многопоточному исполнению программы, а именно распараллеливанию кода с помощью OpenMP. Также список рассматриваемых угроз был ограничен следующими: программа содержит самомодифицирующийся код, программа осуществляет обращения к случайным участкам памяти, программа вызывает запрещенные или неразрешенные функции.

Метод реализуется в три этапа. Первая часть подразумевает обработку кода в статике и модификацию его таким образом, чтобы во время выполнения автоматически осуществлялась верификация. На этом этапе происходит изменение кода таким образом, что программа во время выполнения следит за переменными и функциями, задействованными в многопоточных участках, и проверяет их значения и параметры [2]. Во время второго каждый пользователь устанавливает ограничения, которые он хотел бы наложить на программы, выполняемые на его оборудовании, таким образом создавая своего рода политику безопасности [3]. Для того чтобы код мог быть выполнен, он должен соответствовать политике безопасности как администратора всей среды, так и пользователя, на хосте которого код будет выполняться непосредственно. Третьим этапом является выполнение кода с мониторингом его самим собой.

На основе метода была реализована тестовая модель, по результатам работы которой можно сделать вывод, что основная цель (проверка без параллельных участков без существенных затрат по ресурсам) была достигнута и время, затрачиваемое на проверку, невелико по сравнению со временем, которое затрачивается на вычисления.

Основным достоинством метода является отсутствие необходимости в установлении отношений доверия между сторонами, что чрезвычайно важно при работе с большим количеством непостоянных клиентов. По результатам тестирования можно сказать, что метод может применяться в условиях, описанных выше, для верификации программного обеспечения означенного типа.

СПИСОК ЛИТЕРАТУРЫ:

1. J. Barnes with Praxis Critical Systems Ltd. High Integrity Software. The Spark Approach to Safety and Security. Addison-Wesley, 2003.
2. *Cousot P., Cousot R.* Abstract Interpretation and Application to Logic Programs // Journal of Logic Programming. 1992. 13.
3. *Heitmeyer C., Archer M., Bharadwaj R., Jeffords R.* Tools for Constructing Requirements Specifications: The SCR Toolset at the Age of Ten // Journal of Computer Systems Science and Engineering, 20(1): 19–35, January 2005.
4. *Foster I., Kasselman C.* The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufman Publishers, 2004.

Е. Ю. Родионов, М. В. Сеник

ПРЕДОТВРАЩЕНИЕ НЕСАНКЦИОНИРОВАННОГО ЗАПУСКА ПО: ПОДХОДЫ К ИДЕНТИФИКАЦИИ ПРИЛОЖЕНИЙ

В работе рассматриваются проблемы, связанные с идентификацией приложений в системах предотвращения несанкционированного запуска ПО. На основе существующих подходов к решению данного вопроса предложен новый метод.

В настоящее время трудно найти организации, не уделяющие внимания защите критически важной информации. В подавляющем большинстве случаев меры по защите информации ограничиваются использованием антивирусных программ, межсетевых экранов и различных средств контроля доступа,

которые предоставляют защиту от внешних угроз. Однако, как показывает статистика [1], угрозы, исходящие изнутри организации, в частности от персонала, намного превосходят внешние по уровню причиненного ущерба. Согласно современным стандартам [2], регламентирующим обеспечение информационной безопасности, наибольшим потенциалом для нанесения ущерба предприятию обладает его собственный персонал. Проблема угроз со стороны инсайдеров актуальна для организаций всех отраслей, об этом говорит возрастающая озабоченность компаний. Что касается самих видов деятельности инсайдеров, то, согласно статистическим данным [1], наиболее опасной угрозой является утечка информации.

Для противодействия подобным угрозам активно используются программно-аппаратные средства защиты, осуществляющие блокирование портов рабочих станций и контролирующие запуск приложений на автоматизированных рабочих местах (АРМ). Подобные средства защиты позволяют решить две задачи: во-первых, предотвратить утечку информации, а во-вторых, если утечка все же произошла, получить доказательства, в результате чего нарушителя можно привлечь к ответственности. Разработка таких систем является отнюдь не тривиальной задачей. Данная работа посвящена средствам защиты, осуществляющим контроль над запуском приложений, и одному из главных вопросов при решении такой задачи — идентификации приложений. Остановимся на данном вопросе и подробно рассмотрим подходы к его решению.

В первую очередь, хотелось бы заметить, что система предотвращения несанкционированного запуска ПО обеспечивает замкнутость программной среды, предоставляя надежную защиту от различных видов нежелательных программ: даже при проникновении в защищаемую среду как локально, так и из сети они не причинят вред из-за того, что не смогут быть запущены в принципе. Для достижения желаемого результата необходимо, прежде всего, выбрать модель разграничения доступа и определить способ идентификации программ.

Первым способом является использование разграничения прав запуска приложений, для определенного каталога и для определенного пользователя. Например, можно разрешить запуск исполняемых файлов, находящихся в некотором каталоге, и в то же время запретить в него запись, т. е. изменение этого каталога и его содержимого. Однако это является недостаточным, так как, например, интерпретатор при запуске сценария осуществляет фактически не запуск, а чтение файла. В то же время пользователи системы для выполнения своих должностных обязанностей должны иметь права на чтение и запись файлов, следовательно, они могут разместить сценарий в определенном каталоге и выполнить его с помощью интерпретатора. Таким образом, возникает необходимость рассматривать запущенные в системе процессы не только как объекты, но и как субъекты доступа. Такой подход позволяет разрешить процессам интерпретатора чтение только из одного каталога. В этом случае мы получим механизм защиты, аналогичный механизму ограничения программной среды, так как множество всех исполняемых файлов и сценариев будет ограничено одним или несколькими каталогами, а пользователи не смогут изменять их содержимое. Однако данный метод ставит задачу определения необходимых для нормальной работы процесса файлов и, соответственно, каталогов.

Ограничение программной среды на основе размещения в специальных каталогах, разрешенных к запуску, исполняемых файлов не предоставляет гибкости при администрировании и конфигурировании системы защиты. Более удобным и, наверное, более естественным способом было бы составление в явном виде списка доступных для запуска приложений. Такой подход обеспечивает удобство не только при администрировании системы безопасности, но и при использовании приложения на рабочем месте сотрудников, так как разрешенное приложение может быть запущено, даже если потребуется перенести соответствующий исполняемый файл. Т. е. управление рабочими местами сотрудников и управление системой защиты оказываются разделены. Однако для корректного

создания списка необходимо определить свойства файла, по которым можно было бы однозначно его идентифицировать. Очевидно, что для этого недостаточно использовать имя файла.

Наиболее подходящей характеристикой является хэш-код исполняемого файла. Проверка хэш-кода исключает возможность подмены и поэтому обладает высокой надежностью. При этом контролируется именно содержание файла, что в конечном счете и является целью контроля запуска. Кроме того, при хранении списка хэш-кодов нет необходимости в обеспечении его секретности, и поэтому он может передаваться в открытом виде на клиентские машины. Также хранение нескольких хэш-кодов для разных версий программ позволяет контролировать обновление программного обеспечения.

Недостатком такой схемы идентификации приложений является необходимость затрачивать ресурсы системы для вычисления хэш-функции, и это тем чувствительнее для производительности системы, чем чаще необходимо запускать те или иные процессы или подгружать библиотеки. Особенно это касается библиотек, так как наиболее часто используемые библиотеки могут проецироваться в адресное пространство большого количества процессов.

Методом, смежным с приведенным, является проверка электронно-цифровой подписи исполняемых файлов и библиотек, позволяющая, не затрачивая большой объем памяти на хранение хэш-кодов, указывать как доверенные большое число приложений. Причем администратору системы безопасности нет необходимости подробно ознакомляться с работой каждого продукта доверенного производителя, что существенно облегчает работу по настройке системы контроля запуска приложений. В то же время этот подход ставит задачу организации доверия к сертификату производителя файла и должен опираться на инфраструктуру открытых ключей. У этого метода тот же недостаток, что и у проверки хэш-кода, — падение производительности.

Из вышесказанного следует, что ни один из приведенных методов не удовлетворяет полностью требованиям по влиянию на производительность защищаемой системы, надежности и удобству конфигурирования системы контроля доступа. Поэтому более предпочтителен подход, основанный на комбинации различных методов. Его суть состоит в минимальном использовании ресурсоемких средств (хэш-функций и цифровой подписи). При идентификации файла оптимизация возможна только в случае его несоответствия списку разрешенных. Т. е. первыми должны быть проверены наиболее простые характеристики: имя, размер и т. п., и только в случае успеха должен быть проверен хэш-код. Вообще, для выбора метода контроля для различных файлов можно разделить все исполняемые файлы, сценарии и библиотеки на три группы:

– часто загружаемые в память;

– исполняемые файлы ОС и другие исполняемые файлы, изменение которых (перемещение, удаление, обновление) происходит редко;

– исполняемые файлы, непосредственно связанные с производственной деятельностью сотрудников, и прочие исполняемые файлы.

Таким образом, при создании политики запуска приложений следует начинать с контроля на основе доверенных каталогов. Этим методом, прежде всего, должны быть охвачены стандартные библиотеки и другие файлы первой и второй групп. В силу того, что этот подход к проверке наиболее производителен, его нужно использовать настолько широко, насколько это возможно, но не в ущерб удобству работы с системой безопасности, учитывая возможность изменения структуры каталогов и доверия к исполняемым файлам. Для оставшейся части файлов должна быть создана политика, основанная на перечне разрешенных к запуску файлов.

Таким образом, модель разграничения доступа и идентификации исполняемых файлов определяет архитектуру системы предотвращения несанкционированного запуска ПО. Предложенный метод идентификации сочетает в себе преимущества существующих подходов и в то же время позволяет избежать некоторых неудобств, связанных с ними.

СПИСОК ЛИТЕРАТУРЫ:

1. Инсайдерские угрозы в России `09. URL: http://perimetrix.ru/downloads/rp/PTX_Personal_Data_2009.pdf.
2. СТО БР ИББС-1.0-2008. Обеспечение информационной безопасности организаций банковской системы Российской Федерации. Общие положения.

Е. Ю. Родионов, М. В. Сеник

ПРЕДОТВРАЩЕНИЕ НЕСАНКЦИОНИРОВАННОГО ЗАПУСКА ПО: ПОДХОДЫ К ИДЕНТИФИКАЦИИ ПРИЛОЖЕНИЙ

В работе рассматриваются проблемы, связанные с идентификацией приложений в системах предотвращения несанкционированного запуска ПО. На основе существующих подходов к решению данного вопроса предложен новый метод.

В настоящее время трудно найти организации, не уделяющие внимания защите критически важной информации. В подавляющем большинстве случаев меры по защите информации ограничиваются использованием антивирусных программ, межсетевых экранов и различных средств контроля доступа, которые предоставляют защиту от внешних угроз. Однако, как показывает статистика [1], угрозы, исходящие изнутри организации, в частности от персонала, намного превосходят внешние по уровню причиненного ущерба. Согласно современным стандартам [2], регламентирующим обеспечение информационной безопасности, наибольшим потенциалом для нанесения ущерба предприятию обладает его собственный персонал. Проблема угроз со стороны инсайдеров актуальна для организаций всех отраслей, об этом говорит возрастающая озабоченность компаний. Что касается самих видов деятельности инсайдеров, то, согласно статистическим данным [1], наиболее опасной угрозой является утечка информации.

Для противодействия подобным угрозам активно используются программно-аппаратные средства защиты, осуществляющие блокирование портов рабочих станций и контролирующие запуск приложений на автоматизированных рабочих местах (АРМ). Подобные средства защиты позволяют решить две задачи: во-первых, предотвратить утечку информации, а во-вторых, если утечка все же произошла, получить доказательства, в результате чего нарушителя можно привлечь к ответственности. Разработка таких систем является отнюдь не тривиальной задачей. Данная работа посвящена средствам защиты, осуществляющим контроль над запуском приложений, и одному из главных вопросов при решении такой задачи — идентификации приложений. Остановимся на данном вопросе и подробно рассмотрим подходы к его решению.

В первую очередь, хотелось бы заметить, что система предотвращения несанкционированного запуска ПО обеспечивает замкнутость программной среды, предоставляя надежную защиту от различных видов нежелательных программ: даже при проникновении в защищаемую среду как локально, так и из сети они не причинят вред из-за того, что не смогут быть запущены в принципе. Для достижения желаемого результата необходимо, прежде всего, выбрать модель разграничения доступа и определить способ идентификации программ.

Первым способом является использование разграничения прав запуска приложений, для определенного каталога и для определенного пользователя. Например, можно разрешить запуск исполняемых файлов, находящихся в некотором каталоге, и в то же время запретить в него запись,