网关到网守 (H.235) 与网守到网守(IZCT) 安全故障排除指南

Contents

Introduction

H.323网络有不同的种类配置和呼叫流。本文与介入网守的H.323网络讨论大多安全性问题。本文汇总方式每个功能工作和如何排除故障它与在大多的一说明调试。本文不讨论VoIP整体安全。

本文包括这些功能：

- **域内网关到网守安全**—此安全根据H.235， H.323呼叫由网守验证，授权，并且路由。网守认为实际上知道和可信的实体网关不验证它，当网关设法向它登记时。
- **网守安全的域间网守**—使用**区域间清除令牌(IZCT)，**此安全包括正在验证和授权在Internet Telephone Service Provider (ITSP)之间管理域的H.323呼叫。本文包括终止的网守发送在其位置确认(LCF)消息的一标记仅的部分，以便验证answerCall Admission Request (ARQ)。位置请求(LRQ)验证在此功能没有包括。LRQ验证是定于将来Cisco IOS软件版本功能。

定义

缩略语	定义

ARQ	准入请求—从一个H.323终端的一注册、准入和状态协议(RAS)发送的消息到请求接纳建立呼叫的网守。
ACF	准入确认—从网守发送的RAS信息到确认呼叫的接受的终端。
ARJ	准入拒绝—一个RAS信息从网守到拒绝准入请求的终端。
CAT	Cisco访问令牌— H.235清楚的令牌。
CHAP	挑战握手验证协议—使用挑战的认证协议。
GCF	关守确认—从网守发送的RAS信息到确认网守的发现的H.323终端。
GRQ	关守请求—从H.323终端发送的RAS信息发现网守。
H.235	安全的ITU H系列(H.323和其他H.245-based)多媒体终端的建议和加密。
IZCT	区域间清除令牌— IZCT在始发关守生成，当LRQ启动时或ACF将为在ITSP管理域内的一区域内部呼叫发送。
LRQ	位置请求—从网守发送的RAS信息到下一跳网守或呼叫段跟踪和路由呼叫。
RAS	注册、接纳和状态—允许网守执行注册、终端的接纳和状态检查的协议。
RCF	注册确认—从网守发送的RAS信息到确认注册的终端。
RRJ	注册拒绝—从拒绝注册请求的网守发送的RAS信息。
RRQ	注册请求—从终端发送的RAS信息到请求向它登记的网守。
RIP	进展中的请求—从网守发送的RAS信息到陈述的发送方呼叫进展中。

域内网关到网守安全

H.323是ITU建议地址保护在不安全网络的实时通信。这介入注意事项两个主要部分：验证和保密性。有验证的两种类型，根据H.235 ：

- 不要求在通信实体之间的前期联系方式的基于对称加密的认证。

- 基于能力有某前期共享机密(进一步被参考作为基于的订阅)，提供基于预订的验证两表：密码证书

令牌中传输的时间戳

时间戳用于防止重放攻击。所以，它是需要的供相互可接受的参考计时(从派生时间戳哪些)。相当数量可接受时间反称性是本地实施问题。

Cisco 如何实现 H.235 建议

思科使用质询握手验证协议(CHAP)象的认证机制作为基本类型其网关对其网守H.235实施。这允许您有效利用验证、授权和统计(AAA)，使用现有的功能执行实际验证。也意味着网守没有要求访问网关ID、用户帐户客户编号、密码和管脚数据库。方案根据H.235，第10.3.3部分。它描述作为与散列的基于预订的密码。

然而，而不是使用H.225 cryptoTokens，此方法以被填充的字段使用H.235 clearTokens适当地为了用在RADIUS上。此标记指Cisco访问令牌(CAT)。您在网守可总是执行验证本地而不是使用RADIUS服务器。

使用cryptoTokens要求网守维护或有某个方式获取所有的密码用户和网关。这是因为cryptoToken的令牌的字段指定这样正在验证实体需要密码生成比较已接收一个的其自己的标记。

完全Cisco网守忽略cryptoTokens。然而，支持H.235第10.3.3部分使用cryptoTokens验证网关的vocalTek网守和人。Cisco网守使用CAT验证网关。因为网关不知道什么类型的网守谈与，在RRQ发送两个。在GRQ的authenticationCapability是为cryptoToken并且表明MD5切细是认证机制(虽然CAT也使用MD5)。

参考Cisco H.323网关安全和记帐增强功能欲知更多信息。

如何配置安全等级

- 终端或注册级安全当注册安全启用在网守，网关在所有重量级的RRQ消息要求包括CAT。CAT，在这种情况下，包含验证网关到网守的信息。网守格式化消息到验证在标记包含的信息的RADIUS服务器。它响应回到有Access-Accept或访问拒绝的网守。这，反过来，响应到有RCF或RRJ的网关。如果呼叫从一个顺利地验证的网关然后发出，使用网关密码，该网关收到从网守的ACF后生成新的CAT。此CAT是相同的到在注册时生成的那个除了时间戳。它在流出的设置信息安置。目的地网关在目的地端ARQ解压缩从设置信息的标记并且安置它。在发送目的地端ACF前，网守使用RADIUS验证始发网关。这防止认识一个网关地址从使用它避免安全机制和访问公共交换电话网(PSTN)的一个未验证的终端。所以，在此级别，没有需要包括所有令牌在产生的ARQ。选择[no]从网守命令行界面(CLI)的**security token required-for registration**配置网守。命令的选项不造成网守不再检查在RAS消息的令牌。命令的选项不造成网关不再生成RAS消息的令牌。
- 每呼叫级别安全每呼叫安全构件在注册级安全。除会议注册安全需求之外，当每呼叫安全在网守时，启用网关在所有始发端ARQ消息也要求包括访问令牌。标记在这种情况下包含识别网关用户到网守的信息。使用在网关的交互语音应答(IVR)脚本此信息得到。此输入他们的用户ID和PIN的提示用户从键盘，在他们发出呼叫前。在产生的ARQ包含的CAT由RADIUS在终端或注册级安全验证以与描述相似的方式前。在它接收ACF后，网关生成新的CAT使用其密码并且在H.225设置信息内发送它到终端网关。选择[no]**安全令牌required-for全部**从网守CLI配置网守。命令的选项不造成网守不再检查在RAS消息的令牌。选择[no]**安全密码<PASSWORD>级别每**

呼叫从网关CLI配置网关。命令的选项不造成网关不再生成RAS消息的令牌。

- 所有成水平安全这在为呼叫需要的为注册和所有RAS消息允许网关包括CAT。所以，它是上述两个级别的组合。使用此选项， CAT的验证在ARQ消息的根据做一呼叫用户的帐号和PIN。总计发送的CAT的验证其他RAS消息根据为网关配置的密码。所以，它类似于每呼叫级别。选择[no]**安全令牌required-for全部**从网守CLI配置网守。命令的选项不造成网守不再检查在RAS消息的令牌。选择[no]**安全密码**从网关CLI的**<PASSWORD>级别全部**配置网关。命令的选项不造成网关不再生成RAS消息的令牌。

没有 IVR 时在每呼叫级别上的 H.235 用法

H.235在一个每呼叫级别上不可能使用没有IVR。如果没有收集帐户和PIN的IVR，网关需要发送ARQ，不用清楚的令牌(但是与加密令牌)。因为Cisco网守只接受清楚的令牌，呼叫由有安全拒绝原因的网守拒绝。

此示例显示从没有配置为了IVR能收集帐户和PIN的**始发网关**收集的**h225 asn1**调试**(OGW)**。RRQ消息有一个清楚的令牌，但是ARQ不。ARJ消息被退还的到网关。

```
Mar 4 01:31:24.358: H235 OUTGOING ENCODE BUFFER::= 61 000100C0
2B955BEB 08003200 32003200 32000006 006F0067 00770000
Mar 4 01:31:24.358:
Mar 4 01:31:24.358: RAS OUTGOING PDU ::=
value RasMessage ::= registrationRequest :
  {
   requestSeqNum 29
   protocolIdentifier { 0 0 8 2250 0 3 }
   discoveryComplete FALSE
   callSignalAddress
   {
   }
   rasAddress
   {
    ipAddress :
    {
     ip 'AC100D0F'H
     port 57514
    }
   }
   terminalType
   {
    mc FALSE
    undefinedNode FALSE
   }
   gatekeeperIdentifier {"ogk1"}
   endpointVendor
   {
    vendor
    {
     t35CountryCode 181
     t35Extension 0
     manufacturerCode 18
    }
   }
   timeToLive 60
   tokens
   !--- Clear Token is included in the RRQ message. {

    {
     tokenOID { 1 2 840 113548 10 1 2 1 }
```

```
     timeStamp 731208684
     challenge 'F57C3C65B59724B9A45C93F98CCF9E45'H
     random 12
     generalID {"ogw"}
    }
   }
   cryptoTokens
   {
    cryptoEPPwdHash :
     {
      alias h323-ID : {"ogw"}
      timeStamp 731208684
      token
      {
       algorithmOID { 1 2 840 113549 2 5 }
       paramS
       {
       }
       hash "D7F85666AF3B881ADD876DD61C20D5D9"
      }
     }
    }
    keepAlive TRUE
    endpointIdentifier {"81F5E24800000001"}
    willSupplyUUIEs FALSE
    maintainConnection TRUE
   }
Mar 4 01:31:24.370: RAS OUTGOING ENCODE BUFFER::= 0E 40001C06 0008914A
00030000 0100AC10 0D0FE0AA 0003006F 0067006B 003100B5 00001212 EF000200
3B2F014D 000A2A86 4886F70C 0A010201 C02B955B EB10F57C 3C65B597 24B9A45C
93F98CCF 9E45010C 06006F00 67007700 002A0104 02006F00 670077C0 2B955BEB
082A8648 86F70D02 05008080 D7F85666 AF3B881A DD876DD6 1C20D5D9 0180211E
00380031 00460035 00450032 00340038 00300030 00300030 00300030 00300031
01000180
Mar 4 01:31:24.378: h323chan_dgram_send:Sent UDP msg. Bytes sent: 173 to
172.16.13.35:1719
Mar 4 01:31:24.378: RASLib::GW_RASSendRRQ:
3640-1#debug RRQ (seq# 29) sent to 172.16.13.35
Mar 4 01:31:24.462: h323chan_chn_process_read_socket
Mar 4 01:31:24.462: h323chan_chn_process_read_socket: fd (2) of type CONNECTED has data
Mar 4 01:31:24.462: h323chan_chn_process_read_socket: h323chan accepted/connected
Mar 4 01:31:24.462: h323chan_dgram_recvdata:rcvd from [172.16.13.35:1719] on so
ck[2]
Mar 4 01:31:24.466: RAS INCOMING ENCODE BUFFER::= 12 40001C06 0008914A
00030006 006F0067 006B0031 1E003800 31004600 35004500 32003400 38003000
30003000 30003000 30003000 310F8A01 0002003B 01000180
Mar 4 01:31:24.466:
Mar 4 01:31:24.466: RAS INCOMING PDU ::=
value RasMessage ::= registrationConfirm :
  {
   requestSeqNum 29
   protocolIdentifier { 0 0 8 2250 0 3 }
   callSignalAddress
   {
   }
   gatekeeperIdentifier {"ogk1"}
   endpointIdentifier {"81F5E24800000001"}
   alternateGatekeeper
   {
   }
   timeToLive 60
   willRespondToIRR FALSE
   maintainConnection TRUE
  }
```

```
Mar 4 01:31:24.470: RCF (seq# 29) rcvd
Mar 4 01:32:00.220: H225 NONSTD OUTGOING PDU ::=
value ARQnonStandardInfo ::=
  {
   sourceAlias
   {
   }
   sourceExtAlias
   {
   }
   callingOctet3a 129
   interfaceSpecificBillingId "ISDN-VOICE"
  }
Mar 4 01:32:00.220: H225 NONSTD OUTGOING ENCODE BUFFER::= 80 000008A0
01810B12 4953444E 2D564F49 4345
Mar 4 01:32:00.220:
Mar 4 01:32:00.220: H235 OUTGOING ENCODE BUFFER::= 61 000100C0
2B955C0F 08003200 32003200 32000006 006F0067 00770000
Mar 4 01:32:00.224:
Mar 4 01:32:00.224: RAS OUTGOING PDU ::=
value RasMessage ::= admissionRequest :
  {
   requestSeqNum 30
   callType pointToPoint : NULL
   callModel direct : NULL
   endpointIdentifier {"81F5E24800000001"}
   destinationInfo
   {
    e164 : "3653"
   }
   srcInfo
   {
    e164 : "5336",
    h323-ID : {"ogw"}
   }
   bandWidth 1280
   callReferenceValue 5
   nonStandardData
   {
    nonStandardIdentifier h221NonStandard :
    {
     t35CountryCode 181
     t35Extension 0
     manufacturerCode 18
    }
    data '80000008A001810B124953444E2D564F494345'H
   }
   conferenceID 'E1575DA6175611CC8014A6051561649A'H
   activeMC FALSE
   answerCall FALSE
   canMapAlias TRUE
   callIdentifier
    {
     guid 'E1575DA6175611CC8015A6051561649A'H
    }
    cryptoTokens
    !--- Only cryptoTokens are included, no clear ones. {
     cryptoEPPwdHash :
     {
      alias h323-ID : {"ogw"}
      timeStamp 731208720
      token
      {
       algorithmOID { 1 2 840 113549 2 5 }
```

```
      paramS
      {
      }
      hash "105475A4C0A833E7DE8E37AD3A8CDFF"
     }
    }
   }
   willSupplyUUIEs FALSE
  }
Mar 4 01:32:00.236: RAS OUTGOING ENCODE BUFFER::= 27 88001D00 F0003800
31004600 35004500 32003400 38003000 30003000 30003000 30003000 31010180
69860201 80866940 02006F00 67007740 05000005 40B50000 12138000 0008A001
810B1249 53444E2D 564F4943 45E1575D A6175611 CC8014A6 05156164 9A056120
01801100 E1575DA6 175611CC 8015A605 1561649A 2A010402 006F0067 0077C02B
955C0F08 2A864886 F70D0205 00808010 5475A4C0 A833E7DE 8E370AD3 A8CDFF01 00
Mar 4 01:32:00.240: h323chan_dgram_send:Sent UDP msg. Bytes sent: 170 to
172.16.13.35:1719
Mar 4 01:32:00.240: RASLib::GW_RASSendARQ: ARQ (seq# 30) sent to 172.16.13.35
Mar 4 01:32:00.312: h323chan_chn_process_read_socket
Mar 4 01:32:00.312: h323chan_chn_process_read_socket: fd (2) of type CONNECTED has data
Mar 4 01:32:00.312: h323chan_chn_process_read_socket: fd (2) of type CONNECTED has data
Mar 4
3640-1#01:32:00.312: h323chan_chn_process_read_socket: h323chan accepted/connected
Mar 4 01:32:00.312: h323chan_dgram_recvdata:rcvd from [172.16.13.35:1719] on so
ck[2]
Mar 4 01:32:00.312: RAS INCOMING ENCODE BUFFER::= 2C 001D8001 00
Mar 4 01:32:00.312:
Mar 4 01:32:00.312: RAS INCOMING PDU ::=
value RasMessage ::= admissionReject :
!--- ARQ is rejected with a security denial reason. {
   requestSeqNum 30
   rejectReason securityDenial : NULL
  }
Mar 4 01:32:00.312: ARJ (seq# 30) rcvd
```

参考思科H.235占和安全性增强的配置任务部分Cisco网关关于IVR配置的更多信息。

主要问题

您需要关注的主要问题包括：

- 网关和网守的配置
- 在网守和RADIUS服务器的RADIUS配置
- 网络时间协议(NTP) —您必须有在所有网关和网守的同一时间。由于认证信息包括时间戳，重要的是所有思科H.323网关和网守(或执行验证)的其他实体同步。必须同步使用NTP，思科H.323网关。
- 软件故障由于bug

不同级别的调试和呼叫流

因为所有级别安全包括注册和每呼叫案件，实验室设置此练习的该安全级别。注册部分和一次正常VoIP呼叫的呼叫流在配置里解释此处。

Note: 此处配置不完成。更多命令在debug输出之间跟随。设计显示什么问题能发生，如果不检查所有事例如配置、NTP和RADIUS。另外，网关在网守验证本地，以便您能发现什么值为网关ID和密码设置。此配置被剪断，以便仅相关的配置显示。

```
!
interface Ethernet0/0
 ip address 172.16.13.15 255.255.255.224
 half-duplex
 h323-gateway voip interface
 h323-gateway voip id gka-1 ipaddr 172.16.13.35 1718
 !--- The gatekeeper name is gka-1. h323-gateway voip h323-id gwa-1@cisco.com
 !--- The gateway H323-ID is gwa-1@cisco.com. h323-gateway voip tech-prefix 1# ! ! gateway
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
 exec-timeout 0 0
 password ww
 logging synchronous
end
!--- No NTP is configured. !--- The snipped gatekeeper configuration is like this: ! aaa new-
model aaa authentication login default local aaa authentication login h323 local aaa
authorization exec default local aaa authorization exec h323 local aaa accounting connection
h323 start-stop group radius ! username gwa-1 password 0 2222 username gwa-2 password 0 2222 !
gatekeeper zone local gka-1 cisco.com 172.16.13.35 security token required-for all
!--- The gatekeeper is configured for the "All level security". no shutdown ! ! line con 0 exec-
timeout 0 0 line aux 0 line vty 0 4 password ww line vty 5 15 ! no scheduler max-task-time no
scheduler allocate ntp master
!--- This gatekeeper is set as an NTP master. ! end
```

这些调试在本例中打开：

- debug ras
- debug h225 asn1
- debug radius
- debug aaa authentication
- debug aaa authorization

发生的第一件事是网关发送GRQ到网守和网守发送GCF到网关。网关然后发送RRQ并且等待RCF或RRJ。

在先前配置中，网关没有为为令牌是需要的任何安全级别设置，以便其GRQ不运载**authenticationCapability**。然而，当此输出显示，网守仍然退还GCF ：

```
*Mar 2 13:32:45.413: RAS INCOMING ENCODE BUFFER::= 00 A0000006
0008914A 000200AC 100D0FD2 C6088001 3C050401 00204002 00006700 6B006100
2D003102 400E0067 00770061 002D0031 00400063 00690073 0063006F 002E0063
006F006D 0080CC
*Mar 2 13:32:45.421:
*Mar 2 13:32:45.425: RAS INCOMING PDU ::=

value RasMessage ::= gatekeeperRequest :
  {
   requestSeqNum 1
   protocolIdentifier { 0 0 8 2250 0 2 }
   rasAddress ipAddress :
   {
    ip 'AC100D0F'H
    port 53958
   }
   endpointType
   {
```

```
    gateway
    {
     protocol
     {
      voice :
      {
       supportedPrefixes
       {
        {
        prefix e164 : "1#"
        }
       }
      }
     }
    }
    mc FALSE
    undefinedNode FALSE
   }
   gatekeeperIdentifier {"gka-1"}
   endpointAlias
   {
    h323-ID : {"gwa-1@cisco.com"},
!--- The H.323-ID of the gateway is gwa-1@cisco.com. e164 : "99" } } *Mar 2 13:32:45.445: RAS
OUTGOING PDU ::= value RasMessage ::= gatekeeperConfirm :
  {
   requestSeqNum 1
   protocolIdentifier { 0 0 8 2250 0 3 }
   gatekeeperIdentifier {"gka-1"}
   rasAddress ipAddress :
   {
    ip 'AC100D23'H
    port 1719
   }
  }
 !--- The gateway sends an RRQ message to the gatekeeper with the !--- IP address sent in the
GCF. This RRQ does not carry any Token information !--- because security is not configured on
the gateway. *Mar 2 13:32:45.477: RAS INCOMING ENCODE BUFFER::= 0E C0000106 0008914A 00028001
00AC100D 0F06B801 00AC100D 0FD2C608 80013C05 04010020 40000240 0E006700 77006100 2D003100
40006300 69007300 63006F00 2E006300 6F006D00 80CC0800 67006B00 61002D00 3100B500 00120E8A
02003B01 000100 *Mar 2 13:32:45.489: *Mar 2 13:32:45.493: RAS INCOMING PDU ::= value RasMessage
::= registrationRequest :
  {
  requestSeqNum 2
  protocolIdentifier { 0 0 8 2250 0 2 }
  discoveryComplete TRUE
  callSignalAddress
  {
   ipAddress :
   {
    ip 'AC100D0F'H
    port 1720
   }
  }
  rasAddress
  {
   ipAddress :
   {
    ip 'AC100D0F'H
    port 53958
   }
  }
  terminalType
  {
   gateway
```

```
   {
    protocol
    {
     voice :
     {
      supportedPrefixes
      {
       {
        prefix e164 : "1#"
       }
      }
     }
    }
   }
   mc FALSE
   undefinedNode FALSE
  }
  terminalAlias
  {
   h323-ID : {"gwa-1@cisco.com"},
   e164 : "99"
  }
  gatekeeperIdentifier {"gka-1"}
  endpointVendor
  {
   vendor
   {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
   }
  }
  timeToLive 60
  keepAlive FALSE
  willSupplyUUIEs FALSE
 }
 !--- Since the gateway does not include any tokens and !--- the gatekeeper is set to
authenticate !--- all endpoints and calls, the gatekeeper rejects the gateway request. !--- It
sends an RRJ with the securityDenial reason.


*Mar 2 13:32:45.525: RAS OUTGOING PDU ::=

value RasMessage ::= registrationReject :
  {
   requestSeqNum 2
   protocolIdentifier { 0 0 8 2250 0 3 }
   rejectReason securityDenial : NULL
!--- Gatekeeper rejects the RRQ with security denial reason. gatekeeperIdentifier {"gka-1"} }
*Mar 2 13:32:45.529: RAS OUTGOING ENCODE BUFFER::= 14 80000106 0008914A 00038301 00080067
006B0061 002D0031 *Mar 2 13:32:45.533: !--- Configure the security password 2222 level all
command on the gateway. !--- The configuration of the gateway appears like this:

!
gateway
 security password 0356095954 level all
!--- The password is hashed. ! !--- As soon as you make this change the gateway !--- sends a new
GRQ to the gatekeeper. !--- You see the authenticationCapability and algorithmOIDs.

*Mar 2 13:33:15.551: RAS INCOMING ENCODE BUFFER::= 02 A0000206
0008914A 000200AC 100D0FD2 C6088001 3C050401 00204002 00006700 6B006100
2D003102 400E0067 00770061 002D0031 00400063 00690073 0063006F 002E0063
006F006D 0080CC0C 30020120 0A01082A 864886F7 0D0205
*Mar 2 13:33:15.563:
```

```
*Mar 2 13:33:15.567: RAS INCOMING PDU ::=
value RasMessage ::= gatekeeperRequest :
  {
   requestSeqNum 3
   protocolIdentifier { 0 0 8 2250 0 2 }
   rasAddress ipAddress :
   {
    ip 'AC100D0F'H
    port 53958
   }
   endpointType
   {
    gateway
    {
     protocol
     {
      voice :
      {
       supportedPrefixes
       {
        {
         prefix e164 : "1#"
        }
       }
      }
     }
    }
    mc FALSE
    undefinedNode FALSE
   }
   gatekeeperIdentifier {"gka-1"}
   endpointAlias
   {
    h323-ID : {"gwa-1@cisco.com"},
    e164 : "99"
   }
   authenticationCapability
   {
    pwdHash : NULL
   }
   algorithmOIDs
   {
    { 1 2 840 113549 2 5 }
   }
  }
```

这解释某些在GRQ的消息：

- **authenticationCapability** —此字段只有值pwdHash。它表明MD5切细是认证机制。
- **algorithmOIDs** —算法对象ID它在这种情况下运载值(是消息摘要5散列的对象ID的1 2 840 113549 2 5)。(1 2 840 113549 2 5)翻译对iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) md5(5)。因此思科执行MD5密码散列。Rsadsi代表RSA Data Security公司RSA Data Security，公司的开放式系统互联(OSI)对象标识符是1.2.840.113549 (0x2a、0x86、0x48、0x86、0xf7， 0x0d在十六进制)，如注册由美国国家标准局(ANSI)。

网守再发送GCF作为上一条留言。网关然后发送使用验证的一RRQ以某些字段描述令牌。传送的**asn1 RRQ**信息在本例中表示。

```
*Mar 2 13:33:15.635: RAS INCOMING ENCODE BUFFER::= 0E C0000306
0008914A 00028001 00AC100D 0F06B801 00AC100D 0FD2C608 80013C05 04010020
40000240 0E006700 77006100 2D003100 40006300 69007300 63006F00 2E006300
```

```
6F006D00 80CC0800 67006B00 61002D00 3100B500 00120EEA 02003B47 014D000A
2A864886 F70C0A01 0201C02B 92A53610 B9D84DAE 58F6CB4B 5EE5DFB6 B92DD281
01011E00 67007700 61002D00 31004000 63006900 73006300 6F002E00 63006F00
6D000042 01040E00 67007700 61002D00 31004000 63006900 73006300 6F002E00
63006F00 6DC02B92 A536082A 864886F7 0D020500 80802B21 B94F3980 ED12116C
56B79F4B 4CDB0100 0100
*Mar 2 13:33:15.667:
*Mar 2 13:33:15.671: RAS INCOMING PDU ::=
value RasMessage ::= registrationRequest :
  {
   requestSeqNum 4
   protocolIdentifier { 0 0 8 2250 0 2 }
   discoveryComplete TRUE
   callSignalAddress
   {
    ipAddress :
     {
      ip 'AC100D0F'H
      port 1720
     }
    }
    rasAddress
    {
     ipAddress :
     {
      ip 'AC100D0F'H
      port 53958
     }
    }
    terminalType
    {
     gateway
     {
      protocol
      {
       voice :
       {
        supportedPrefixes
        {
         {
          prefix e164 : "1#"
         }
        }
       }
      }
     }
     mc FALSE
     undefinedNode FALSE
    }
    terminalAlias
    {
     h323-ID : {"gwa-1@cisco.com"},
     e164 : "99"
    }
    gatekeeperIdentifier {"gka-1"}
    endpointVendor
    {
     vendor
     {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
     }
    }
```

```
    timeToLive 60
    tokens
!--- Clear Token, or what is called CAT. {

     {
      tokenOID { 1 2 840 113548 10 1 2 1 }
      timeStamp 731030839
      challenge 'B9D84DAE58F6CB4B5EE5DFB6B92DD281'H
      random 1
      generalID {"gwa-1@cisco.com"}
     }
    }
    cryptoTokens
!--- CryptoToken field. {
     cryptoEPPwdHash :
     {
      alias h323-ID : {"gwa-1@cisco.com"}
      timeStamp 731030839
      token
      {
       algorithmOID { 1 2 840 113549 2 5 }
       paramS
       {
       }
       hash "2B21B94F3980ED12116C56B79F4B4CDB"
      }
     }
    }
    keepAlive FALSE
    willSupplyUUIEs FALSE
   }
```

在答复讨论前，某些上述RRQ消息的相关字段解释此处。有令牌的两种类型：清楚的令牌或CAT)和加密令牌。

如上所述， Cisco网守忽略加密令牌。然而，网关仍然发送两个，因为不知道到什么类型的网守谈。因为其他供应商也许使用加密令牌，网关发送两个。

这解释ClearToken语法。

- **tokenOID** —识别标记的对象ID。
- **时间戳**—网关的当前世界统一时间(UTC)时期。秒钟从00:00 1/1/1970 UTC。它使用作为暗示的CHAP质询，好象最初来自网守。
- **挑战**—网关生成的16字节MD5消息摘要使用这些字段：挑战= [random + GW/User Password + timeStamp] MD5哈希RADIUS执行此计算(因为认识随机数、网关密码和CHAP质询)确定什么挑战应该是：CHAP答复= [CHAP ID + UserPassword + CHAP Challenge] MD5哈希
- **随机**— RADIUS用于的一字节值识别此特定的请求。网关增加一个可变模块256每认证请求的能满足此RADIUS要求。
- **generalID** —根据安全级别和种类或用户帐户客户编号的网关H323-ID RAS信息。

Note: 所有这些字段是重要。然而，更多注意给对时间戳和generalID。在这种情况下，时间戳是**731030839，**并且generalID是**gwa-1@cisco.com**。

在RRQ的cryptoToken包含关于生成标记的网关的信息。这包括是在网关配置的H.323 ID)的网关ID(和网关密码。

此字段包含为CryptoH323Token字段定义的其中一个cryptoToken类型指定在H.225。目前，支持的cryptoToken的唯一的类型是cryptoEPPwdHash。

这些字段在cryptoEPPwdHash字段内包含：

- **别名**—网关别名，是网关的H.323 ID。
- **时间戳**—当前时间时间标记。
- **标记**—消息摘要5 (MD5)-encoded PwdCertToken。此字段包含这些项目：**时间戳**—同cryptoEPPwdHash的时间戳一样。**密码**—网关的密码。**generalID** —网关别名和在cryptoEPPwdHash包括的那个一样。**tokenID** —对象ID。

查看从网守的答复在本例中。

```
*Mar 2 13:33:15.723: RAS OUTGOING PDU ::=

value RasMessage ::= registrationReject :
  {
   requestSeqNum 4
   protocolIdentifier { 0 0 8 2250 0 3 }
   rejectReason securityDenial : NULL
!--- The gatekeeper rejects the RRQ with securityDenial reason.

   gatekeeperIdentifier {"gka-1"}
  }

*Mar 2 13:33:15.727: RAS OUTGOING ENCODE BUFFER::= 14 80000306 0008914A
00038301 00080067 006B0061 002D0031
*Mar 2 13:33:15.731:
```

RRQ由网守拒绝。对此的原因是，因为没有在网关的配置里设置的NTP。网守检查标记的时间戳发现是否在可接受的时间范围内相对其自己的时间。目前此窗口是+/- 30秒在网守附近的UTC时期。

此窗口的令牌的外部造成网守丢弃此消息。这防止重放攻击设法重新使用一被监听的标记的人。实际上，所有网关和网守需要使用NTP避免这次偏离问题。网守发现在标记的时间戳在可接受的时间范围其时间内。所以，它不检查与RADIUS验证网关。

网关为指向网守的NTP然后配置作为Ntp master，因此网关和网守有同一时间。当这发生时，网关送回一新的RRQ和这次网守回复到与RRJ的新的RRQ。

这些调试是从网守。看到调试的运行网守是否去认证阶段。

```
Mar 2 13:57:41.313: RAS INCOMING ENCODE BUFFER::= 0E C0005906 0008914A
00028001 00AC100D 0F06B801 00AC100D 0FD2C608 80013C05 04010020 40000240
0E006700 77006100 2D003100 40006300 69007300 63006F00 2E006300 6F006D00
80CC0800 67006B00 61002D00 3100B500 00120EEA 02003B47 014D000A 2A864886
F70C0A01 0201C02B 9367D410 7DD4C637 B6DD4E34 0883A7E5 E12A2B78 012C1E00
67007700 61002D00 31004000 63006900 73006300 6F002E00 63006F00 6D000042
01040E00 67007700 61002D00 31004000 63006900 73006300 6F002E00 63006F00
6DC02B93 67D4082A 864886F7 0D020500 8080ED73 946B13E9 EAED6F4D FED13478
A6270100 0100
Mar 2 13:57:41.345:
Mar 2 13:57:41.349: RAS INCOMING PDU ::=
value RasMessage ::= registrationRequest :
  {
   requestSeqNum 90
   protocolIdentifier { 0 0 8 2250 0 2 }
   discoveryComplete TRUE
   callSignalAddress
   {
    ipAddress :
    {
```

```
   ip 'AC100D0F'H
   port 1720
  }
 }
 rasAddress
 {
  ipAddress :
  {
   ip 'AC100D0F'H
   port 53958
  }
 }
 terminalType
 {
  gateway
  {
   protocol
   {
    voice :
    {
     supportedPrefixes
     {

      {
      prefix e164 : "1#"
     }
    }
   }
  }
 }
 mc FALSE
 undefinedNode FALSE
}
terminalAlias
{
 h323-ID : {"gwa-1@cisco.com"},
 e164 : "99"
}
gatekeeperIdentifier {"gka-1"}
endpointVendor
{
 vendor
 {
  t35CountryCode 181
  t35Extension 0
  manufacturerCode 18
 }
}
timeToLive 60
tokens
{

 {
  tokenOID { 1 2 840 113548 10 1 2 1 }
  timeStamp 731080661
  challenge '7DD4C637B6DD4E340883A7E5E12A2B78'H
  random 44
  generalID {"gwa-1@cisco.com"}
 }
}
cryptoTokens
{
 cryptoEPPwdHash :
 {
```

```
    alias h323-ID : {"gwa-1@cisco.com"}
    timeStamp 731080661
    token
    {
     algorithmOID { 1 2 840 113549 2 5 }
     paramS
     {
     }
     hash "ED73946B13E9EAED6F4DFED13478A627"
    }
   }
  }
  keepAlive FALSE
  willSupplyUUIEs FALSE
 }

Mar 2 13:57:41.401: AAA: parse name=<no string> idb type=-1 tty=-1
Mar 2 13:57:41.405: AAA/MEMORY: create_user (0x81416060) user='gwa-1@cisco.com'
ruser='NULL' ds0=0port='NULL' rem_addr='NULL' authen_type=CHAP
service=LOGIN priv=0 initial_task_id='0'
Mar 2 13:57:41.405: AAA/AUTHEN/START (2845574558): port='' list='h323'
action=LOGIN service=LOGIN
Mar 2 13:57:41.405: AAA/AUTHEN/START (2845574558): found list h323
Mar 2 13:57:41.405: AAA/AUTHEN/START (2845574558): Method=LOCAL
Mar 2 13:57:41.405: AAA/AUTHEN (2845574558): User not found, end of method list
Mar 2 13:57:41.405: AAA/AUTHEN (2845574558): status = FAIL
!--- Authentication fails. The user is not found on the list. Mar 2 13:57:41.405:
voip_chapstyle_auth: astruct.status = 2 Mar 2 13:57:41.405: AAA/MEMORY: free_user (0x81416060)
user='gwa-1@cisco.com' ruser='NULL' port='NULL' rem_addr='NULL' authen_type=CHAP service=LOGIN
priv=0 Mar 2 13:57:41.409: RAS OUTGOING PDU ::= value RasMessage ::= registrationReject :
  {
   requestSeqNum 90
   protocolIdentifier { 0 0 8 2250 0 3 }
   rejectReason securityDenial : NULL
   gatekeeperIdentifier {"gka-1"}
  }
```

在配置NTP以后，网守仍然拒绝RRQ。这时，然而，它通过该网关的认证过程。网守拒绝RRQ，因为用户(此处网关ID)在有效用户列表没有找到RADIUS的。网关在网守的配置里验证本地。在用户列表您看到gwa-1。然而，因为RRQ的用户是gwa-1@cisco.com，那不是正确的用户。

并且，一旦用户名gwa-1@cisco.com密码0 2222命令在网守配置，网守确认RRQ，并且网关注册。

在此实验室，另一个网关(gwa-2)注册到同一网守(gka-1)。呼叫由gwa-1@cisco.com被做到gwa-2发现ARQ、ACF和设置信息如何查找。

收集的这些调试是从始发和终接网关(gwa-2)。

- debug h225 asn1
- debug ras
- debug voip ccapi inout

某些的说明调试消息包括。

在您打印从产生/终端网关前的调试，此文本说明呼叫流：

1. 当设置信息从PSTN时接收，网关发送ARQ对并且接收从网守的ACF。
2. 当网关接收ACF时，使用网关密码、H323-ID别名和当前时间，网关生成CAT。标记在呼叫控制块(CCB)安置。
3. 当网关传送设置信息到终端网关时，从CCB获取访问令牌并且在一clearToken的

nonStandardParameter字段在设置信息内的安置它。

4. 终端网关从设置信息去除标记，从nonStandardParameter转换它到CAT，并且放置它到ARQ。
5. 网守检查标记的时间戳发现是否在可接受的时间范围内相对其自己的时间。目前此窗口是+/-30秒在网守附近的UTC时期。此窗口的令牌的外部造成网守丢弃此消息。这导致呼叫拒绝。
6. 如果标记是可接受，网守格式化RADIUS访问请求包，填写适当的属性验证CHAP质询并且发送它到RADIUS服务器。
7. 根据假设网关的别名知道在服务器使用别名、密码和CHAP质询从网守，服务器找出密码关联与此别名并且产生其自己的CHAP答复。如果其CHAP答复匹配从网守接收的那个，服务器发送访问接受数据包到网守。如果他们不配比，或者，如果网关的别名不在服务器数据库，服务器送回访问拒绝数据包到网守。
8. 网守回应到有ACF的网关，如果接收访问接受或者与原因代码**securityDenial的**一ARJ，如果接收访问拒绝。如果网关接收ACF，呼叫连接。

此示例显示从始发网关的调试。

Note: 设置的**h225 asn1**调试不在本例中，因为这是一样如在按照始发网关示例的终端网关示例中看到。

```
Mar 2 19:39:07.376: cc_api_call_setup_ind (vdbPtr=0x6264AB2C,
callInfo={called=3653,called_oct3=0x81,calling=,calling_oct3=0x81,calling_oct3a=0x0,
calling_xlated=false,subscriber_type_str=RegularLine,fdest=1,peer_tag=5336,
prog_ind=3},callID=0x61DDC2A8)
Mar 2 19:39:07.376: cc_api_call_setup_ind type 13 , prot 0
Mar 2 19:39:07.376: cc_process_call_setup_ind (event=0x6231F0C4)
Mar 2 19:39:07.380: >>>>CCAPI handed cid 30 with tag 5336          to app "DEFAULT"
Mar 2 19:39:07.380: sess_appl: ev(24=CC_EV_CALL_SETUP_IND), cid(30), disp(0)
Mar 2 19:39:07.380: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(30), disp(0)
Mar 2 19:39:07.380: ssaCallSetupInd
Mar 2 19:39:07.380: ccCallSetContext (callID=0x1E, context=0x6215B5A0)
Mar 2 19:39:07.380: ssaCallSetupInd cid(30), st(SSA_CS_MAPPING),oldst(0),
ev(24)ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
Mar 2 19:39:07.380: ssaCallSetupInd finalDest cllng(1#5336), clled(3653)
Mar 2 19:39:07.380: ssaCallSetupInd cid(30), st(SSA_CS_CALL_SETTING),oldst(0),
ev(24)dpMatchPeersMoreArg result= 0
Mar 2 19:39:07.380: ssaSetupPeer cid(30) peer list: tag(3653) called number (3653)
Mar 2 19:39:07.380: ssaSetupPeer cid(30), destPat(3653), matched(4), prefix(),
peer(62664554), peer->encapType (2)
Mar 2 19:39:07.380: ccCallProceeding (callID=0x1E, prog_ind=0x0)
Mar 2 19:39:07.380: ccCallSetupRequest (Inbound call = 0x1E, outbound peer =3653,
dest=, params=0x62327730 mode=0, *callID=0x62327A98, prog_ind = 3)
Mar 2 19:39:07.380: ccCallSetupRequest numbering_type 0x81
Mar 2 19:39:07.380: ccCallSetupRequest encapType 2 clid_restrict_disable 1
null_orig_clg 1 clid_transparent 0 callingNumber 1#5336
Mar 2 19:39:07.380: dest pattern 3653, called 3653, digit_strip 0
Mar 2 19:39:07.380: callingNumber=1#5336, calledNumber=3653, redirectNumber=
display_info= calling_oct3a=0
Mar 2 19:39:07.384: accountNumber=, finalDestFlag=1,
guid=6aef.3a87.165c.11cc.8040.d661.b74f.9390
Mar 2 19:39:07.384: peer_tag=3653
Mar 2 19:39:07.384: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653,called_oct3=0x81, calling=1#5336,calling_oct3=0x81,
calling_xlated=false, subscriber_type_str=RegularLine, fdest=1,
voice_peer_tag=3653},mode=0x0) vdbPtr type = 1
Mar 2 19:39:07.384: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653, called_oct3 0x81, calling=1#5336,calling_oct3
0x81, calling_xlated=false, fdest=1, voice_peer_tag=3653}, mode=0x0, xltrc=-5)
Mar 2 19:39:07.384: ccSaveDialpeerTag (callID=0x1E, dialpeer_tag=0xE45)
```

```
Mar 2 19:39:07.384: ccCallSetContext (callID=0x1F, context=0x621545DC)
Mar 2 19:39:07.384: ccCallReportDigits (callID=0x1E, enable=0x0)
Mar 2 19:39:07.384: cc_api_call_report_digits_done (vdbPtr=0x6264AB2C,
callID=0x1E, disp=0)
Mar 2 19:39:07.384: sess_appl: ev(52=CC_EV_CALL_REPORT_DIGITS_DONE), cid(30),disp(0)
Mar 2 19:39:07.384: cid(30)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_REPORT_DIGITS_DONE)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(1)
Mar 2 19:39:07.384: -cid2(31)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING)
Mar 2 19:39:07.384: ssaReportDigitsDone cid(30) peer list: (empty)
Mar 2 19:39:07.384: ssaReportDigitsDone callid=30 Reporting disabled.
Mar 2 19:39:07.388: H225 NONSTD OUTGOING PDU ::=
value ARQnonStandardInfo ::=
  {
   sourceAlias
   {
   }
   sourceExtAlias
   {
   }
   interfaceSpecificBillingId "ISDN-VOICE"
  }
Mar 2 19:39:07.388: H225 NONSTD OUTGOING ENCODE BUFFER::= 80 00000820
0B124953 444E2D56 4F494345
Mar 2 19:39:07.388:
Mar 2 19:39:07.388: H235 OUTGOING ENCODE BUFFER::= 61 000100C0 2B93B7DA
08003200 32003200 3200001E 00670077 0061002D 00310040 00630069 00730063
006F002E 0063006F 006D0000
Mar 2 19:39:07.392:
Mar 2 19:39:07.392: RAS OUTGOING PDU ::=
value RasMessage ::= admissionRequest :
!--- The ARQ is sent to the gatekeeper. {
   requestSeqNum 549
   callType pointToPoint : NULL
   callModel direct : NULL
   endpointIdentifier {"8155346000000001"}
   destinationInfo
   {
    e164 : "2#3653"
   }
   srcInfo
   {
    e164 : "1#5336",
    h323-ID : {"gwa-1@cisco.com"}
   }
   bandWidth 640
   callReferenceValue 15
   nonStandardData
   {
    nonStandardIdentifier h221NonStandard :
    {
     t35CountryCode 181
     t35Extension 0
     manufacturerCode 18
    }
    data '80000008200B124953444E2D564F494345'H
   }
   conferenceID '6AEF3A87165C11CC8040D661B74F9390'H
   activeMC FALSE
   answerCall FALSE
   canMapAlias TRUE
   callIdentifier
   {
    guid '6AEF3A87165C11CC8041D661B74F9390'H
   }
```

```
   tokens
!--- Token is included since there is an all level of security.

   {

    {
     tokenOID { 1 2 840 113548 10 1 2 1 }
     timeStamp 731101147
     challenge '1CADDBA948A8291C1F134035C9613E3E'H
     random 246
     generalID {"gwa-1@cisco.com"}
    }
   }
   cryptoTokens
   {
    cryptoEPPwdHash :
    {
     alias h323-ID : {"gwa-1@cisco.com"}
     timeStamp 731101147
     token
     {
      algorithmOID { 1 2 840 113549 2 5 }
      paramS
      {
      }
      hash "5760B7B4877335B7CD24BD24E4A2AA89"
     }
    }
   }
   willSupplyUUIEs FALSE
  }

Mar 2 19:39:07.408: RAS OUTGOING ENCODE BUFFER::= 27 88022400 F0003800
31003500 35003300 34003600 30003000 30003000 30003000 30003000 31010280
50698602 02804086 69400E00 67007700 61002D00 31004000 63006900 73006300
6F002E00 63006F00 6D400280 000F40B5 00001211 80000008 200B1249 53444E2D
564F4943 456AEF3A 87165C11 CC8040D6 61B74F93 9004E320 01801100 6AEF3A87
165C11CC 8041D661 B74F9390 48014D00 0A2A8648 86F70C0A 010201C0 2B93B7DA
101CADDB A948A829 1C1F1340 35C9613E 3E0200F6 1E006700 77006100 2D003100
40006300 69007300 63006F00 2E006300 6F006D00 00420104 0E006700 77006100
2D003100 40006300 69007300 63006F00 2E006300 6F006DC0 2B93B7DA 082A8648
86F70D02 05008080 5760B7B4 877335B7 CD24BD24 E4A2AA89 0100
Mar 2 19:39:07.412: h323chan_dgram_send:Sent UDP msg. Bytes sent: 291 to
172.16.13.35:1719

Mar 2 19:39:07.416: RASLib::GW_RASSendARQ: ARQ (seq# 549) sent to 172.16.13.35
Mar 2 19:39:07.432: h323chan_dgram_recvdata:rcvd from [172.16.13.35:1719] on sock[1]

Mar 2 19:39:07.432: RAS INCOMING ENCODE BUFFER::= 2B 00022440 028000AC
100D1706 B800EF1A 00C00100 020000
Mar 2 19:39:07.432:
Mar 2 19:39:07.432: RAS INCOMING PDU ::=
value RasMessage ::= admissionConfirm :
!--- Received from the gatekeeper with no tokens. {
   requestSeqNum 549
   bandWidth 640
   callModel direct : NULL
   destCallSignalAddress ipAddress :
   {
    ip 'AC100D17'H
    port 1720
   }
   irrFrequency 240
   willRespondToIRR FALSE
```

```
   uuiesRequested
    {
     setup FALSE
     callProceeding FALSE
     connect FALSE
     alerting FALSE
     information FALSE
     releaseComplete FALSE
     facility FALSE
     progress FALSE
     empty FALSE
    }
   }

Mar 2 19:39:07.436: ACF (seq# 549) rcvd
```

此示例显示从**终端网关(TGW)的**调试。注意TGW设置第二个段，自从获得了ACF，并且呼叫连接。

```
Mar 2 19:39:07.493: PDU DATA = 6147C2BC


value H323_UserInformation ::=
  {
   h323-uu-pdu
   {
    h323-message-body setup :
    {
     protocolIdentifier { 0 0 8 2250 0 2 }
     sourceAddress
     {
      h323-ID : {"gwa-1@cisco.com"}
!--- Setup is sent from gwa-1@cisco.com gateway. } sourceInfo { gateway { protocol { voice : {
supportedPrefixes { { prefix e164 : "1#" } } } } } mc FALSE undefinedNode FALSE } activeMC FALSE
conferenceID '6AEF3A87165C11CC8040D661B74F9390'H conferenceGoal create : NULL callType
pointToPoint : NULL sourceCallSignalAddress ipAddress : { ip 'AC100D0F'H port 11032 }
callIdentifier { guid '6AEF3A87165C11CC8041D661B74F9390'H } tokens
!--- Setup includes the Clear Token (CAT). {
      {
       tokenOID { 1 2 840 113548 10 1 2 1 }
       timeStamp 731101147
       challenge 'AFBAAFDF79446B9D8CE164DB8C111A87'H
       random 247
       generalID {"gwa-1@cisco.com"}
       nonStandard
       {
        nonStandardIdentifier { 0 1 2 4 }
        data '2B93B7DBAFBAAFDF79446B9D8CE164DB8C111A87...'H
       }
      }
     }
     fastStart
     {
      '0000000C6013800A04000100AC100D0F4673'H,
      '400000060401004C6013801114000100AC100D0F...'H
     }
     mediaWaitForConnect FALSE
     canOverlapSend FALSE
    }
    h245Tunneling TRUE
    nonStandardControl
    {

     {
```

```
      nonStandardIdentifier h221NonStandard :
      {
       t35CountryCode 181
       t35Extension 0
       manufacturerCode 18
      }
      data 'E0010200010415040390 90A31803A983811E0285...'H
     }
    }
   }
  }

RAW_BUFFER::=
E0 01020001 04150403 9090A318 03A98381 1E028583 70058133 36353302 80060004
00000003
Mar 2 19:39:07.509: PDU DATA = 6147F378
value H323_UU_NonStdInfo ::=
  {
   version 2
   protoParam qsigNonStdInfo :
   {
    iei 4
    rawMesg '04039090A31803A983811E028583700581333635...'H
   }
   progIndParam progIndIEinfo :
   {
    progIndIE '00000003'H
   }
  }


PDU DATA = 6147F378


value ARQnonStandardInfo ::=
  {
   sourceAlias
   {
   }
   sourceExtAlias
   {
   }
  }


RAW_BUFFER::=
00 0000
Mar 2 19:39:07.517: RAW_BUFFER::=
61 000100C0 2B93B7DA 08003200 32003200 3200000A 00670077 0061002D
00320000
Mar 2 19:39:07.517: PDU DATA = 6147C2BC
value RasMessage ::= admissionRequest :
!--- An answer ARQ is sent to the gatekeeper to authenticate the caller. {
   requestSeqNum 22
   callType pointToPoint : NULL
   callModel direct : NULL
   endpointIdentifier {"81F5989C00000002"}
   destinationInfo
   {
    e164 : "2#3653"
   }
   srcInfo
   {
    e164 : "1#5336"
```

```
    }
    srcCallSignalAddress ipAddress :
    {
     ip 'AC100D0F'H
     port 11032
    }
    bandWidth 640
    callReferenceValue 2
    nonStandardData
    {
     nonStandardIdentifier h221NonStandard :
     {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
     }
     data '000000'H
    }
    conferenceID '6AEF3A87165C11CC8040D661B74F9390'H
    activeMC FALSE
    answerCall TRUE
    canMapAlias FALSE
    callIdentifier
    {
     guid '6AEF3A87165C11CC8041D661B74F9390'H
    }
    tokens
!--- CAT is included. {

     {
      tokenOID { 0 4 0 1321 1 2 }
      timeStamp 731101147
      challenge 'AFBAAFDF79446B9D8CE164DB8C111A87'H
      random 247
      generalID {"gwa-1@cisco.com"}
     }
    }
    cryptoTokens
    {
     cryptoEPPwdHash :
     {
      alias h323-ID : {"gwa-2"}
      timeStamp 731101147
      token
      {
       algorithmOID { 1 2 840 113549 2 5 }
       paramS
       {
       }
       hash "8479E7DE63AC17C6A46E9E19659568"
      }
     }
    }
    willSupplyUUIEs FALSE
   }
RAW_BUFFER::=
27 98001500 F0003800 31004600 35003900 38003900 43003000 30003000 30003000
30003000 32010280 50698601 02804086 6900AC10 0D0F2B18 40028000 0240B500
00120300 00006AEF 3A87165C 11CC8040 D661B74F 939044E3 20010011 006AEF3A
87165C11 CC8041D6 61B74F93 9044014D 00060400 8A290102 C02B93B7 DA10AFBA
AFDF7944 6B9D8CE1 64DB8C11 1A870200 F71E0067 00770061 002D0031 00400063
00690073 0063006F 002E0063 006F006D 00002E01 04040067 00770061 002D0032
C02B93B7 DA082A86 4886F70D 02050080 808479E7 0DE63AC1 7C6A46E9 E1965905
680100
```

```
Mar 2 19:39:07.533: h323chan_dgram_send:Sent UDP msg. Bytes sent: 228
        to 172.16.13.35:1719


Mar 2 19:39:07.533: RASLib::GW_RASSendARQ: ARQ (seq# 22) sent to 172.16.13.35
Mar 2 19:39:07.549: h323chan_dgram_recvdata:rcvd from [172.16.13.35:1719]
on sock[1]
RAW_BUFFER::=
2B 00001540 028000AC 100D1706 B800EF1A 00C00100 020000
Mar 2 19:39:07.549: PDU DATA = 6147C2BC
value RasMessage ::= admissionConfirm :
!--- ACF is received from the gatekeeper. {
   requestSeqNum 22
   bandWidth 640
   callModel direct : NULL
   destCallSignalAddress ipAddress :
   {
    ip 'AC100D17'H
    port 1720
   }
   irrFrequency 240
   willRespondToIRR FALSE
   uuiesRequested
   {
    setup FALSE
    callProceeding FALSE
    connect FALSE
    alerting FALSE
    information FALSE
    releaseComplete FALSE
    facility FALSE
    progress FALSE
    empty FALSE
   }
  }

Mar 2 19:39:07.553: ACF (seq# 22) rcvd
Mar 2 19:39:07.553: cc_api_call_setup_ind (vdbPtr=0x61BC92EC,
callInfo={called=2#3653,called_oct3=0x81,calling=1#5336,calling_oct3=0x81,
calling_oct3a=0x0,subscriber_type_str=Unknown, fdest=1 peer_tag=5336,
prog_ind=3},callID=0x6217CC64)
Mar 2 19:39:07.553: cc_api_call_setup_ind type 0 , prot 1
Mar 2 19:39:07.553: cc_api_call_setup_ind (vdbPtr=0x61BC92EC,
callInfo={called=2#3653, calling=1#5336, fdest=1 peer_tag=5336},
callID=0x6217CC64)
Mar 2 19:39:07.553: cc_process_call_setup_ind (event=0x61E1EAFC)
Mar 2 19:39:07.553: >>>>CCAPI handed cid 9 with tag 5336 to app "DEFAULT"
Mar 2 19:39:07.553: sess_appl: ev(25=CC_EV_CALL_SETUP_IND), cid(9), disp(0)
Mar 2 19:39:07.553: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(9), disp(0)
Mar 2 19:39:07.553: ssaCallSetupInd
Mar 2 19:39:07.553: ccCallSetContext (callID=0x9, context=0x62447A28)
Mar 2 19:39:07.553: ssaCallSetupInd cid(9), st(SSA_CS_MAPPING),oldst(0),
ev(25)ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
Mar 2 19:39:07.553: ssaCallSetupInd finalDest cllng(1#5336), clled(2#3653)
Mar 2 19:39:07.553: ssaCallSetupInd cid(9), st(SSA_CS_CALL_SETTING),oldst(0),
ev(25)dpMatchPeersMoreArg result= 0
Mar 2 19:39:07.557: ssaSetupPeer cid(9) peer list: tag(3653)
called number (2#3653)
Mar 2 19:39:07.557: ssaSetupPeer cid(9), destPat(2#3653), matched(5),
prefix(21), peer(620F1EF0), peer->encapType (1)
Mar 2 19:39:07.557: ccCallProceeding (callID=0x9, prog_ind=0x0)
Mar 2 19:39:07.557: ccCallSetupRequest (Inbound call = 0x9, outbound peer
=3653, dest=, params=0x61E296C0 mode=0, *callID=0x61E299D0, prog_ind = 3)
Mar 2 19:39:07.557: ccCallSetupRequest numbering_type 0x81
```

```
Mar 2 19:39:07.557: dest pattern 2#3653, called 2#3653, digit_strip 1
Mar 2 19:39:07.557: callingNumber=1#5336, calledNumber=2#3653,
redirectNumber=display_info= calling_oct3a=0
Mar 2 19:39:07.557: accountNumber=, finalDestFlag=1,
guid=6aef.3a87.165c.11cc.8040.d661.b74f.9390
Mar 2 19:39:07.557: peer_tag=3653
Mar 2 19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C, dest=,
callParams={called=2#3653,called_oct3=0x81, calling=1#5336,calling_oct3=0x81,
subscriber_type_str=Unknown, fdest=1, voice_peer_tag=3653},mode=0x0) vdbPtr
type = 6
Mar 2 19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C, dest=,
callParams={called=2#3653, called_oct3 0x81, calling=1#5336,calling_oct3 0x81,
fdest=1, voice_peer_tag=3653}, mode=0x0, xltrc=-4)
Mar 2 19:39:07.557: ccSaveDialpeerTag (callID=0x9, dialpeer_tag=
Mar 2 19:39:07.557: ccCallSetContext (callID=0xA, context=0x6244D9EC)
Mar 2 19:39:07.557: ccCallReportDigits (callID=0x9, enable=0x0)
```

网关 IOS 问题

在同一个实验室， IOS镜像12.2(6a)在OGW装载。当呼叫被做时，被注意OGW仍然发送根据其密码的清楚的令牌，即使网关没有配置为了IVR能收集Account/PIN。另外，为所有级别配置的网守接受该呼叫。这在Cisco Bug ID CSCdw43224 (仅限注册用户)描述。

带备选终点的安全性

如被提及前在本文，端到端电话安全带有在RAS/H.225消息的clearTokens字段发送的使用访问令牌。当启用这样安全时，来源网关转发从在ACF的网守接收的访问令牌对在H.225设置信息的目的地H.323终端。此目的地H.323终端然后转发在设置信息接收的访问令牌对在其准入请求的网守。通过该执行，它给远程关守能力承认根据访问令牌的正确性的呼叫。访问令牌的内容是至生成它的实体。为了最小化安全漏洞和防护装置防御中间人攻击，网守能编码在访问令牌的目的地特定信息。这意味着，当alternateEndpoints在ACF时提供，网守能为指定的每alternateEndpoint提供一个分开的访问令牌。

当它第一尝试建立连接时，在ACF clearToken字段与地址的接收在destCallSignalAddress字段的Cisco网关发送访问令牌。如果此尝试不成功，并且Cisco网关继续进行对尝试连接备选终点，使用相关的访问令牌(如果是可用的)从alternateEndpoints列表。如果在ACF接收的alternateEndpoints列表不包括访问令牌，但是ACF包括访问令牌， Cisco网关在所有尝试包括此访问令牌连接备选终点。

OSP 令牌支持

目前Cisco网关只支持开放结算协议(OSP)和其令牌。没有在网守的支持。网关认可从结算服务器接收的OSP令牌并且插入他们到Q.931设置信息到终端网关。

各端点或区域的不同安全级别

目前您无法配置每个终端或区域的不同的安全级别。安全等级是为该网守管理的所有区域。功能请求可以为这样问题打开。

域间网守到网守安全

网守安全的域间网守提供能力验证根据一个每跳跃基本类型的域内部和领域间关守对关守请求。这意味着目的地关守终止CAT并且生成新的，如果网守决定向前转发LRQ。如果网守检测一个无效LRQ签名通过发送位置拒绝(LRJ)响应。

实现网守对网守安全

始发关守生成IZCT，当LRQ启动时或ACF将在一区域内部呼叫的情况下发送。此标记通过其路由路径被横断。沿路径，每网守如果需要，更新目的地关守ID和来源关守ID，反映区域信息。终止的网守生成一标记用其密码。此标记是运载的上一步在位置确认(LCF)消息和通过对OGW。OGW在H.225设置信息包括此标记。当TGW收到标记时，在ARQ answerCall转发并且由终止的网守(TGK)验证，不用任何需要对于RADIUS服务器。

认证类型根据与散列的密码正如ITU H.235所描述。特别地，加密方法是与密码散列的MD5。

IZCT的目的将知道LRQ是否从异地到达，从区域，并且从哪载波。它也用于通过标记到在LCF的OGW从TGK。在IZCT格式内，此信息要求：

- srcCarrierID —源载波ID
- dstCarrierID —目的地载波ID
- intCarrierID —媒介载波ID
- srcZone —源区域
- dstZone —目的地区域
- 区域间类型
  INTRA_DOMAIN_CISCOINTER_DOMAIN_CISCOINTRA_DOMAIN_TERM_NOT_CISCOINTER_DOMAIN_ORIG_NOT_CISCO

此功能优良运作，不用任何需要对于从网关或载波敏感路由(CSR)服务器的载波ID。在这样案件中，关于载波ID的字段是空的。此处示例不包括任何载波ID。对于详细呼叫流，版本和平台支持和配置，参考内部域网守安全增强功能。

网守配置

IZCT功能要求在网守的此配置。

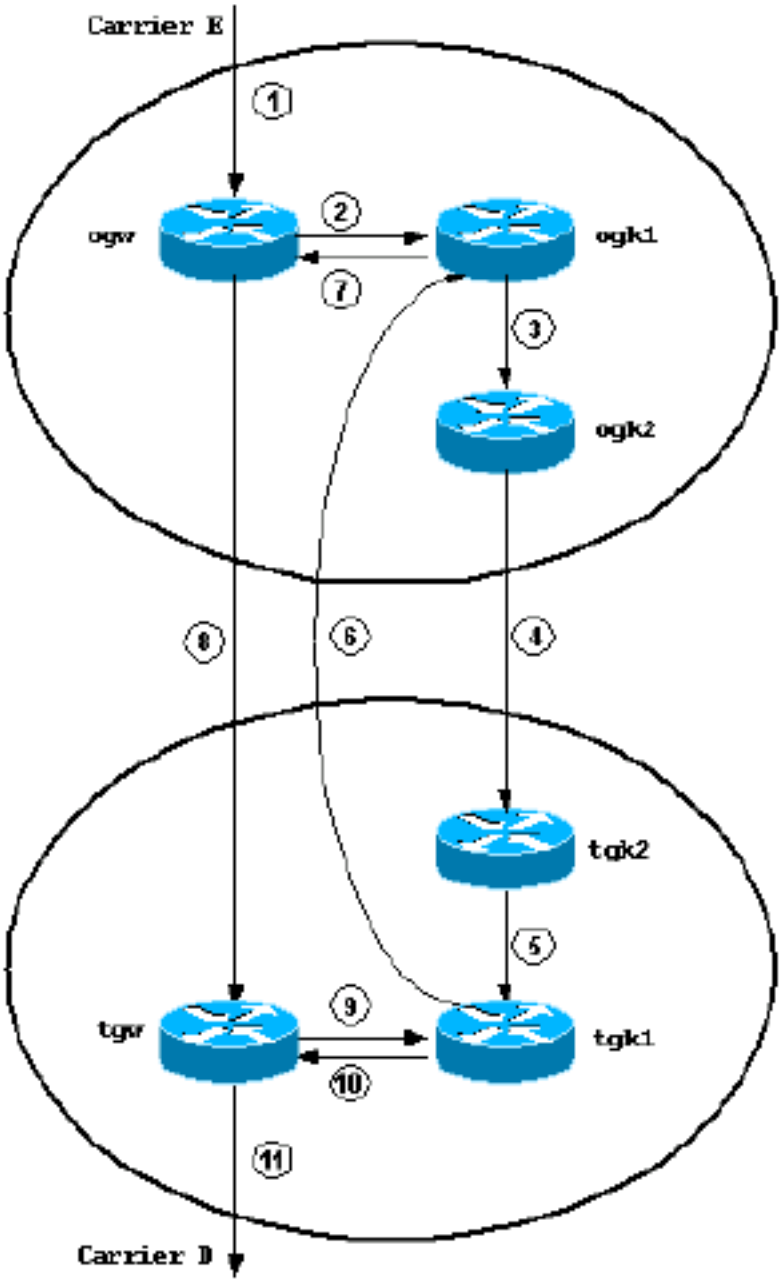
```
Router(gk-config)#
[no] security izct password <PASSWORD>
```

密码需要是六个到八个字符。识别哪个区域在象这样的一个异地：

```
Router(config-gk)#
zone remote other-gatekeeper-name other-domain-name other-gatekeeper-ip-address [port-number]
[cost cost-value [priority priority-value]] [foreign-domain]
```

IZCT 呼叫流

此图表显示IZCT流。



在此配置中，网关的名称和网守是相同的象那些在IZCT呼叫流程图中，但是与小写一起使用。呼叫流在配置以后解释，与调试说明。

要解释IZCT功能和呼叫流，第一示例没有域内网关到网守安全。在那以后，有TGW不能生成IZCT的示例，以便TGK1拒绝呼叫。这是为了显示功能运作如设计。所有这些设置根据拓扑在IZCT呼叫流程图中。

示例 1：网守的呼叫流对仅网守安全

此示例显示所有网关和网守相关的配置。

OGW配置	TGW配置
Router(config-gk)# **zone remote** *other-gatekeeper-name other-domain-name other-gatekeeper-ip-address*	Router(config-gk)# **zone remote** *other-gatekeeper-name other-domain-name other-gatekeeper-ip-address*

[*port-number*] [**cost** *cost-value* [**priority** *priority-value*]] [**foreign-domain**]	[*port-number*] [**cost** *cost-value* [**priority** *priority-value*]] [**foreign-domain**]
OGK1配置	**TGK1配置**
Router(config-gk)# **zone remote** *other-gatekeeper-name other-domain-name other-gatekeeper-ip-address* [*port-number*] [**cost** *cost-value* [**priority** *priority-value*]] [**foreign-domain**]	Router(config-gk)# **zone remote** *other-gatekeeper-name other-domain-name other-gatekeeper-ip-address* [*port-number*] [**cost** *cost-value* [**priority** *priority-value*]] [**foreign-domain**]
OGK2配置	**TGK2配置**
Router(config-gk)# **zone remote** *other-gatekeeper-name other-domain-name other-gatekeeper-ip-address* [*port-number*] [**cost** *cost-value* [**priority** *priority-value*]] [**foreign-domain**]	Router(config-gk)# **zone remote** *other-gatekeeper-name other-domain-name other-gatekeeper-ip-address* [*port-number*] [**cost** *cost-value* [**priority** *priority-value*]] [**foreign-domain**]

带调试的呼叫流

这些示例使用调试解释呼叫流。

1. 载波的E一个用户告诉载波的D.一个用户。

```
Mar 4 15:31:19.989: cc_api_call_setup_ind
 (vdbPtr=0x6264ADF0, callInfo={called=3653,
called_oct3=0x80,calling=4085272923,calling_oct3=0x21,calling_oct3a=0x80
calling_xlated=false,subscriber_type_str=RegularLine,fdest=1,peer_tag=5336,
prog_ind=0},callID=0x6219F9F0)
Mar 4 15:31:19.993: cc_api_call_setup_ind type 13 , prot 0
Mar 4 15:31:19.993: cc_process_call_setup_ind (event=0x6231A6B4)
Mar 4 15:31:19.993: >>>>CCAPI handed cid 7 with tag 5336 to app "DEFAULT"
Mar 4 15:31:19.993: sess_appl: ev(24=CC_EV_CALL_SETUP_IND), cid(7), disp(0)
Mar 4 15:31:19.993: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(7), disp(0)
Mar 4 15:31:19.993: ssaCallSetupInd
Mar 4 15:31:19.993: ccCallSetContext (callID=0x7, context=0x621533F0)
Mar 4 15:31:19.997: ssaCallSetupInd cid(7), st(SSA_CS_MAPPING),oldst(0),
ev(24) ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
Mar 4 15:31:19.997: ssaCallSetupInd finalDest cllng(4085272923), clled(3653)
Mar 4 15:31:19.997: ssaCallSetupInd cid(7), st(SSA_CS_CALL_SETTING),oldst(0),
ev(24)dpMatchPeersMoreArg result= 0
Mar 4 15:31:19.997: ssaSetupPeer cid(7) peer list: tag(3653) called number (3653)
Mar 4 15:31:19.997: ssaSetupPeer cid(7), destPat(3653), matched(4), prefix(),
peer(626640B0), peer->encapType (2)
Mar 4 15:31:19.997: ccCallProceeding (callID=0x7, prog_ind=0x0)
Mar 4 15:31:19.997: ccCallSetupRequest (Inbound call = 0x7, outbound peer=3653,
dest=,
   params=0x62327730 mode=0, *callID=0x62327A98, prog_ind = 0)
Mar 4 15:31:19.997: ccCallSetupRequest numbering_type 0x80
Mar 4 15:31:19.997: ccCallSetupRequest encapType 2 clid_restrict_disable 1 null
_orig_clg 0 clid_transparent 0 callingNumber 4085272923
Mar 4 15:31:19.997: dest pattern 3653, called 3653, digit_strip 0
Mar 4 15:31:19.997: callingNumber=4085272923, calledNumber=3653, redirectNumber
= display_info= calling_oct3a=80
```

```
Mar 4 15:31:19.997: accountNumber=, finalDestFlag=1,
guid=221b.686c.17cc.11cc.8010.a049.e052.4766
Mar 4 15:31:19.997: peer_tag=3653
Mar 4 15:31:19.997: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653,called_oct3=0x80, calling=4085272923,calling_oct3=0x21,
calling_xlated=false, subscriber_type_str=RegularLine, fdest=1, voice_peer_tag=365
3},mode=0x0) vdbPtr type = 1
Mar 4 15:31:19.997: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653, called_oct3 0x80, calling=4085272923,calling_oct3 0x21,
calling_xlated=false, fdest=1, voice_peer_tag=3653}, mode=0x0, xltrc=-5)
Mar 4 15:31:20.001: ccSaveDialpeerTag (callID=0x7, dialpeer_tag=0xE45)
Mar 4 15:31:20.001: ccCallSetContext (callID=0x8, context=0x6215388C)
Mar 4 15:31:20.001: ccCallReportDigits (callID=0x7, enable=0x0)
```

2. 因为始发网关的dialpeer (tag=3653)为RAS配置，发送ARQ对OGK1。

```
Mar 4 15:31:20.001: H225 NONSTD OUTGOING PDU ::=


value ARQnonStandardInfo ::=
  {
   sourceAlias
   {
   }
   sourceExtAlias
   {
   }
   callingOctet3a 128
   interfaceSpecificBillingId "ISDN-VOICE"
  }

Mar 4 15:31:20.005: H225 NONSTD OUTGOING ENCODE BUFFER::= 80 000008A0
01800B12 4953444E 2D564F49 4345
Mar 4 15:31:20.005:
Mar 4 15:31:20.005: RAS OUTGOING PDU ::=

value RasMessage ::= admissionRequest :
!--- ARQ is sent out to ogk1. {
  requestSeqNum 1109
  callType pointToPoint : NULL
  callModel direct : NULL
  endpointIdentifier {"81567A4000000001"}
  destinationInfo
  {
   e164 : "3653"
  }
  srcInfo
  {
   e164 : "4085272923",
   h323-ID : {"ogw"}
  }
  bandWidth 640
  callReferenceValue 4
  nonStandardData
  {
   nonStandardIdentifier h221NonStandard :
   {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
   }
   data '80000008A001800B124953444E2D564F494345'H
  }
  conferenceID '221B686C17CC11CC8010A049E0524766'H
  activeMC FALSE
```

```
  answerCall FALSE
  canMapAlias TRUE
  callIdentifier
  {
   guid '221B686C17CC11CC8011A049E0524766'H
  }
  willSupplyUUIEs FALSE
 }

Mar 4 15:31:20.013: RAS OUTGOING ENCODE BUFFER::= 27 88045400 F0003800
31003500 36003700 41003400 30003000 30003000 30003000 30003000 31010180
69860204 8073B85A 5C564002 006F0067 00774002 80000440 B5000012 13800000
08A00180 0B124953 444E2D56 4F494345 221B686C 17CC11CC 8010A049 E0524766
04E02001 80110022 1B686C17 CC11CC80 11A049E0 52476601 00
Mar 4 15:31:20.017: h323chan_dgram_send:Sent UDP msg. Bytes sent: 130 to
172.16.13.35:1719

Mar 4 15:31:20.017: RASLib::GW_RASSendARQ: ARQ (seq# 1109) sent to
172.16.13.35
```

3. 当OGK1接收ARQ时，确定目的地由远程区域OGK2服务。它然后识别IZCT必要(通过CLI ：**安全izct密码<pwd>**)。OGK1创建IZCT的收益在LRQ前发送。它然后发送IZCT和LRQ对OGK2并且送回RIP消息到OGW。

```
Mar  4 15:31:19.927: H225 NONSTD OUTGOING PDU ::=
value LRQnonStandardInfo ::=
  {
   ttl 6
   nonstd-callIdentifier
   {
    guid '221B686C17CC11CC8011A049E0524766'H
   }
   callingOctet3a 128
   gatewaySrcInfo
   {
    e164 : "4085272923",
    h323-ID : {"ogw"}
   }
  }

Mar 4 15:31:19.935: H225 NONSTD OUTGOING ENCODE BUFFER::= 82 86B01100
221B686C 17CC11CC 8011A049 E0524766 01801002 048073B8 5A5C5640
02006F00 670077
Mar 4 15:31:19.939:
Mar 4 15:31:19.939: PDU ::=

value IZCToken ::=
!--- The gatekeeper creates and sends out the IZCT. {
   iztInterZoneType intraDomainCisco : NULL
!--- The destination is in the same domain, it is intraDomainCisco type. iztSrcZone "ogk1"
 !--- The source zone is ogk1. )


Mar 4 15:31:19.943: ENCODE BUFFER::= 07 00C06F67 6B310473 72630464
73740469 6E74
Mar 4 15:31:19.947:
Mar 4 15:31:19.947: RAS OUTGOING PDU ::=

value RasMessage ::= locationRequest :
!--- LRQ is sent out to ogk2. { requestSeqNum 2048
   destinationInfo
   {
    e164 : "3653"
```

```
   }
   nonStandardData
   {
    nonStandardIdentifier h221NonStandard :
    {
     t35CountryCode 181
     t35Extension 0
     manufacturerCode 18
    }
    data '8286B01100221B686C17CC11CC8011A049E05247...'H
   }
   replyAddress ipAddress :
   {
    ip 'AC100D23'H
    port 1719
   }
   sourceInfo
   {
    h323-ID : {"ogk1"}
   }
   canMapAlias TRUE
   tokens
!--- The IZCT is included. {

    {
     tokenOID { 1 2 840 113548 10 1 0 }
     nonStandard
     {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '0700C06F676B3104737263046473740469 6E74'H
     }
    }
   }
  }

Mar 4 15:31:19.967: RAS OUTGOING ENCODE BUFFER::= 4A 8007FF01 01806986
40B50000
 12288286 B0110022 1B686C17 CC11CC80 11A049E0 52476601 80100204 8073B85A
5C56400
2 006F0067 007700AC 100D2306 B70BA00B 01400300 6F006700 6B003101
802B0100 80092A
86 4886F70C 0A010009 2A864886 F70C0A01 00130700 C06F676B 31047372
63046473 74046
96E 74
Mar 4 15:31:19.983:
Mar 4 15:31:19.987: IPSOCK_RAS_sendto: msg length 122 from 172.16.13.35:1719
 to 172.16.13.14: 1719
Mar 4 15:31:19.987: RASLib::RASSendLRQ: LRQ (seq# 2048) sent to 172.16.13.14
Mar 4 15:31:19.987: RAS OUTGOING PDU ::=

value RasMessage ::= requestInProgress :
!--- RIP message is sent back to OGW. {
   requestSeqNum 1109
   delay 9000
  }

Mar 4 15:31:19.991: RAS OUTGOING ENCODE BUFFER::= 80 05000454 2327
Mar 4 15:31:19.991:
Mar 4 15:31:19.991: IPSOCK_RAS_sendto: msg length 7 from 172.16.13.35:1719 to
172.16.13.15: 57076
Mar 4 15:31:19.991: RASLib::RASSendRIP: RIP (seq# 1109) sent to 172.16.13.15
```

4. 当OGK2接收LRQ时，检查IZCT。从配置它发现LRQ neets也包含IZCT。OGK2通过更改izctSrcZone和izctDstZone是然后创建一新的IZCT ogk2并且转发LRQ对TGK2。在它派出

LRQ对TGK2后，退还RIP消息对OGK1。如果网守是集群的一部分，集群名称使用SrcZone或DstZone。

```
Mar  4 15:31:20.051: RAS OUTGOING PDU ::=


value RasMessage ::= requestInProgress :
!--- RIP message is sent back to OGK1. { requestSeqNum 2048
   delay 6000
  }

Mar 4 15:31:20.055: RAS OUTGOING ENCODE BUFFER::= 80 050007FF 176F
Mar 4 15:31:20.055:
Mar 4 15:31:20.055: IPSOCK_RAS_sendto: msg length 7 from 172.16.13.14:1719 to
172.16.13.35: 1719
Mar 4 15:31:20.059: RASLib::RASSendRIP: RIP (seq# 2048) sent to 172.16.13.35
Mar 4 15:31:20.059: H225 NONSTD OUTGOING PDU ::=

value LRQnonStandardInfo ::=
  {
   ttl 5
   nonstd-callIdentifier
   {
    guid '221B686C17CC11CC8011A049E0524766'H
   }
   callingOctet3a 128
   gatewaySrcInfo
   {
    e164 : "4085272923",
    h323-ID : {"ogw"}
   }
  }

Mar 4 15:31:20.063: H225 NONSTD OUTGOING ENCODE BUFFER::= 82 06B01100
221B686C 17CC11CC 8011A049 E0524766 01801002 048073B8 5A5C5640 02006F00 670077
Mar 4 15:31:20.072:
Mar 4 15:31:20.072: PDU ::=

value IZCToken ::=
  {
   izctInterZoneType intraDomainCisco : NULL
!--- This is still intraDomain since message OGK1 is !--- not a foreign domain.
izctSrcZone "ogk2"
!--- ScrZone and DstZone become ogk2. izctDstZone "ogk2"
     }

Mar 4 15:31:20.076: ENCODE BUFFER::= 47 00C06F67 6B32066F 676B3204 73726304
64737404 696E74
Mar 4 15:31:20.080:
Mar 4 15:31:20.080: RAS OUTGOING PDU ::=

value RasMessage ::= locationRequest :
!--- The LRQ is forwarded to TGK2. { requestSeqNum 2048 destinationInfo { e164 : "3653" }
nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension
0 manufacturerCode 18 } data '8206B01100221B686C17CC11CC8011A049E05247...'H } replyAddress
ipAddress : { ip 'AC100D23'H port 1719 } sourceInfo { h323-ID : {"ogk1"} } canMapAlias TRUE
tokens
!--- IZCT is included. {
    {
     tokenOID { 1 2 840 113548 10 1 0 }
     nonStandard
     {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '4700C06F676B32066F676B320473726304647374...'H
```

```
     }
    }
   }
  }

Mar 4 15:31:20.104: RAS OUTGOING ENCODE BUFFER::= 4A 8007FF01
01806986 40B50000 12288206 B0110022 1B686C17 CC11CC80 11A049E0 52476601
80100204 8073B85A 5C564002 006F0067 007700AC 100D2306 B70BA00B 01400300
6F006700 6B003101 80300100 80092A86 4886F70C 0A010009 2A864886 F70C0A01
00184700 C06F676B 32066F67 6B320473 72630464 73740469 6E74
Mar 4 15:31:20.120:
Mar 4 15:31:20.120: IPSOCK_RAS_sendto: msg length 127 from 172.16.13.14:1719
to 172.16.13.16: 1719
Mar 4 15:31:20.124: RASLib::RASSendLRQ: LRQ (seq# 2048) sent to 172.16.13.16
```

5. TGK2确定LRQ来自异地。它更新与其自己的ID的IZCT's dstZone和interZoneType作为INTER_DOMAIN_CISCO。它然后创建新的CAT并且通过更新IZCT和LRQ对TGK1。TGK2对待LRQ接收作为在这两个方案之一的一区域的区域：TGK2's远程区域列表不包含LRQ接收的区域。TGK2's远程区域列表包含LRQ接收的区域。区域标记用外部域标志。它然后送回请求进展中消息到OGK1。

```
Mar 4 15:31:20.286: RAS OUTGOING PDU ::=
value RasMessage ::= requestInProgress :
!--- The RIP message is sent back to !--- OGK1 since lrq-forward queries are configured on
OGK2 and TGK2. { requestSeqNum 2048 delay 6000 } Mar 4 15:31:20.286: RAS OUTGOING ENCODE
BUFFER::= 80 050007FF 176F Mar 4 15:31:20.286: Mar 4 15:31:20.286: IPSOCK_RAS_sendto: msg
length 7 from 172.16.13.16:1719 to 172.16.13.35: 1719 Mar 4 15:31:20.286:
RASLib::RASSendRIP: RIP (seq# 2048) sent to 172.16.13.35 Mar 4 15:31:20.286: H225 NONSTD
OUTGOING PDU ::= value LRQnonStandardInfo ::= { ttl 4 nonstd-callIdentifier { guid
'221B686C17CC11CC8011A049E0524766'H } callingOctet3a 128 gatewaySrcInfo { e164 :
"4085272923", h323-ID : {"ogw"} } } Mar 4 15:31:20.290: H225 NONSTD OUTGOING ENCODE
BUFFER::= 81 86B01100 221B686C 17CC11CC 8011A049 E0524766 01801002 048073B8 5A5C5640
02006F00 670077 Mar 4 15:31:20.290: Mar 4 15:31:20.290: PDU ::= value IZCToken ::=
!--- The IZCT information. {
   izctInterZoneType interDomainCisco : NULL
!--- The zone type is interDomain since the OGK2 !--- in a foreign domain is configured in
TGK2. izctSrcZone "ogk2"
!--- SrcZone is still ogk2.  izctDstZone "tgk2"
!--- DstZone changed to tgk2. }

Mar 4 15:31:20.294: ENCODE BUFFER::= 47 20C06F67 6B320674 676B3204
73726304 64737404 696E74
Mar 4 15:31:20.294:
Mar 4 15:31:20.294: RAS OUTGOING PDU ::=
value RasMessage ::= locationRequest :
!--- LRQ is sent to TGK1. {
   requestSeqNum 2048
   destinationInfo
   {
    e164 : "3653"
   }
    nonStandardData
    {
     nonStandardIdentifier h221NonStandard :
     {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
     }
     data '8186B01100221B686C17CC11CC8011A049E05247...'H
    }
    replyAddress ipAddress :
    {
```

```
     ip 'AC100D23'H
     port 1719
    }
    sourceInfo
    {
     h323-ID : {"ogk1"}
    }
    canMapAlias TRUE
    tokens
!--- The IZCT is included. {

     {
      tokenOID { 1 2 840 113548 10 1 0 }
      nonStandard
      {
       nonStandardIdentifier { 1 2 840 113548 10 1 0 }
       data '4720C06F676B320674676B320473726304647374...'H
      }
     }
    }
   }

Mar 4 15:31:20.302: RAS OUTGOING ENCODE BUFFER::= 4A 8007FF01 01806986
40B50000 12288186 B0110022 1B686C17 CC11CC80 11A049E0 52476601 80100204
8073B85A 5C564002 006F0067 007700AC 100D2306 B70BA00B 01400300 6F006700
6B003101 80300100 80092A86 4886F70C 0A010009 2A864886 F70C0A01 00184720
C06F676B 32067467 6B320473 72630464 73740469 6E74
Mar 4 15:31:20.306:
Mar 4 15:31:20.306: IPSOCK_RAS_sendto: msg length 127 from 172.16.13.16:1719
to 172.16.13.41: 1719
Mar 4 15:31:20.306: RASLib::RASSendLRQ: LRQ (seq# 2048) sent to 172.16.13.41
```

6. 通常TGK1更新IZCT's dstCarrierID到载波E，路由进程取决于。然而，因为没有使用载波，您看不到那。TGK1生成一哈希标记用IZCT's密码。它发送与更新IZCT的一LCF在它对OGK1。此iztHash用于验证TGK1从TGW接收的answerCall ARQ，当以后收到从OGW时的VoIP设置信息。

```
Mar  4 15:31:20.351: PDU ::=
value IZCToken ::=
!--- IZCT with a hash is generated to be sent back to TGK2. {
   izctInterZoneType interDomainCisco : NULL
   izctSrcZone "ogk2"
   izctDstZone "tgk2"
   izctTimestamp 731259080
   izctRandom 3
   izctHash '5A7D5E18AA658A6A4B4709BA5ABEF2B9'H
  }

Mar 4 15:31:20.355: ENCODE BUFFER::= 7F 20C06F67 6B320674 676B32C0 2B9620C7
0103105A 7D5E18AA 658A6A4B 4709BA5A BEF2B904 73726304 64737404 696E74
Mar 4 15:31:20.355:
Mar 4 15:31:20.355: RAS OUTGOING PDU ::=
value RasMessage ::= locationConfirm :
!--- LCF is sent back to OGK1 since lrq-forward queries !--- are configured on OGK2 and
TGK2. {
   requestSeqNum 2048
   callSignalAddress ipAddress :
   {
    ip 'AC100D17'H
    port 1720
   }
   rasAddress ipAddress :
   {
    ip 'AC100D17'H
```

```
    port 55762
   }
   nonStandardData
   {
    nonStandardIdentifier h221NonStandard :
    {
     t35CountryCode 181
     t35Extension 0
     manufacturerCode 18
    }
    data '000140020074006700770600740067006B003101...'H
   }
   destinationType
   {
    gateway
    {
     protocol
     {
      voice :
      {
       supportedPrefixes
       {
       }
      }
     }
    }
    mc FALSE
    undefinedNode FALSE
   }
   tokens
!--- The IZCT is included. {
    {
     tokenOID { 1 2 840 113548 10 1 0 }
     nonStandard
     {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '7F20C06F676B320674676B32C02B9620C7010310...'H
      }
     }
    }
   }

Mar 4 15:31:20.367: RAS OUTGOING ENCODE BUFFER::= 4F 07FF00AC
100D1706 B800AC10 0D17D9D2 40B50000 122F0001 40020074 00670077 06007400
67006B00 31011001 40020074 00670077 00AC100D 1706B800 00000000 00000000
00104808 0880013C 05010000 48010080 092A8648 86F70C0A 0100092A 864886F7
0C0A0100 307F20C0 6F676B32 0674676B 32C02B96 20C70103 105A7D5E 18AA658A
6A4B4709 BA5ABEF2 B9047372 63046473 7404696E 74
Mar 4 15:31:20.371:
Mar 4 15:31:20.371: IPSOCK_RAS_sendto: msg length 154 from 172.16.13.41:1719 to
172.16.13.35: 1719
Mar 4 15:31:20.371: RASLib::RASSendLCF: LCF (seq# 2048) sent to 172.16.13.35
```

7. OGK1在ACF解压缩从LCF的IZCT并且发送它对OGW。

```
Mar 4 15:31:20.316: PDU ::=
value IZCToken ::=
!--- The extracted IZCT. {
   izctInterZoneType interDomainCisco : NULL
   izctSrcZone "ogk2"
   izctDstZone "tgk2"
   izctTimestamp 731259080
   izctRandom 3
   izctHash '5A7D5E18AA658A6A4B4709BA5ABEF2B9'H
```

```
 }

Mar 4 15:31:20.324: ENCODE BUFFER::= 7F 20C06F67 6B320674 676B32C0 2B9620C7 0103105A
7D5E18AA 658A6A4B 4709BA5A BEF2B904 73726304 64737404 696E74
Mar 4 15:31:20.328:
Mar 4 15:31:20.332: RAS OUTGOING PDU ::=
value RasMessage ::= admissionConfirm :
!--- ACF is sent back to OGW with the hashed IZCToken. {
   requestSeqNum 1109
   bandWidth 640
   callModel direct : NULL
   destCallSignalAddress ipAddress :
   {
    ip 'AC100D17'H
    port 1720
   }
    irrFrequency 240
    tokens
!--- The IZCT is included. {

     {
      tokenOID { 1 2 840 113548 10 1 0 }
      nonStandard
      {
       nonStandardIdentifier { 1 2 840 113548 10 1 0 }
       data '7F20C06F676B320674676B32C02B9620C7010310...'H
      }
     }
    }
    willRespondToIRR FALSE
    uuiesRequested
    {
     setup FALSE
     callProceeding FALSE
     connect FALSE
     alerting FALSE
     information FALSE
     releaseComplete FALSE
     facility FALSE
     progress FALSE
     empty FALSE
    }
   }

Mar 4 15:31:20.352: RAS OUTGOING ENCODE BUFFER::= 2B 00045440 028000AC 100D1706
B800EF1A 08C04801 0080092A 864886F7 0C0A0100 092A8648 86F70C0A 0100307F 20C06F67
6B320674 676B32C0 2B9620C7 0103105A 7D5E18AA 658A6A4B 4709BA5A BEF2B904 73726304
64737404 696E7401 00020000
Mar 4 15:31:20.364:
Mar 4 15:31:20.364: IPSOCK_RAS_sendto: msg length 97 from 172.16.13.35:1719 to
172.16.13.15: 57076
Mar 4 15:31:20.368: RASLib::RASSendACF: ACF (seq# 1109) sent to 172.16.13.15
```

8. OGW发送IZCT到在H.225设置信息的TGW。

```
Mar 4 15:31:20.529: H225.0 OUTGOING PDU ::=
value H323_UserInformation ::=
  {
   h323-uu-pdu
   {
    h323-message-body setup :
!--- H.225 SETUP message is sent to TGW. { protocolIdentifier { 0 0 8 2250 0 2 }
sourceAddress { h323-ID : {"ogw"} } sourceInfo { gateway { protocol { voice : {
supportedPrefixes { { prefix e164 : "1#" } } } } } } mc FALSE undefinedNode FALSE } activeMC
```

```
FALSE conferenceID '221B686C17CC11CC8010A049E0524766'H conferenceGoal create : NULL
callType pointToPoint : NULL sourceCallSignalAddress ipAddress : { ip 'AC100D0F'H port
11003 } callIdentifier { guid '221B686C17CC11CC8011A049E0524766'H } tokens
!--- The hashed IZCT information is included in the setup message. {
     {
      tokenOID { 1 2 840 113548 10 1 0 }
      nonStandard
      {
       nonStandardIdentifier { 1 2 840 113548 10 1 0 }
       data '7F20C06F676B320674676B32C02B9620C7010310...'H
      }
     }
    }
    fastStart
    {
     '0000000C6013800A04000100AC100D0F4125'H,
     '4000000604010040C6013801114000100AC100D0F...'H
    }
    mediaWaitForConnect FALSE
    canOverlapSend FALSE
   }
   h245Tunneling TRUE
   nonStandardControl
   {
    {
     nonStandardIdentifier h221NonStandard :
     {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
     }
     data '6001020001041F04038090A31803A983816C0C21...'H
    }
   }
  }
 }
```

9. TGW通过IZCT对在ARQ answerCall的TGK1。

```
Mar 4 15:31:20.613:
Mar 4 15:31:20.613: RAS OUTGOING PDU ::=
value RasMessage ::= admissionRequest :
!--- ARQ answerCall type is sent to TGK1. {
   requestSeqNum 78
   callType pointToPoint : NULL
   callModel direct : NULL
   endpointIdentifier {"617D829000000001"}
   destinationInfo
   {
    e164 : "3653"
   }
   srcInfo
   {
    e164 : "4085272923",
    h323-ID : {"ogw"}
   }
   srcCallSignalAddress ipAddress :
   {
    ip 'AC100D0F'H
    port 11003
   }
   bandWidth 1280
   callReferenceValue 3
   nonStandardData
   {
```

```
   nonStandardIdentifier h221NonStandard :
   {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
   }
   data '80000008800180'H
  }
  conferenceID '221B686C17CC11CC8010A049E0524766'H
  activeMC FALSE
  answerCall TRUE
  canMapAlias TRUE
  callIdentifier
  {
   guid '221B686C17CC11CC8011A049E0524766'H
  }
  tokens
!--- The hashed IZCToken information is included. {
   {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
     nonStandardIdentifier { 1 2 840 113548 10 1 0 }
     data '7F20C06F676B320674676B32C02B9620C7010310...'H
    }
   }
  }
  willSupplyUUIEs FALSE
 }
```

10. TGK1成功验证目的地IZCT。这是因为TGK1生成在IZCT的哈希，并且退还ACF到TGW。

```
Mar 4 15:31:20.635:
Mar 4 15:31:20.635: PDU ::=
value IZCToken ::=
!--- The extracted IZCT from the ARQ to be validated. {
   izctInterZoneType interDomainCisco : NULL
   izctSrcZone "ogk2"
   izctDstZone "tgk2"
   izctTimestamp 731259080
   izctRandom 3
   izctHash '5A7D5E18AA658A6A4B4709BA5ABEF2B9'H
  }

Mar 4 15:31:20.639: RAS OUTGOING PDU ::=
value RasMessage ::= admissionConfirm :
!--- After the IZCT is validated, ACF is sent back to TGW. {
   requestSeqNum 78
   bandWidth 1280
   callModel direct : NULL
   destCallSignalAddress ipAddress :
   {
    ip 'AC100D17'H
    port 1720
   }
   irrFrequency 240
   willRespondToIRR FALSE
   uuiesRequested
   {
    setup FALSE
    callProceeding FALSE
    connect FALSE
    alerting FALSE
    information FALSE
    releaseComplete FALSE
```

```
    facility FALSE
    progress FALSE
    empty FALSE
   }
  }
```

11. 在接收ACF后，TGW建立呼叫往载波D。**示例 2：因为TGW不能解压缩从收到的SETUP消息的IZCT呼叫失败。**此示例根据拓扑和配置和示例1一样。在本例中，TGW的软件更改对不支持IZCT的版本。在这种情况下，TGW不能解压缩从设置信息的IZCT。这造成TGK1拒绝与安全拒绝断开原因的呼叫。因为呼叫流是相同的象示例1，此示例显示设置信息、ARQ和仅ARJ在TGW。

```
Mar 4 19:50:32.346: PDU DATA = 6147C2BC
value H323_UserInformation ::=
  {
   h323-uu-pdu
   {
    h323-message-body setup :
!--- H.225 SETUP message is received with a token included. { protocolIdentifier { 0 0 8
2250 0 2 } sourceAddress { h323-ID : {"ogw"} } sourceInfo { gateway { protocol { voice : {
supportedPrefixes { { prefix e164 : "1#" } } } } } mc FALSE undefinedNode FALSE } activeMC
FALSE conferenceID '56CA67C817F011CC8014A049E0524766'H conferenceGoal create : NULL
callType pointToPoint : NULL sourceCallSignalAddress ipAddress : { ip 'AC100D0F'H port
11004 } callIdentifier { guid '56CA67C817F011CC8015A049E0524766'H } tokens
!--- Hashed IZCT is included. {
     {
      tokenOID { 1 2 840 113548 10 1 0 }
      nonStandard
      {
       nonStandardIdentifier { 1 2 840 113548 10 1 0 }
       data '7F20C06F676B320674676B32C02B965D85010410...'H
      }
     }
    }
    fastStart
    {
     '0000000C6013800A04000100AC100D0F45D9'H,
     '400000060401004C6013801114000100AC100D0F...'H
    }
    mediaWaitForConnect FALSE
    canOverlapSend FALSE
   }
   h245Tunneling TRUE
   nonStandardControl
   {
    {
     nonStandardIdentifier h221NonStandard :
     {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
     }
     data '6001020001041F04038090A31803A983816C0C21...'H
    }
   }
  }
 }

RAW_BUFFER::=
60 01020001 041F0403 8090A318 03A98381 6C0C2180 34303835 32373239 32337005
80333635 33
Mar 4 19:50:32.362: PDU DATA = 6147F378
value H323_UU_NonStdInfo ::=
  {
```

```
   version 2
   protoParam qsigNonStdInfo :
   {
    iei 4
    rawMesg '04038090A31803A983816C0C2180343038353237...'H
   }
  }


PDU DATA = 6147F378
value ARQnonStandardInfo ::=
  {
   sourceAlias
   {
   }
   sourceExtAlias
   {
   }
   callingOctet3a 128
  }


RAW_BUFFER::=
 80 00000880 0180
 Mar 4 19:50:32.366: PDU DATA = 6147C2BC
value RasMessage ::= admissionRequest :
!--- ARQ is sent out. There is no token in it. {
   requestSeqNum 23
   callType pointToPoint : NULL
   callModel direct : NULL
   endpointIdentifier {"617D829000000001"}
   destinationInfo
   {
    e164 : "3653"
   }
    srcInfo
   {
    e164 : "4085272923"
   }
   srcCallSignalAddress ipAddress :
   {
    ip 'AC100D0F'H
    port 11004
   }
   bandWidth 640
   callReferenceValue 1
   nonStandardData
   {
    nonStandardIdentifier h221NonStandard :
    {
     t35CountryCode 181
     t35Extension 0
     manufacturerCode 18
    }
    data '80000008800180'H
   }
   conferenceID '56CA67C817F011CC8014A049E0524766'H
   activeMC FALSE
   answerCall TRUE
   canMapAlias FALSE
   callIdentifier
   {
    guid '56CA67C817F011CC8015A049E0524766'H
   }
```

```
   willSupplyUUIEs FALSE
  }

RAW_BUFFER::=
27 98001600 F0003600 31003700 44003800 32003900 30003000 30003000 30003000
30003000 31010180 69860104 8073B85A 5C5600AC 100D0F2A FC400280 000140B5
00001207 80000008 80018056 CA67C817 F011CC80 14A049E0 52476644 E0200100
110056CA 67C817F0 11CC8015 A049E052 47660100
Mar 4 19:50:32.374: h323chan_dgram_send:Sent UDP msg. Bytes sent: 117 to
172.16.13.41:1719
Mar 4 19:50:32.374: RASLib::GW_RASSendARQ: ARQ (seq# 23) sent to 172.16.13.41
Mar 4 19:50:32.378: h323chan_dgram_recvdata:rcvd from [172.16.13.41:1719]
on sock[1]
RAW_BUFFER::=
2C 00168001 00
Mar 4 19:50:32.378: PDU DATA = 6147C2BC
value RasMessage ::= admissionReject :
!--- ARJ is received with a reason of security denial. {
   requestSeqNum 23
   rejectReason securityDenial : NULL
  }

Mar 4 19:50:32.378: ARJ (seq# 23) rcvd
```

Related Information

- 内部域网守安全增强功能