*Журов П. М, ГУ МФТИ*

**Особенности контроля взаимодействия клиента VMware vSphere 6.5 с vCenter**

На данный момент виртуализация является одной из самых быстроразвивающихся областей в сфере информационных технологий. Преимущественно виртуальные системы призваны заменить классические сервера, поэтому их задачей помимо прочего является и обработка большого количества данных. Но, как и в любой системе, работающей с данными, в виртуальной инфраструктуре необходимо обеспечить доверенную среду. Одним из важнейших инструментов обеспечения доверенной среды в виртуальной инфраструктуре является разграничение доступа.

Может возникнуть вопрос, почему именно на разграничении доступа делается акцент, ведь помимо него существует множество других задач. Ответ в том, что на данный момент это одна из самых актуальных проблем в вопросе безопасности виртуальных инфраструктур. В последнее время во всей сфере IT в целом и в области виртуализации в частности, наблюдается рост количества утечек данных. Самые крупные из них [1] связаны именно с неправильным разграничением доступа. Поэтому так важно обеспечить возможность гибкой и удобной настройки прав пользователей.

Лидером в области виртуализации на данный момент является VMware с продуктом vSphere. Но, даже несмотря на все достоинства данного ПО, в нём есть существенный недостаток с точки зрения безопасности, а именно: администратор отвечает как за управление инфраструктурой, так и за назначение прав доступа пользователям к её элементам. Это влечет за собой высокую трудоёмкость решения следующей проблемы: если в системе есть несколько сегментов, то критически возрастает вероятность преднамеренной или случайной утечки данных из одного сегмента в другой [2, 3, 4]. Для её решения необходимо средство, позволяющее разграничить доступ к элементам инфраструктуры для всех пользователей, включая администратора.

На сегодняшний день на рынке существует всего три продукта с подобным функционалом. Но один из них [5] не имеет сертификата ФСТЭК (а значит, не может применяться в государственных информационных системах), а второй [6] – не работает напрямую с vSphere (использует сторонний клиент), что вынуждает администратора изменить привычный порядок работы.

Третий комплекс [7] соответствует всей нормативной базе и не меняет порядка работы управляющего персонала, но в связи с заявлением VMware о том, что в следующей версии продукта [8] будет использоваться новый клиент управления на основе HTML5, что влечет за собой изменение принципа взаимодействия веб-клиента с сервером, можно сделать вывод, что существующие решения более не применимы, а значит, проблема остается актуальной. Данную проблему так же усугубляет то, что у новой версии нет открытого API, а следовательно, у разработчиков нет возможности быстро переделать существующие решения под новый клиент.

В то же время именно последний случай – ПАК «Сегмент-В.» – представляется наиболее целесообразным для доработки, поскольку в его случае не требуется коренной пересмотр решения. В статье описаны основные аспекты этой доработки, в рамках которой был проведен анализ трафика передаваемого от HTML5 клиента vSphere на vCenter, частично выделен протокол управления, а затем рассмотрены особенности, которые можно использовать для контроля доступа.

**Рассмотрение существующей модели разграничения доступа**

Прежде всего, стоит описать модель разграничения доступа для виртуальной инфраструктуры vSphere [9], которая используется в ПАК «Сегмент-В.». Для этого выделим несколько ключевых особенностей данной инфраструктуры, которые эта модель использует.

Самым главным элементом vSphere является центральный сервер виртуальной инфраструктуры – VCSA (vCenter Server Appliance). К нему подключаются различные гипервизоры, на нем создаются хранилища и сети и с помощью него же происходит взаимодействие со всеми этими элементами. Для управления этим сервером используется веб-интерфейс.

Данная модель разграничения доступа использует принцип, схожий с принципом атаки man-in-the-middle: веб-клиент рассматривается в качестве внешнего элемента ВИ, сам vCenter – внутреннего, а посередине устанавливается прокси-сервер, который анализирует трафик и пропускает команды управления, основываясь на заранее записанных в него политиках доступа. Это позволяет реализовать принцип «разделяй и властвуй»: настройками политик доступа занимается администратор безопасности инфраструктуры (АБИ), у которого при этом нет прав системного администратора, и наоборот – системный администратор не имеет возможности настраивать политики доступа.

Реализация на данном прокси-сервере, например, мандатной системы разграничения доступа, решит как проблему сегментированности, так и проблему суперпользователя.

**Механизм**

Какими же качествами должен обладать механизм, который реализует эту модель.

В первую очередь АБИ должен иметь возможность удаленно настраивать политики доступа по защищенному соединению. Помимо удобства это так же даст возможность АБИ быстро вносить изменения в политики в критических ситуациях.

Во-вторых, использование данного механизма не должно нарушать привычный порядок работы пользователей инфраструктуры и системного администратора/системных администраторов в частности.

И наконец, крайне желательно, чтобы данный механизм обладал модульностью и масштабируемостью. Под модульностью здесь подразумевается то, что при изменении протокола управления VCSA достаточно будет изменить лишь какую-то часть механизма для продолжения его эксплуатации, а не переделывать весь механизм полностью. Это означает, что каждый модуль должен выполнять строго одну функцию, и никак не влиять на работу остальных. Масштабируемость же в данном контексте означает возможность беспрепятственно добавлять обработку новых функций VCSA.

**Создание механизма**

Для создания данного механизма необходимо сначала провести анализ трафика между веб-клиентом vSphere и VCSA. Для этого, в первую очередь был создан стенд, содержащий в себе все ключевые элементы виртуальной инфраструктуры: несколько гипервизоров, часть которых объединена в различные кластеры, несколько хранилищ, сетей и пользователей. Все это необходимо, чтобы как можно подробнее изучить протокол управления.

Для анализа данных, передаваемых с клиента на сервер, использовался язык python и библиотека mitmproxy, которая позволяет реализовать прокси-сервер для прослушивания трафика. В vSphere используется протокол HTTPS, который исключает возможность прослушивания трафика, но в силу того, что имеется доступ к серверу VCSA, из которого можно взять сертификат и ключ TLS, данное ограничение не помешало реализации.

Новый HTML5 клиент использует для управления протокол JSON RPC, в котором клиент передает команды в формате JSON и в таком же виде получает их от сервера. Эта информация использовалась в дальнейшем для обработки запросов.

Далее, путем воспроизведения действий пользователя в ВИ и журналирования трафика, проходящего через прокси-сервер, были выделены следующие особенности протокола управления:

1. Операции, в которых происходят изменения в элементах инфраструктуры (например, изменение настроек виртуальной машины) всегда выполняются с помощью POST запросов, и такие операции однозначно идентифицируются по ключам JSON словаря.
2. Операции, направленные на получение информации об инфраструктуре, всегда выполняются с помощью GET запроса, и однозначно идентифицируются по своему URL.
3. Информация о пользователе, совершившем действие, всегда хранится в заголовке запроса в открытом виде (обратите внимание, даже не в виде идентификатора сессии).
4. Информацию об объекте, с которым работает пользователь, можно получить двумя путями:
	1. В большинстве случаев id объекта содержится в URL запроса.
	2. Иногда, например, когда объектов несколько, информация о них содержится в теле запроса.

Эти особенности позволяют однозначно идентифицировать из любого запроса объект, субъект и операцию.

С использованием данной информации и описанных выше требований к механизму, была разработана его архитектура. В первую очередь было принято решение разделить прокси-сервер на модули, каждый из которых выполнял бы одну строго определенную задачу. Взаимодействие между модулями должно происходить так, чтобы при замене одного модуля не приходилось никак изменять другой. Получившаяся архитектура представлена на рисунке ниже (Рис. 1).



*Рис. 1 Архитектура прокси-сервера*

Принцип работы прокси-сервера, следующий: Inspector перехватывает запрос и передает его в Handler (обработчик). Тот, пользуясь особенностями протокола управления, описанными выше, получает из запроса данные, необходимые для инициализации объектов, пользователя и операции, затем инициализирует их с помощью модуля Adapter, передает возвращенные адаптером объекты в модуль Logic, который принимает решение о допустимости данной операции. Именно это решение возвращается из Handler’а в Inspector. Под инициализацией здесь подразумевается создание объектов (программных), которые содержат в себе пользователя/элемент ВИ/операцию и все соответствующие ему/ей атрибуты безопасности. Рассмотрим каждый модуль немного подробнее:

1. Модуль Inspector отвечает за перехват трафика. В нем подключается библиотека mitmproxy, с её помощью перехватывается запрос, который передается в модуль Handler, а тот возвращает в ответ да/нет (пропустить запрос или нет). Если запрос заблокирован (Handler вернул нет), Inspector заменяет ответ сервера на ответ с ошибкой.
2. Модуль Handler является связующим звеном между остальными модулями. Он отвечает за получение необходимой информации из запроса, которая включает в себя пользователя, операцию, вид операции и объекты, участвующие в операции (их может и не быть).
3. Модуль Adapter используя локальную базу данных, в которую записаны политики доступа и данные, полученные из Handler’а. Затем он инициализирует объекты и пользователя с соответствующими им атрибутами безопасности (уровни и метки доступа, а также список допустимых операций для пользователя) и операцию. В конце адаптер возвращает инициализированные объекты в Handler.

1. Модуль Logic получает от Handler’а объекты, пользователя и операцию со всеми соответствующими атрибутами и принимает решение о предоставлении/не предоставлении доступа данному пользователю к данной операции с данными объектами.

**Анализ полученного механизма**

1. В первую очередь возникает вопрос к архитектуре – зачем передавать все данные об объектах через Handler, когда можно напрямую передавать их из SQL adapter в Logic, а затем сразу возвращать решение в Inspector.

Это обусловлено тем, что операции бывают нескольких видов: операции без объектов, операции с одним объектом и операции с несколькими объектами. В зависимости от вида операции, данные, передаваемые в модуль Logic, разные, и сделать передачу данных из Adaprter’а в Logic напрямую – значит, добавить в адаптер функцию разделения вида операций, что нарушает принцип модульности (данный модуль должен отвечать только за взаимодействие с базой данных).

1. Также может возникнуть еще один вопрос – как Handler определяет тип операции, и как он выполняет требование масштабируемости, если при добавлении нового функционала придется менять и сам Handler.

Это реализовано с помощью файла operation keys. Указанный файл включает в себя словари, которые содержат:

* 1. Соответствия между ключами из словаря POST запроса и операциями.
	2. Соответствия между URL’ами и операциями.
	3. Список операций, получаемых несколькими путями (например, изменение параметров сети может быть различным: от изменения физического адаптера, до изменения названия порт-группы. Все эти операции являются различными с точки зрения управления, но одной и той же операцией – «Изменение конфигурации сети» – с точки зрения разграничения доступа).
	4. Словарь, который содержит в себе названия операций с несколькими объектами в качестве ключей, и путями к получению данных объектов из тела запроса в качестве значений.
	5. Набор операций, в которых не используются объекты.

При обработке запроса Handler пользуется данным файлом для выделения нужной информации из запроса, а именно: название операции и id объекта/объектов. Поэтому при изменении запроса для какой-либо операции, достаточно будет поменять соответствующую запись в одном из словарей данного файла, и, аналогично, при добавлении какой-либо операции – добавить соответствующую запись.

**Заключение**

Как было сказано в начале, проблема разграничения доступа в виртуальных инфраструктурах является сейчас очень актуальной. Несмотря на существование работающей теоретической модели, которая позволяет решить её, реализация данной модели оказалась недостаточно гибкой для того, чтобы сохранить работоспособность после изменений протокола управления vCenter. В данной работе показано одно из возможных решений проблемы сохранения работоспособности в случае изменений в виртуальной инфраструктуре, а также выделены особенности этого решения, которые позволяют успешно его реализовать.

**СПИСОК ЛИТЕРАТУРЫ:**

* + - 1. “Топ-5 крупнейших утечек c начала года” Kaspersky Lab [Электронный ресурс]. Режим доступа:  <https://www.kaspersky.ru/blog/data-leaks-2017/18993/>, свободный (дата обращения: 12.04.2018).
			2. Приказ №17. Об утверждении требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах: утвержден ФСТЭК России 11 февраля 2013 г.
			3. Приказ №21. Об утверждении состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных: утвержден ФСТЭК России 18 февраля 2013 г.
			4. *Угаров Д. В., Постоев Д. А.* Проблемы реализации разграничения доступа к функциям управления виртуальных сред // Вопросы защиты информации: Научно-практический журнал/ФГУП «ВИМИ», М., 2016г., Вып.3, №114. С. 34–35.
			5. ESXi Security [Электронный ресурс]. Режим доступа https://www.hytrust.com/solutions/private-cloud-controls/esxi/, свободный (дата обращения: 12.04.2018).
			6. vGate [Электронный ресурс]. Режим доступа https://www.securitycode.ru/products/vgate/, свободный (дата обращения: 12.04.2018).
			7. ПАК Сегмент-В. [Электронный ресурс]. Режим доступа http://www.accord.ru/segment-v.html/, свободный (дата обращения: 12.04.2018).
			8. Goodbye, vSphere Web Client! [Электронный ресурс]. Режим доступа https://blogs.vmware.com/vsphere/2017/08/goodbye-vsphere-web-client.html, свободный (дата обращения: 12.04.2018).
			9. *Конявская С. В., Угаров Д. В., Постоев Д. А.* Инструмент контроля доступа к средствам управления виртуальной инфраструктурой // Information Security/Информационная безопасность. М., 2016. № 2. С. 9.