CPAR 8.0でのカスタムスクリプトの設定

内容

概要

このドキュメントでは、スクリプトと拡張ポイントを使用してCisco Prime Access Registrar(CPAR)8.0の動作をカスタマイズする方法について説明します。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- CPAR 8.0管理

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- CentOS 6.5 64ビットにCPAR 8.0をインストール

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

背景説明

CPARは、内部スクリプトと外部スクリプトの両方で変更できます。スクリプトはC/C++/Java/TCLで記述できます。スクリプトを使用して、RADIUS、TACACS、およびDIAMETERパケットの処理を変更できます。スクリプトは、拡張ポイントでCPARで参照できます。拡張ポイントは、一部の設定要素の下に表示される設定/属性で、スクリプトを参照できます。リファレンスガイドに従って、CPARはカスタムスクリプトによるデータの損失や破損などの責任を負いません。

ネットワークデバイス設定の2つの拡張ポイントの例を次に示します

```
[ //localhost/Radius/Clients/piborowi ]
    Name = piborowi
    Description =
    Protocol = tacacs-and-radius
    IPAddress = 192.168.255.15
    SharedSecret = <encrypted>
    Type = NAS
    Vendor =
    IncomingScript~ =                          // Extension point for incomming traffic
    OutgoingScript~ =                          // Extension point for outgoing traffic
    EnableDynamicAuthorization = FALSE
    NetMask =
    EnableNotifications = FALSE
    EnforceTrafficThrottling = TRUE
```

CPARアドミニストレーションガイドによると、複数の拡張ポイントを使用できます。着信スクリプトは、次の各拡張ポイントで参照できます。

- RADIUS サーバ
- ベンダー（即時クライアント）
- クライアント（個別のNAS）
- NAS-Vendor-Behind-the-Proxy
- Client-Behind-the-Proxy
- リモートサーバ（タイプRADIUS）
- サービス

認証または認可スクリプトは、次の各拡張ポイントで参照できます。

- グループ認証
- ユーザ認証
- グループ認可
- ユーザ認証

発信スクリプトは、次の各拡張ポイントで参照できます。

- サービス
- Client-Behind-the-Proxy
- NAS-Vendor-Behind-the-Proxy
- クライアント（個別のNAS）
- NASベンダー
- RADIUS サーバ

複数の拡張ポイントがあるため、CPARがスクリプトを実行する順序を理解することが重要です。29の使用可能なスクリプティング/拡張ポイントの順序については、管理者ガイドの表7-1を参照してください。

内部スクリプトは、CPAR CLI(aregcmd)で直接設定されるスクリプトです。 外部ファイルやプログラミングの知識は必要ありません。外部スクリプトは、オペレーティングシステム（CENTOSまたはRHEL）のファイルに保存され、CPAR CLIで参照されます。

設定

発信トラフィックの内部スクリプト

内部スクリプトでは、次の修飾子を使用できます。

1. +rsp: – 応答の追加と属性

2.-rsp: – 応答から属性を削除

を選択します。#rsp: – 属性を新しい値に置き換えます

4.上記はreq(要求/受信パケットとenv（これは環境ディクショナリ）に使用できます。 例+req:または – env:

/Radius/Scriptsの下に内部スクリプトを追加します。Access-Acceptパケットとともに返される2つの追加AVPを設定します。Filter-IdおよびVendor-Specific（音声ドメインに参加）。

```
--> ls -R

[ //localhost/Radius/Scripts/addattr ]
    Name = addattr
    Description =
    Language = internal
    Statements/
        1. +rsp:Filter-Id=PhoneACL
        2. +rsp:Cisco-AVPair=device-traffic-class=voice

--> ls -R

[ Services/local-users ]
    Name = local-users
    Description =
    Type = local
    IncomingScript~ =
    OutgoingScript~ = addattr
    OutagePolicy~ = RejectAll
    OutageScript~ =
    UserList = Default
    EnableDeviceAccess = True
    DefaultDeviceAccessAction~ = DenyAll
    DeviceAccessRules/
        1. switches
```

ローカルradclientを使用してテストします。

```
--> simple
```

```
p011
--> p011 send
p014
--> p014
Packet: code = Access-Accept, id = 18, length = 64, attributes =
        Filter-Id = PhoneACL
        Cisco-AVPair = device-traffic-class=voice
```

トレース：

```
07/31/2019 10:31:26.254: P2363: Running Service local-users's OutgoingScript: addattr
07/31/2019 10:31:26.254: P2363: Internal Script for 1   +rsp:Filter-Id=PhoneACL : Filter-Id =
PhoneACL
07/31/2019 10:31:26.254: P2363: Setting value PhoneACL for attribute Filter-Id
07/31/2019 10:31:26.254: P2363: Trace of Response Dictionary
07/31/2019 10:31:26.254: P2363: Trace of Access-Request packet
07/31/2019 10:31:26.254: P2363:    identifier = 18
07/31/2019 10:31:26.254: P2363:    length = 30
07/31/2019 10:31:26.254: P2363:    respauth = fb:63:14:3f:c1:fb:ac:03:7d:16:29:61:ba:ef:13:4f
07/31/2019 10:31:26.254: P2363:    Filter-Id = PhoneACL
07/31/2019 10:31:26.254: P2363: Internal Script for 2   +rsp:Cisco-AVPair=device-traffic-
class=voice : Cisco-AVPair = device-traffic-class=voice
07/31/2019 10:31:26.254: P2363: Setting value device-traffic-class=voice for attribute Cisco-
AVPair
07/31/2019 10:31:26.254: P2363: Trace of Response Dictionary
07/31/2019 10:31:26.254: P2363: Trace of Access-Request packet
07/31/2019 10:31:26.254: P2363:    identifier = 18
07/31/2019 10:31:26.254: P2363:    length = 64
07/31/2019 10:31:26.254: P2363:    respauth = fb:63:14:3f:c1:fb:ac:03:7d:16:29:61:ba:ef:13:4f
07/31/2019 10:31:26.254: P2363:    Filter-Id = PhoneACL
07/31/2019 10:31:26.254: P2363:    Cisco-AVPair = device-traffic-class=voice
```

着信トラフィックの内部スクリプト

user@domain形式のすべてのユーザ名をanonymousに置き換える新しいスクリプトを作成し、使用するサービスの着信スクリプトとして適用します。

設定例:

```
--> cd /Radius/Scripts

--> add test

--> set language internal

--> cd Statements

--> add 1

--> cd 1

--> set statements "#req:User-Name=~(.*)(@[a-z]+.[a-z]+)~\anonymous"

--> ls -R

[ //localhost/Radius/Scripts/test ]
    Name = test
    Description =
    Language = internal
    Statements/
        1. #env:User-Name=~(.*)~anonymous

--> ls -R /Radius/Services/employee-service/

[ /Radius/Services/employee-service ]
```

```
    Name = employee-service
    Description =
    Type = local
    IncomingScript~ = test
    OutgoingScript~ =
    OutagePolicy~ = RejectAll
    OutageScript~ =
    UserList = default
    EnableDeviceAccess = FALSE
    DefaultDeviceAccessAction~ = DenyAll
```

radclientを使用したテスト（ユーザ名がanonymousに変更されているため、要求はおそらく拒否されます）：

```
--> simple




p01e

--> p01e
Packet: code = Access-Request, id = 27, length = 72, attributes =
User-Name = <username>@cisco.com
User-Password = <password>
NAS-Identifier = localhost
NAS-Port = 7
--> p01e send
p020
--> p020
Packet: code = Access-Reject, id = 27, length = 35, attributes =
        Reply-Message = Access Denied
```

トレース：

従業員サービスが実行される前に、3つのスクリプトが呼び出されます。最初にCPARは*CiscoIncomingScriptを呼び出*し、次に*ParseServiceHints*を呼び出します。これはlocalhostクライアント/ネットワークデバイスの設定に関連付けられています。パケットからユーザ名を抽出し、環境辞書に登録します。2番目のスクリプト*test*が呼び出され、環境辞書のユーザ名が<username>からanonymousに変更されます

localhostクライアント：

```
[ //localhost/Radius/Clients/localhost ]
    Name = localhost
    Description =
    Protocol = radius
    IPAddress = 127.0.0.1
    SharedSecret = <encrypted>
    Type = NAS+Proxy
    Vendor = Cisco
    IncomingScript~ = ParseServiceHints
    OutgoingScript~ =
    EnableDynamicAuthorization = FALSE
    NetMask =
    EnableNotifications = FALSE
    EnforceTrafficThrottling = TRUE
```

トレース出力：

```
07/31/2019 11:38:53.522: P2855: PolicyEngine: [SelectPolicy] Successful
07/31/2019 11:38:53.522: P2855: Using Client: localhost
07/31/2019 11:38:53.522: P2855: Using Vendor: Cisco
07/31/2019 11:38:53.522: P2855: Running Vendor Cisco's IncomingScript: CiscoIncomingScript
07/31/2019 11:38:53.522: P2855: Running Client localhost IncomingScript: ParseServiceHints
07/31/2019 11:38:53.522: P2855: Rex: environ->get( "Request-Type" ) -> "Access-Request"
07/31/2019 11:38:53.522: P2855: Rex: environ->get( "Request-Type" ) -> "Access-Request"
07/31/2019 11:38:53.522: P2855: Rex: environ->get( "User-Name" ) -> "<username>"


07/31/2019 11:38:53.522: P2855: Authenticating and Authorizing with Service employee-service
07/31/2019 11:38:53.522: P2855: Running Service employee-service's IncomingScript: test
07/31/2019 11:38:53.522: P2855: Numbered attribute got for the radius / tacacs packet. ignoring
# User-Name
07/31/2019 11:38:53.523: P2855: Numbered attribute got for the radius / tacacs packet. ignoring
# User-Name
07/31/2019 11:38:53.523: P2855: Numbered attribute got for the radius / tacacs packet. ignoring
# User-Name
07/31/2019 11:38:53.523: P2855: Internal Script for 1   #env:User-Name=~(.*)~anonymous : User-
Name = anonymous
07/31/2019 11:38:53.523: P2855: Setting value anonymous for attribute User-Name
07/31/2019 11:38:53.523: P2855: Trace of Environment Dictionary
07/31/2019 11:38:53.523: P2855:           User-Name = anonymous
07/31/2019 11:38:53.523: P2855:           NAS-Name-And-IPAddress = localhost (127.0.0.1)
07/31/2019 11:38:53.523: P2855:           Authorization-Service = employee-service
07/31/2019 11:38:53.523: P2855:           Source-Port = 51169
07/31/2019 11:38:53.523: P2855:           Authentication-Service = employee-service
07/31/2019 11:38:53.523: P2855:           Trace-Level = 1000
07/31/2019 11:38:53.523: P2855:           Destination-Port = 1812
07/31/2019 11:38:53.523: P2855:           Destination-IP-Address = 127.0.0.1
07/31/2019 11:38:53.523: P2855:           Source-IP-Address = 127.0.0.1
07/31/2019 11:38:53.523: P2855:           Enforce-Traffic-Throttling = TRUE
07/31/2019 11:38:53.523: P2855:           Request-Type = Access-Request
07/31/2019 11:38:53.523: P2855:           Script-Level = 6
07/31/2019 11:38:53.523: P2855:           Provider-Identifier = Default
07/31/2019 11:38:53.523: P2855:           Request-Authenticator =
5f:62:5a:72:0f:7b:a2:2a:9c:06:ba:2e:bd:f4:e4:4b
07/31/2019 11:38:53.523: P2855:           Realm = cisco.com
07/31/2019 11:38:53.523: P2855: Getting User anonymous's UserRecord from UserList Default
07/31/2019 11:38:53.523: P2855: Failed to get User anonymous's UserRecord from UserList Default
07/31/2019 11:38:53.523: P2855: Running Vendor Cisco's OutgoingScript: CiscoOutgoingScript
07/31/2019 11:38:53.523: P2855: Trace of Access-Reject packet
07/31/2019 11:38:53.523: P2855:    identifier = 27
07/31/2019 11:38:53.523: P2855:    length = 35
07/31/2019 11:38:53.523: P2855:    respauth = d3:7d:b3:f6:05:47:2c:66:d9:c0:01:7d:67:d7:93:99
07/31/2019 11:38:53.523: P2855:    Reply-Message = Access Denied
07/31/2019 11:38:53.523: P2855: Sending response to 127.0.0.1
```

外部スクリプトの作成

/opt/CSCOar/scripts/radius/tcl/ディレクトリにファイル*nadip*.tclを追加し、次の内容を追加します。

```
[root@piborowi-cpar80-16 tcl]# cat /opt/CSCOar/scripts/radius/tcl/nadip.tcl
proc UpdateNASIP {request response environ} {
$request trace 2 "TCL CUSTOM_SCRIPT Updating NAS IP ADDRESS"
$request trace 2 "Before put: " [ $request get NAS-IP-Address ]
```

```
$request put NAS-IP-Address 1.2.3.4
$request trace 2 "After put: " [ $request get NAS-IP-Address ]
}
```

nadip.tclの*内容*を次の行で説明します。

行#1プロシージャの定義と引数。要求、応答、環境、および3つの使用可能な辞書で、セッション/パケットデータを変更できます。

行#2スクリプトをトレースレベル2として出力するためのデバッグ行。

行#3この値を設定する前に、要求ディクショナリ内のNAS-IP-Address属性の内容。

行#4要求辞書のNas-IP-Address属性を1.2.3.4に設定します。

#5行NAS-IP-Address属性を再度印刷します。

スクリプトが作成され、オペレーティングシステムに保存されたら、スクリプトへのCPAR参照を設定します。言語をTCLに設定します。ファイル名は拡張子を持つ正確なファイル名でなければなりません（この場合はnadip.tclです）。 EntryPointは、スクリプトとして実行するファイル内のプロシージャの名前です。参照がサービス中のCPARスクリプト(incomingScript)を作成し、radclientでテストします。

行#2、#3、#5はトレースで確認できます。

```
--> ls -R /Radius/scripts/nadipaddress/

[ /Radius/Scripts/nadipaddress ]
    Name = nadipaddress
    Description =
    Language = tcl             <<<<<<<<
    Filename = nadip.tcl       <<<<<<<<
    EntryPoint = UpdateNASIP   <<<<<<<<
    InitEntryPoint =
    InitEntryPointArgs =

--> ls -R /Radius/services/employee-service/

[ /Radius/Services/employee-service ]
    Name = employee-service
    Description =
    Type = local
    IncomingScript~ = nadipaddress    <<<<<<<<
    OutgoingScript~ =
    OutagePolicy~ = RejectAll
    OutageScript~ =
    UserList = default
    EnableDeviceAccess = FALSE
    DefaultDeviceAccessAction~ = DenyAll
```

トレース：

```
07/31/2019 13:40:53.615: P3490: Running Service employee-service's IncomingScript: nadipaddress
07/31/2019 13:40:53.615: P3490: TCL CUSTOM_SCRIPT Updating NAS IP ADDRESS
07/31/2019 13:40:53.616: P3490:     Tcl: request trace 2 TCL CUSTOM_SCRIPT Updating NAS IP
ADDRESS -> OK
```

```
07/31/2019 13:40:53.616: P3490:     Tcl: request get NAS-IP-Address -> <empty>
07/31/2019 13:40:53.616: P3490: Before put:
07/31/2019 13:40:53.616: P3490:     Tcl: request trace 2 Before put:   -> OK
07/31/2019 13:40:53.616: P3490:     Tcl: request put NAS-IP-Address 1.2.3.4 -> OK
07/31/2019 13:40:53.616: P3490:     Tcl: request get NAS-IP-Address -> 1.2.3.4
07/31/2019 13:40:53.616: P3490: After put: 1.2.3.4
07/31/2019 13:40:53.616: P3490:     Tcl: request trace 2 After put:  1.2.3.4 -> OK
```