Configure Custom Scripts on CPAR 8.0

Contents

Introduction

This document describes how to customize Cisco Prime Access Registrar (CPAR) 8.0 behavior with the use of scripts and extension points.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- CPAR 8.0 administration

Components Used

The information in this document is based on these software and hardware versions:

- CPAR 8.0 installed on CentOS 6.5 64 bit

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. If your network is live, ensure that you understand the potential impact of any command.

Background Information

CPAR can be modified by both internal and external scripts. Scripts can be written in C/C++/Java/TCL. Scripts can be used to modify processing of RADIUS, TACACS and DIAMETER packets. Scripts can be referenced in CPAR in extension points. Extension points is a setting/attribute that appears under some of the configuration elements and allows to reference a script. As per [reference guide](#) CPAR is not responsible for any data loss, damages, etc caused by custom scripts.

Here is an example of two extension points under network device configuration

```
[ //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
```

According to CPAR administration guide, there are multiple available extension points. An incoming script can be referenced at each of these extension points:

- RADIUS server
- Vendor (of the immediate client)
- Client (individual NAS)
- NAS-Vendor-Behind-the-Proxy
- Client-Behind-the-Proxy
- Remote Server (of type RADIUS)
- Service

An authentication or authorization script can be referenced at each of these extension points:

- Group Authentication
- User Authentication
- Group Authorization
- User Authorization

The outgoing script can be referenced at each of these extension points:

- Service
- Client-Behind-the-Proxy
- NAS-Vendor-Behind-the-Proxy
- Client (individual NAS)
- NAS Vendor
- RADIUS server

It is crucial to understand the order in which scripts are executed by CPAR since there are multiple extension points. Refer to table 7-1 of administrator guide to see the order of 29 available scripting/extension points.

An internal script is a one which is configured directly in CPAR CLI (aregcmd). It does not require any external files and much programming knowledge. An external script is a one that is stored in a file in operating system (CENTOS or RHEL) and is just referenced in CPAR CLI.

Configure

Internal Script For Outgoing Traffic

In internal scripts you can use these modifiers:

1. +rsp: - adds and attribute to response

2. -rsp: - removes attribute from response

3. #rsp: - replaces attribute with new value

4. above can be used for req (request/incomming packet and env, which is environment dictionary). Examples +req: or -env:

Add an internal Script under /Radius/Scripts. Configure two additional AVP to be returned with Access-Accept packet: Filter-Id and Vendor-Specific one (to join voice domain).

```
--> 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
```

Test with the use of local radclient:

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

Traces:

```
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
```

Internal Script For Incoming Traffic

Create a new script that replaces all usernames in format user@domain to anonymous and apply it as incomming script for the service you use.

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
```

Test with radclient (request is most probably rejected because the username is changed to anonymous):

```
--> simple <username>@cisco.com <password>
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
```

Trace:

Before employee-service is executed, three scripts are invoked. First CPAR invokes *CiscoIncomingScript*, then it invokes *ParseServiceHints* which is attached to localhost Client/Network Device configuration. It extracts username from packet and puts it in environment dictionary. Second script, *test* is invoked and username in environment dictionary is changed from <username> to anonymous

localhost client:

```
[ //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
```

Trace output:

```
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
```

Create External Script

Add a file *nadip.tcl* to /opt/CSCOar/scripts/radius/tcl/ directory and add this content:

```
[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 ]
}
```

Content of *nadip.tcl* explained line by line:

Line #1 Procedure definition and arguments. Request, response, environ and three available dictionaries where you can modify session/packet data.

Line #2 Debug line for script to be printed as trace level 2.

Line #3 Content of NAS-IP-Address attribute in request dictionary before you set this value.

Line #4 Set Nas-IP-Address attribute in request dictionary to value 1.2.3.4.

Line #5 Print NAS-IP-Address attribute again.

Once script is created and saved in operating system, configure CPAR reference to the script. Set language as TCL, filename must be exact filename with extension (in this case it is nadip.tcl). EntryPoint is the name of the procedure in the file that you would like to execute as a script. Reference created CPAR script under service (incomingScript) and test with radclient.

Lines #2, #3, #5 can be observed in the trace:

```
--> 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
```

Trace:

```
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
```