При построении автомата используются следующие правила.

Множество состояний автомата A — виртуальные метки, которые ставятся в точках ветвления защищаемого приложения.

Множество входов автомата — переменные, в соответствии со значением которых происходит тот или иной переход по графу потока управления защищаемого приложения.

Множество выходов — код, который необходимо выполнить для того, чтобы перейти из одного состояния автомата в другое.

Функции переходов и выходов данного автомата — табличное представление связей вершин в графе потока управления.

В результате над защищаемым приложением строится конечный автомат, который полностью описывает логику переходов данного приложения.

В дальнейшем алгоритм защиты приложения сводится к тому, что множество состояний автомата, интерпретирующего защищаемое приложение, переносится во внешний аппаратный модуль.

Путем осуществления подобного набора преобразований над кодом приложения злоумышленник для анализа получит набор вершин графа потока управления. В случае, если вершины графа никак не связаны и невозможно выделить некоторые характеристики вершин, позволяющие восстанавливать дуги графа, то сложность восстановления графа будет составлять $O(2^{n^2})$, где n число вершин графа.

СПИСОК ЛИТЕРАТУРЫ:

1. *Чернов А. В.* Анализ запутывающих преобразований программ. URL: http://citforum.ru/ security/articles/analysis/.
2. *Краснопевцев А. А.* Разработка средств автоматизации для переноса байт-кода во внешний аппаратный модуль // Технологии Microsoft в теории и практике. М.: Вузовская книга, 2008. С. 139–141.
3. *Холанд Г., Мак-Гроу Г.* Взлом программного обеспечения, анализ и использование кода. М.: Издательский дом «Вильямс», 2005. – 400 с.: илл.
4. *Букасов В. А, Краснопевцев А. А.* Автоматизация динамической распаковки программ // Материалы 10-й Международной научно-практической конференции. Таганрог. 24–27 июня 2008 г. С. 13–14.

С. В. Ктитров, А. А. Кокуев, Р. Г. Козин

РЕАЛИЗАЦИЯ МЕХАНИЗМА УСТАНОВКИ ПРАВ ПРОЦЕССА ПРИ ЗАПУСКЕ В ОПЕРАЦИОННЫХ СИСТЕМАХ MICROSOFT WINDOWS XP и WINDOWS 7

Требование защиты данных пользователя и сохранения целостности и работоспособности операционной системы приводит к необходимости разделения полномочий пользователей, а следовательно, и запускаемых ими процессов на привилегированные, имеющие административные полномочия, и пользовательские. В ряде ситуаций такое деление не позволяет обеспечить надлежащую защиту данных даже при реализации списков управления доступом.

Рассмотрим задачу передачи данных между пользователями. Предположим, пользователь a наделен привилегиями $R(a)$, пользователь b имеет привилегии $R(b)$, S — административные привилегии, причем $S \cap R(a) = \varnothing$; $S \cap R(b) = \varnothing$; $R(a) \cap R(b) = C$. Если $C \neq \varnothing$, возможен обмен данными между пользователями a и b как через подмножество файловой системы, так и с

использованием процесса с правами, включающими C, причем между пользователями a и b в обе стороны. Ни у одного из пользователей нет возможности ни проконтролировать передаваемые данные, ни предотвратить внесение изменений отправителем в уже переданные данные. Использование модели мандатного доступа решает задачу частично, так как позволяет реализовать лишь однонаправленный поток данных. Модель с общими правами характерна для ОС Windows.

Имеющаяся в ОС UNIX возможность запуска процессов (SUID-программ) с правами другого пользователя решает задачу несанкционированного доступа к передаваемым данным и защиты их от изменений. Действительно, пусть $\exists P : R(P) \subset R(b)$, где P — процесс, доступный только для выполнения пользователем a. Тогда с использованием процесса P возможен как контроль передаваемой информации, так и запись ее в область файловой системы, недоступную пользователю a. В этом случае права C могут быть существенно сокращены, например до объема, определяемого интерфейсом ввода данных для процесса P. Разработчики ОС Windows сознательно отказались от реализации данного механизма ради повышения уровня безопасности, однако его наличие унифицировало бы средства защиты различных ОС и облегчило бы создание кроссплатформенных средств обработки защищенных данных, таких, например, как приведены в работе [1]. Механизм запуска программ от имени другого пользователя RunAs ОС Windows предполагает ввод пароля этого пользователя и не может быть использован для решения данной задачи. Также можно реализовать процесс-посредник с правами $R(P) \supset S$ в форме службы, что несет потенциальную угрозу целостности и стабильности ОС. Примером рассмотренной задачи может служить система тестирования, не использующая модель сетевого взаимодействия, запускаемая учеником, но не позволяющая внести изменения или уничтожить протокол теста.

Другая, не менее важная задача — частичное делегирование административных полномочий, что требует поддержки средствами ОС возможности предоставления полномочий $\hat{S} \subset S$. Механизм UAC (User Access Control) [2] ОС Windows Vista и Windows 7 предполагает контроль предоставления административных полномочий самим пользователем и не обеспечивает гибкого делегирования, которое могло бы быть обеспечено возможностью запуска процесса с правами $R(P) \supset S$ пользователем a: $S \cap R(a) = \varnothing$.

Помимо делегирования административных полномочий, в ряде случаев необходимо оградить запускаемый процесс от пользовательских данных, создавая так называемую «песочницу». Процессы, находящиеся в «песочнице», могут обращаться только к тем данным, к которым им явно разрешено обращаться.

Таким образом, существует класс задач, которые не могут быть легко решены в рамках реализованной в ОС Windows модели предоставления прав доступа.

Рассматриваемая задача может быть решена с использованием разработанной авторами универсальной системной службы, аналогичной сервису RunAs, но не требующей указания пароля пользователя, привилегии которого предоставляются запускаемой программе. Список пользователей, которым разрешается запуск процесса P, формирует сам носитель прав, как контролирующий посредством P поток данных от запустившего P пользователя, так и размещающий с его помощью данные в хранилище, доступа к которому у пользователя процесса нет. При этом владелец P необязательно имеет административные полномочия, а право запуска процесса P делегируется владельцем программы с использованием разработанного авторами приложения, взаимодействующего с системной службой.

Предложенная архитектура позволяет использовать псевдонимы программ для сокрытия исполняемого файла в защищенной области файловой системы. Код системной службы имеет небольшой объем и может быть тщательно проверен и верифицирован. То же относится к программам, делегирующим административные полномочия. Таким образом, в системе не будет

процессов с административными полномочиями кроме верифицированных и будет исключен запуск иных программ с повышенными привилегиями.

Рассмотрим реализацию механизма делегирования прав подробнее. Для изменения прав процесса используется либо хранимый с правами LocalSystem пароль владельца, предоставляемый им самим при регистрации программы в службе и используемый в дальнейшем в системном вызове CreateProcessWithLogonW, либо перехват системных вызовов с использованием внедрения DLL в процесс [2]. В первом случае для запуска процесса требуется программа-посредник, также входящая в комплект программного комплекса, которая выполняется с правами пользователя-клиента и передает запрос службе. Служба проверяет целостность программы-посредника и обрабатывает запрос на основе таблицы предоставляемых полномочий, а затем запускает запрошенную программу с заданными в таблице полномочиями. В этом случае возможно использование псевдонимов программ. Одна и та же программа может иметь несколько псевдонимов для запуска с разными привилегиями. Во втором случае программа-посредник внедряется в системную оболочку, осуществляя перехват системного вызова CreateProcess. Перехват осуществляется подменой функции CreateProcess специальной реализацией, которая передает запрос системной службе. В случае неудачного или ошибочного запроса программа осуществляет вызов процесса, используя штатную версию CreateProcess. Целостность запускаемых программ проверяется с помощью подсчета хэш-сумм и времени последнего изменения. Поэтому при изменении исполняемого файла владелец должен переустановить программу. Следует отметить, что права устанавливать программы, изменяющие полномочия, могут быть предоставлены не всем пользователям, а пользователь, устанавливающий программу, может наделить ее только своими правами.

Таким образом, реализация данного механизма позволяет локально без использования ActiveDirectory реализовать делегирование административных полномочий и предоставить право избирательного делегирования своих полномочий непривилегированным пользователям, что повышает безопасность системы и позволяет снизить нагрузку на администратора системы.

СПИСОК ЛИТЕРАТУРЫ:

1. *Дудаков Н. С., Пирогов Н. Е., Шумилов Ю. Ю.* Кроссплатформенная система безопасного управления хранением динамических данных // Безопасность информационных технологий. 2009. № 3. С. 12–15.
2. *Рихтер Дж., Назар К.* Windows via C++. Программирование на языке Visual C++ / Пер. с англ. М.: Издательство «Русская Редакция»; СПб.: Питер, 2008. – 896 с.

С. Д. Кулик, К. И. Ткаченко, И. А. Лукьянов

ИДЕНТИФИКАЦИЯ ИСПОЛНИТЕЛЯ ТЕКСТОВ ПО ЧАСТОТНО-ГРАММАТИЧЕСКИМ ХАРАКТЕРИСТИКАМ И СИНТАКСИЧЕСКИМ ОСОБЕННОСТЯМ

При разработке систем, являющихся объектами информационной защиты, встают задачи идентификации субъектов — пользователей данных систем (как авторизованных, так и пытающихся получить несанкционированный доступ). Предлагаемый в данной работе подход позволяет получить дополнительные сведения, необходимые для поддержки процесса принятия решения о тождественности субъектов (известного и неизвестного) по разнородным идентифицирующим факторам.