CPAR 8.0에서 사용자 지정 스크립트 구성

목차

소개

이 문서에서는 스크립트 및 확장 포인트를 사용하여 Cisco CPAR(Prime Access Registrar) 8.0 동작을 사용자 지정하는 방법에 대해 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- CPAR 8.0 관리

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- CentOS 6.5 64비트에 설치된 CPAR 8.0

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다.이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다.네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 이해해야 합니다.

배경 정보

CPAR은 내부 스크립트와 외부 스크립트에서 모두 수정할 수 있습니다.스크립트는 C/C++/Java/TCL로 작성할 수 있습니다.스크립트를 사용하여 RADIUS, TACACS 및 DIAMETER 패킷의 처리를 수정할 수 있습니다.확장 포인트에서 CPAR에서 스크립트를 참조할 수 있습니다.확장 지점은 일부 구성 요소 아래에 표시되며 스크립트를 참조할 수 있는 설정/특성입니다.참조 설명서에 따라 CPAR은 맞춤형 스크립트로 인한 데이터 손실, 손상 등에 대해 책임을 지지 않습니다.

다음은 네트워크 디바이스 컨피그레이션 아래의 두 확장 포인트의 예입니다.

```
[ //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 관리 설명서에 따르면 여러 개의 확장 포인트가 사용 가능합니다.들어오는 스크립트는 다음 각 확장 지점에서 참조할 수 있습니다.

- RADIUS 서버
- 공급업체(직접 클라이언트)
- 클라이언트(개별 NAS)
- NAS-Vendor-Behind-the-Proxy
- Client-Behind-the-Proxy
- 원격 서버(RADIUS 유형)
- 서비스

인증 또는 권한 부여 스크립트는 다음 각 확장 지점에서 참조할 수 있습니다.

- 그룹 인증
- 사용자 인증
- 그룹 권한 부여
- 사용자 권한 부여

나가는 스크립트는 다음 각 확장 지점에서 참조할 수 있습니다.

- 서비스
- Client-Behind-the-Proxy
- NAS-Vendor-Behind-the-Proxy
- 클라이언트(개별 NAS)
- NAS 공급업체
- RADIUS 서버

확장 지점이 여러 개이므로 CPAR에서 스크립트를 실행하는 순서를 이해하는 것이 중요합니다.사용 가능한 스크립팅/확장 지점 29개의 순서를 보려면 관리자 가이드 표 7-1을 참조하십시오.

내부 스크립트는 CPAR CLI(aregcmd)에서 직접 구성된 스크립트입니다. 외부 파일과 프로그래밍 지식이 많이 필요하지 않습니다.외부 스크립트는 운영 체제(CENTOS 또는 RHEL)의 파일에 저장되어 CPAR CLI에서만 참조되는 스크립트입니다.

구성

발신 트래픽용 내부 스크립트

내부 스크립트에서 다음 한정자를 사용할 수 있습니다.

1.+rsp:- 응답에 특성 추가

2 .-rsp:- 응답에서 특성 제거

3 .#rsp:- 속성을 새 값으로 바꿉니다.

4.위의 내용은 req(환경 사전인 패킷 및 env)에 사용할 수 있습니다. 예 +req:또는 -env:

/Radius/Scripts 아래에 내부 스크립트를 추가합니다.Access-Accept 패킷과 함께 반환되도록 두 개의 추가 AVP를 구성합니다.Filter-Id 및 Vendor-Specific one(음성 도메인에 가입하기 위한).

```
--> 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
```

로컬 radclient를 사용하여 테스트:

```
--> simple
```

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

추적:

```
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
```

수신 트래픽용 내부 스크립트

모든 사용자 이름을 user@domain 형식으로 익명 형식으로 바꾸고 사용하는 서비스에 대한 인코밍 스크립트로 적용하는 새 스크립트를 만듭니다.

구성:

```
--> 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
```

radclient로 테스트(사용자 이름이 anonymous로 변경되었으므로 요청이 대부분 거부됨):

```
--> 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
```

추적:

직원 서비스가 실행되기 전에 세 개의 스크립트가 호출됩니다.첫 번째 CPAR은 *CiscoIncomingScript를* 호출한 다음 로컬 호스트 클라이언트/네트워크 장치 구성에 연결된 ParseServiceHints를 호출합니다.패킷에서 사용자 이름을 추출하여 환경 사전에 넣습니다.두 번째 스크립트, *테스트*가 호출되고 환경 사전의 사용자 이름이 <username>에서 익명으로 변경됩니다.

로컬 호스트 클라이언트:

```
[ //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
```

추적 출력:

```
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
```

외부 스크립트 만들기

/opt/CSCOar*/scripts/radius/tcl/* 디렉토리에 nadip.tcl 파일을 추가하고 다음 내용을 추가합니다.

```
[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*의 내용은 라인별로 설명합니다.

라인 #1 프로시저 정의 및 인수세션/패킷 데이터를 수정할 수 있는 요청, 응답, 환경 및 3개의 사용 가능한 사전.

줄 #2 추적 레벨 2로 인쇄할 스크립트의 디버그 줄

줄 #3 이 값을 설정하기 전에 요청 사전에서 NAS-IP-Address 특성의 내용.

라인 #4 요청 사전의 Nas-IP-Address 특성을 값 1.2.3.4으로 설정합니다.

행 #5 NAS-IP-Address 속성을 다시 인쇄합니다.

스크립트가 생성되어 운영 체제에 저장되면 스크립트에 대한 CPAR 참조를 구성합니다.언어를 TCL로 설정합니다. 파일 이름은 확장자가 있는 정확한 파일 이름이어야 합니다(이 경우 nadip.tcl). EntryPoint는 스크립트로 실행할 파일의 프로시저 이름입니다.서비스(incomingScript)에서 생성된 CPAR 스크립트 및 radclient를 사용하여 테스트한 참조

#2, #3, #5는 추적에서 확인할 수 있습니다.

```
--> 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
```

추적:

```
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
```