УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2015 Том 15 № 3 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2015 Vol. 15 No 3 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.056.53

АБСТРАКТНЫЕ МОДЕЛИ ВИРТУАЛИЗАЦИИ СИСТЕМЫ

М.Г. Ковешников[a], К.А. Щеглов[b], А.Ю. Щеглов[b]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
[b] ООО «НПП «Информационные технологии в бизнесе», Санкт-Петербург, 194044, Российская Федерация
Адрес для переписки: info@npp-itb.spb.ru

Информация о статье
Поступила в редакцию 08.04.15, принята к печати 17.04.15
doi:10.17586/2226-1494-2015-15-3-483-492
Язык статьи – русский
Ссылка для цитирования: Ковешников М.Г., Щеглов К.А., Щеглов А.Ю. Абстрактные модели виртуализации системы // Научно-технический вестник информационных технологий, механики и оптики. 2015. Т. 15. № 3. С. 483–492.

Аннотация
Представлены результаты исследования защиты системных объектов – системных файлов и файлов пользовательских конфигураций системы и приложений – от несанкционированного к ним доступа, в том числе с целью реализации защиты от атак на отказ в обслуживании. Предложен метод и разработаны абстрактные модели виртуализации системы, на которых исследованы сценарии атак для различных режимов виртуализации. Дана оценка эффективности технологии виртуализации системных средств. Предлагаемая технология основана на реализации перенаправлений запросов доступа к разделяемым между субъектами системным объектам. Смоделированы режимы полной и частичной виртуализации системы: в режиме полной виртуализации создаются копии всех системных объектов доступа, на которые перенаправляются запросы субъектов, включая соответствующие объекты приложений; в режиме частичной виртуализации создаются соответствующие копии лишь части системы, например, только системные объекты приложений. Применительно к различным сценариям атак оценена эффективность альтернативных решений. Рассмотрено запатентованное апробированное техническое решение, реализующее метод виртуализации системы для операционных систем семейства Microsoft Windows, на примере которого показаны возможности и простота администрирования соответствующим образом построенных средств защиты системных объектов. Подтверждена практическая значимость предложенного метода защиты.

Ключевые слова
информационная безопасность, системный объект, защита, отказ в обслуживании, виртуализация системного средства, сценарий атаки, абстрактная модель.

ABSTRACT MODELS FOR SYSTEM VIRTUALIZATION

M.G. Koveshnikov[a], K. A. Shcheglov[b], A. Yu. Shcheglov[b]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
[b] OOO «Scientific Production Enterprise "Information technologies in business», Saint Petersburg, 194044, Russian Federation
Corresponding author: info@npp-itb.spb.ru

Article info
Received 08.04.15, accepted 17.04.15
doi:10.17586/2226-1494-2015-15-3-483-492
Article in Russian
For citation: Koveshnikov M.G., Shcheglov K.A., Shcheglov A.Yu. Abstract models for system virtualization. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol.15, no. 3, pp. 483–492.

Abstract
The paper is dedicated to issues of system objects securing (system files and user system or application configuration files) against unauthorized access including denial of service attacks. We have suggested the method and developed abstract system virtualization models, which are used to research attack scenarios for different virtualization modes. Estimation for system tools virtualization technology effectiveness is given. Suggested technology is based on redirection of access requests to system objects shared among access subjects. Whole and partial system virtualization modes have been modeled. The difference between them is the following: in the whole virtualization mode all copies of access system objects are created whereon subjects' requests are redirected including corresponding application objects; in the partial virtualization mode corresponding copies are created only for part of a system, for example, only system objects for applications. Alternative solutions effectiveness is valued relating to different attack scenarios. We consider proprietary and approved technical solution which implements system virtualization method for Microsoft Windows OS family. Administrative simplicity and capabilities of correspondingly designed system objects security tools are illustrated on this example. Practical significance of the suggested security method has been confirmed.

Keywords
informational security, system object, security, denial of service, system tool virtualization, attack scenario, abstract model.

Введение

Для современных корпоративных информационных систем характерна одновременная работа на различных уровнях конфиденциальности. Как правило, один и тот же сотрудник на одном и том же компьютере должен обрабатывать информацию различных уровней конфиденциальности, в различных режимах (сессиях) [1] с различными требованиями к защите информации. Для обеспечения необходимого уровня информационной безопасности в этом случае возможны два решения:

– изолирование сессий обработки информации на защищаемом компьютере, где сессия определяется учетной записью, что является единственно возможным корректным решением для сессионного контроля доступа [1];
– организация такого режима работы приложений, который бы минимизировал риск потерь от успешной атаки на критичную сессию (менее защищенную, например, созданную для обработки открытой информации, в том числе предполагающую неконтролируемый доступ к ресурсам Интернет) или на критичное приложение.

Для изолирования сессий по данным (в том числе с целью предотвращения утечек конфиденциальной информации в результате реализации на том же компьютере незащищенной обработки открытой информации) успешно может применяться технология контроля доступа к создаваемым объектам – как к создаваемым файлам, так и к данным, помещаемым в буфер обмена [2, 3]. Такая технология основана на реализации запатентованного технического решения [4].

С системными файловыми объектами ситуация значительно сложнее, хотя задача защиты от отказа в обслуживании, связанная с несанкционированным удалением или модификацией системных файлов и файлов пользовательских конфигураций, не менее актуальна. Реализация разграничительной политики доступа к системным объектам путем настройки прав доступа пользователей к системным файлам и к файлам пользовательских конфигураций представляет собою крайне трудоемкую практическую задачу [5]. С этой целью можно использовать технологию виртуализации машин, предполагающую использование избыточной вычислительной мощности серверов [6–8]. Эффект защиты в данном случае достигается за счет того, что для каждого пользователя может быть создана своя виртуальная машина [9, 10]. Однако эта технология крайне избыточна и существенно загружает вычислительные ресурсы. Кроме того, ее прямое назначение – совсем иное, а решение задачи защиты в данном случае можно рассматривать лишь как некий дополнительный эффект.

Для решения задачи защиты в настоящей работе предлагается технология виртуализации системных средств.

Технология виртуализации системных средств

Под виртуальной системой будем понимать систему, состоящую из копий системных файлов, создаваемую для субъекта доступа, сессия которого изолируется, на которую перенаправляются запросы доступа субъекта к оригинальной (базовой) системе.

Сначала рассмотрим предлагаемую технологию в общем случае. Пусть исходно на системном диске (C:) установлена операционная система (ОС) и приложения, и пусть в системе заведены два интерактивных пользователя User 1 и User 2, сессии которых требуется разделить.

Создадим копии системного диска C: на дисках D: и E: Для этого скопируем соответствующие системные и скрытые файлы, что достаточно просто сделать. С учетом выполненной процедуры копирования далее будем говорить, что на диске C: установлена базовая система, на дисках D: и E: – созданные нами виртуальные системы.

Заметим, что виртуальные системы не обязательно создавать на отдельных дисках, можно их создать в отдельных каталогах того же диска C:.

Теперь зададим правила перенаправления запросов к файловым объектам. Для пользователя User 1 это будет правило перенаправления доступа к диску C: на диск D:, для User 2 – правило перенаправления доступа к диску C: на диск E:. Реализуя разграничительную политику доступа к файловым объектам, запретим пользователю User 1 доступ к диску E:, а пользователю User 2 – к диску D:. В результате этого получим схему виртуализации системы, приведенную на рис. 1.

Рассмотрим работу системы. С базовой системой может взаимодействовать только системный пользователь, запросы к базовой системе интерактивных пользователей User 1 и User 2 перенаправляются к соответствующим виртуальным системам. Таким образом, если пользователь User 1 запускает с базовой системы какое-либо приложение, например браузер, реально оно будет запущено с диска D:. При последующей работе приложения любое его обращение с правами пользователя User 1 к базовой системе будет перенаправляться к соответствующей виртуальной системе. При этом приложение, исходно инсталлированное на диск C:, будет работать корректно, поскольку все перенаправления его запросов доступа «прозрачны» для приложения, в частности, корректно будут сохраняться временные данные, cookies, настройки, кэшированная информация и т.д.

Рис. 1. Схема виртуализации системы: ПО – программное обеспечение средства защиты

Отметим, что если пользователь User 2 запускает с базовой системы какое-либо приложение, например, тот же браузер, реально оно будет запущено уже с диска E:. При этом это приложение, инсталлированное на диск C:, будет работать корректно, поскольку все перенаправления его запросов доступа «прозрачны» для приложения, но в этом случае они уже будут сохраняться на диске E:.

Соответствующим образом запретим пользователю доступ User 1 к диску E: (с этим диском работает пользователь User 2), а пользователю User 2 – к диску D:.

Таким образом, получаем полную изолированность сессий по пользователям, в том числе и изолированность системного пользователя; успешная атака, осуществленная на приложение, запущенное каким-либо интерактивным пользователем, не приведет к несанкционированной модификации (или удалению элементов) как базовой системы, так и виртуальных систем иных пользователей. Важно, что в результате виртуализации системы мы всегда имеем доверенную операционную систему (базовую систему), с которой осуществляется загрузка и с которой может быть восстановлена виртуальная система, подвергшаяся успешной атаке, поскольку с правами интерактивного пользователя, под которыми запускаются приложения, доступ к ней осуществить невозможно.

Достоинства данной технологии для решения рассматриваемой задачи защиты достаточно очевидны: запускаются только одна операционная система (базовая система) и только одно средство защиты, реализующее перенаправление запросов доступа к файловым объектам (рис. 1), что практически не приводит к дополнительной загрузке вычислительных ресурсов и не усложняет задачу администрирования. Как видим, принципиально меняется и собственно идея виртуализации: запускаются не отдельные виртуальные машины для пользователей, а только одна система, виртуализуется лишь та часть системы (для каждого пользователя, включая системного, – своя), с которой непосредственно должен работать пользователь.

Техническое решение, реализующее используемый для виртуализации системы метод перенаправления запросов доступа к неразделяемым файловым объектам, авторами запатентовано [11], практически реализовано в комплексной системе защиты информации «Панцирь+» для ОС Microsoft Windows (далее будем использовать для иллюстраций интерфейсы, разработанные для этой системы защиты) [12] и апробировано.

Интерфейс задания и отображения заданных правил перенаправления запросов доступа к файловым объектам, реализованных в данной системе защиты, показан на рис. 2.

В качестве объектов доступа, в отношении которых будет применено заданное правило перенаправления запросов доступа, могут выступать файлы, каталоги, логические диски.

Для упрощения задачи администрирования в качестве субъекта доступа в правилах перенаправления используется сущность «профиль» (рис. 2). В один профиль объединяются субъекты, для которых назначаются одинаковые правила перенаправления к файловым объектам.

Отметим, что техническое решение [11] исходно разработано и используется для разделения между субъектами доступа не разделяемых системой и приложениями файловых объектов, например, каталогов, используемых для временного хранения файлов. Без подобного решения невозможно корректно реализовать разграничительную политику доступа, в частности, разделительную, предполагающую полное изолирование сессий – режимов обработки информации различных уровней конфиденциальности.

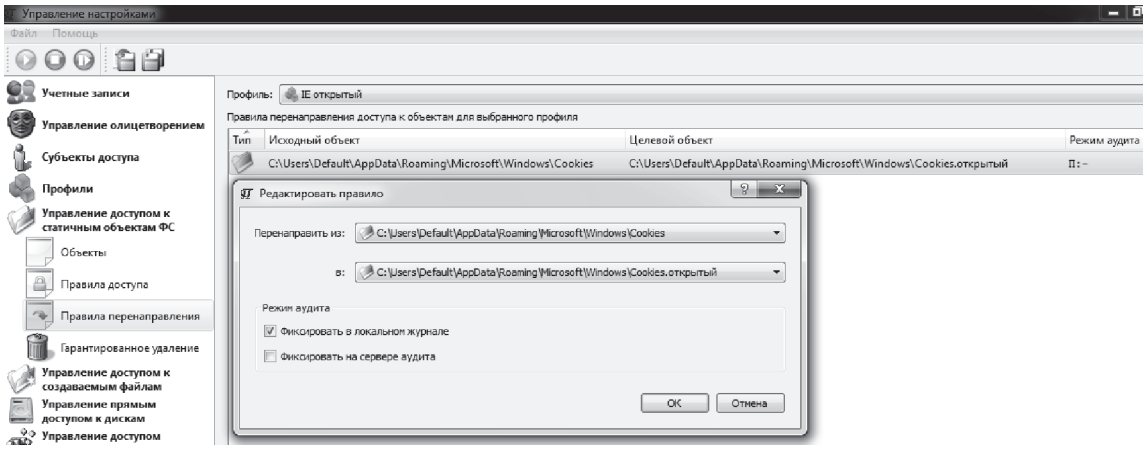

Рис. 2. Интерфейс задания и отображения заданных правил перенаправления запросов доступа к файловым объектам

Абстрактная модель системы

Построим абстрактные модели системы и виртуальной системы, на которых рассмотрим различные стратегии атак, и оценим эффективность предлагаемой технологии виртуальных системных средств для решения рассматриваемой задачи защиты.

Пусть на операционной системе OS имеются следующие субъекты (Subjects):

- *sp* – процесс системы;
- *adm* – администратор;
- u_1 – первый пользователь;
- u_2 – второй пользователь.

Это можно записать следующим образом: $Subjects = \{sp, adm, u_1, u_2\}$.

Также на OS установлены следующие объекты (*Objects*):

- *S* – системные файлы, которые содержат:
  - *SE* – исполняемые файлы операционной системы,
  - *CF* – конфигурационные файлы (различные текстовые и бинарные файлы);
- *P* – пользовательские приложения;
- *UF* – пользовательские файлы.

Тогда $Objects = S \cup P \cup UF$, где $S = SE \cup CF$; $P = \{p_1,\ldots, p_N\}$, где p_i – установленное приложение $i \in 1..N$, *N* – количество приложений, установленных в системе; $SE = \{se_1,\ldots, se_M\}$, где se_i – установленное системное приложение $i \in 1..M$; $CF = \{cf_1,\ldots,cf_K\}$, где cf_i – конфигурационный файл, $i \in 1..K$; $UF = \{uf_1,\ldots,uf_J\}$, где uf_i – пользовательский файл, $i \in 1..J$.

При работе с операционной системой OS пользователь u_1 имеет область доступа $CONF_1$ (конфигурация первого пользователя), куда входят:

- P_1 – подмножество программ *P*, доступное для u_1;
- CF_1 – подмножество конфигурационных файлов *CF*, доступное для u_1;
- UF_1 – подмножество *UF*: файлы, хранящие пользовательские настройки пользователя u_1.

Аналогично пользователь u_2 имеет область доступа $CONF_2$ (конфигурация второго пользователя), куда входят:

- P_2 – подмножество *P*, доступное для u_2;
- CF_2 – подмножество *CF*, доступное для u_2;
- UF_2 – файлы, хранящие пользовательские настройки пользователя u_2.

Это можно записать следующим образом:

$CONF_1 = P_1 \cup CF_1 \cup UF_1$; $CONF_2 = P_2 \cup CF_2 \cup UF_2$.

Следует отметить, что в общем случае множества $P_1 \cap P_2$, $CF_1 \cap CF_2$, $UF_1 \cap UF_2$ могут быть непустыми: $P_1 \cap P_2 \neq \varnothing$ (используются те же программы), $CF_1 \cap CF_2 \neq \varnothing$ (для учетных записей не разделяются конфигурационные файлы, например, для приложения), $UF_1 \cap UF_2 \neq \varnothing$ (не разделяются пользовательские настройки, например, для приложения).

Обозначим символом $x \rightarrow A$ отношение, состоящее в том, что для любого объекта из *A* субъект *x* может читать этот объект и, если это исполняемый файл, может его запускать:

$x \rightarrow A: \forall a \in A$, *x* может прочесть (исполнить) *a*, где $x \in Subjects$, $A \subseteq Objects$.

Обозначим символом $x \Rightarrow A$ отношение, состоящее в том, что для любого объекта из *A* субъект *x* может читать и изменять этот объект:

$x \Rightarrow A$: $\forall a \in A$, x может прочесть (и исполнить) и изменить a, где $x \in$ Subjects, $A \subseteq Objects$.

Из приведенных обозначений следует, что если $x \Rightarrow A$, то $x \rightarrow A$, обратное неверно. Тогда можно записать правила доступа всех субъектов следующим образом:

$sp \Rightarrow Objects$

$adm \Rightarrow Objects$

$u_1 \Rightarrow CONF_1$ и $u_1 \rightarrow SE \cup (P \setminus P_1)$

$u_2 \Rightarrow CONF_2$ и $u_2 \rightarrow SE \cup (P \setminus P_2)$

Каждому пользователю доступна для записи своя конфигурация, а для чтения и запуска – системные и пользовательские приложения, установленные на систему.

Рассмотрим различные сценарии атак.

Сценарий 1. Пользователь u_1 является злоумышленником.

Пользователь u_1 может модифицировать $CONF_1$, которая состоит из P_1, UF_1, CF_1. Так как $P_1 \cap P_2 \neq \varnothing$, $CF_1 \cap CF_2 \neq \varnothing$, $UF_1 \cap UF_2 \neq \varnothing$, т.е. могут существовать программы, конфигурационные файлы и пользовательские файлы, общие для нескольких пользователей, то пользователь u_1 может изменить настройки пользователя u_2.

Сценарий 2. Происходит заражение системного исполняемого файла операционной системы.

Пусть заражен некоторый системный исполняемый файл $se_1 \in SE$. Так как $u_1 \rightarrow SE$ и $u_2 \rightarrow SE$, то $u_1 \rightarrow se_1$ и $u_2 \rightarrow se_1$. В результате оба пользователя оказываются под воздействием заражения и используют зараженный файл se_1.

Сценарий 3. Происходит заражение пользовательского исполняемого файла.

Пусть заражен некоторый пользовательский исполняемый файл $p_1 \in P$. Если только один пользователь имеет доступ к этому файлу, т.е. $p_1 \in P_1 \setminus P_2$ или $p_1 \in P_2 \setminus P_1$, тогда только один из пользователей u_1 или u_2 соответственно оказываются под влиянием зараженного файла. Если же исполняемый файл доступен обоим пользователям, т.е. $p_1 \in P_1 \cap P_2$, то оба пользователя оказываются под воздействием зараженного файла.

Сценарий 4. Кража информации из конфигурационных файлов.

Пусть пользователь u_1 запускает исполняемый файл $e_1 \in SE \cup P_1$, который является зараженным. Тогда приложение e_1 выполняется с правами пользователя u_1 и имеет доступ к следующим файлам: $e_1 \Rightarrow CONF_1$ (где $CONF_1 = P_1 \cup CF_1 \cup UF_1$) и $e_1 \rightarrow SE \cup (P \setminus P_1)$. Так как $P_1 \cap P_2$, $CF_1 \cap CF_2$, $UF_1 \cap UF_2$ могут быть непустыми, может произойти кража данных не только одного пользователя, но и другого пользователя, не запускавшего зараженный файл.

Сценарий 5. Оказывается заражен исполняемый файл, выполняемый системным процессом *sp* или администратором.

Пусть файл $f \in SE$ является зараженным, далее он запускается системным процессом или администратором. Так как и администратор, и системный процесс имеют полный доступ к системе ($sp \Rightarrow Objects$, $adm \Rightarrow Objects$), то вся система оказывается доступна для процесса, запущенного из файла f.

Абстрактные модели виртуализации системы

Полная виртуализация системы. При реализации технологии виртуализации системного средства исходная абстрактная модель системы изменяется следующим образом.

Субъекты системы остаются теми же: $Subject = \{sp, adm, u_1, u_2\}$. Однако, так как виртуализация системы предполагает создание копий, на которые перенаправляются запросы пользователя, то множество объектов операционной системы меняется. А именно, для каждого пользователя создается своя копия системы: $Objects_{all} = Objects \cup Objects_1 \cup Objects_2$ – множество всех объектов, установленных на операционной системе. При этом $Objects = S \cup P \cup UF$ – множество объектов, изначально установленных на систему. В случае полного копирования файлов операционной системы копии этих файлы попадают во все множества файлов, доступные для каждого из пользователей u_1 и u_2. В случае частичного копирования только некоторое подмножество этих файлов попадает во множества доступных файлов для пользователей u_1 и u_2. Здесь подмножества определяются таким же образом, как и без использования виртуальных систем, т.е. $S = SE \cup CF$, а $P = \{p_1, \ldots, p_N\}$, где p_i – установленное приложение $i \in 1..N$, N – количество приложений, установленных в системе. $SE = \{se_1, \ldots, se_M\}$, где se_i – установленное системное приложение $i \in 1..M$, $CF = \{cf_1, \ldots, cf_K\}$, cf_i – конфигурационный файл, $i \in 1..K$, $UF = \{uf_1, \ldots, uf_J\}$, uf_i – пользовательский файл, $i \in 1..J$.

Рассмотрим случай полного копирования файлов операционной системы (полная виртуализация системы) и опишем множество объектов, доступных для пользователей u_1 и u_2.

Обозначим $\approx$ отношение, состоящее в том, что один объект является копией другого. Пусть объект o_1 был создан операцией копирования объекта o_2, тогда $o_1 \approx o_2$.

$Objects_1 = S_1 \cup P_1 \cup UF_1$, где $S_1 \approx S$, $P_1 \approx P$, $UF_1 \approx UF$.

$Objects_2 = S_2 \cup P_2 \cup UF_2$, где $S_2 \approx S$, $P_2 \approx P$, $UF_2 \approx UF$.

Здесь $S_t = SE_t \cup CF_t$, $t \in \{1,2\}$ – индекс пользователя (для u_1: $t = 1$, для u_2: $t = 2$), а $P_t = \{p_{t1},\ldots, p_{tN}\}$, где p_{ti} – установленное приложение $i \in 1..N$, N – количество приложений, установленных в виртуальной системе для пользователя t. $SE_t = \{se_{t1},\ldots, se_{tM}\}$, где se_{ti} – установленное системное приложение $i \in 1..M$, $CF_t = \{cf_{t1},\ldots,cf_{tK}\}$, где cf_{ti} – конфигурационный файл, $i \in 1..K$, $UF_t = \{uf_{t1},\ldots,uf_{tJ}\}$, uf_{ti} – пользовательский файл, $i \in 1..J$.

Следующим этапом организации работы в виртуальной системе являются правила перенаправления (которые также будем обозначать ≈). Пусть выполняются перенаправления всех операций чтения и (или) записи файлов из множества O_1 на их копию O_2 | $O_2 \approx O_1$ для пользователя u:

$$O_1 \xrightarrow{L,u} O_2,$$

где $L = \{r\}$, или $L = \{w\}$, или $L = \{r,w\}$ – множество типов операций, r – операция чтения, w – операция записи. Тогда можно записать все правила перенаправления для исходной системы в виртуальные системы в следующем виде:

$$Objects \xrightarrow{\{r,w\},u_1} Objects_1,$$

$$Objects \xrightarrow{\{r,w\},u_2} Objects_2.$$

Для каждого из пользователей u_1, u_2 выделяется своя копия системы, и все запросы пользователя на чтение и запись перенаправляются на нее.

Учитывая, что перенаправление выполняется до проверки прав доступа к объекту, для пользователей u_1 и u_2 можно назначить следующие права:

$u_1 \Rightarrow Objects_1$,

$u_2 \Rightarrow Objects_2$.

Рассмотрим сценарии атак применительно к виртуальным системам с полной виртуализацией.

Сценарий 1. Пользователь u_1 является злоумышленником.

Пользователь u_1 может модифицировать множество файлов $Objects_1$, которое является копией множества $Objects$. Однако конфигурация пользователя u_2 остается недоступной для пользователя u_1, так как $Objects_1 \cap Objects_2 = \varnothing$. В сравнении с обычным использованием системы применение виртуальных систем позволяет обезопасить пользователей системы от злоумышленной модификации одним из пользователей чужих конфигураций.

Сценарий 2. Происходит заражение системного исполняемого файла операционной системы.

Пусть заражен некоторый системный исполняемый файл $se_1 \in SE$, $SE \subseteq Objects$. Так как $u_1 \Rightarrow Objects_1$, $u_2 \Rightarrow Objects_2$ и ни один из пользователей не имеет доступа к $Objects$, то для каждого из пользователей u_1 и u_2 имеется своя копия файла se_1, а именно: $se_{11} \approx se_1$ и $se_{12} \approx se_1$, где $se_{11} \in SE_1$, $se_{12} \in SE_2$, $SE_1 \subseteq Objects_1$, $SE_2 \subseteq Objects_2$. Правила перенаправления настроены таким образом, что при обращении обоих пользователей к любому файлу из $Objects$ доступ фактически осуществляется к файлу из копии, специально созданной для каждого из них. Поэтому каждый пользователь обращается к файлам se_{11} и se_{12}, которые не были заражены. Каждый из пользователей имеет доступ только к своей копии файла se_1:

$u_1 \rightarrow se_{11}$ и $u_2 \rightarrow se_{12}$.

$$se_1 \xrightarrow{\{r,w\},u_1} se_{11},$$

$$se_1 \xrightarrow{\{r,w\},u_2} se_{12}.$$

В результате ни один из пользователей не оказывается под воздействием заражения, так как они используют копии файла se_1. В сравнении с использованием оригинальной системы пользователи оказываются защищенными от данного типа угроз.

Сценарий 3. Происходит заражение пользовательского исполняемого файла.

Пусть заражен некоторый пользовательский исполняемый файл $p_1 \in P$. Так как $P \subseteq Objects$, то ни один из пользователей u_1, u_2 не имеет доступа к файлу p_1. Ввиду включенного перенаправления каждый из них использует свою копию этого файла $p_{11} \approx p_1$, $p_{12} \approx p_1$:

$$p_1 \xrightarrow{\{r,w\},u_1} p_{11},$$

$$p_2 \xrightarrow{\{r,w\},u_2} p_{12}.$$

Следовательно, ни один из пользователей не оказывается под влиянием заражения. По сравнению с использованием оригинальной системы отсутствует вероятность того, что зараженный файл используют один или более пользователей системы.

Рассмотрим другой вариант этого сценария, при котором зараженный файл находится в одной из копий системы. Пусть заражен пользовательский исполняемый файл p_{11} из копии $Objects_1$ одного из пользователей u_1: $p_{11} \in P_1$, $P_1 \subseteq Objects_1$, $p_{11} \approx p_1$, где $p_1 \in P$, $P \subseteq Objects$. Единственный пользователь, который имеет доступ к этому файлу – это u_1: $u_1 \rightarrow p_{11}$. Так как пользователь u_2 не имеет доступа к файлу p_{11} и имеет свою копию этого файла p_{12}, $p_{12} \approx p_1$, то он работает только со своей копией файла и не оказывается под

влиянием заражения. Таким образом, виртуальные системы локализуют заражение и предотвращают распространение заражения на других пользователей системы.

Сценарий 4. Кража информации из конфигурационных файлов

Пусть пользователь u_1 запускает исполняемый файл, который является зараженным, $e_{11} \in SE_1 \cup P_1$, $SE_1 \subseteq Objects_1$, $P_1 \subseteq Objects_1$. Тогда приложение e_{11} выполняется с правами пользователя u_1 и имеет доступ к файлам $e_{11} \Rightarrow Objects_1$. Таким образом, под воздействием оказываются только файлы пользователя u_1, файлы остальных пользователей, например пользователя u_2, недоступны, так как хранятся в $Objects_2$, к которому у пользователя u_1 нет доступа. Таким образом, виртуальные системы устраняют вероятность кражи информации о чужой конфигурации.

Сценарий 5. Оказывается заражен исполняемый файл, выполняемый системным процессом *sp* или администратором.

Пусть файл $f \in SE$ является зараженным. Он запускается системным процессом или администратором. Так как и администратор, и системный процесс имеют полный доступ к системе ($sp \Rightarrow Objects_{all}$, $adm \Rightarrow Objects_{all}$), то вся система оказывается доступной для процесса, запущенного из файла *f*. Однако процессу *f* необходимо определить, что применяются виртуальные системы и что необходимо заразить файлы копии. Это не обеспечивает надежную защиту, но, тем не менее, создает возможность сохранить файлы копий нетронутыми.

Частичная виртуализация системы. Частичная виртуализация предполагает создание копий с последующим перенаправлением к ним запросов доступа субъектов не всего системного средства, а некоторой его части. При реализации частичной виртуализации абстрактная модель виртуальной системы изменяется следующим образом.

Субъекты системы остаются теми же: $Subjects = \{sp, adm, u_1, u_2\}$. Однако, так как виртуализация системы предполагает создание копий части системы, на которые перенаправляются запросы пользователей, то множество объектов операционной системы меняется. Для каждого пользователя создается своя копия части системы: $Objects_{all} = Objects \cup Copy_1 \cup Copy_2$ – множество всех объектов, установленных на операционной системе.

Множество объектов, изначально установленных на систему, определяется аналогично предыдущим случаям: $Objects = S \cup P \cup UF$. Так как выполняется частичное копирование, то только часть файлов попадает в копию для каждого из пользователей u_1 и u_2. Какие файлы переносить в копию, а какие – нет, определяется необходимым уровнем изоляции конфигураций пользователей.

$Copy_1$ и $Copy_2$ состоят из файлов, скопированных из множества *Objects*. Обозначим подмножества файлов, которые были скопированы из множества *Objects*, как $Copy_{i1}$ и $Copy_{i2}$, $Copy_{i1} \subseteq Objects$, $Copy_{i2} \subseteq Objects$, тогда $Copy_1 \approx Copy_{i1}$ и $Copy_2 \approx Copy_{i2}$. Если же копии $Copy_{i1}$ и $Copy_{i2}$ совпадают с множеством *Objects* объектов, изначально установленных на систему, то это – случай полного копирования.

Опишем подмножества файлов из изначальной системы, которые копируются для пользователей u_1 и u_2 соответственно, следующим образом:

- $Copy_{i1} = S_{i1} \cup P_{i1} \cup UF_{i1}$; каждое из множеств S_{i1}, P_{i1}, UF_{i1} может быть пустым;
- $Copy_{i2} = S_{i2} \cup P_{i2} \cup UF_{i2}$; каждое из множеств S_{i2}, P_{i2}, UF_{i2} также может быть пустым.

Отметим, что копируемые файлы могут как быть совершенно различными, так и совпадать для разных пользователей, т.е. $Copy_{i1} \cap Copy_{i2} \supseteq \varnothing$. Тогда частичные копии для пользователей u_1 и u_2 можно описать так:

$Copy_1 = S_{c1} \cup P_{c1} \cup UF_{c1}$, где $S_{c1} \approx S_{i1}$, $P_{c1} \approx P_{i1}$, $UF_{c1} \approx UF_{i1}$,

$Copy_2 = S_{c2} \cup P_{c2} \cup UF_{c2}$, где $S_{c2} \approx S_{i2}$, $P_{c2} \approx P_{i2}$, $UF_{c2} \approx UF_{i2}$.

Также для использования виртуальных систем следует назначить правила перенаправления. Перенаправление осуществляется таким образом, что все операции пользователя к подмножеству скопированных файлов оригинальной системы фактически адресуются к копии, созданной для этого пользователя. Правила перенаправления для этого случая запишем следующим образом:

$$Copy_{i1} \xrightarrow{\{r,w\},u_1} Copy_1,$$

$$Copy_{i2} \xrightarrow{\{r,w\},u_2} Copy_2.$$

Учитывая, что перенаправление выполняется до проверки прав доступа к объекту, для пользователей u_1 и u_2 назначаются следующие права:

$u_1 \Rightarrow Copy_1 \cup CF_1 \setminus CF_{i1} \cup P_1 \setminus P_{i1} \cup UF_1 \setminus UF_{i1}$ и $u_1 \rightarrow SE \setminus SE_{i1} \cup (P \setminus P_{i1}) \setminus P_1$,

$u_2 \Rightarrow Copy_2 \cup CF_2 \setminus CF_{i2} \cup P_2 \setminus P_{i2} \cup UF_2 \setminus UF_{i2}$ и $u_2 \rightarrow SE \setminus SE_{i2} \cup (P \setminus P_{i2}) \setminus P_2$.

Оригинальные права пользователя сохраняются с тем лишь отличием, что доступ к скопированным файлам осуществляется к копии, и оригинальные файлы становятся недоступными для пользователя. Тем не менее, пользователю добавляются права на полный доступ к специально созданной для него частичной копии.

Рассмотрим сценарии атак применительно к виртуальным системам с частичной виртуализацией.

Сценарий 1. Пользователь u_1 является злоумышленником.

Пользователь u_1 может модифицировать множество файлов $Copy_1$, которое является копией множества *Objects*. Вся конфигурация пользователя u_2, скопированная для виртуальной системы, остается недоступной для пользователя u_1, так как $Copy_1 \cap Copy_2 = \varnothing$, даже если были скопированы одинаковые файлы. Тем не менее, если файлы, которые не были скопированы пользователем u_1, совпадают с файлами, которые не участвуют в виртуальной системе пользователя u_2, т.е. $(CF_1 \setminus CF_{i1}) \cap (CF_2 \setminus CF_{i2}) \neq \varnothing$, или $(P_1 \setminus P_{i1}) \cap (P_2 \setminus P_{i2}) \neq \varnothing$, или $(UF_1 \setminus UF_{i1}) \cap (UF_2 \setminus UF_{i2}) \neq \varnothing$, то пользователь u_1 сможет модифицировать файлы из пересечения. Таким образом, частичная копия позволяет устранить пересекающиеся части конфигураций пользователей, а остальная часть системы остается не защищенной от такой угрозы.

Сценарий 2. Происходит заражение системного исполняемого файла операционной системы.

Пусть заражен некоторый системный исполняемый файл $se_1 \in SE$, $SE \subseteq Objects$. Если $se_1 \in Objects \setminus Copy_{i1}$ или $se_1 \in Objects \setminus Copy_{i2}$, т.е. этот файл не содержится во множестве файлов, скопированных одним или двумя пользователями, то один или оба пользователя используют зараженный файл. Если файл se_1 был скопирован для виртуальной системы пользователя, этот пользователь использует оригинальную, не подвергшуюся заражению копию этого файла. Таким образом, если пользователи используют виртуальные системы с частичной копией и оригинальная версия была сохранена в эту копию, то заражение исходного файла никак не влияет на работу пользователей. Однако, если файл не был скопирован, то пользователи будут использовать зараженный файл.

Сценарий 3. Происходит заражение пользовательского исполняемого файла.

Пусть заражен некоторый пользовательский исполняемый файл $p_1 \in P$. Рассмотрим данный сценарий для пользователя u_1. Для пользователя u_2 можно записать аналогичные утверждения с изменением рассматриваемых множеств доступных файлов и заменой индексов в выражениях. Аналогично предыдущему сценарию, необходимо определить, принадлежит ли файл p_1 множеству скопированных файлов пользователя из оригинальной системы P_{i1}. Если $p_1 \in P_{i1}$, то существует файл p_{c1} такой, что p_{c1} принадлежит частичной копии пользователя u_1 – $Copy_{c1}$ и, в частности, множеству пользовательских приложений P_{c1}, и выполняется $p_{c1} \approx p_1$. Ввиду включенного перенаправления пользователь u_1 использует файл p_{c1} при обращении к p_1:

$$p_1 \xrightarrow{\{r,w\},u_1} p_{c1}.$$

Следовательно, пользователь не оказывается под влиянием заражения.

Если же $p_1 \notin P_{c1}$, то пользователь u_1 использует зараженный файл, так как перенаправление в виртуальную систему не выполняется для этого файла. Таким образом, пользователь не подвержен риску, если зараженные файлы были ранее скопированы в виртуальную копию.

В другом варианте этого сценария зараженный файл находится в одной из частичных копий системы. Пусть заражен пользовательский исполняемый файл p_{c1} из копии $Copy_1$ одного из пользователей u_1: $p_{c1} \in P_{c1}$, $P_{c1} \subseteq Copy_1$, $p_{c1} \approx p_1$, где $p_1 \in P$, $P \subseteq Objects$. Единственный пользователь, который имеет доступ к этому файлу – это u_1: $u_1 \rightarrow p_{c1}$. Так как пользователь u_2 не имеет доступа к файлу p_{c1} и использует либо оригинальный файл p_1, либо свою копию p_{12}, $p_{12} \approx p_1$, то он не имеет доступа к зараженному файлу p_{c1}. Таким образом, виртуальные системы локализуют заражение при частичном копировании и предотвращают распространение заражения на других пользователей системы.

Сценарий 4. Кража информации из конфигурационных файлов.

Пусть пользователь u_1 запускает исполняемый файл, который является зараженным. Возможны два случая, когда $f \in SE_1 \cup P_1$, $SE_1 \subseteq Copy_1$, $P_1 \subseteq Copy_1$, т.е. f запускается из копии системы, либо когда $f \in SE \cup P$, $SE \subseteq Objects$, $P \subseteq Objects$. В любом из этих случаев приложение f выполняется с правами пользователя u_1 и имеет доступ к следующим файлам: $f \Rightarrow Copy_1 \cup CF_1 \setminus CF_{i1} \cup P_1 \setminus P_{i1} \cup UF_1 \setminus UF_{i1}$ и $u_1 \rightarrow SE \setminus SE_{i1} \cup (P \setminus P_{i1}) \setminus P_1$. В результате под воздействием оказываются не только файлы пользователя u_1, но и, возможно, файлы остальных пользователей, например пользователя u_2, если конфигурации этих пользователей пересекаются и не были скопированы для обоих из них. Остальные же данные пользователя u_2, которые хранятся в $Copy_2$, недоступны для кражи, так как пользователь u_1 не имеет к ним доступа. Таким образом, виртуальные системы с частичным копированием снижают количество данных, которые могут быть доступны остальным пользователям для кражи.

Сценарий 5. Оказывается заражен исполняемый файл, выполняемый системным процессом *sp* или администратором.

Пусть файл $f \in SE$ является зараженным. Он запускается системным процессом или администратором. Так как и администратор, и системный процесс имеют полный доступ к системе ($sp \Rightarrow Objects_{all}$, $adm \Rightarrow Objects_{all}$), то вся система оказывается доступной для процесса, запущенного из файла *f*. Однако процессу *f* необходимо определить, что применяются виртуальные системы. Кроме того, так как копирование выполнено частично, то процессу *f* нужно определить, где расположены копии, и что их тоже необходимо заразить. Это, также как и в случае полного копирования, не является надежным методом защиты, но, тем не менее, создает возможность сохранить файлы копий нетронутыми.

Исследование построенных абстрактных моделей виртуализации системы позволяет сделать следующие выводы.

1. При реализации полной виртуализации системы достигается и полное изолирование системных и конфигурационных файлов системы и приложений. Доступ к конфигурационным файлам любого пользователя возможен только для этих пользователей, при этом все заражения локализуются его виртуальной системой.
2. Применение частичной виртуализации позволяет найти компромисс между объемом создаваемых копий виртуализируемой системы и защищенностью пользовательской конфигурации. Как было показано при рассмотрении сценариев атак, при частичном копировании большую роль играют следующие обстоятельства: имеется ли копия оригинального файла, который был заражен; какие из файлов были созданы копированием. Чем полнее копия, тем ниже шанс, что пользователь будет использовать зараженный не по его вине файл и что его конфигурация будет изменена или считана другим пользователем. В качестве рекомендации можно предложить переносить в виртуальную систему все системные объекты и ключевые пользовательские приложения, а также конфигурационные файлы, которые находятся в общем доступе для нескольких пользователей.

Заключение

В работе при изложении и проведении исследований эффективности технологии виртуализации системных средств в качестве субъекта доступа рассматривался пользователь (учетная запись). Однако при реализации разграничительной политики доступа субъектов к объектам в современной информационной системе субъект доступа уже должен задаваться сущностью «пользователь, процесс» – какой пользователь и каким процессом запрашивает доступ к объекту. Для реализации подобной схемы контроля и разграничения прав доступа может использоваться запатентованное техническое решение [13]. Именно процесс в современной информационной системе с той или иной вероятностью (различающейся на порядки для различных процессов) несет в себе угрозу атаки как по причине потенциальной возможности выявления уязвимости [14], так и по причине возможности наделения процесса вредоносными свойствами [15]. Поэтому различным процессам в общем случае должны назначаться различные права доступа к объектам (в противном случае они одинаковы и наследуют права доступа к объектам запустившего их пользователя).

С учетом сказанного задачу виртуализации системы (полную или частичную) можно рассмотреть применительно к процессу как к субъекту доступа. Виртуализация системы может реализовываться для критичных процессов (запускаемых одним пользователем), для которых по тем или иным причинам высока вероятность реализации успешной атаки. Однако эти вопросы выходят за рамки настоящей работы и являются предметом дальнейших исследований авторов.

Литература

1. Щеглов К.А., Щеглов А.Ю. Метод сессионного контроля доступа к файловым объектам. Вопросы практической реализации // Вестник компьютерных и информационных технологий. 2014. № 8 (122). С. 54–60.
2. Щеглов К.А., Щеглов А.Ю. Новый подход к защите данных в информационной системе // Изв. вузов. Приборостроение. 2015. Т. 58. № 3. С. 157–166. doi: 10.17586/0021-3454-2015-58-3-157-166
3. Щеглов К.А., Щеглов А.Ю. Защита от атак на уязвимости приложений // Информационные технологии. 2014. № 9. С. 34–39.
4. Щеглов А.Ю., Щеглов К.А. Система контроля доступа к файлам на основе их автоматической разметки. Патент РФ № 2524566. Бюл. 2014. № 21.
5. Щеглов К.А., Щеглов А.Ю. Контроль доступа к статичным файловым объектам // Вопросы защиты информации. 2012. № 2 (97). С. 12–20.
6. Kim S.-K., Ma S.-Y., Moon J. A novel secure architecture of the virtualized server system // Journal of Supercomputing. 2015. doi: 10.1007/s11227-015-1401-4
7. Jithin R., Chandran P. Virtual Machine Isolation // Communications in Computer and Information Science. 2014. V. 420 CCIS. P. 91–102. doi: 10.1007/978-3-642-54525-2_8
8. Pektas A., Acarman T. A dynamic malware analyzer against virtual machine aware malicious software // Security and Communication Networks. 2014. V. 7. N 12. P. 2245–2257. doi: 10.1002/sec.931
9. Luo X., Yang L., Hao D., Liu F., Wang D. On data and virtualization security risks and solutions of cloud computing // Journal of Networks. 2014. V. 9. N 3. P. 571–581. doi: 10.4304/jnw.9.3.571-581
10. Win T.Y., Tianfield H., Mair Q. Virtualization security combining mandatory access control and virtual machine introspection // Proc. IEEE/ACM 7th Int. Conf. on Utility and Cloud Computing, UCC 2014. London, 2015. P. 1004–1009. doi: 10.1109/UCC.2014.165

11. Щеглов А.Ю., Щеглов К.А. Система переформирования объекта в запросе доступа. Патент РФ № 2538918. Бюл. 2015. № 1.
12. Щеглов А.Ю., Павличенко И.П., Корнетов С.В., Щеглов К.А. Комплексная система защиты информации «Панцирь+» для ОС Microsoft Windows. Свидетельство о государственной регистрации программы для ЭВМ №2014660889.
13. Щеглов А.Ю., Щеглов К.А. Система контроля доступа к ресурсам компьютерной системы с субъектом доступа «пользователь, процесс». Патент РФ № 2534599. Бюл. 2014. № 33.
14. Щеглов К.А., Щеглов А.Ю. Защита от атак на уязвимости приложений. Модели контроля доступа // Вопросы защиты информации. 2013. № 2 (101). С. 36–43.
15. Щеглов К.А., Щеглов А.Ю. Защита от атак со стороны приложений, наделяемых вредоносными функциями. Модели контроля доступа // Вопросы защиты информации. 2012. № 4 (99). С. 31–36.

Ковешников Михаил Геннадьевич – студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, Mike_35_92@mail.ru

Щеглов Константин Андреевич – менеджер по развитию, ООО «НПП «Информационные технологии в бизнесе», Санкт-Петербург, 194044, Российская Федерация, scheglov.konstantin@gmail.com

Щеглов Андрей Юрьевич – доктор технических наук, профессор, генеральный директор, ООО «НПП «Информационные технологии в бизнесе», Санкт-Петербург, 194044, Российская Федерация, info@npp-itb.spb.ru

Michael G. Koveshnikov – student, ITMO University, Saint Petersburg, 197101, Russian Federation, Mike_35_92@mail.ru

Konstantin A. Shcheglov – development manager, OOO «Scientific Production Enterprise «Information technologies in business», Saint Petersburg, 194044, Russian Federation, scheglov.konstantin@gmail.com

Andrey Yu. Shcheglov – D.Sc., Professor, General Director, OOO «Scientific Production Enterprise «Information technologies in business», Saint Petersburg, 194044, Russian Federation info@npp-itb.spb.ru