Configurar scripts personalizados en CPAR 8.0

Contenido

Introducción

Este documento describe cómo personalizar el comportamiento de Cisco Prime Access Registrar (CPAR) 8.0 con el uso de scripts y puntos de extensión.

Prerequisites

Requirements

Cisco recomienda que tenga conocimiento sobre estos temas:

- CPAR 8.0 administración

Componentes Utilizados

La información que contiene este documento se basa en las siguientes versiones de software y hardware.

- CPAR 8.0 instalado en CentOS 6.5 de 64 bits

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Antecedentes

El CPAR se puede modificar mediante scripts internos y externos. Las secuencias de comandos se pueden escribir en C/C++/Java/TCL. Las secuencias de comandos se pueden utilizar para modificar el procesamiento de los paquetes RADIUS, TACACS y DIAMETER. Se puede hacer referencia a las secuencias de comandos en el CPAR en los puntos de extensión. Los puntos de extensión son un valor de configuración/atributo que aparece debajo de algunos de los elementos de configuración y permite hacer referencia a un script. Según la guía de referencia, el CPAR no

es responsable de ninguna pérdida de datos, daños, etc. causados por guiones personalizados.

Este es un ejemplo de dos puntos de extensión en la configuración del dispositivo de red

```
[ //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
```

Según la guía de administración del CPAR, hay varios puntos de extensión disponibles. Se puede hacer referencia a un script entrante en cada uno de estos puntos de extensión:

- servidor RADIUS
- Proveedor (del cliente inmediato)
- Cliente (NAS individual)
- NAS-Vendor-Behind-the-Proxy
- Cliente-Detrás-del-Proxy
- Servidor remoto (de tipo RADIUS)
- Servicio

Se puede hacer referencia a un script de autenticación o autorización en cada uno de estos puntos de extensión:

- Autenticación de grupo
- Autenticación de usuario
- Autorización de grupo
- Autorización del usuario

Se puede hacer referencia al script saliente en cada uno de estos puntos de extensión:

- Servicio
- Cliente-Detrás-del-Proxy
- NAS-Vendor-Behind-the-Proxy
- Cliente (NAS individual)
- Proveedor NAS
- servidor RADIUS

Es fundamental comprender el orden en que el CPAR ejecuta los scripts, ya que hay varios puntos de extensión. Consulte la tabla 7-1 de la guía del administrador para ver el orden de 29 puntos de secuencias de comandos/extensión disponibles.

Una secuencia de comandos interna es una que se configura directamente en CPAR CLI (aregcmd). No requiere ningún archivo externo ni muchos conocimientos de programación. Una secuencia de comandos externa es una que se almacena en un archivo del sistema operativo (CENTOS o RHEL) y se hace referencia a ella en CPAR CLI.

Configurar

Script Interno Para El Tráfico Saliente

En las secuencias de comandos internas puede utilizar estos modificadores:

1. +rsp: - agrega y atribuye a la respuesta

2. -rsp: - quita el atributo de la respuesta

3. #rsp: - reemplaza el atributo con un nuevo valor

4. se puede utilizar para la solicitud (paquete de solicitud/entrada y env, que es el diccionario de entorno). Ejemplos +req: o -env:

Agregue un script interno en /Radius/Scripts. Configure dos AVP adicionales que se devolverán con el paquete Access-Accept: ID de filtro y específico del proveedor (para unirse al dominio de voz).

```
--> 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
```

Prueba con el uso de radclient local:

```
--> simple




p011
--> p011 send
```

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

Rastreos:

```
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
```

Script Interno Para El Tráfico Entrante

Cree una nueva secuencia de comandos que reemplace todos los nombres de usuario en el formato user@domain al anónimo y aplíquela como secuencia de comandos entrante para el servicio que utiliza.

Configure

```
--> 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
```

Prueba con radclient (lo más probable es que la solicitud se rechace porque el nombre de usuario se cambia a anónimo):

```
--> 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
```

Seguimiento:

Antes de ejecutar el servicio de empleado, se invocan tres secuencias de comandos. El primer CPAR invoca *CiscoIncomingScript*, luego invoca *ParseServiceHinks* que se asocia a la configuración de LocalHost Client/Network Device. Extrae el nombre de usuario del paquete y lo pone en el diccionario de entorno. Se llama a la segunda secuencia de comandos, *prueba* y se cambia el nombre de usuario en el diccionario de entorno de **<username>** a anonymous

cliente localhost:

```
[ //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
```

Salida de seguimiento:

```
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
```

Crear script externo

Agregue un archivo *nadip.tcl* a /opt/CSCOar/scripts/radius/tcl/ directory y agregue este contenido:

```
[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 ]
}
```

Contenido de *nadip.tcl* explicado línea por línea:

Línea n.º 1 Definición de procedimiento y argumentos. Solicitud, respuesta, entorno y tres diccionarios disponibles donde puede modificar datos de sesión/paquete.

Línea n.º 2 de depuración para que la secuencia de comandos se imprima como nivel 2 de seguimiento.

Línea Nº 3 Contenido del atributo NAS-IP-Address en el diccionario de solicitudes antes de establecer este valor.

Línea n.º 4 Establezca el atributo Nas-IP-Address en el diccionario de solicitudes en el valor 1.2.3.4.

Línea n.º 5 Vuelva a imprimir el atributo NAS-IP-Address.

Una vez que se haya creado y guardado el script en el sistema operativo, configure la referencia CPAR al script. Establezca el idioma como TCL, el nombre de archivo debe ser el nombre exacto con la extensión (en este caso es nadip.tcl). EntryPoint es el nombre del procedimiento del archivo que desea ejecutar como script. Haga referencia al script CPAR creado en service (incomingScript) y realice la prueba con radclient.

Las líneas 2, 3 y 5 se pueden observar en el seguimiento:

```
--> 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
```

Seguimiento:

```
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
```