Научная статья

УДК 004.451
DOI 10.25205/1818-7900-2022-20-1-81-96

Разработка PowerShell-сценариев для автоматизации процессов администрирования ОС Windows. Автоматизация процесса создания и распространения SSL-сертификатов на серверы

Надежда Александровна Федькова [1]
Дмитрий Андреевич Федьков [2]

[1, 2] Брянский государственный аграрный университет
Брянск, Россия

[1] voytova.nady@yandex.ru
[2] dmitrijfedkov@gmail.com

Аннотация

Рассматривается процесс автоматизации администрирования ОС Windows, в частности автоматизация процесса создания и распространения SSL-сертификатов на серверы. Для разработки применяется встроенная командная оболочка PowerShell. Приведены теоретические аспекты изучаемой проблематики: понятие SSL-сертификатов, назначение хранилища Vault и сервиса Consul. Представлены результаты разработки в виде листингов кода.

Ключевые слова

администрирование, MS Windows, SSL-сертификат, PowerShell

Благодарности

Работа выполнена при поддержке ФГБОУ ВО «Брянский ГАУ»

Для цитирования

Федькова Н. А., *Федьков Д. А.* Разработка PowerShell-сценариев для автоматизации процессов администрирования ОС Windows. Автоматизация процесса создания и распространения SSL-сертификатов на серверы // Вестник НГУ. Серия: Информационные технологии. 2022. Т. 20, № 1. С. 81–96. DOI 10.25205/1818-7900-2022-20-1-81-96

Development of PowerShell Scripts for Automating Windows OS Administration Processes. Automating the Process of Creating and Distributing SSL Certificates to Servers

Nadezhda A. Fedkova [1], **Dmitry A. Fedkov** [2]

[1, 2] Bryansk State Agrarian University
Bryansk, Russia

[1] voytova.nady@yandex.ru
[2] dmitrijfedkov@gmail.com

Abstract

The article describes the process of automating Windows administration, in particular, automating the process of creating and distributing SSL certificates to servers. The built-in command shell PowerShell is used for development.

Theoretical aspects of the subject are given: the concept of SSL certificates, the purpose of Vault storage and Consul service. The results of development in the form of code listings are presented.

Keywords

administration, MS Windows, SSL certificate, PowerShell

Acknowledgements

The work was carried out with the support of FSBEI HE “Bryansk SAU”

For citation

Fedkova N. A., Fedkov D. A. Development of PowerShell Scripts for Automating Windows OS Administration Processes. Automating the Process of Creating and Distributing SSL Certificates to Servers. *Vestnik NSU. Series: Information Technologies*, 2022, vol. 20, no. 1, pp. 81–96. (in Russ.) DOI 10.25205/1818-7900-2022-20-1-81-96

Введение

Большинство серверов используется в качестве хостинга для сервисов, а большинство сервисов для безопасной передачи данных используют HTTPS протокол [1–4]. Для работы сервиса на протоколе HTTPS ему необходимы сертификат с открытым и сертификат закрытым ключом. Сертификат открытого ключа подтверждает принадлежность данного открытого ключа владельцу сайта. Сертификат открытого ключа и сам открытый ключ посылаются клиенту при установлении соединения. Закрытый ключ используется для расшифровки сообщений от клиента. Так как сервисы взаимодействую между собой в одном локальном центре обработки данных, для создания сертификатов используются частный центр сертификации Microsoft Certificate Authority Service.

Центры сертификации (Certificate Authorities / CA), выпускают цифровые сертификаты – поддающиеся проверке небольшие файлы данных, содержащие идентификационные данные, которые помогают веб-сайтам, пользователям и устройствам представлять свою подлинную личность в Интернете (подлинную, потому что ЦС проверил личность). ЦС играют важную роль в функционировании Интернета и в осуществлении прозрачных, надежных транзакций в сети. Ежегодно ЦС выпускают миллионы цифровых сертификатов, которые используются для защиты информации, шифрования миллиардов транзакций и обеспечения безопасной связи.

SSL-сертификат – это популярный тип цифрового сертификата, который связывает данные о владельце веб-сервера (веб-сайта) с криптографическими ключами. Эти ключи используются в протоколе SSL/TLS для активации безопасной сессии между браузером и веб-сервером, на котором размещен SSL-сертификат. Для того чтобы браузер мог доверять SSL-сертификату и установить SSL/TLS-сессию без предупреждений безопасности, SSL-сертификат должен содержать доменное имя использующего его веб-сайта, быть выданным надежным центром сертификации и не иметь истекшего срока действия.

Выполнение процессов работы с сертификатами можно осуществлять «в ручном» режиме, выполняя настройки с помощью пользовательского интерфейса. Но в случае большой серверной инфраструктуры этот процесс становится слишком трудоемким и занимает много времени. С целью оптимизации этого процесса целесообразна автоматизация процессов «ручного» администрирования. Автоматизация в среде Windows выполняется с помощью интегрированного скриптового языка PowerShell.

Предложенная ниже разработка является одним из компонентов системы автоматизации серверной инфраструктуры на базе ОС Windows Server. Ядром этой системы является сервис каталога сервисов, который состоит из пользовательского web-интерфейса, а также HTTP API. В данном каталоге хранится информация о серверах и web-сервисах компании, которые размещаются на этих серверах. Для повышения безопасности необходимо перевести все сетевые взаимодействия между web-сервисами с использованием шифрования, для этого требуются цифровые сертификаты, реализующие возможность создания безопасного шифрованного соединения по протоколу HTTPS. Для инфраструктуры, построенной на базе серверов с ОС Windows Server, актуально использовать центр сертификации Microsoft Certificate

Authority Service. Остается выбрать автоматизированный способ выпуска и доставки сертификатов на сервера. Это можно сделать штатными средствами ОС Windows с применением групповых политик, чтобы каждый сервер выпускал самостоятельно сертификаты для всех имеющихся web-сервисов, но, когда в компании сотни серверов и сотни сервисов, это многократно увеличивает количество выпускаемых сертификатов, которые по факту ни чем не отличаются. Например, для 100 серверов и 500 сервисов выпускается 50 000 сертификатов, а добавление одного нового сервера приводит к выпуску еще 500 сертификатов и т. д. На каждом сервере одновременно не размещаются все 500 сервисов, они равномерно разделены на группы, но при использовании механизма групповых политик этим невозможно управлять, чтобы выпускать сертификаты только для тех web-сервисов, которые располагаются на определенных серверах, а также эту систему сложно мониторить в случае сбоев в ее работы. В связи с этим для высоконагруженных систем целесообразно разработать отдельный компонент автоматизации в виде PowerShell-сценария. Данное решение является гибким, так как в дальнейшем можно реализовать любую специфическую для копания функциональность выпуска и доставки сертификатов. Также в результате тесной интеграции с сервисом каталога сервисов имеется возможность запросить средствами HTTP API информацию о имеющихся в компании web-сервисах и серверах, на которых данные сервисы размещены. С учетом этой информации для каждого web-сервиса выпускается только один сертификат, а после доставляется только на те сервера, где размещен этот сервис, и в случае с примером выше для 100 серверов и 500 сервисов будет выпущено всего 500 сертификатов. Все чувствительные данные, в том числе конфигурация скрипта, хранятся в хранилище секретов Vault, а не просто в виде обычного файла рядом со скриптом или в самом скрипе, что повышает безопасность. Только пользователи с особыми привилегиями могут просматривать или изменять данную конфигурацию. При необходимости можно доработать скрипт, для взаимодействия с любым другим центром сертификации, который имеет API для выпуска сертификатов.

Автоматизация процесса создания и распространения SSL-сертификатов на серверы

Для автоматизации процесса создания и распространения сертификатов на серверы предлагается разработка PowerShell-сценария – скрипта DeployCerts.ps1, который будет выполняться на одном из серверов управления. Его работа будет заключаться в следующем:

- получение информации из каталога сервисов;
- выпуск новых сертификатов в центре сертификации;
- обновление устаревающих сертификатов в центре сертификации;
- распространение сертификатов сервисов по серверам;
- обновление информации о сертификатах в каталоге сервисов.

Запуск скрипта должен выполняться с помощью планировщика задач операционной системы Windows каждые 15 минут.

На рис. 1. показан общий механизм работы скрипта DeployCerts.ps1.

Далее будут описаны детали реализации.

Организация взаимодействия с хранилищем секретов Vault

В первую очередь скрипту необходимо выполнить чтение конфигурационных данных. Данные конфигурации находятся в «хранилище секретов» Vault. Vault – это инструмент для безопасного доступа к секретам. Секрет – это все, к чему необходимо жестко контролировать доступ, например ключи API, пароли или сертификаты. Vault предоставляет единый интерфейс к любому секрету, обеспечивая жесткий контроль доступа и ведя подробный журнал аудита.

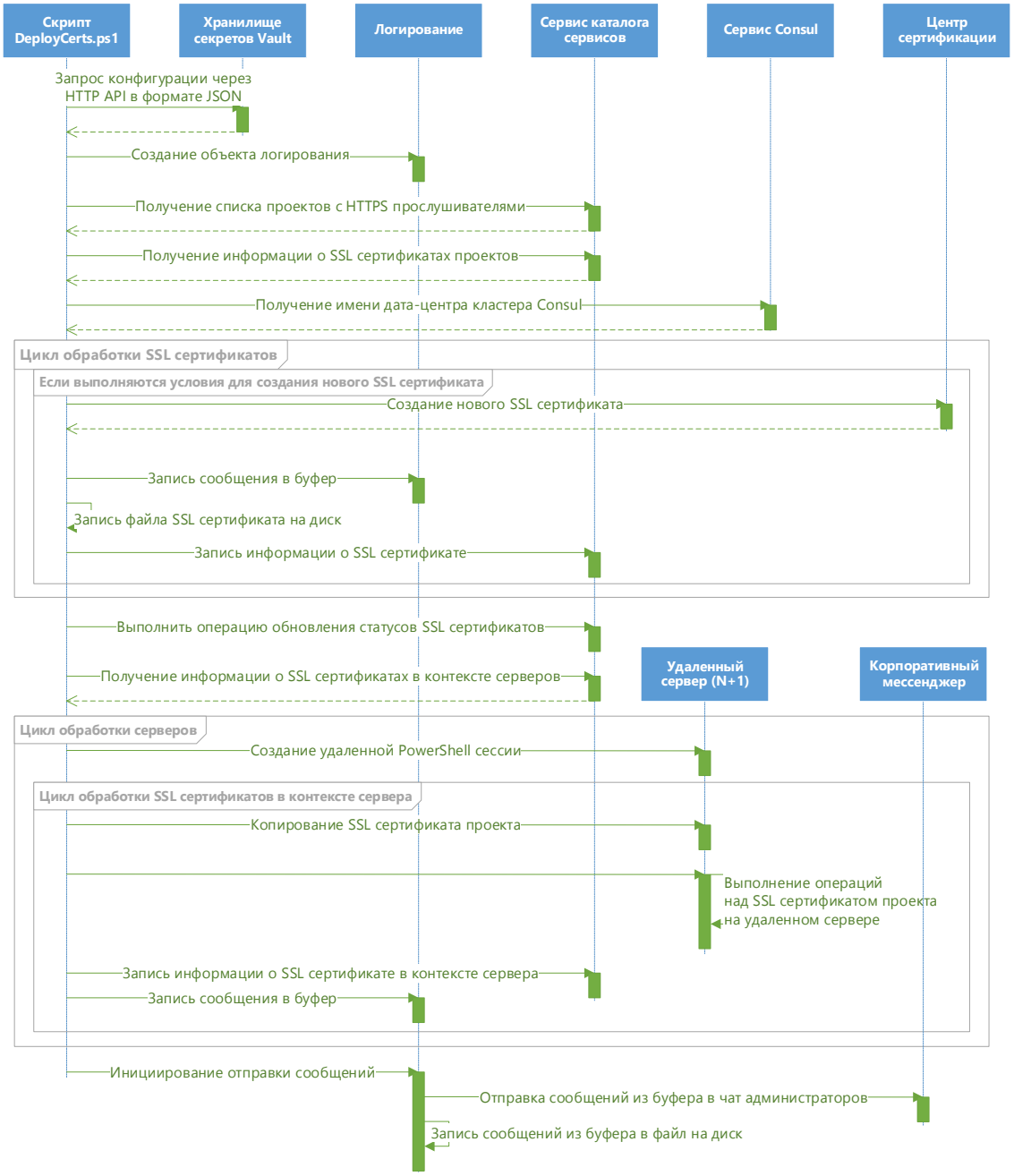

Рис. 1. Механизм работы скрипта DeployCerts.ps1
Fig. 1. The mechanism of the DeployCerts.ps1 script

Современные информационные системы требуют доступа к множеству секретов: учетные данные базы данных, API-ключи для внешних сервисов, учетные данные для взаимодействия сервис-ориентированной архитектуры и т. п. Понять, кто и к каким секретам имеет доступ, очень сложно, и это зависит от конкретной платформы. Добавить к этому продление ключей, безопасное хранение и подробные журналы аудита практически невозможно без специального решения. В таких ситуациях помогает Vault.

Конфигурация скрипта DeployCerts.ps1 в хранилище Vault хранится в формате JSON.

Описание структуры конфигурации

В свойстве «AdditionalDomains» указывается список доменных имен или список шаблонов доменных имен, которые будут добавлены в свойство сертификата в поле «Дополнительное имя субъекта / Subject Alternative Name» во время его выпуска в центре сертификации. Скрипт при разборе значений строк DNS имен, вместо шаблонов {Pattern} подставит значения.

{WinDomainFqdn} – полное имя домена AD, к которому присоединен компьютер, выполняющий скрипт (компьютер, с которого распространяются сертификаты).

{ConsulDcName} – имя DC Consul, к которому подключен агент компьютера, на котором выполняется скрипт (компьютер, с которого распространяются сертификаты).

{ProjectName} – имя проекта.

В значении свойства «AdditionalDomains» может быть указан пустой массив или массив строк, а также это свойство вообще может отсутствовать, если в нем нет необходимости, это не приведет к ошибке выполнения скрипта.

В свойстве «ChatId» указывается идентификатор корпоративного чата, для уведомлений обслуживающего персонала о работе скрипта. В случае создания новых сертификатов или обновлении старых, а также при возникновении ошибок во время работы, скрипт отправляет сообщения в указанный в конфигурации чат. Данная функция является полезной при возникновении ошибок в работе скрипта и позволяет быстро диагностировать и устранить проблему. Свойство принимает строку.

Свойство «PassPfx» хранит пароль, которым шифруется pfx файл сертификата, выпускаемый в центре сертификации. Это обязательное свойство конфигурации. Если оно отсутствует, скрипт выдаст исключение при чтении конфигурации. Данное свойство должно быть строкой.

В свойстве «RenewalPeriodDays» указывается количество дней до конца срока действия сертификата (в этот период будет производиться обновление сертификатов). Это необязательное свойство, если оно не заполнено, то по умолчанию будет использоваться значение 30. Данное свойство должно быть числом.

Чтение конфигурации выполняется с помощью функции GetConfigFromVault, которая в случае успешного чтения конфигурации вернет экземпляр класса DeployCertsConfig (рис. 2).

```
function GetConfigFromVault # возвращает объект типа DeployCertsConfig
{
    [CmdletBinding()]
    Param()

    [PSCustomObject]$configData = $null
    [DeployCertsConfig]$deployCertsConfig = $null
    try {
        $configData = Get-AdmAbtVltData -SecretPath "[illegible]"
    } catch {
        throw "I couldn't read the parameters from Vault."
    }
    if(!$configData) {
        throw "An empty configuration object is obtained."
    }
    try {
        $deployCertsConfig = [DeployCertsConfig]::new($configData)
    } catch {
        throw ("Configuration initialization error. Detail info: ({0})" -f $_.ToString())
    }
    return $deployCertsConfig
}
```

Рис. 2. Листинг функции GetConfigFromVault (чтение конфигурации)
Fig. 2. Listing of the Get Config From Vault function (reading the configuration)

Функция выполняет вызов командлета Get-AdmAbtVltData, который выполняет чтение конфигурации в формате JSON из Vault через HTTP API, затем десериализацию из JSON в объект типа PSCustomObject и возвращает функции GetConfigFromVault. Если во время чтения конфигурации возникли ошибки, или командлет Get-AdmAbtVltData вернул пустой объект, будет выдано исключение, и скрипт прервет выполнение. Иначе полученный объект будет передан в качестве параметра в конструктор класса DeployCertsConfig. В конструкторе класса DeployCertsConfig будет выполнена проверка свойств объекта, полученного в параметре $_ConfigObjectVlt.

Если свойство $_ConfigObjectVlt.AdditionalDomains существует и содержит больше нуля элементов, то все элементы массива этого свойства будут записаны в одноименное свойство экземпляра класса.

Далее проверяется свойство $_ConfigObjectVlt.PassPfx. Если это свойство не существует или его значение пустое, будет выдано исключение, что прервет выполнение скрипта, иначе значение будет преобразовано в зашифрованную строку и присвоено свойству PasswordPfx экземпляра класса. Последним проверяется свойство $_ConfigObjectVlt.RenewalPeriodDays, если свойство существует и его значение не равно нулю, то значение будет присвоено свойству экземпляра класса RenewalPeriod, иначе свойству ничего не будет присвоено, и будет использоваться значение по умолчанию, равное 30. Если в конструкторе не возникло исключений, он вернет функции GetConfigFromVault экземпляр класса DeployCertsConfig.

Логирование

После успешного чтения конфигурации будет создан экземпляр класса writeProcessing Class.

Данный класс предназначен, для записи сообщений статуса в буфер, форматирование поступающих сообщений HTML тегами в соответствии с их статусом (цвет, полужирное начертание), а также отправку данных из буфера в чат корпоративного мессенджера.

Для создания экземпляра класса writeProcessingClass вызывается единственный конструктор, который принимает два параметра, это идентификатор чата и заголовок сообщения, отправляемого в чат. Заголовок сразу будет отформатирован и записан в общий буфер сообщений. Также в конструкторе будет инициализировано поле LogFilePath, в данное поле будет записан путь к файлу журнала. Метод Write класса writeProcessingClass предназначен для записи сообщения в буфер, этот метод принимает три аргумента: строку, которую необходимо записать в буфер, значение перечисления writeProcessingEnum (это перечисление отвечает за статус сообщения), последний параметр имеет тип bool, и, если он является true, сообщение будет отформатировано как полужирное. Метод Push выполняет запись буфера сообщений в файл, путь которого определен в свойстве LogFilePath, а также выполняет отправку буфера сообщений в чат, предварительно обработав буфер функцией ThisScript.Format Message (рис. 3).

```
function ThisScript.FormatMessage
{
    [CmdletBinding()]
    Param(
        [Parameter(Mandatory=$True)][ValidateNotNullOrEmpty()][Alias("Messages")][string[]]$Messages_fnc0
    )
    [timespan]$timeExecScriptSec = ((Get-Date)-$thisScriptStrtTime)
    $Messages_fnc0 += "[color=Silver][size=10]Execution time: {0} sec.[/size][/color]" -f $timeExecScriptSec.TotalSeconds
    [string]$outFunction0 = $null
    $outFunction0 = [string]::Join("`n", $Messages_fnc0)
    return $outFunction0
}
```

Рис. 3. Листинг функции ThisScript.FormatMessage (оценка времени выполнения скрипта)
Fig. 3. Listing of This Script.FormatMessage function (evaluation of script execution time)

Функция ThisScript.FormatMessage принимает один параметр, который является буфером, сформированным в классе writeProcessingClass. Функция добавляет в «подвал» сообщения информацию о том, сколько времени прошло между запуском скрипта и текущим временем, т. е. эта строка, показывает, сколько времени заняло выполнение скрипта. Далее функция преобразует буфер, который является массивом строк, в одну единственную строку и возвращает эту строку вызывающему методу.

Операции с сервисом каталога сервисов

Далее в глобальной области видимости создается экземпляр класса обработчика и помещается в переменную $serviceManagerHandler, с помощью которого выполняется взаимодействие с HTTP API сервиса каталога сервисов. Внутренняя структура класса обработчика сервиса каталога сервисов, рассматриваться не будет, с ним мы работает как с «черным ящиком».

После этого необходимо вызвать метод $serviceManagerHandler.GetProjectsContainsSsl Binding(), который вернет массив строк, данные строки содержат наименования проектов, у которых в СКС имеются прослушивали с протоколом HTTPS.

Если функция вернет пустой массив, следовательно, нет сервисов для обработки, и в этом случае скрипт завершает работу. Если список сервисов был получен, то необходимо загрузить информацию о доступных сертификатах из сервиса каталога сервисов, это делается с помощью вызова функции $serviceManagerHandler.GetProjectCertificates(). Вызов функции вернет массив объектов, свойства объектов массива представлены в табл. 1.

Таблица 1

Свойства объекта сертификата

Table 1

Certificate Object Properties

Имя свойства	Тип	Пример значения	Описание
certId	int	100	Числовой идентификатор сертификата
projectName	string	Project	Имя сервиса
thumbprint	string	C8CCE305D3D0282860882E2ADA1CB7F54CC5FDA0	Значение хэша сертификата
certExpires	DateTime	2022-06-22T00:00:00	Время истечения сертификата
oldThumbprint	string	6A26BC92CE7611C209078C85CD1A5059F98AFD45	Значение хэша сертификата предшествующего, текущему сертификату
certFileName	string	4c7392c9-2925-4aeb-90d9-e2e95fc23527.pfx	Имя файла сертификата

После этого будет выполнено получение имени дата центра кластера Consul, к которому присоединен сервер и на котором выполняется скрипт DeployCerts.ps1.

Получение информации из сервиса Consul

Consul – это решение для Service Mesh, предоставляющее полнофункциональную плоскость управления с функциями обнаружения, конфигурации и сегментации сервисов. Каждая из этих функций может использоваться отдельно по мере необходимости, или они могут использоваться вместе для создания полнофункционального Service Mesh. Consul требует наличия плоскости данных и поддерживает как прокси, так и встроенную модель интеграции.

Consul поставляется с простым встроенным прокси, чтобы все работало «из коробки», но также поддерживает интеграцию прокси сторонних производителей, таких как Envoy. Consul разработан таким образом, чтобы быть «дружественным» как для сообщества DevOps, так и для разработчиков приложений, что делает его идеальным для современных «эластичных» инфраструктур.

Базовая архитектура Consul. Consul – это распределенная, высокодоступная система. Каждый узел, предоставляющий услуги Consul, запускает агента Consul. Запуск агента не требуется для обнаружения других сервисов или получения / установки данных ключей / значений. Агент отвечает за проверку работоспособности служб на узле, а также самого узла.

Агенты взаимодействуют с одним или несколькими серверами Consul. На серверах Consul хранятся и реплицируются данные. Серверы сами выбирают «лидера». Хотя Consul может функционировать с одним сервером, рекомендуется использовать от 3 до 5 серверов, чтобы избежать сценариев отказа, приводящих к потере данных. Для каждого центра обработки данных рекомендуется кластер серверов Consul.

Серверы поддерживают каталог, который формируется путем агрегирования информации, предоставленной агентами. Каталог поддерживает высокоуровневое представление кластера, включая информацию о том, какие службы доступны, какие узлы выполняют эти службы, информацию о состоянии здоровья и многое другое.

Компоненты инфраструктуры, которым необходимо обнаружить другие службы или узлы, могут запросить любой из серверов Consul или любой из агентов Consul. Агенты направляют запросы на серверы автоматически.

В каждом центре данных работает кластер серверов Consul. Когда выполняется запрос на обнаружение или настройку службы между центрами данных, локальные серверы Consul направляют запрос в удаленный центр данных и возвращают результат. Это имя необходимо получить, так как оно дальше будет использоваться как заполнитель в шаблонах DNS имен, для поля сертификата «Дополнительное имя субъекта / Subject Alternative Name». Получение имени дата центра кластера Consul выполняется с помощью функции Get-ConsulLocalDc (рис. 4).

```
function Get-ConsulLocalDc
{
    [CmdletBinding()]
    Param()
    try
    {
        return (Invoke-RestMethod `
                    -UseBasicParsing `
                    -Method Get `
                    -Uri "http://localhost:8500/v1/agent/self" `
                    -ErrorAction Stop).Config.Datacenter
    }
    catch
    {
        throw "Error reading Consul configuration. Detaild error: $_"
    }
}
```

Рис. 4. Листинг функции Get-ConsulLocalDc
(получение имени дата центра кластера Consul)

Fig. 4. Listing features Get-Consullocaldc
(name name data center cluster Consul)

Данная функция выполняет HTTP запрос в API агента Consul. Вызываемый метод agent / self возвращает информацию о конфигурации и членах локального агента в формате JSON. Config элемент содержит подмножество конфигурации. Из данного элемента будет выбрано свойство Datacenter, которое будет возвращено функцией.

Далее в глобальной области видимости скрипта необходимо инициализировать переменную $certDnsNamesBuilder ссылкой на экземпляр класса CertDnsNamesBuilder. Создание экземпляра класса CertDnsNamesBuilder выполняется с помощью конструктора, принимающего один параметр, который является хэш-таблицей, где в качестве ключей, записаны имена заполнителей DNS имен, а в качестве значений – значения этих заполнителей.

Первый вызов метода AddDnsNames на экземпляре класса CertDnsNamesBuilder выполняет добавление массива строк DNS имен, используемых по умолчанию. Второй вызов метода, выполняет добавление массива строк DNS имен, определенных в конфигурации в свойстве AdditionalDomains (конфигурация и ее свойства были рассмотрены ранее).

Метод AddDnsNames выполняет добавление массива строк DNS имен в свойство Data экземпляра класса CertDnsNamesBuilder. Данное свойство имеет тип System.Collections.Generic.HashSet[string], который позволяет организовывать строки в список, значения которого не повторяются.

Цикл обработки SSL сертификатов

Далее следует цикл (рис. 5), в котором выполняется обработка сертификатов. В начале цикла обновляется имя сертификата, в значении ключа ProjectName хэш-таблицы свойства Masks экземпляра класса CertDnsNamesBuilder.

```
1   for ($i=0; $i -lt $projectNames.Count; $i++)
2   {
3       [string]$projectNameInFor = $projectNames[$i]
4       $certDnsNamesBuilder.Masks["ProjectName"] = $projectNameInFor
5
6       [PSCustomObject]$certificateSm = $null
7       $certificateSm = $certificatesInfoFromSm | Where-Object {$_.ProjectName -eq $projectNameInFor}
8
9       [System.Security.Cryptography.X509Certificates.X509Certificate2]$creationCert = $null
10      #region Создание сертификата
11      if(!$certificateSm) { # сертификата нет в SM
12          $writeClass.Write(("Create certificate for project `"{0}`":" -f $projectNameInFor), 2, $True)
13          [string]$certName = "{0}.pfx" -f (New-Guid).ToString()
14
15          try
16          {
17              $creationCert = ThisScript.CreateCert   -ProjctName $projectNameInFor `
18                                                      -CertName $certName `
19                                                      -CertPass $deployCertsConfig.PasswordPfx `
20                                                      -DomainName $fqdnDomainName `
21                                                      -CertDnsNames $certDnsNamesBuilder.GetDnsNames()
22              $writeClass.Write(("Request and creation of the certificate - OK"), 2, $False)
23          }
24          catch
25          {
26              $writeClass.Write($_.ToString(), 1, $False)
27              continue
28          }
29
30          try {
31              $isAddInSm = {
32                  [bool]$resultWriteDataInSm = $False
33                  $resultWriteDataInSm = $serviceManagerHandler.AddProjectCertificate(
34                      $projectNameInFor, $creationCert.Thumbprint, $creationCert.NotAfter, $certName)
35
36                  return $resultWriteDataInSm
37              }
38
39              if(!$isAddInSm.InvokeReturnAsIs()) {
40                  throw "The project is not found or has no SSL binding."
41              }
42
43              $writeClass.Write(("Writing data in SM - OK"), 2, $False)
44          } catch {
45              $writeClass.Write($_.ToString(), 1, $False)
46          }
47      }
48      #endregion
49      #region Обновление сертификата···
85      #endregion
86  }
```

Рис. 5. Листинг цикла (создание сертификата)
Fig. 5. Cycle listing (certificate creation)

Затем выполняется поиск информации о сертификате в массиве, на который ссылается переменная $certificatesInfoFromSm. Если объект не был найден, будет выполнена попытка выпуска сертификата в центре сертификации (рис. 5 строки 10–48). Эта операция выполняется с помощью функции ThisScript.CreateCert (рис. 6).

```
function ThisScript.CreateCert
{
    # function 1
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$True)]
        [ValidateNotNullOrEmpty()]
        [Alias("ProjectName")]
        [string]$ProjctName_f1
        ,
        [Parameter(Mandatory=$True)]
        [ValidateNotNullOrEmpty()]
        [Alias("CertName")]
        [string]$CertName_f1
        ,
        [Parameter(Mandatory=$True)]
        [ValidateNotNullOrEmpty()]
        [Alias("CertPass")]
        [securestring]$CertPass_f1
        ,
        [Parameter(Mandatory=$True)]
        [ValidateNotNullOrEmpty()]
        [Alias("DomainName")]
        [string]$DomainName_f1
        ,
        [Parameter(Mandatory=$True)]
        [ValidateNotNullOrEmpty()]
        [Alias("CertDnsNames")]
        [string[]]$CertDnsNames_f1
    )

    $paramRequestCert = @{
        Template="AdminAbiitProjects";
        SubjectName=("CN={0}.{1}" -f $ProjctName_f1, $DomainName_f1).ToLower();
        DnsName=$CertDnsNames_f1;
        CertStoreLocation="Cert:\CurrentUser\My"
    }

    $resultRequestCert = Get-Certificate @paramRequestCert -ErrorAction Stop
    [System.Security.Cryptography.X509Certificates.X509Certificate2]$certObj = Get-Item -Path ("Cert:\CurrentUser\My\{0}" -f $resultRequestCert.Certificate.Thumbprint)
    [void]($certObj | Export-PfxCertificate -FilePath ("C:\Certificates\{0}" -f $CertName_f1) -Password $CertPass_f1 -Force -ErrorAction Stop)
    [void]($certObj | Remove-Item -Force)
    return $resultRequestCert.Certificate
}
```

Рис. 6. Листинг функции ThisScript.CreateCert (функция выпуска сертификата в центре сертификации Microsoft Certificate Authority)
Fig. 6. Listing of the ThisScript function.CreateCert (certificate issuance function in the Microsoft Certificate Authority)

Функции будут переданы имя проекта, имя файла сертификата, пароль сертификата, который был получен из конфигурации, имя домена Active Directory, к которому присоединен компьютер, на котором выполняется скрипт, а также список DNS имен, который предварительно будет отформатирован методом GetDnsNames экземпляра класса CertDnsNames Builder. Данный метод выполняет форматирование строк, хранящихся в списке свойства Data. Форматирование выполняет замену подстрок {ProjectName}, {WinDomainFqdn}, {ConsulDc Name} в обрабатываемой строке, значениями из хэш-таблицы, на которую ссылается свойство Masks (табл. 2).

Таблица 2

Хэш таблица с набором данных в свойстве Mask

Table 2

Hash table with a set of data in the Mask property

Имя ключа	Имя значения
ProjectName	TestProject
WinDomainFqdn	contoso.lan
ConsulDcName	dc1

Функция выполнит запрос на выпуск сертификата в центре сертификации с помощью командлета Get-Certificate, в случае успеха, данный командлет запишет сертификат в хранили-

ще сертификатов пользователя, после этого сертификат вместе с закрытым ключом будет экспортирован в директорию C:\Certificates в формате pfx, контейнер сертификата будет зашифрован с помощью пароля, который был получен из файла конфигурации. После экспорта сертификата в файл, сертификат будет удален из хранилища сертификатов пользователя. В данном случае хранилище сертификатов используется в качестве промежуточного хранилища. Заключительным этапом работы функции ThisScript.CreateCert является возврат объекта сертификата с типом System.Security.Cryptography.X509Certificates.X509 Certificate2.

Например, если в свойстве Mask хранится хэш-таблица с набором данных из табл. 2, тогда получим следующие результаты форматирования, указанные в табл. 3.

Таблица 3

Результаты форматирования

Table 3

Formatting results

Исходная строка	Результирующая строка
*.{WinDomainFqdn}	*.contoso.lan
*.node.consul	*.node.consul
*.node.{ConsulDcName}.consul	*.node.dc1.consul
*.{ProjectName}.service.consul	*.TestProject.service.consul
*.{ProjectName}.service.{ConsulDcName}.consul	*.TestProject.service.dc1.consul
*.{ProjectName}.query.consul	*.TestProject.query.consul

После этого выполнение скрипта вернется обратно в глобальную область и продолжит выполнение цикла. Далее будет выполнена запись в сервис каталога сервисов информации о выпущенном сертификате. Если условие в 11 строке (см. рис. 6) не сработало, т. е. сертификат был найден в массиве, на который ссылается переменная $certificatesInfoFromSm, выполнение скрипта перейдет к выполнению блока else (рис. 7), в котором будет выполнена попытка обновить сертификат.

Условие в строке 52 проверяет, что дата срока действия сертификата превышает значение, указанное в конфигурации, если это условие выполняется, будет выполнено обновление сертификата. Обновление заключается в создании нового сертификата в центре сертификации с помощью функции ThisScript.CreateCert и с последующим обновлением данных о сертификате в СКС (см. рис. 5, строки 67–82).

Далее описанные операции выполняются в цикле для каждого проекта. После завершения цикла обработки сертификатов будет выполнен метод $serviceManagerHandler.Update CertificateStatus() (рис. 8).

Этот метод не возвращает данных. Он запускает в СКС процесс обновления статусов сертификатов для записей компьютеров. К примеру, если был обновлен сертификат проекта, то у записей компьютеров изменится статус обновленного сертификата. После этого значение переменной $certificatesInfoFromSm, которая хранит массив объектов сертификатов, будет перезаписано обновленными значениями.

Далее в переменную $certsAndServers с помощью метода $serviceManagerHandler.Get ServersRequireCertificateUpdates() будет записан массив объектов, эти объекты хранят информацию о компьютерах и статусах сертификатов, принадлежащих этим компьютерам. Данный метод возвращает только те объекты компьютеров, которым требуется записать новые сертификаты или обновить старые, свойства возвращаемого объекта указаны в табл. 4.

```powershell
for ($i=0; $i -lt $projectNames.Count; $i++)
{
    [string]$projectNameInFor = $projectNames[$i]
    $certDnsNamesBuilder.Masks["ProjectName"] = $projectNameInFor

    [PSCustomObject]$certificateSm = $null
    $certificateSm = $certificatesInfoFromSm | Where-Object {$_.ProjectName -eq $projectNameInFor}

    [System.Security.Cryptography.X509Certificates.X509Certificate2]$creationCert = $null
    #region Создание сертификата ···
    #endregion
    #region Обновление сертификата
    else
    {
        if((Get-Date).AddDays($deployCertsConfig.RenewalPeriod) -ge [DateTime]$certificateSm.CertExpires) {
            $writeClass.Write(("Renew certificate for project `"{0}`":" -f $projectNameInFor), 2, $True)
            try {
                $creationCert = ThisScript.CreateCert `
                                    -ProjctName     $projectNameInFor `
                                    -CertName       $certificateSm.CertFileName `
                                    -CertPass       $deployCertsConfig.PasswordPfx `
                                    -DomainName     $fqdnDomainName `
                                    -CertDnsNames   $certDnsNamesBuilder.GetDnsNames()
                $writeClass.Write(("Request and creation of the certificate - OK"), 2, $False)
            } catch {
                $writeClass.Write($_.ToString(), 1, $False)
                continue
            }

            try {
                $isUpdateInSm = {
                    [bool]$resultWriteDataInSm = $False
                    $resultWriteDataInSm = $serviceManagerHandler.UpdateProjectCertificate(
                        $projectNameInFor, $creationCert.Thumbprint, $creationCert.NotAfter, $certificateSm.CertFileName, $certificateSm.Thumbprint)
                    return $resultWriteDataInSm
                }

                if(!$isUpdateInSm.InvokeReturnAsIs()) {
                    throw "The project is not found or has no SSL binding."
                }

                $writeClass.Write(("Writing data in SM - OK"), 2, $False)
            } catch {
                $writeClass.Write($_.ToString(), 1, $False)
            }
        }
    }
    #endregion
}
```

Рис. 7. Листинг цикла (обновление сертификата)
Fig. 7. Cycle listing (certificate renewal)

```powershell
try {
    [void]($serviceManagerHandler.UpdateCertificateStatus())
} catch {
    $writeClass.Write($_.ToString(), 1, $False)
}

$certificatesInfoFromSm = $null
try {
    $certificatesInfoFromSm = $serviceManagerHandler.GetProjectCertificates()
} catch {
    $writeClass.Write($_.ToString(), 0, $True)
}

try {
    # метод возвращает только те сертификаты и сервера, которым требуется обновление
    [PSCustomObject[]]$certsAndServers = $serviceManagerHandler.GetServersRequireCertificateUpdates() | Where-Object {$_.OSVersion -ne "Linux"}
} catch {
    $writeClass.Write($_.ToString(), 1, $False)
}
```

Рис. 8. Листинг кода метода serviceManagerHandler.UpdateCertificateStatus()
(обновление статусов сертификатов)
Fig. 8. Listing Koda method serviceManagerHandler.UpdateCertificateStatus()
(updated status certificate)

Таблица 4

Свойства объекта «компьютер-сертификат»

Table 4

Properties of the “computer-certificate” object

Имя свойства	Тип	Пример значения	Описание
certId	int	100	Числовой идентификатор сертификата
serverName	string	test-server1	Имя сервера
status	int	1	Числовой статус сертификата для сервера

На основе объектов «компьютер-сертификат» (см. табл. 4.) будут сформированы новые объекты. Данные объекты будут группировать информацию о сертификатах по каждому компьютеру (табл. 5), назовем этот объект «сертификаты компьютера».

Таблица 5

Свойства объекта «сертификаты компьютера»

Table 5

Properties of the “Computer certificates” object

Имя свойства	Тип	Пример значения	Описание
ComputerName	string	test-server1	Имя компьютера
CertsInfo	object[]	–	Информация о сертификатах, принадлежащих компьютеру
HasErrors	bool	false	Возникали ли ошибки при подключении к компьютеру.

Создание объектов «сертификаты компьютера» будет выполняться с помощью циклической конструкции, листинг которой приведен на рис. 9.

```
[PSCustomObject[]]$certsOnServers = $null

if($certsAndServers.Count -gt 0) {
    foreach ($srvObj in ($certsAndServers | Group-Object -Property ServerName | Select-Object -Property Name, Group)) {
        [PSCustomObject[]]$certsInfo = $null
        foreach($cert in $srvObj.Group) {
            $certsInfo += [PSCustomObject]([ordered]@{
                CertID=$cert.CertID;
                CertStatus=$cert.Status;
                DeployStatus=$False;
                UpdateStatus="NotRequired"
            })
        }

        $certsOnServers += [PSCustomObject]([ordered]@{
            ComputerName=$srvObj.Name;
            CertsInfo=$certsInfo;
            HasErrors=$False
        })
    }
} elseif ($writeClass.ErrorCounter -gt 0) {
    $writeClass.Push()
    exit 0
} else {
    exit 0
}
```

Рис. 9. Листинг блока кода (формирование массива объектов соответствия сервер-сертификаты)
Fig. 9. Code block listing (forming an array of server-certificate compliance objects)

Новые объекты «сертификаты компьютера» помещаются в массив, на который ссылается переменная $certsOnServers, далее объекты этого массива будут обработаны в цикле, переменная цикла имеет название $serverObj. Вначале будет выполнена попытка создать удаленную powershell сессию для компьютера, имя которого указанно в свойстве $serverObj. ComputerName. Если по каким-то причинам эта операция завершится с ошибкой, то свойству $serverObj.HasErrors будет присвоено значение $true, выполнение цикла сразу вернется к началу, и начнется обработка следующего элемента массива.

Если операция создания удаленной powershell сессии будет завершена успешно, то переменная $psSession будет инициализирована объектным представлением этой сессии, которая в дальнейшем будет использоваться для подключения к уделенному компьютеру, для копирования сертификатов проектов и выполнении операций над этими сертификатами на удаленном компьютере.

Во вложенном цикле выполнятся перебор объектов «информация о сертификате», хранящихся в свойстве объекта $serverObj.CertsInfo. Каждый перебираемый объект помещается в переменную цикла $srvCertObj. В первую очередь выполняется копирование сертификата с компьютера, на котором выполняется данный скрипт, на удаленный компьютер во временную директорию. Копирование выполняется с помощью удаленной powershell сессии, созданной ранее.

Если копирование было выполнено успешно, на удаленном компьютере для скопированного сертификата будет выполнен ряд операций. Вначале будет выполнена проверка того, что скопированный сертификат отсутствует в хранилище сертификатов компьютера, и если сертификат отсутствует, то будет произведена установка сертификата из файла в хранилище сертификатов. Установка выполняется с помощью командлета Import-PfxCertificate. После успешной установки сертификата в хранилище данному сертификату необходимо применить ACL для закрытого ключа, такие же как и у других сертификатов для этого проекта, при условии, что эти сертификаты существуют. Access Control List или ACL – список управления доступом, который определяет, кто или что может получать доступ к объекту (программе, процессу или файлу) и какие именно операции разрешено или запрещено выполнять субъекту (пользователю, группе пользователей). Это необходимо, чтобы в дальнейшем приложение, для которого предназначается этот сертификат, смогло создать HTTPS прослушиватель. Агрегация ACL из других сертифкатов и дальнейшее применение агрегированных ACL к новому сертификату выполняется с помощью функции Merge-CertificateAcl. Заключительной операцией является удаление сертификата из временной директории. После этого свойству $srvCertObj.DeployStatus присваивается значение $true.

Затем проверяется статус сертификата, который хранится в свойстве $srvCertObj.Cert Status (информацию о нем можно найти в табл. 5). Если статус сертификата равен 2, то сертификату требуется обновление. Этот статус присваивается сертификату, если проекту ранее уже был выдан сертификат и взамен устаревшего сертификата был выпущен новый сертификат. Рассмотрим логику для тех сертификатов, которым требуется обновление. Вначале с помощью функции Move-CertificateAcl для закрытого ключа нового сертификата будут применены такие же ACL, как и для закрытого ключа старого сертификата. Это очень похоже на операцию, выполняемую функцией Merge-CertificateAcl, рассмотренной ранее, но в отличие от нее ACL копируются из определенного сертификата, тогда как Merge-CertificateAcl делает агрегацию ACL по всем сертификатам проекта. Далее выполняется поиск старого отпечатка сертификата в веб-сервере HTTP.sys для HTTPS прослушивателей. Если HTTPS прослушиватель со старым отпечатком сертификата будет найден, то для этого прослушивателя в вебсервере HTTP.sys будет заменен отпечаток сертификата на новый. После этой операции веб-сервер HTTP.sys сможет найти и загрузить новый сертификат из хранилища по обновленной информации об отпечатке.

Если операции установки и обновления сертификата прошли успешно, то выполняется запрос в СКС для изменения статуса сертификата для проекта на определенном сервере. По-

сле выполнения запроса сервиса каталога сервисов изменит статус сертификата на 1, что будет означать, что сертификат находится в актуальном состоянии, и при последующих запусках скрипта он не будет обрабатываться до тех пор, пока у сертификата не истечет срок действия. По завершении обработки всех сертификатов во внутреннем цикле будет закрыта и удалена PowerShell-сессия для освобождения ресурсов. Далее внешний цикл повторяется для остальных компьютеров.

Завершающим этапом работы является уведомление системных администраторов о результатах работы скрипта. Уведомления отправляются, если были созданы новые сертификаты или обновлены старые, а также если во время работы возникали ошибки. Данные уведомления отправляются в специальный чат корпоративного мессенджера.

Заключение

Для автоматизации процесса создания и распространения SSL-сертификатов на серверы было предложено разработать скрипт, который выполняется на одном из серверов управления. Его работа заключается в получении информации из каталога сервисов; выпуске новых сертификатов в центре сертификации; обновлении устаревающих сертификатов в центре сертификации; распространении сертификатов сервисов по серверам; обновлении информации о сертификатах в каталоге сервисов.

Список литературы

1. **Айвенс К.** Внедрение, управление и поддержка сетевой инфраструктуры MS Windows Server 2003: Учеб. пособие. М.: Интернет-университет информационных технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. 914 с. URL: http://www.iprbookshop.ru/102009.html
2. **Коньков К. А., Карпов В. Е.** Основы операционных систем: Учебник. М.: Интернет-университет информационных технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. 346 с. URL: http://www.iprbookshop.ru/102031.html
3. **Костюк А. И., Беспалов Д. А.** Администрирование баз данных и компьютерных сетей: Учеб. пособие. Ростов н/Д; Таганрог: Изд-во ЮФУ, 2020. 127 с. URL: http://www.iprbookshop.ru/107941.html
4. **Мейер Б.** Инструменты, алгоритмы и структуры данных: Учеб. пособие. М.: Интернет-университет информационных технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. 540 с. URL: http://www.iprbookshop.ru/102012.html

References

1. **Ivens K.** Implementation, management and support of the network infrastructure of MS Windows Server 2003: tutorial. Moscow, Internet University of Information Technologies (INTUIT), Ai Pi Ar Media, 2021, 914 p. (in Russ.) URL: http://www.iprbookshop.ru/102009.html
2. **Konkov K. A., Karpov V. E.** Fundamentals of operating systems: textbook. Moscow, Internet University of Information Technologies (INTUIT), Ai Pi Ar Media, 2021, 346 p. (in Russ.) URL: http://www.iprbookshop.ru/102031.html. – EBS "IPRbooks"
3. **Kostyuk A. I., Bespalov D. A.** Administration of databases and computer networks: tutorial. Rostov on Don, Taganrog, Publishing House of the Southern Federal University, 2020, 127 p. (in Russ.) URL: http://www.iprbookshop.ru/107941.html
4. **Meyer B.** Tools, algorithms and data structures: tutorial. Moscow, Internet University of Information Technologies (INTUIT), Ai Pi Ar Media, 2021, 540 p. (in Russ.) URL: http://www.iprbookshop.ru/102012.html

Информация об авторах

Федькова Надежда Александровна, кандидат экономических наук, доцент
Author ID 724535
Федьков Дмитрий Андреевич, студент магистратуры

Information about the Authors

Fedkova Nadezhda Alexandrovna, Candidate of Economic Sciences
Author ID 724535
Fedkov Dmitry Andreevich, Master's Student

Статья поступила в редакцию 01.12.2021;
одобрена после рецензирования 01.02.2022; *принята к публикации 01.02.2022*
The article was submitted 01.12.2021;
approved after reviewing 01.02.2022; *accepted for publication 01.02.2022*