Guía de solución de problemas de seguridad relativos a Gateway a Gatekeeper (H.235) y Gatekeeper a Gatekeeper (IZCT)

Contenido

Introducción

Las redes de H.323 tienen diferentes tipos de configuraciones y de flujos de llamada. Este documento discute la mayor parte de los problemas de seguridad con las redes de H.323 que implican a los porteros. Este documento resume la manera los trabajos de cada característica y cómo resolverla problemas con una explicación en la mayor parte de los debugs. Este documento no dirige la seguridad general del VoIP.

Este documentos abarca estas características:

- **Seguridad de gateway a gatekeeper dentro del dominio** — Esta Seguridad se basa en el H.235, en el cual las llamadas de H.323 son autenticadas, autorizadas, y ruteadas por un portero. Consideran al portero haber sabido y una entidad confiable en cierto modo que el gateway no la autentica cuando el gateway intenta registrarse con él.
- **Gatekeeper entre dominios a la seguridad del gatekeeper** — Esta Seguridad cubre la autenticidad y autorizar de las llamadas de H.323 entre los dominios administrativos de los

proveedores de servicios de telefonía por Internet (ITSP) usando el **InterZone Clear Token (IZCT)**. Este documentos abarca solamente la porción donde el portero terminal envía un token en su mensaje del Location Confirmation (LCF) de modo que autentique el pedido de admisión del answerCall (ARQ). La validación del Location Request (LRQ) no se incluye en esta característica. La validación de LRQ es una característica programada para una versión de software futura de Cisco IOS®.

Definiciones

Siglas	Definición
ARQ	Pedido de admisión — Un mensaje del protocolo de registración, admisión y estado (RAS) enviado de un punto final de H.323 a un portero que solicita una admisión para establecer una llamada.
ACF	Admission Confirm — Un mensaje RAS enviado del portero al punto final que confirma la aceptación de una llamada.
ARJ	Admission Rejection — Un mensaje RAS del portero al punto final que rechaza el pedido de admisión.
CAT	Cisco Access Token — El token no cifrado H.235.
GRIETA	Challenge Handshake Authentication Protocol — Un protocolo de autenticación donde se utiliza un desafío.
GCF	Gatekeeper Confirm — Un mensaje RAS enviado de un portero al punto final de H.323 que confirma la detección del portero.
GRQ	Gatekeeper Request — Un mensaje RAS enviado de un punto final de H.323 para descubrir al portero.
H.235	Recomendación ITU para seguridad y cifrado para los terminales multimedia de las H-series (el H.323 y el otro H.245-based).
IZCT	InterZone Clear Token — Un IZCT se genera en el Gatekeeper de origen cuando se inicia un LRQ o un ACF está a punto de ser enviado para una llamada del intrazone dentro del dominio administrativo ITSP.
LRQ	Location Request — Un mensaje RAS enviado de un portero al portero o al tramo de llamada del salto siguiente para localizar y para rutear la llamada.
RAS	Registro, admisión, y estatus — Un protocolo que permite que un portero realice el registro, la admisión, y las revisiones de estado del punto final.
RCF	El registro confirma — Un mensaje RAS enviado del portero al punto final que confirma el registro.
RRJ	Registration Reject — Un mensaje RAS enviado del portero que rechaza el pedido de inscripción.

RRQ	Pedido de inscripción — Un mensaje RAS enviado del punto final al portero que pide para registrarse con él.
RIP	Petición en curso — Un mensaje RAS enviado de un portero al remitente que estado la llamada está en curso.

Puerta de enlace dentro del dominio para seguridad del control de acceso

H.323 es una recomendación de ITU que dirige la sujeción de la comunicación a través de redes no seguras en tiempo real. Esto implica dos áreas importantes de preocupación: autenticación y aislamiento. Hay dos tipos de autenticación, según el H.235:

- Autenticación simétrica basada en encripción que no requiere ningún contacto anterior entre las entidades de comunicación.
- De acuerdo con la capacidad de tener cierto secreto compartido previó (referido más lejos como suscripción basada), dos formas de autenticación suscripción-basada se proporcionan:contraseñacertificado

Sello de fecha/hora pasado en los token

Un sello de fecha/hora se utiliza para prevenir los ataques con paquetes copiados. Por lo tanto, es necesario para la referencia aceptable a mutuamente - al tiempo (de cuál para derivar los sellos de fecha/hora). El periodo de la posición oblicua aceptable del tiempo es una cuestión de implementación local.

Cómo Cisco implementa la recomendación H.235

Cisco utiliza un Challenge Handshake Authentication Protocol (CHAP) - como el esquema de autenticación como la base para su gateway a su implementación del portero H.235. Esto permite que usted leverage el Authentication, Authorization, and Accounting (AAA), usando la funcionalidad existente para realizar la autenticación real. También significa que no requieren al portero tener acceso a una base de datos de ID de gateway, a los números de cuenta de usuario, a las contraseñas, y a los contactos. El esquema se basa en el H.235, la sección 10.3.3. Se describe como contraseña suscripción-basada con el picado.

Sin embargo, en vez de usar el cryptoTokens H.225, este método utiliza los clearTokens H.235 con los campos poblados apropiadamente para el uso con el RADIUS. Este token se refiere como el Cisco Access Token (CAT). Usted puede realizar siempre la autenticación localmente en el portero en vez de usar a un servidor de RADIUS.

El uso del cryptoTokens requiere que el portero mantenga o tiene cierta manera de adquirir las contraseñas para todos los usuarios y gatewayes. Esto es porque el campo simbólico del cryptoToken se especifica tales que la entidad de autenticidad necesita la contraseña generar su propio token contra el cual comparar recibido.

Los gatekeeperes de Cisco ignoran totalmente el cryptoTokens. Sin embargo, gatekeeperes vocalTek y otros que soportan H.235 el uso de la sección 10.3.3 el cryptoTokens de autenticar el

gateway. Los gatekeeperes de Cisco utilizan el CAT para autenticar el gateway. Puesto que el gateway no sabe con qué tipo de portero habla, envía ambos en el RRQ. El authenticationCapability en el GRQ está para el cryptoToken e indica que el picado MD5 es el mecanismo de autenticación (aunque el CAT también utiliza el MD5).

Refiera a la Seguridad de gateway y a las mejoras en la contabilidad de Cisco H.323 para más información.

Cómo configurar los niveles de seguridad

- Punto final o Seguridad de nivel de inscripciónCon la seguridad de registro habilitada en el portero, el gateway se requiere para incluir un CAT en todos los mensajes RRQ pesados. El CAT, en este caso, contiene la información que autentica el gateway sí mismo al portero. El portero formata un mensaje a un servidor de RADIUS que autentique la información contenida en el token. Responde de nuevo al portero con un access-accept o el Access-Reject. Esto, a su vez, responde al gateway con un RCF o un RRJ.Si una llamada entonces se pone de un gateway con éxito autenticado, ese gateway genera un nuevo CAT tras el recibo de un ACF del portero que usa la contraseña de gateway. Este CAT es idéntico al que está generado durante el registro a excepción del sello de fecha/hora. Se coloca en el mensaje setup saliente. El gateway de destino extrae el token del mensaje setup y lo coloca en el lado de destino ARQ. El portero utiliza el RADIUS para autenticar el gateway de origen antes de que envíe el lado de destino ACF. Esto previene un punto final NON-autenticado que conozca el direccionamiento de un gateway de usarlo para evitar el plan de seguridad y de acceder el Public Switched Telephone Network (PSTN).Por lo tanto, en este nivel, no hay necesidad de incluir ningunos tokens en los ARQ que originan.Teclee **[no] el security token required-for registration del** comando line interface(cli) del portero de configurar al portero. *La ninguna* opción del comando hace al portero marcar no más para saber si hay tokens en los mensajes RAS.Del tipo **punto final llano del <PASSWORD> de la contraseña de seguridad [no] del** gateway CLI para configurar el gateway. *La ninguna* opción del comando hace el gateway generar no más los tokenes para mensajes RAS.
- Seguridad por llamada llanaEstructuras de la seguridad por llamada sobre la Seguridad de nivel de inscripción. Además de los requisitos de seguridad de registro de la reunión, un gateway también se requiere para incluir los tokenes de acceso en todos los mensajes ARQ del lado de origen cuando la seguridad por llamada se habilita en el portero. El token en este caso contiene la información que identifica al usuario del gateway al portero. Esta información se obtiene usando un script de la respuesta de voz interactiva (IVR) en el gateway. Esto indica a los usuarios que ingresen su identificación del usuario y el PIN del telclado numérico antes de que pongan una llamada.El CAT contenido en el ARQ que origina es autenticado por el RADIUS igual que descrito anterior en la punto final o la Seguridad de nivel de inscripción. Después de que reciba el ACF, el gateway genera un nuevo CAT usando su contraseña y lo envía dentro del mensaje setup H.225 al gateway de terminación.Teclee **[no] el token de seguridad requerir-para todos del** portero CLI para configurar al portero. *La ninguna* opción del comando hace al portero marcar no más para saber si hay tokens en los mensajes RAS.Del tipo **<PASSWORD> de la contraseña de seguridad [no] llano por llamada del** gateway CLI para configurar el gateway. *La ninguna* opción del comando hace el gateway generar no más los tokenes para mensajes RAS.
- Todos nivelan la SeguridadEsto permite que el gateway incluya un CAT en todos los mensajes RAS necesarios para el registro y para las llamadas. Por lo tanto, es una

combinación de los dos niveles antedichos. Con esta opción, la validación de CAT en los mensajes ARQ se basa en el número de cuenta y el PIN del usuario que hace una llamada. La validación del CAT enviado en todos los otros mensajes RAS se basa en la contraseña configurada para el gateway. Por lo tanto, es similar al por llamada llano.Teclee **[no] el token de seguridad requerir-para todos del** portero CLI para configurar al portero. *La ninguna* opción del comando hace al portero marcar no más para saber si hay tokens en los mensajes RAS.Del tipo **nivel todo del <PASSWORD> de la contraseña de seguridad [no] del** gateway CLI para configurar el gateway. *La ninguna* opción del comando hace el gateway generar no más los tokenes para mensajes RAS.

Utilización de H.235 en un nivel por llamadas sin IVR

El H.235 no se puede utilizar en un por llamada llano sin el IVR. Si no hay IVR para recoger una cuenta y el PIN, el gateway necesita enviar el ARQ sin un token no cifrado (pero con un token de criptografía). Puesto que el gatekeeper de Cisco valida solamente los tokenes no cifrados, la llamada es rechazada por el portero con una razón de la denegación de seguridad.

Este ejemplo muestra los debugs del **asn1 del h225** recogidos de un **gateway de origen (OGW)** que no se configure para que un IVR recoja la cuenta y el PIN. El mensaje RPQ tiene un token no cifrado, pero el ARQ no hace. Un mensaje ARJ se devuelve al gateway.

```
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
```

Refiera a la sección de las tareas de configuración de las estadísticas y de las mejoras de la seguridad de Cisco H.235 para los gatewayes de Cisco para más información sobre la configuración de IVR.

Aspectos importantes

Los aspectos importantes que usted necesita ser referido a incluyen:

- Configuración del gateway y del portero
- Configuración de RADIUS en el portero y el servidor de RADIUS
- Network Time Protocol (NTP) — Usted debe tener el mismo tiempo en todos los gatewayes y porteros. Porque la información de autenticación incluye un grupo fecha/hora, es importante que todo el Cisco el Gateways H.323 y los porteros (o la otra entidad que realiza la autenticación) esté sincronizado. Cisco Gateways H.323 se debe sincronizar usando el NTP.
- Falla de software debido a un bug

Depuraciones y flujo de llamada para los diferentes niveles

Puesto que toda la Seguridad llana cubre el registro y por llamada los casos, el laboratorio se configura con ese nivel de seguridad para este ejercicio. Los flujos de llamada para la parte de la inscripción y una llamada VoIP normal se explican en la configuración aquí.

Nota: La configuración aquí no es completa. Los comandos more siguen entre las salidas de los debugs. Se diseña para mostrar qué problema puede suceder si usted no marca todas las cosas tales como configuración, NTP, y RADIUS. Además, el gateway se autentica en el portero localmente de modo que usted pueda ver que lo que valora se fijan para la identificación del gateway y la contraseña. Se corta esta configuración para solamente mostrar la configuración relacionada.

```
!
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
```

Estos debugs se giran en este ejemplo:

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

La primera cosa que ocurre es que el gateway envía un GRQ al portero y al portero envía un GCF al gateway. El gateway después envía un RRQ y espera un RCF o el RRJ.

En la configuración previa, el gateway no se fija para ningún nivel de seguridad de modo que su GRQ no lleve ningún **authenticationCapability** que sea necesario para los tokens. Sin embargo, el portero todavía devuelve un GCF mientras que esta salida muestra:

```
*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 }
   }
  }
```

Esto explica algunos de los mensajes en el GRQ:

- **authenticationCapability** — Este campo tiene solamente el valor del pwdHash. Indica que el

picado MD5 es el mecanismo de autenticación.

- **algorithmOIDs** — La identificación de objeto del algoritmo en este caso lleva el valor (1 2 840 113549 2 5) que es el ID del objeto para el picado de la publicación de mensaje 5.(1 2 840 113549 2 5) traduce iso(1) al miembro-body(2) US(840) rsadsi(113549) digestAlgorithm(2) md5(5). Por lo tanto Cisco hace el picado de la contraseña MD5.El identificador de objeto del Open Systems Interconnection (OSI) de Inc. RSA Data Security, Inc.'s de Rsa Data Security de la significa de Rsadsi es 1.2.840.113549 (0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d en el maleficio), según lo registrado por el XXX (ANSI).

El portero envía otra vez un GCF como el mensaje anterior. El gateway entonces envía un RRQ con ciertos campos para describir los tokens que utiliza para ser autenticado. **El mensaje RPQ del asn1** se envía que se muestra en este ejemplo.

```
*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
   }
```

Antes de que se discuta la respuesta, algunos de los campos relacionados en el mensaje RPQ antedicho se explican aquí. Hay dos tipos de tokens: un token no cifrado o un CAT) y un token de criptografía.

Como se mencionó antes, los gatekeeperes de Cisco ignoran los tokenes de criptografía. Sin embargo, el gateway todavía envía ambos porque no sabe a qué tipo de portero está hablando. Puesto que los otros vendedores pudieron utilizar los tokenes de criptografía, el gateway envía ambos.

Esto explica la sintaxis de ClearToken.

- **tokenOID** — El ID del objeto para identificar el token.
- **grupo fecha/hora** — La época del tiempo universal coordinado actual (UTC) del gateway. Segundos desde 00:00 1/1/1970 UTC.Se utiliza como Grieta-desafío implicado como si hubiera venido inicialmente del portero.
- **desafío** — Una publicación de mensaje 16-byte MD5 generada por el gateway usando estos campos:hash del desafío = del [random + GW/User Password + timeStamp] MD5El RADIO realiza este cálculo (puesto que conoce el número aleatorio, la contraseña de gateway, y el desafío de la GRIETA) para determinar lo que debe ser el desafío: Hash de la respuesta = del [CHAP ID + UserPassword + CHAP Challenge] MD5 de la GRIETA
- **al azar** — Un valor del octeto usado por el RADIUS para identificar esta petición determinada.El gateway incrementa un módulo variable del 256 para que cada pedido de autenticación satisfaga este requisito RADIUS.
- **generalID** — El gateway H323-ID o el número de cuenta de usuario basado en el nivel de seguridad y el tipo de mensaje RAS.

Nota: Todos estos campos son importantes. Sin embargo, más atención se presta al sello de fecha/hora y al generalID. En este caso, el sello de fecha/hora es **731030839** y el generalID es **gwa-1@cisco.com**.

El cryptoToken en el RRQ contiene la información sobre el gateway que genera el token. El incluye la identificación del gateway (que es H.323 ID configurado en el gateway) y la contraseña de gateway.

Este campo contiene uno de los tipos del cryptoToken definidos para el campo CryptoH323Token especificado en el H.225. Actualmente, el único tipo de cryptoToken soportado es el cryptoEPPwdHash.

Estos campos se contienen dentro del campo del cryptoEPPwdHash:

- **alias** — El gateway alias, que es H.323 ID del gateway.
- **grupo fecha/hora** — El sello de la hora actual.
- **token** — La publicación de mensaje 5 (MD5)-encoded PwdCertToken. Este campo contiene estos elementos:**grupo fecha/hora** — Lo mismo que el grupo fecha/hora del cryptoEPPwdHash.**contraseña** — La contraseña del gateway.**generalID** — El mismo gateway alias que el que está incluido en el cryptoEPPwdHash.**tokenID** — El ID del objeto.

Vea la respuesta del portero en este ejemplo.

```
*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:
```

El RRQ es rechazado por el portero. La razón de esto es porque no había NTP fijado en la

configuración del gateway. El portero marca el sello de fecha/hora del token para ver si está dentro de una ventana aceptable en relación con su propio tiempo. Esta ventana es actualmente +/- 30 segundos alrededor de la hora UTC del portero.

Un exterior simbólico de esta ventana hace al portero desechar este mensaje. Esto previene los ataques con paquetes copiados alguien que intenta reutilizar un token snooped. En la práctica, todos los gatewayes y porteros necesitan utilizar el NTP para evitar este problema de desviación del vez. El portero encuentra que el sello de fecha/hora en el token está dentro de la ventana aceptable de su tiempo. Por lo tanto, no marca con el RADIUS para autenticar el gateway.

El gateway entonces se configura para el NTP que señala al portero como el master NTP, de modo que el gateway y el portero tengan el mismo tiempo. Cuando ocurre esto, el gateway envía un nuevo RRQ y este vez el portero contesta de nuevo al nuevo RRQ con un RRJ.

Estos debugs son del portero. Funcionamiento de los debugs para ver si el portero va a la fase de autenticación.

```
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"}
  }
```

Después de configurar el NTP, el portero todavía rechaza el RRQ. Esta vez, sin embargo, pasa con el proceso de autenticación para ese gateway. El portero rechaza el RRQ porque no encuentran al usuario (aquí la identificación del gateway) en la lista de usuarios válidos en el RADIUS. El gateway se autentica localmente en la configuración del portero. En la lista de usuario usted ve gwa-1. Sin embargo, ése no es el usuario correcto puesto que el usuario en el RRQ es gwa-1@cisco.com.

También, una vez que el comando 2222 de la contraseña 0 de gwa-1@cisco.com del nombre de usuario se configura en el portero, el portero confirma el RRQ y se registra el gateway.

En este laboratorio, otro gateway (gwa-2) se registra al mismo portero (gka-1). Una llamada se hace de gwa-1@cisco.com a gwa-2 para considerar cómo el ARQ, el ACF, y los mensajes setup miran.

Estos debugs recogidos son del gateway de origen y destino (gwa-2).

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

Una explicación de algunos de los mensajes del debug es incluida.

Antes de que usted imprima los debugs de originar/gateway de terminación, este texto explica el flujo de llamada:

1. Cuando un mensaje setup se recibe del PSTN, el gateway envía un ARQ a y recibe un ACF del portero.
2. Cuando el gateway recibe el ACF, el gateway genera un CAT usando la contraseña de gateway, el H323-ID alias, y la hora actual. El token se coloca en el Bloqueo de control de llamada (CCB).
3. Cuando el gateway envía el mensaje setup al gateway de terminación, extrae el token de acceso del CCB y lo pone en el campo del nonStandardParameter de un clearToken dentro del mensaje setup.
4. El gateway de terminación quita el token del mensaje setup, lo convierte de un nonStandardParameter en un CAT, y lo coloca en el ARQ.
5. El portero marca el sello de fecha/hora del token para ver si está dentro de una ventana aceptable en relación con su propio tiempo. Esta ventana es actualmente +/- 30 segundos alrededor de la hora UTC del portero. Un exterior simbólico de esta ventana hace al portero desechar este mensaje. Esto hace la llamada ser rechazada.
6. Si el token es aceptable, el portero formata un paquete de pedidos del acceso a RADIUS,

completa los atributos apropiados para verificar un desafío de la GRIETA y los envía a un servidor de RADIUS.

7. De acuerdo con la suposición que el gateway alias está sabido en el servidor, el servidor localiza la contraseña asociada a este alias y genera su propia respuesta de la GRIETA usando el alias, la contraseña, y el desafío de la GRIETA del portero. Si su respuesta de la GRIETA hace juego el que está recibido del portero, el servidor envía un acceso valida el paquete al portero. Si no hacen juego, o si el gateway alias no está en la base de datos de servidor, el servidor envía un paquete del rechazo de acceso de nuevo al portero.
8. El portero responde al gateway con un ACF si recibe un acceso valida, o un ARJ con el código de la causa **securityDenial** si recibe un rechazo de acceso. Si el gateway recibe un ACF, la llamada está conectada.

Este ejemplo muestra el debug del gateway de origen.

Nota: Los debugs del **asn1 del h225** para la configuración no están en este ejemplo puesto que es lo mismo como se ve en el ejemplo de terminación del gateway que sigue el ejemplo de la gateway de origen.

```
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
```

Este ejemplo muestra los debugs del **gateway de terminación (TGW)**. Note que el TGW ha configurado la segunda pierna desde que consiguió el ACF, y la llamada está conectada.

```
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 'E00102000104150403 9090A31803A983811E0285...'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)
```

Problemas con el IOS de la gateway

En el mismo laboratorio, la imagen del IOS 12.2(6a) se carga en el OGW. Cuando se hace una llamada, se nota que el OGW todavía envía un token no cifrado basado en su contraseña aunque el gateway no se configura para que el IVR recoja el Account/PIN. Además, el portero configurado para todo el nivel valida esa llamada. Esto se documenta en el Id. de bug Cisco CSCdw43224 (clientes registrados solamente).

Seguridad con puntos finales alternativos

Según lo mencionado anterior en este documento, la Seguridad de la llamada entre extremos se proporciona el uso de los tokenes de acceso que se envían en el campo de los clearTokens en los mensajes RAS/H.225. Al habilitar tal Seguridad, el gateway de origen adelante el token de acceso recibido del portero en un ACF al punto final de H.323 del destino en el mensaje setup H.225. Este punto final de H.323 del destino entonces adelante el token de acceso recibido en el mensaje setup al portero en su pedido de admisión. De esta manera, da a gatekeeper remoto la capacidad de admitir las llamadas basadas en la validez del token de acceso. El contenido del token de acceso está hasta la entidad que lo genera. Para minimizar a las brechas en la seguridad y guardar contra los ataques del intermediario, los porteros pueden codificar la información específica del destino en el token de acceso. Esto significa que cuando los alternateEndpoints se proporcionan en un ACF, el portero puede proporcionar un token de acceso separado para cada alternateEndpoint especificado.

Cuando primero intenta establecer una conexión, el gateway de Cisco envía el token de acceso que ha recibido en el campo del clearToken del ACF con el direccionamiento en el campo de los destCallSignalAddress. Si esta tentativa es fracasada y el gateway de Cisco procede a las conexiones de la tentativa con un punto final alterno, utiliza el token de acceso asociado (si está disponible) de la lista alternateEndpoints. Si la lista alternateEndpoints recibida en el ACF no incluye los tokenes de acceso, pero el ACF incluye un token de acceso, el gateway de Cisco incluye este token de acceso en todas las tentativas de conectar con un punto final alterno.

Soporte de Token OSP

Actualmente el Open Settlement Protocol (OSP) y sus tokens se soportan solamente en los gatewayes de Cisco. No hay soporte en el portero. El gateway reconoce los tokenes OSP recibidos de un servidor de asentamiento y los inserta en el mensaje setup del q.931 a un gateway de terminación.

Diferentes niveles de seguridad para cada punto final o zona

Usted no puede actualmente configurar diversos niveles de seguridad para cada punto final o zona. El nivel de seguridad está para todas las zonas manejadas por ese portero. Una petición de la característica se puede abrir para tal problema.

Controlador de acceso entre dominios a Seguridad de controlador de acceso

El gatekeeper entre dominios a la seguridad del gatekeeper proporciona la capacidad de validar las peticiones del portero-a-portero del intradomain y del interdomain sobre una base del por-salto. Esto significa que el gatekeeper de destino termina el CAT y genera un nuevo si el portero decide remitir el LRQ hacia adelante. Si el portero detecta una firma inválida LRQ responde enviando un Location Reject (LRJ).

Implemente al portero a la seguridad del gatekeeper

El Gatekeeper de origen genera un IZCT cuando se inicia un LRQ o un ACF está a punto de ser enviado en caso de una llamada de la intra-zona. Este token se atraviesa a través de su trayectoria de ruteo. A lo largo de la trayectoria, cada portero pone al día el gatekeeper de destino ID y/o el ID de gatekeeper de la fuente en caso necesario, para reflejar la información de zona. El portero terminal genera un token con su contraseña. Este token se lleva detrás en los mensajes del Location Confirmation (LCF) y se pasa al OGW. El OGW incluye este token en el mensaje setup H.225. Cuando el TGW recibe el token, se remite en la llamada de respuesta de ARQ y es validado por el portero terminal (TGK) sin ninguna necesidad de un servidor de RADIUS.

Basan al tipo de autenticación en la contraseña con el picado según lo descrito en ITU H.235. Específicamente, el método de encripción es MD5 con el picado de la contraseña.

El propósito del IZCT es saber si el LRQ ha llegado de un dominio externo, del cual zona, y de qué portador. También se utiliza para pasar un token al OGW en el LCF del TGK. Dentro del formato IZCT, se requiere esta información:

- srcCarrierID — Identificación de la portadora de origen
- dstCarrierID — Identificación de la portadora de destino
- intCarrierID — Identificación de la portadora intermedia
- srcZone — Zona de origen
- dstZone — Zona de destino
- tipo entre zonasINTRA_DOMAIN_CISCOINTER_DOMAIN_CISCOINTRA_DOMAIN_TERM_NOT_CISCOINTER_DOMAIN_ORIG_NOT_CISCO

Esta característica trabaja muy bien sin ninguna necesidad de un ID de la portadora del gateway o de un servidor del Carrier Sensitive Routing (CSR). En tal caso, los campos sobre el ID de la portadora están vacíos. Los ejemplos aquí no incluyen ningún ID de la portadora. Para un flujo de llamada detallado, la versión y el Soporte de la plataforma, y las configuraciones, refieren al Mejora de la seguridad de acceso y autenticación entre dominios.

Configuración de Gatekeeper (control de acceso)

La característica IZCT requiere esta configuración en el portero.

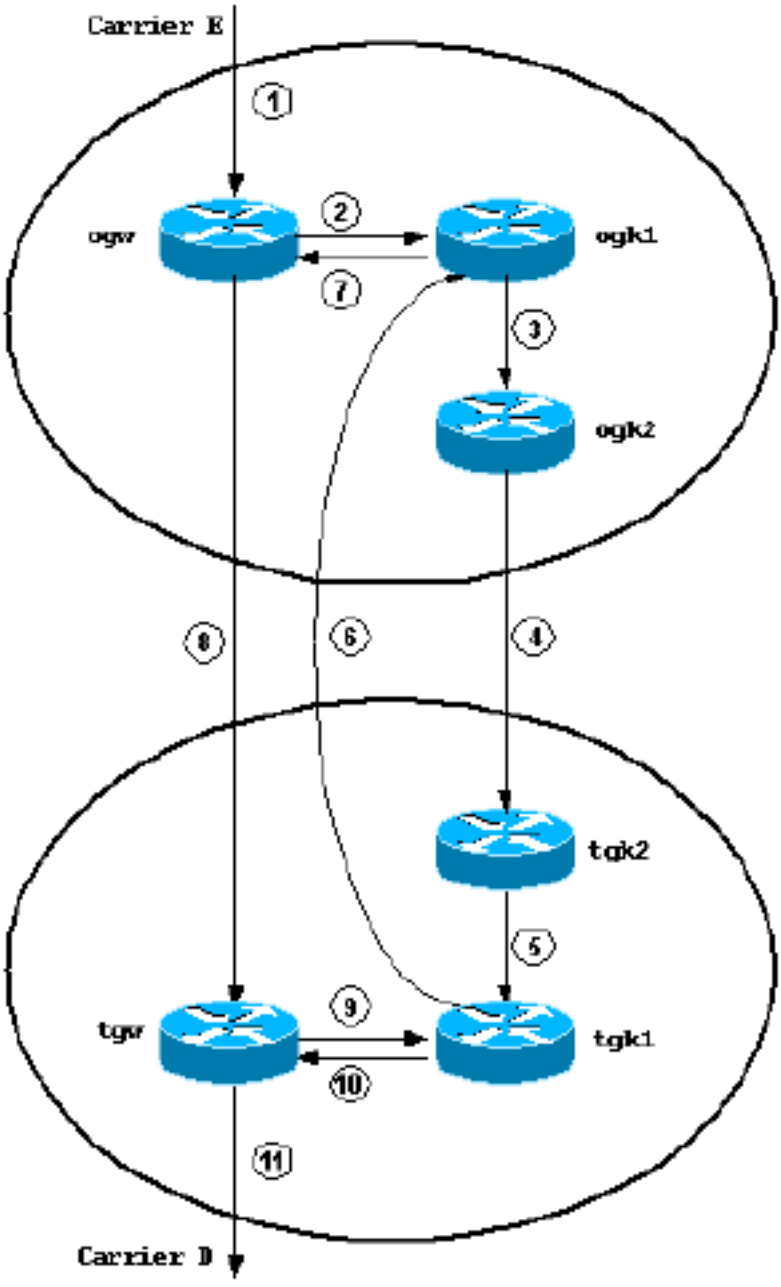
```
Router(gk-config)#
[no] security izct password <PASSWORD>
```

La contraseña necesita ser seis a ocho caracteres. Identifique qué zona está en un dominio externo como esto:

```
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]
```

Flujo de llamada IZCT

Este diagrama muestra el flujo IZCT.



En esta configuración, los nombres de los gatewayes y los porteros son lo mismo que ésos usados en el diagrama de flujo de llamadas IZCT pero con la minúscula. El flujo de llamada se explica después de la configuración, con las explicaciones del debug.

Para explicar la característica y el flujo de llamada IZCT, el primer ejemplo no tiene la seguridad de gateway a gatekeeper dentro del dominio. Después de eso, hay ejemplos donde no está capaz el TGW de generar el IZCT de modo que el TGK1 rechace la llamada. Éste es mostrar que la característica trabaja según lo diseñado. Todas estas configuraciones se basan en la topología en el diagrama de flujo de llamadas IZCT.

Ejemplo 1: Flujo de llamada para el portero a la seguridad del gatekeeper solamente

Este ejemplo muestra las configuraciones relacionadas de todos los gatewayes y porteros.

Configuración de OGW	Configuración TGW
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]**
Configuración OGK1	**Configuración TGK1**
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]**
Configuración OGK2	**Configuración TGK2**
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]**

Flujo de llamadas con depuración

Estos ejemplos utilizan los debugs para explicar el flujo de llamada.

1. Un usuario en el portador E llama a un usuario en el portador D.

```
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. Puesto que el dialpeer del gateway de origen (tag=3653) se configura para el RAS, envía un ARQ al 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. Cuando el OGK1 recibe el ARQ, determina que el destino es mantenido por la zona remota OGK2. Entonces identifica que un IZCT está necesitado (con el CLI: **security izct password <pwd>**). El OGK1 procede a crear el IZCT antes de que se envíe el LRQ. Entonces manda el IZCT y el LRQ al OGK2 y envía un mensaje del RIP de nuevo al 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. {
   izctInterZoneType intraDomainCisco : NULL
!--- The destination is in the same domain, it is intraDomainCisco type. izctSrcZone "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. Cuando el OGK2 recibe el LRQ, marca el IZCT. De la configuración encuentra que los neets LRQ también para contener un IZCT. El OGK2 entonces crea un nuevo IZCT cambiando el izctSrcZone y el izctDstZone para ser ogk2 y adelante el LRQ al TGK2. Después de que envíe el LRQ al TGK2, devuelve un mensaje del RIP al OGK1.Si los porteros son parte de al cluster, el nombre de clúster se utiliza para el SrcZone o el 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. El TGK2 determina que el LRQ viene de un dominio externo. Pone al día el dstZone IZCT con su propio ID e interZoneType como INTER_DOMAIN_CISCO. Después crea un nuevo CAT y pasa el IZCT actualizado y el LRQ al TGK1.El TGK2 trata la zona de la cual un LRQ se recibe como zona del dominio externo en cualquiera de estos dos escenarios:La lista de zonas remotas TGK2 no contiene la zona de la cual se recibe un LRQ.La lista de zonas remotas TGK2 contiene la zona de la cual se recibe un LRQ. La zona se marca con un indicador del dominio externo.Entonces envía una petición el mensaje en curso de nuevo al 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. El TGK1 pone al día normalmente el dstCarrierID IZCT al portador E, que es determinado

por el proceso de ruteo. Sin embargo, puesto que no se utiliza ningunos portadores, usted no ve eso. El TGK1 genera un token del hash con la contraseña IZCT. Envía un LCF con el IZCT actualizado en él al OGK1. Este izctHash se utiliza para autenticar el answerCall ARQ que el TGK1 recibe del TGW cuando el más adelante recibe el mensaje setup VoIP del OGW.

```
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. El OGK1 extrae el IZCT del LCF y lo envía en un ACF al 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. El OGW envía el IZCT al TGW en el mensaje setup H.225.

```
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,
     '400000060401004C6013801114000100AC100D0F...'H
    }
    mediaWaitForConnect FALSE
    canOverlapSend FALSE
   }
   h245Tunneling TRUE
   nonStandardControl
   {
    {
     nonStandardIdentifier h221NonStandard :
     {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
```

```
      }
      data '6001020001041F04038090A31803A983816C0C21...'H
     }
    }
   }
  }
```

9. El TGW pasa el IZCT al TGK1 en una llamada de respuesta de ARQ.

```
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. El TGK1 autentica el destino IZCT con éxito. Esto es porque el TGK1 genera el hash en el

IZCT y devuelve un ACF al 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. El TGW establece la llamada hacia el portador D después de que reciba el ACF.**Ejemplo 2: La llamada falló porque el TGW no puede extraer el IZCT del mensaje de configuración recibido.**Este ejemplo se basa en la misma topología y configuración que el ejemplo 1. En este ejemplo, el software del TGW se cambia a una versión donde el IZCT no se soporta. En tal caso, el TGW no puede extraer el IZCT del mensaje setup. Esto hace el TGK1 rechazar la llamada con un motivo de desconexión de la denegación de seguridad.Este ejemplo muestra solamente el mensaje setup, el ARQ, y el ARJ en el TGW puesto que el flujo de llamada es lo mismo que el ejemplo 1.

```
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
```

Información Relacionada

- Mejora de la seguridad de acceso y autenticación entre dominios