представление в виде hex-кода бинарных файлов).Все пространство процессов характеризуется модулем TIB(Task Information Block) и SIB(Segment Information Block) , является замкнутым и самодостаточным, адекватной описанию состояний процессора на аппаратном уровне. Внутренний цикл работы процессора определяется временными интервалами. При выполнении одной инструкции другие запросы блокируются на время выполнения задачи ядра ОС.

Мультиплексирование каналов связи с внешним оборудованием позволяет создавать очередь параллельной обработки (конвейер).На уровне декодирования идет преобразование 2*2*2 в последовательную выборку разрядов , т.е. 2+2+2 =6 , тогда время выполнения команды (T0- T5) занимает максимально 3-4 такта , 1 такт на контроль выполнения и 1 такт для подачи сигнала стробирования , чтобы открыть доступ к кристаллу процессора. В исполнительной системе объект (object)[6] – это отдельный образец статически определенного типа объектов, существующий во время выполнения. Тип объектов, иногда называемый классом объектов, включает определенный системой тип данных, объектные сервисы, работающие с образцами этого типа, и набор атрибутов объекта. Атрибут объекта - это поле данных внутри объекта, частично определяющее его состояние. Объектные сервисы – способы манипулирования объектами – обычно считывают или изменяют атрибуты объектов. Windows NT использует объекты для унификации представления и управления системными ресурсами. Каждый системный ресурс, который могут совместно использовать несколько процессов, такой, как файл, память или физическое устройство, реализован как объект и обрабатывается объектными сервисами. Доступ ОС к ресурсам и работа с ними унифицированы. Создание, удаление и ссылка на объект осуществляется с использованием описателей (handle) объектов. Контроль использования ресурсов сводится к отслеживанию создания и использования объектов. Для всех объектов контроль доступа к ним осуществляется одинаково с помощью подсистемы защиты. Два процесса совместно используют объект [7] тогда, когда каждый из них открыл его описатель. ОС может отслеживать количество описателей, открытых для данного объекта, чтобы определить, действительно ли они все еще используется, и может удалить объекты, которые более не используются. Микроядро и слой абстрагирования от Оборудования (HAL) изолируют подсистемы Исполнительной Системы от конкретной архитектуры процессора. Другой аспект независимости от архитектуры состоит в том, что правильно написанный драйвер (общающийся с внешним миром только посредством функций, предоставляемых различными компонентами исполнительной системы) переносим между всеми поддерживаемыми NT – платформами на

уровне исходных текстов. HAL обеспечивает поддержку и отвечает за предоставление стандартного интерфейса к ресурсам процессора, которые могут меняться в зависимости от модели внутри одного семейства процессоров. Возможность замены слоя HAL обеспечивает всем вышележащим слоям операционной системы независимость от аппаратной архитектуры. Ядро NT — это стержень всей активности Windows NT. Оно управляет процессором, планируя потоки и направляя их на выполнение, реагирует на прерывания и исключения и реализует низкоуровневые механизмы синхронизации, которые используются им самим и другими частями исполнительной системы. Объект-поток ядра содержится внутри объекта-потока исполнительной системы и содержит информацию, необходимую ядру для направления потока на исполнение. Аналогично ядро реализует минимальную версию объекта-процесса, называемую объект-процесс ядра (kernel process object).Кроме того, в объекте – процессе ядра находится указатель на каталог таблиц страниц процесса (используется для отслеживания виртуального адресного пространства процесса), общее время выполнения всех потоков процесса, базовый диспетчерский приоритет процесса по умолчанию и набор по умолчанию процессоров, на которых могут исполняться потоки процесса, – так называемое процессорное сродство (processor affinity). Управление информацией, хранящейся в объект – процессе ядра, осуществляет исключительно ядро. Другие части исполнительной системы могут считывать или изменять ее, только вызывая функции ядра. Объект-поток ядра более сложен, чем объект-процесс ядра. Он содержит некоторую очевидную информацию, такую как процессорное сродство потока (неполное подмножество сродства процесса) и общее время выполнения потока. Сюда также входят базовый приоритет потока (который может отличаться от базового приоритета по умолчанию для процесса) и его текущий приоритет. Особенно важный элемент данных, хранящихся в объекте-потоке ядра, — это диспетчерское состояние потока. В любой момент времени поток может находиться в одном из шести состояний; из них только одно делает его кандидатом на выполнение. Уязвимость при передачи системных запросов к ядру ОС заключается в том , что по умолчанию приложения , запущенные на правах администратора, имеют право на запуск(Execute) и контроль выполнения списков задач, провести переполнение системного буфера и оказаться в пространстве системных процессов ядра. SP2 поддерживает функцию DEP (предотвращение запуска кода, объявленного как данные), позволяющую, например, помечать области памяти, выделенные для данных.Это является определенной преградой для атак, использующих переполнение буфера.Функция DEP в SP2 требует аппаратной поддержки и работает только в процессоре фирмы Advanced Micro Devices с защитой неисполняемых страниц или в процессоре Intel с битовой функцией Execute Disable. Пакет SP2 обеспечивает программную поддержку DEP в коде ядра Windows XP. Данный механизм сбрасывает право владения при переключении задач (стоит пометка в системных таблицах о явно заданном неразрешении выполнения загружаемых страниц кода 3-го кольца защиты). Это изолирует пространство данных и кода при переключении контекста задач при переходе (3 ---- > 0) полностью контролировать процесс выполнения задачи. Однако остается неопределенность логики работы системы на уровне ядра. Если процесс или задачи уже загружены в основную память и получили управление , то некорректные действия программ в привилегированном режиме приведут к критическому сбою защиты ядра. Дело в том , что два бита определяют таблицу переходов процессора при переключении контекста активных задач (механизм скрытой логики , реализованной аппаратно на платформе Intel). При передачи кода на выполнение процессору происходит его приведение к BDC- формату(представление в виде тетраграмм).Вместо 3 битов поле прав

при передаче на ДШ процессора будет представлять поле из 4 битов . Приписываемый ноль не изменяет значение, однако преобразование на уровне ядра (проверка AR – поля на аппаратном уровне) установить ?0 в 00 , процессор определит все данные , передаваемые ему на уровне ядра как код программ , подлежащих выполнению , потому любые сбои приведут к немедленной остановке(отказу) всей системы. Таким образом ,неопределенность описания программных механизмов управления аппаратно – реализованным защищенным режимом Intel- процессора приводит к подобным результатам. Приведем общий отчет о проведенном тестировании с использованием отладчика Soft Ice 4.5 в виде таблицы. Загрузчик ОС бинарного кода программ на выполнение работает с 8-битным кодовыми тегами, а ДШ(дешифратор) процессора с тетрадами передаваемых кодов .Сопряжение внутреннего и внешнего циклов обработки прерываний кодируется с помощью уже описанных мною внутренних ps_ таблиц процессора, которые несут информацию о сегментах кода и управление по их выполнению(SIB информация, а ею как раз не владеет модуль ntoskrnl.exe, оперирующий только с TIB –блоками и целостность кодов ядра с помощью модуля KTEB (Kernel Envirement Execution Block .Вид KTEB по умолчанию приведен ниже. Неопределенность в старших битах и ведет к остановке процессов ядра при неверных обращениях (неадекватность модели).XXXX – адрес вектора обработки прерывания , старшие биты равны нулю, так как мы реализуем прямое обращение к микроядру с помощью аппаратно – реализованной поддержки прямой обработки прерываний процессором (механизм GENINT32 и TaskG32).

Если процессору явно указать и присвоить s =1 , процесс ядра потеряет признак системности(право System_Owner) и не сможет модифицировать код программ , загруженных в основную физическую память(p=0).Если бит s равен 1 , то трансляции кода с ассемблера в коды машинных команд , будет обнаружен значащий бит в старших разрядах (16-31 линии системной шины) , что приведет передачи к выполнению операций в области пользовательских процессов программный внешний цикл обработки , а не младшие адреса системной шины , по которой передается код сразу на внутреннюю шину процессора. Значит, при программировании системы необходимо требовать повышения системности в организации связки программный уровень – аппаратный уровень.

При проведении экспериментов использовались специализированные инструментальные средства, позволяющие восстановить структуру кода ядра. Ниже приведено полное описание структур кода микроядра, модулей ntoskernel.exe и взаимосвязь данных полных описателей Full_INFORMATION_QEUE (Full-INFORMATION_RESP)на уровне диспетчера процессов и редуцированных списков динамических процессов(неполная структура описателей)-READ_QEUE_AND X (WRITE_QEUE_ANDX) на уровне микроядра(сопряжение проектов верхнего и нижнего уровней).

Описание работы процессора и его взаимодействие с ядром ОС Windows NT 5.1

Базовые приоритеты id={0-7} идентифицируют объекты-процессы ядра, ,выполняющие рутину ввода-вывода данных на внутреннюю шину процессора DB0-DB7, с ARPL-структурой доступа на уровне asm_ спецификаций(ассемблерного кода).Приоритет id=8(normal) описывает приложение 3-го кольца защиты ,взаимодействующее с системной оболочкой.

На данном уровне Explorer.exe и его расширения формируют очередь запросов на обращение к системным модулям ядра, формирует буфер обмена данными.

Приоритет id=9(above_normal) описывает приложения фонового контроля задач (Winlogon.exe), реализует перехват всех системных обращений приложений рабочего стола.

Приоритет id=10(high_level) описывает привилегированные модули smss.exe.csrss.exe, которые реализуют динамическую среду вызовов клиентских запросов, взаимодействие с сервисами доменов.Начиная с id=11, система идентифицирует код как процесс системного окружения (system_Idle).

На данном уровне реализуется механизм Native API вызовов, взаимодействие непосредственно с верхним интерфейсом ядра ОС Windows NT.Id=12 описывает работу менеджера объектов, механизмов защиты объектов (VMM-Интерфейс, базовые элементы защиты). Id=13 реализует идентификацию потоков задач драйверов низкого уровня, взаимодействие с портами IO-среды . Id=14 характеризует TSR-код контроллера ввода-вывода, DMA-интерфейс.Id=15 идентифицирует самый привилегированный процесс в ядре (программа-монитор, распределяющая ресурсы процессора, переключение задач, формирование управляющих тегов.

В научной литературе подробно описываются разработки по построению синтаксиса языка экспертных систем безопасности. Однако существует неполнота описания синтаксическими методами(задание определенных правил и процедур контроля) семантики модели процессов.

Чтобы доказать полноту и непротиворечивость высокой степени защищенности предлагаемой мною метода защиты, нужно проанализировать матрицу отношений связи информации, найти P-полного описания семантических моделей Hal-среды Windows и разрабатываемого мною монитора безопасности. Матрица несимметричная, она не нормализована.

Чтобы отражение было рефлексивным, необходимо включение единичной главной или вспомогательной диагоналей. ДНФ-форма карты процессов Карно 10^01 || 11^01=00||01=01 – устойчивый уровень процессов.

Состояние 00000 является запрещенным, так как определитель матрицы равен в данном случае нулю(условие вырождения).Процессор запрещает напрямую обращаться к ПЗУ и менять содержимое теневых регистров.

Таким образом, чтобы исключить тупиковую ветвь матрицы состояний, необходимо ввести дополнительный контроль за контекстом активных задач ядра.

Рассмотрим далее сам процесс выполнения кодов на уровне ядра ОС Windows процессором..

В научной литературе описываются лишь 2 уровня представления кодирования атрибутов доступа: на уровне системной оболочки 3-битная структура доступа к файлам (RWE – права на чтение, запись, выполнение) и 4-битная структура менеджера доступа к объектам ОС Windows (Owner-владелец + RWE).

Я же обнаружил еще 2 новых «слоя» контроля контекста задач: кодирование аттрибутов защиты на уровне менеджера виртуальной памяти и подсистемы ввода-вывода представляет собою 5 –битную структуру.

VPL-признак виртуализации	Owner	E	W	R

На уровне диспетчера HAL-среды и объектов микроядра данная структура имеет расширение до 6-битного вектора.

S бит	VPL-флаг	Owner	E	W	R

Таким образом, обобщенный тег кодирования кода на уровне процессора (уже не логические данные объекта, а аппаратная спецификация) представляет собой следующую структуру:

CR0 (регистры контекста задач)	CR1	CR2	CR3	CR4 CR5
Установка системной защиты страниц памяти ядра ntoskernel.exe (явная инициализация бита S) Системный контроль	Реализует Установку флага VPL (переноса данных, операций расширения, Определяет область виртуализации окна процессов Контроль виртуализации	Установка флага NT (Next Task), передача управления другому процессу Проверка условий передачи управления parent- процессу	Установка флага G (G (generate), выполнение вложенной задачи Проверка условий передачи управления дочернему процессу	Передает данные DR3 – DR4 для отладки кода драйверов (ARPL-структура доступа) cpl { 00 или 11 } Проверка привилегий доступа

Теперь посмотрим , все ли состояния логических переходов устойчивы .В процессе выполнения процессор сам присваивает динамический приоритет(контекст семантики), что расширяет поля представления синтаксических моделей языка. Я наглядно раскрою причину неполноты требований основной теоремы безопасности к модулям ядра Windows Nt 5.1.

Пусть на вход процессора поступает тег {1110 00 1110 00 0011} , где 0011- пользовательский процесс , 1110 – процесс ядра , 00 – разделитель записей. По умолчанию присваивается нулевой вектор. Ситуация характеризует передачу недокументированного вызова вируса из 3-го кольца защиты в нулевое. Супервизор индексирует тег ровно на 1 сдвиг влево.

Мы получили конечный статус. Нет в системе дополнительной степени защиты. Любое исключение, десинхронизация событий, мутации объектов ядра приведет к останову системы.

Для разрешения данной коллизии необходимо вводить дополнительный контроль SwapContextHooking и PspCidHooking с перехватом точки входа модуля ntoskernel.exe при стартовой загрузке. Мой алгоритм явным образом присваивает S VPL –структуре значение и разделяет код данных и код программ в любом сегменте активационной записи процессора. Уязвимости на нечеткой границе распределения решаются.

Если не вводит данный критерий, то любой код может свободно получать управление путем изменения динамического контекста задач. Я не отслеживаю системные вызовы, как многие программы защиты ведущих компаний-разработчиков, а использую сам принцип семантических запросов к ЦП на низком уровне и использую свой PE-загрузчик кода библиотек функций перехвата, сканирование Cr0 и VPL-вектора, ввожу анализ дельта-параметра (детектирование каждой функции нотификации планировщика переключения задач ядра).

В регистр EDI заносится адрес так необходимой нам переменной. Бинарный код выглядит так – 0xBF00E0548057, где 0x80540E00, адрес переменной, который копируется в регистр EDI. На этих предположениях основан эвристический метод определения адреса массива.

Модуль process.sys реализует работу с опкодом процессора, внутренними таблицами, PE_loader.sys реализует загрузку зашифрованного модуля библитек перехвата (с введением имитовставки) и определяет точку вхождения в imagine_path модуля ntoskernel.exe, позволяет контролировать выполнение всех процессов. Нельзя контролировать загрузку всех кодов в системе, так как есть недокументированные интерфейсы, но выполнение можно контролировать на уровне тегов данных ядра, поскольку все вызовы модуля ntoskernel.exe транслируются в семантический запрос.

Мой оригинальный алгоритм основан на низкоуровневой спецификации. В ходе исследования были обнаружены новые 4 класса уязвимостей модулей ядра Windows NT 5.1. Сочетание системно-методологического подхода и имитационного моделирования позволило раскрыть базовые закономерности функционирования ядра ОС и скрытые каналы доступа.

Более того, автор рассмотрел операционную систему как организационно-технологическую, представил правовые, организационные, технические и технологические аспекты разрешения коллизий (трудностей).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Искусственный интеллект: Справочник в 3-х книгах. – М.: Радио и связь, 1990. Книга 2. Модели и методы / Под ред. Д.А. Поспелова. – 304 с.

2. Горелик А.Л., Гуревич И.Б., Скрипкин В.А. Современное состояние проблемы распознавания: Некоторые аспекты. - М.: Радио и связь, 1985. – 160 с., ил. – (Кибернетика).

3. Горелик А.Л., Скрипкин В.А. Методы распознавания: Учебное пособие для вузов. – 3-е изд., перераб. и доп. - М.: Высш. шк., 1989. – 232 с..

4. Теория вероятностей: Учеб. для вузов / А.В. Печинкин, О.И. Тескин, Г.М. Цветкова и др. Под ред. В.С. Зарубина, А.П. Крищенко. – М.: Изд-во МГТУ им. Н.Э. Баумана, 1999. – 456 с. (Сер. Математика в техническом университете; Вып. XVI).

5. Heckerman D. A tutorial on learning Bayesian Networks: Technical Report MSR-TR-95-06. – Microsoft Research: Advanced Technology Division, 1996. – 58 p.

6. Androutsopoulos et al. An evaluation of naive Bayesian anti-spam filtering. // Proceedings of the 11th European Conference on Machine Learning. – 2000. – pp. 9–17.

7. Elkan C. Boosting and naive Bayesian learning: Technical Report No. CS97-557.– Department of Computer Science and Engineering, University of California, San Diego.– 1997.– 11p.

8. Умрюхин Е.А. Механизмы мозга: информационная модель и оптимизация обучения. – М., 1999. – 96 с.

9. Ганецкий М.А. Методы программирования // Труды №1 молодых учёных, аспирантов и студентов «Информатика и системы управления». – М: МГТУ, 2002. – 460 с.

А.Г. Лысенко
Россия, г. Санкт - Петербург, ГОУ ВПО Санкт-Петербургский
Государственный политехнический университет

СИСТЕМАТИЗАЦИЯ УГРОЗ ГИБРИДНЫХ СЕТЕЙ

1. Введение

Быстрое развитие беспроводной технологии привлекло внимание множество компаний, которые стали использовать мобильные технологии наряду с фиксированными. Введем определение гибридной сети. Гибридная компьютерная сеть – сеть, в которой используются фиксированные и мобильные технологии.

Для корпоративных клиентов наиболее частый вариант использования мобильных устройства – это их подключение к корпоративной сети.

Мобильные пользователи, подключаясь к фиксированному сегменту, образуют мобильный сегмент корпоративной сети, что в целом составляет гибридную