Gateway to Gatekeeper (H.235) and Gatekeeper to Gatekeeper (IZCT) Security Troubleshooting Guide

Índice

Introdução

As redes de H.323 têm tipos diferentes das configurações e dos fluxos de chamadas. Este documento discute a maioria dos interesses de segurança com as redes de H.323 que envolvem porteiros. Este documento resume a maneira trabalhos de cada característica e como pesquisá-la defeitos com uma explicação na maioria do debuga. Este documento não endereça a segurança total de VoIP.

Este capas de documento estas características:

- **Gateway de intradomínio para a segurança do gatekeeper** — Esta Segurança é baseada no H.235, em que os atendimentos de H.323 são autenticados, autorizados, e distribuídos por um porteiro. O porteiro é considerado sabido e uma entidade confiável de um certo modo que o gateway não o autentica quando o gateway tenta se registrar com ele.
- **Gatekeeper de interdomínio à segurança de gatekeeper** — Esta Segurança cobre a autenticação e a autorização de atendimentos de H.323 entre os campos administrativos dos

provedores de serviços de telefone pelo Internet (ITSP) que usam o **InterZone Clear Token (IZCT)**. Este capas de documento somente a parcela onde o porteiro de terminação envia um token em sua mensagem do Location Confirmation (LCF) de modo que autentique o pedido de admissão da chamada de resposta (ARQ). A validação do Location Request (LRQ) não é incluída nesta característica. A validação LRQ é uma característica programada para um software release futuro de Cisco IOS®.

Definições

Acrônimo	Definição
ARQ	Pedido de admissão — Uma mensagem do Registro, Admissão e Protocolo de Status (RAS) enviada de um valor-limite de H.323 a um porteiro que peça uma admissão para estabelecer um atendimento.
ACF	Admission Confirm — Um mensagem RAS enviado do porteiro ao valor-limite que confirma a aceitação de um atendimento.
ARJ	Admission Rejection — Um mensagem RAS do porteiro ao valor-limite que rejeita o pedido de admissão.
CAT	Cisco Access Token — O clear token H.235.
RACHADURA	Protocolo de autenticação de cumprimento do desafio — Um protocolo de autenticação onde um desafio seja usado.
GCF	Gatekeeper Confirm — Um mensagem RAS enviado de um porteiro ao valor-limite de H.323 que confirma a descoberta do porteiro.
GRQ	Gatekeeper Request — Um mensagem RAS enviado de um valor-limite de H.323 para descobrir o porteiro.
H.235	Recomendação ITU de segurança e criptografia para terminais de multimídia das H-séries (H.323 e o outro H.245-based).
IZCT	InterZone Clear Token — Um IZCT está gerado no gatekeeper de origem quando um LRQ é iniciado ou um ACF está a ponto de ser enviada para um atendimento da intrazona dentro do campo administrativo ITSP.
LRQ	Location Request — Um mensagem RAS enviado de um porteiro ao porteiro ou ao trecho de chamada do salto seguinte para seguir e distribuir o atendimento.
RAS	Registro, admissão, e estado — Um protocolo que permita que um porteiro execute o registro, a admissão, e as verificações de status do valor-limite.
RCF	O registro confirma — Um mensagem RAS enviado do porteiro ao valor-limite que confirma

	o registro.
RRJ	Registration Reject — Um mensagem RAS enviado do porteiro que rejeita a requisição de registro.
RRQ	Requisição de registro — Um mensagem RAS enviado do valor-limite ao porteiro que pede para se registrar com ele.
RIP	Pedido em andamento — Um mensagem RAS enviado de um porteiro ao remetente que indica o atendimento é em andamento.

Gateway intradomínio para segurança do Gatekeeper

H.323 é uma recomendação ITU que os endereços que fixam uma comunicação em redes inseguras do tempo real. Isto envolve duas áreas principal do interesse: autenticação e privacidade. Há dois tipos de autenticação, de acordo com o H.235:

- Autenticação baseada em criptografia simétrica que não exige nenhum contato prévio entre as entidades comunicante.
- Baseado na capacidade para ter algum segredo anterior compartilhado (provido mais como a assinatura baseada), dois formulários da autenticação assinatura-baseada são fornecidos:senhacertificado

Selo de tempo passado nos tokens

Um selo de tempo é usado para impedir ataques de replay. Consequentemente, é precisado para a referência aceitável a mutuamente - ao tempo (de qual para derivar selos de tempo). A quantidade de enviesamento aceitável do tempo é uma matéria da aplicação local.

Como a Cisco implementa as recomendações de H.235

Cisco usa um protocolo de autenticação de cumprimento do desafio (RACHADURA) - como o método de autenticação como a base para seu gateway a sua aplicação do porteiro H.235. Isto permite que você leverage o Authentication, Authorization, and Accounting (AAA), usando a funcionalidade existente para executar a autenticação real. Igualmente significa que o porteiro não está exigido ter o acesso ao base de dados de IDS de gateway, números de conta de usuário, senhas, e pinos. O esquema é baseado no H.235, a seção 10.3.3. É descrito como a senha assinatura-baseada com hashing.

Contudo, em vez de usar os cryptoTokens H.225, este método usa os clearTokens H.235 com os campos povoados apropriadamente para o uso com RAIO. Este token é referido como o Cisco Access Token (CAT). Você pode sempre executar a autenticação localmente no porteiro em vez de usar um servidor Radius.

O uso dos cryptoTokens exige que o porteiro mantém ou tem alguma maneira de adquirir senhas para todos os usuários e gateways. Isto é porque o campo simbólico do cryptoToken é especificado tais que a entidade de autenticação precisa a senha de gerar seu próprio token contra que para comparar recebido.

Os gatekeeperes Cisco ignoram completamente cryptoTokens. Contudo, gatekeeperes vocalTek e outro que apoiam H.235 o uso da seção 10.3.3 os cryptoTokens autenticar o gateway. Os gatekeeperes Cisco usam o CAT para autenticar o gateway. Desde que o gateway não sabe que tipo de porteiro fala, envia ambos no RRQ. O authenticationCapability no GRQ é para o cryptoToken e indica que o hashing MD5 é o mecanismo da autenticação (embora o CAT igualmente usa o MD5).

Refira a segurança de gateway e os aprimoramentos de relatório de Cisco H.323 para mais informação.

Como configurar os níveis de segurança

- Valor-limite ou segurança no nível de registroCom a segurança de registro permitida no porteiro, o gateway é exigido para incluir um CAT em todos os mensagens RRQ pesado. O CAT, neste caso, contém a informação que autentica o gateway próprio ao porteiro. O porteiro formata uma mensagem a um servidor Radius que autentique a informação contida no token. Responde de volta ao porteiro com uma aceitação de acesso ou Rejeição de acesso. Isto, por sua vez, responde ao gateway com um RCF ou um RRJ.Se um atendimento é colocado então de um gateway com sucesso autenticado, esse gateway gerencie um CAT novo após recepção de um ACF do porteiro que usa a senha de gateway. Este CAT é idêntico a esse gerado durante o registro à exceção do selo de tempo. É colocado no mensagem setup que parte. O gateway de destino extrai o token do mensagem setup e coloca-o no lado de destino ARQ. O porteiro usa o RAIO para autenticar o gateway de origem antes que envie o lado de destino ACF. Isto impede um valor-limite NON-autenticado que conheça o endereço de um gateway do usar para contornar o esquema de segurança e de alcançar a rede telefônica pública comutada (PSTN).Consequentemente, neste nível, não há nenhuma necessidade de incluir nenhuns tokens nos ARQ de origem.Datilografe o **security token required-for registration do [no] do** comando line interface(cli) do porteiro configurar o porteiro. *Nenhuma* opção do comando faz com que o porteiro já não verifique para ver se há tokens nos mensagens RAS.Datilografe a **<PASSWORD> da senha de segurança do [no] o valor-limite nivelado do** gateway CLI para configurar o gateway. *Nenhuma* opção do comando faz com que o gateway já não gerencia tokens de mensagens RAS.
- Segurança por chamada niveladaConstruções da segurança por chamada em cima da segurança no nível de registro. Além do que exigências de segurança de registro da reunião, um gateway está exigido igualmente para incluir accesss token em todos os mensagens de ARQ do lado de origem quando a segurança por chamada é permitida no porteiro. O token contém neste caso a informação que identifica o usuário do gateway ao porteiro. Esta informação é obtida usando um script da resposta de voz interativa (IVR) no gateway. Isto alerta usuários inscrever seu usuário - identificação e PIN do teclado numérico antes que coloquem um atendimento.O CAT contido no ARQ de origem é autenticado pelo RAIO da mesma forma como descrito mais cedo no valor-limite ou na segurança no nível de registro. Depois que recebe o ACF, o gateway gerencie um CAT novo usando sua senha e envia-o dentro do mensagem setup H.225 ao gateway de terminação.Datilografe o **token de segurança do [no] exigir-para tudo do** porteiro CLI para configurar o porteiro. *Nenhuma* opção do comando faz com que o porteiro já não verifique para ver se há tokens nos mensagens RAS.Datilografe a **<PASSWORD> da senha de segurança do [no] o por-atendimento nivelado do** gateway CLI para configurar o gateway. *Nenhuma* opção do comando faz com que o gateway já não gerencia tokens de mensagens RAS.

- Todos nivelam a Segurançalsto permite que o gateway inclua um CAT em todos os mensagens RAS necessários para o registro e para atendimentos. Consequentemente, é uma combinação dos dois níveis acima. Com esta opção, a validação de CAT nos mensagens de ARQ é baseada no account number e no PIN do usuário que faz um atendimento. A validação do CAT enviado em todos os outros mensagens RAS é baseada na senha configurada para o gateway. Consequentemente, é similar ao por chamada em nível.Datilografe o **token de segurança do [no] exigir-para tudo do** porteiro CLI para configurar o porteiro. *Nenhuma* opção do comando faz com que o porteiro já não verifique para ver se há tokens nos mensagens RAS.Datilografe o **nível todo do <PASSWORD> da senha de segurança do [no] do** gateway CLI para configurar o gateway. *Nenhuma* opção do comando faz com que o gateway já não gerencia tokens de mensagens RAS.

Uso de H235 em um nível por chamada sem IVR

O H.235 não pode ser usado em um por-atendimento em nível sem IVR. Se não há nenhum IVR para recolher uma conta e um PIN, o gateway precisa de enviar o ARQ sem um clear token (mas com um token de criptografia). Desde que o gatekeeper Cisco aceita somente cleares token, o atendimento é rejeitado pelo porteiro com uma razão da recusa de segurança.

Este exemplo mostra que o **asn1 h225** debuga recolhido de um **gateway de origem (OGW)** que não seja configurado para que um IVR recolha a conta e o PIN. O mensagem de RRQ tem um clear token, mas o ARQ não faz. Um mensagem de ARJ é enviado para trás ao 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
```

Refira a seção das tarefas de configuração da contabilidade e dos aprimoramentos de segurança de Cisco H.235 para Cisco gateway para obter mais informações sobre da configuração de IVR.

Problemas principais

As questões principal que você precisa de ser estado relacionado com incluem:

- Configuração do gateway e do porteiro
- Configuração RADIUS no porteiro e no servidor Radius
- Network Time Protocol (NTP) — Você deve ter o mesmo tempo em todos os gateways e porteiros. Porque a informação da autenticação inclui um timestamp, é importante que todo o Cisco Gateways H.323 e porteiros (ou a outra entidade que executa a autenticação) esteja sincronizado. Cisco Gateways H.323 deve ser sincronizado usando o NTP.
- Falha de software devido a um erro

Depurações e fluxo de chamada para níveis diferentes

Desde que toda a Segurança nivelada cobre casos do registro e do por-atendimento, o laboratório estabelece-se com esse nível de segurança para este exercício. Os fluxos de chamadas para a parte de registro e uma chamada VoIP normal são explicados na configuração aqui.

Nota: A configuração aqui não está completa. Os comandos more seguem entre os resultados do debug. Projeta-se mostrar que problema pode acontecer se você não verifica todas as coisas tais como a configuração, o NTP, e o RAIO. Além, o gateway é autenticado no porteiro localmente de modo que você possa ver que o que avalia são ajustados para o ID de gateway e a senha. Esta configuração é cortada de modo que somente a configuração relacionada seja mostrada.

```
!
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
```

Estes debugam são girados sobre neste exemplo:

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

A primeira coisa que ocorre é que o gateway envia um GRQ ao porteiro e ao porteiro envia um GCF ao gateway. O gateway então envia um RRQ e espera um RCF ou o RRJ.

Na configuração precedente, o gateway não é ajustado para nenhum nível de segurança de modo que seu GRQ não leve nenhum **authenticationCapability** que é precisado para os tokens. Contudo, o porteiro ainda envia para trás um GCF enquanto esta saída mostra:

```
*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 } } }
```

Isto explica algumas das mensagens no GRQ:

- **authenticationCapability** — Este campo tem somente o valor do pwdHash. Indica que o hashing MD5 é o mecanismo da autenticação.
- **algorithmOIDs** — A identificação de objeto do algoritmo neste caso leva o valor (1 2 840 113549 2 5) que é a identificação de objeto para o hashing do message digest 5.(1 2 840 113549 2 5) traduz iso(1) ao membro-body(2) US(840) rsadsi(113549) digestAlgorithm(2) md5(5). Daqui Cisco faz o hashing da senha MD5.Rsadsi representa Rsa Data Security identificador de objeto do Open Systems Interconnection (OSI) s de Inc. Rsa Data Security, Inc. 'é 1.2.840.113549 (0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d encantam dentro), como registrado pelo american national standards institute (ANSI).

O porteiro envia outra vez um GCF como o mensagem anterior. O gateway envia então um RRQ com determinados campos para descrever os tokens que se use para ser autenticado. **O mensagem de RRQ do asn1** que é enviado é mostrado neste exemplo.

```
*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 que a resposta esteja discutida, alguns dos campos relacionados no mensagem de RRQ acima estão explicados aqui. Há dois tipos de tokens: um clear token ou um CAT) e um token de criptografia.

Como mencionado antes, os gatekeeperes Cisco ignoram os tokens de criptografia. Contudo, o gateway ainda envia ambos porque não sabe a que tipo de porteiro está falando. Desde que os outros fornecedores puderam usar tokens de criptografia, o gateway envia ambos.

Isto explica a sintaxe Clear Token.

- **tokenOID** — A identificação de objeto para identificar o token.
- **timeStamp** — A época do tempo universal coordenado atual (UTC) do gateway. Segundos desde 00:00 1/1/1970 de UTC.Éusado como um Rachadura-desafio implicado como se tinha vindo inicialmente do porteiro.
- **desafio** — Um message digest 16-byte MD5 gerado pelo gateway usando estes campos:mistura do desafio = do [random + GW/User Password + timeStamp] MD5O RAIO executa este cálculo (desde que conhece o número aleatório, a senha de gateway, e o desafio da RACHADURA) para determinar o que o desafio deve ser: Mistura da resposta = do [CHAP ID + UserPassword + CHAP Challenge] MD5 da RACHADURA
- **aleatório** — Um valor do byte usado pelo RAIO para identificar este pedido particular.O gateway incrementa um módulo variável do 256 para que cada pedido de autenticação satisfaça este requisito de RADIUS.
- **generalID** — O gateway H323-ID ou o número de conta de usuário baseado no nível de segurança e no tipo de mensagem RAS.

Nota: Todos estes campos são importantes. Contudo, mais atenção é dada ao selo de tempo e ao generalID. Neste caso, o selo de tempo é **731030839** e o generalID é **gwa-1@cisco.com**.

O cryptoToken no RRQ contém a informação sobre o gateway que gerencie o token. Isto inclui o ID de gateway (que é H.323 ID configurado no gateway) e a senha de gateway.

Este campo contém um dos tipos do cryptoToken definidos para o campo CryptoH323Token especificado no H.225. Atualmente, o único tipo de cryptoToken apoiado é o cryptoEPPwdHash.

Estes campos são contidos dentro do campo do cryptoEPPwdHash:

- **aliás** — O gateway aliás, que é H.323 ID do gateway.
- **timestamp** — O selo de horas atual.
- **token** — O message digest 5 (MD5)-encoded PwdCertToken. Este campo contém estes artigos:**timestamp** — O mesmos que o timestamp do cryptoEPPwdHash.**senha** — A senha do gateway.**generalID** — O mesmo gateway aliás que esse incluído no cryptoEPPwdHash.**tokenID** — A identificação de objeto.

Veja a resposta do porteiro neste exemplo.

```
*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:
```

O RRQ é rejeitado pelo porteiro. A razão para esta é porque não havia nenhum NTP ajustado na configuração do gateway. O porteiro verifica o selo de tempo do token para ver se está dentro de uma janela aceitável relativo a seu próprio tempo. Atualmente este indicador é +/- 30 segundos em torno da época UTC do porteiro.

Uma parte externa simbólica deste indicador faz com que o porteiro rejeite esta mensagem. Isto impede ataques de replay de alguém que tenta reutilizar um token snooped. Na prática, todos os gateways e porteiros precisam de usar o NTP para evitar este problema de desvio do tempo. O porteiro encontra que o selo de tempo no token está dentro da janela aceitável de seu tempo. Consequentemente, não verifica com o RAIO para autenticar o gateway.

O gateway é configurado então para o NTP que aponta ao porteiro como o mestre NTP, de modo que o gateway e o porteiro tenham o mesmo tempo. Quando isto ocorre, o gateway envia um RRQ novo e esta vez o porteiro responde de volta ao RRQ novo com um RRJ.

Estes debugam são do porteiro. Debuga a corrida para ver se o porteiro vai à fase de autenticação.

```
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"} }
```

Após ter configurado o NTP, o porteiro ainda rejeita o RRQ. Esta vez, contudo, examina o processo de autenticação para esse gateway. O porteiro rejeita o RRQ porque o usuário (aqui o ID de gateway) não é encontrado na lista de usuários válidos no RAIO. O gateway é autenticado localmente na configuração do porteiro. Na lista de usuários você vê gwa-1. Contudo, aquele não é o usuário correto desde que o usuário no RRQ é gwa-1@cisco.com.

Também, uma vez que o comando 2222 da senha 0 username gwa-1@cisco.com é configurado no porteiro, o porteiro confirma o RRQ e o gateway é registrado.

Neste laboratório, um outro gateway (gwa-2) é registrado ao mesmo porteiro (gka-1). Um atendimento é feito de gwa-1@cisco.com a gwa-2 para considerar como o ARQ, o ACF, e os mensagens setup olham.

Estes debugam recolhido são do gateway de origem e finalização (gwa-2).

- debug h225 asn1
- debugar ras

- debug voip ccapi inout

Uma explicação de algumas das mensagens debugar é incluída.

Antes que você imprimir debugar da origem/gateway de terminação, este texto explicar o fluxo de chamadas:

1. Quando um mensagem setup é recebido do PSTN, o gateway envia um ARQ a e recebe um ACF do porteiro.
2. Quando o gateway recebe o ACF, o gateway gerencie um CAT usando a senha de gateway, o H323-ID aliás, e as horas atual. O token é colocado no bloco de controle da chamada (CCB).
3. Quando o gateway envia o mensagem setup ao gateway de terminação, recupera o access token do CCB e coloca-o no campo do nonStandardParameter de um clearToken dentro do mensagem setup.
4. O gateway de terminação remove o token do mensagem setup, converte-o de um nonStandardParameter em um CAT, e coloca-o no ARQ.
5. O porteiro verifica o selo de tempo do token para ver se está dentro de uma janela aceitável relativo a seu próprio tempo. Atualmente este indicador é +/- 30 segundos em torno da época UTC do porteiro. Uma parte externa simbólica deste indicador faz com que o porteiro rejeite esta mensagem. Isto faz com que o atendimento seja rejeitado.
6. Se o token é aceitável, o porteiro formata um pacote de requisição do acesso radius, preenche os atributos apropriados para verificar um desafio da RACHADURA e envia-os a um servidor Radius.
7. Baseado na suposição que o gateway aliás está sabido no server, o server encontra a senha associada com este pseudônimo e gerencie sua própria resposta da RACHADURA usando o pseudônimo, a senha, e o desafio da RACHADURA do porteiro. Se sua resposta da RACHADURA combina esse recebido do porteiro, o server envia um acesso aceita o pacote ao porteiro. Se não combinam, ou se o gateway aliás não está no base de dados de server, o server envia um pacote da rejeição de acesso de volta ao porteiro.
8. O porteiro responde ao gateway com um ACF se recebe um acesso aceita, ou um ARJ com o código de causa **securityDenial** se recebe uma rejeição de acesso. Se o gateway recebe um ACF, o atendimento está conectado.

Este exemplo mostra debugar do gateway de origem.

Nota: O asn1 h225 debuga para a instalação não está neste exemplo desde que é o mesmo como visto no exemplo de gateway de terminação que segue o exemplo de gateway de origem.

```
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 exemplo mostra que debuga do **gateway de terminação (TGW)**. Observe que o TGW estabeleceu o segundo pé desde que obteve o ACF, e o atendimento é conectado.

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 'E0010200010415040390090A31803A983811E0285...'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)

Problema de IOS do gateway

No mesmo laboratório, a imagem IOS 12.2(6a) é carregada no OGW. Quando um atendimento é feito, observa-se que o OGW ainda envia um clear token baseado em sua senha mesmo que o gateway não seja configurado para que o IVR recolha o Account/PIN. Além, o porteiro configurado para todo o nível aceita esse atendimento. Isto é documentado na identificação de bug Cisco CSCdw43224 (clientes registrados somente).

Segurança com pontos finais alternativos

Como mencionado mais cedo neste documento, a Segurança da chamada ponta a ponta é fornecida com o uso dos accesss token que são enviados no campo dos clearTokens nas mensagens RAS/H.225. Ao permitir tal Segurança, o gateway de origem encaminha o access token recebido do porteiro em um ACF ao valor-limite de H.323 do destino no mensagem setup

H.225. Este valor-limite de H.323 do destino encaminha então o access token recebido no mensagem setup ao porteiro em seu pedido de admissão. Fazendo isso, dá ao gatekeeper remoto a capacidade para admitir os atendimentos baseados na validez do access token. Os índices do access token são até a entidade que o gerencie. A fim minimizar furos de segurança e guardá-los contra ataques que envolva pessoas, os porteiros podem codificar a informação específica do destino no access token. Isto significa que quando os alternateEndpoints são fornecidos em um ACF, o porteiro pode fornecer um access token separado para cada alternateEndpoint especificado.

Quando tenta primeiramente estabelecer uma conexão, o Cisco gateway envia o access token que recebeu no campo do clearToken do ACF com o endereço no campo dos destCallSignalAddress. Se esta tentativa é mal sucedida e o Cisco gateway continua às conexões da tentativa com um ponto final alternado, usa o access token associado (se está disponível) da lista de pontos finais alternativos. Se a lista de pontos finais alternativos recebida no ACF não inclui accesss token, mas o ACF inclui um access token, o Cisco gateway inclui este access token em todas as tentativas de conectar com um ponto final alternado.

Suporte a OSP Token

Atualmente o Open Settlement Protocol (OSP) e seus tokens são apoiados somente em Cisco gateway. Não há nenhum apoio no porteiro. O gateway reconhece os tokens OSP recebidos de um servidor de instalação e introdu-los no mensagem setup Q.931 a um gateway de terminação.

Níveis diferentes de segurança de cada ponto final ou zona

Atualmente você é incapaz de configurar níveis de segurança diferentes para cada valor-limite ou zona. O nível de segurança é para todas as zonas controladas por esse porteiro. Um pedido da característica pode ser aberto para tal edição.

Gatekeeper interdomínio para segurança do gatekeeper

O gatekeeper de interdomínio à segurança de gatekeeper fornece a capacidade para validar pedidos do porteiro-à-porteiro do intradomain e do interdomain em uma base do por-salto. Isto significa que o gatekeeper de destino termina o CAT e gerencie um novo se o porteiro decide enviar os LRQ avançados. Se o porteiro detecta uma assinatura inválida LRQ responde enviando um Location Reject (LRJ).

Porteiro do implementar à segurança de gatekeeper

O gatekeeper de origem gerencie um IZCT quando um LRQ é iniciado ou um ACF está a ponto de ser enviada em caso de um atendimento da intra-zona. Este token é atravessado através de seu caminho de roteamento. Ao longo do trajeto, cada porteiro atualiza o gatekeeper de destino ID e/ou o ID de Gatekeeper da fonte caso necessário, para refletir a informação da zona. O porteiro de terminação gerencie um token com sua senha. Este token é levado para trás nas mensagens do Location Confirmation (LCF) e passado ao OGW. O OGW inclui este token no mensagem setup H.225. Quando o TGW recebe o token, está encaminhado no ARQ answerCall e validado pelo porteiro de terminação (TGK) sem nenhuma necessidade para um servidor Radius.

O tipo do autenticação é baseado na senha com hashing como descrito em ITU H.235.

Especificamente, o método de criptografia é MD5 com hashing da senha.

A finalidade do IZCT é saber se o LRQ chegou de um domínio internacional, de que zona, e de do que portador. É usado igualmente para passar um token ao OGW no LCF do TGK. Dentro do formato IZCT, esta informação é exigida:

- srcCarrierID — Identificação de provedor de origem
- dstCarrierID — Identificação da portadora de destino
- intCarrierID — Identificação da portadora intermediária
- srcZone — Zona de origem
- dstZone — Zona de destino
- tipo de interzonaINTRA_DOMAIN_CISCOINTER_DOMAIN_CISCOINTRA_DOMAIN_TERM_NOT_CISCOINTER_DOMAIN_ORIG_NOT_CISCO

Esta característica trabalha muito bem sem nenhuma necessidade para um portador ID do gateway ou de um server do Carrier Sensitive Routing (CSR). Em tal caso, os campos sobre o portador ID estão vazios. Os exemplos aqui não incluem nenhum portador ID. Para um fluxo de chamadas detalhado, libere e suporte a plataforma, e as configurações, referem Aprimoramentos do gatekeeper para segurança interdomínios.

Configuração de gatekeeper

A característica IZCT exige esta configuração no porteiro.

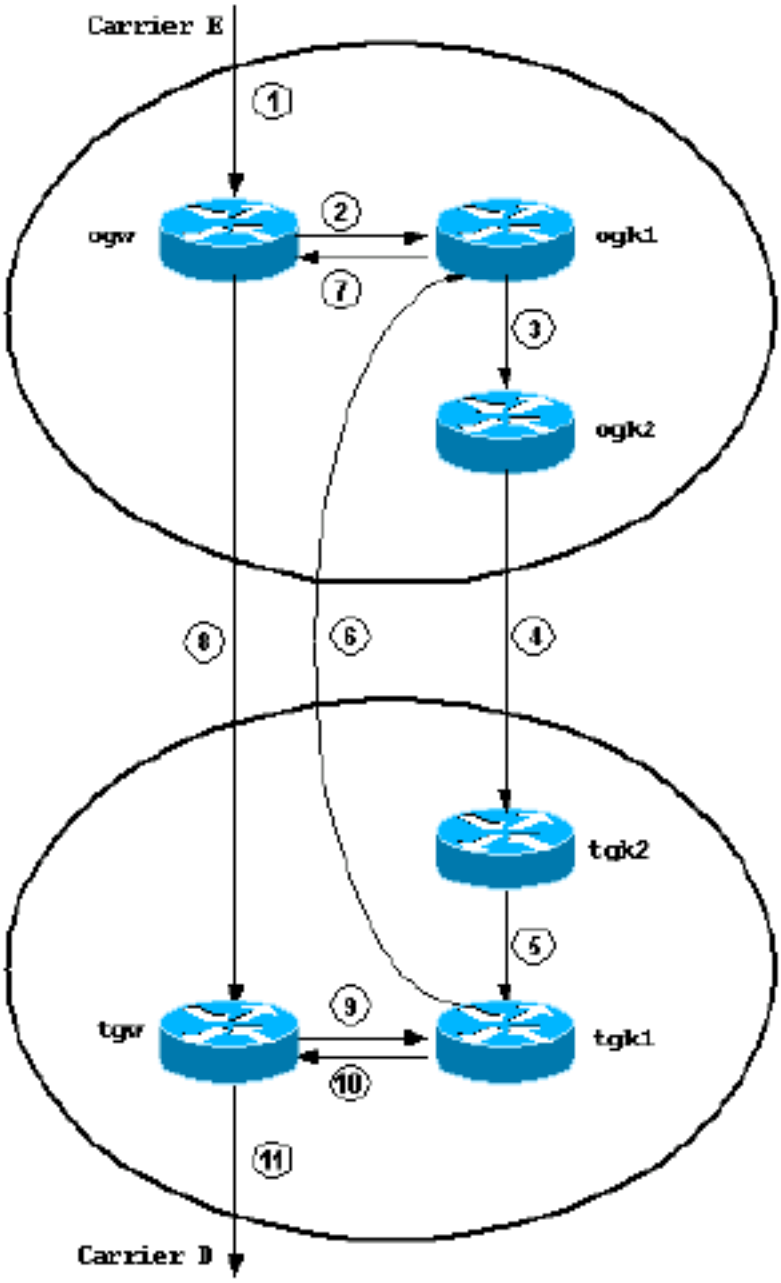
```
Router(gk-config)#[no] security izct password <PASSWORD>
```

A senha precisa de ser seis a oito caráteres. Identifique que zona está em um domínio internacional como esta:

```
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]
```

Fluxo de chamada IZCT

Este diagrama mostra o fluxo IZCT.



Nesta configuração, os nomes dos gateways e os porteiros são os mesmos que aqueles usados no diagrama de fluxo de chamadas IZCT mas com caixa baixa. O fluxo de chamadas é explicado após a configuração, com debuga explicações.

Para explicar a característica e o fluxo de chamadas IZCT, o primeiro exemplo não tem o gateway de intradomínio para a segurança do gatekeeper. Após isso, há os exemplos onde o TGW não pode gerar o IZCT de modo que o TGK1 rejeite o atendimento. Este é mostrar que a característica trabalha como projetado. Todas estas instalações são baseadas na topologia no diagrama de fluxo de chamadas IZCT.

Exemplo 1: Fluxo de chamadas para o porteiro à segurança de gatekeeper somente

Este exemplo mostra as configurações relacionadas de todos os gateways e porteiros.

Configuração de OGW	Configuração TGW
! hostname ogw !controller E1 3/0	hostname tgw ! controller E1 0

```
  pri-group timeslots 1-
2,16
!
interface Ethernet0/0
 ip address 172.16.13.15
255.255.255.224
 half-duplex
 h323-gateway voip
interface
 h323-gateway voip id ogk1
ipaddr
172.16.13.35 1718
 h323-gateway voip h323-id
ogw
 h323-gateway voip tech-
prefix 1#
!
voice-port 3/0:15
!
dial-peer voice 5336 pots
 incoming called-number .
 destination-pattern 5336
 direct-inward-dial
 port 3/0:15
 prefix 21
!
dial-peer voice 3653 voip
 incoming called-number .
 destination-pattern 3653
 session target ras
 dtmf-relay h245-
alphanumeric
 codec g711ulaw
!
gateway
!
ntp clock-period 17178791
ntp server 172.16.13.35
end
```

```
 clock source line primary
 ds0-group 0 timeslots 1-2
type
r2-digital r2-compelled
!
interface Ethernet0
 ip address 172.16.13.23
255.255.255.224
 h323-gateway voip interface
 h323-gateway voip id tgk1
ipaddr
172.16.13.41 1718
 h323-gateway voip h323-id
tgw
 h323-gateway voip tech-
prefix 2#
!
voice-port 0:0
 compand-type a-law
!
dial-peer voice 3653 pots
 application test1
 incoming called-number .
 destination-pattern 3653
 port 0:0
 prefix 21
!
dial-peer voice 5336 voip
 incoming called-number .
 destination-pattern 5336
 session target ras
 dtmf-relay h245-
alphanumeric
 codec g711ulaw
!
gateway
!
ntp clock-period 17179814
ntp server 172.16.13.35
end
```

Configuração OGK1	Configuração TGK1
!	!
hostname ogk1	hostname tgk1
!	!
interface Ethernet0/0	interface Ethernet0/0
ip address 172.16.13.35 255.255.255.224	ip address 172.16.13.41 255.255.255.224
half-duplex	ip directed-broadcast
!	half-duplex
gatekeeper	!
zone local ogk1 domainA.com 172.16.13.35	gatekeeper
zone remote ogk2 domainA.com 172.16.13.14 1719	zone local tgk1 domainB.com 172.16.13.41
zone prefix ogk2 36*	zone remote tgk2 domainB.com 172.16.13.16 1719
zone prefix ogk1 53*	zone prefix tgk1 36*
security izct password 111222	zone prefix tgk2 53*
gw-type-prefix 1#* default-technology	security izct password 111222
	gw-type-prefix 2#* default-technology
	no shutdown

no shutdown ! ! no scheduler max-task-time no scheduler allocate ntp master ! end	 ! ntp clock-period 17179797 ntp server 172.16.13.35 ! end

Configuração OGK2	Configuração TGK2
! hostname ogk2 ! interface Ethernet0/0 ip address 172.16.13.14 255.255.255.224 full-duplex ! gatekeeper zone local ogk2 domainA.com zone remote ogk1 domainA.com 172.16.13.35 1719 zone remote tgk2 domainB.com 172.16.13.16 1719 foreign- domain zone prefix tgk2 36* zone prefix ogk1 53* security izct password 111222 lrq forward-queries no shutdown ! ntp clock-period 17208242 ntp server 172.16.13.35 ! end	! hostname tgk2 ! interface Ethernet0/0 ip address 172.16.13.16 255.255.255.224 half-duplex ! gatekeeper zone local tgk2 domainB.com zone remote tgk1 domainB.com 172.16.13.41 1719 zone remote ogk2 domainA.com 172.16.13.14 1719 foreign- domain zone prefix tgk1 36* zone prefix ogk2 53* security izct password 111222 lrq forward-queries no shutdown ! ntp clock-period 17179209 ntp server 172.16.13.35 ! end

Fluxo de chamada com depurações

Estes exemplos usam debugam para explicar o fluxo de chamadas.

1. Um usuário no portador E chama um usuário no 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. Desde que o dialpeer do gateway de origem (tag=3653) é configurado para o RAS, envia um ARQ ao 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. Quando o OGK1 recebe o ARQ, determina que o destino está prestado serviços de manutenção pela zona remota OGK2. Identifica então que um IZCT está precisado (com o CLI: **<pwd> da senha do izct da Segurança**). O OGK1 continua criar o IZCT antes que o LRQ esteja enviado. Envia então o IZCT e o LRQ para fora ao OGK2 e envia uma mensagem do RASGO de volta ao 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
'0700C06F676B31047372630464737404696E74'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. Quando o OGK2 recebe o LRQ, verifica o IZCT. Da configuração encontra que os neets LRQ para conter igualmente um IZCT. O OGK2 cria então um IZCT novo mudando o izctSrcZone e o izctDstZone para ser ogk2 e para a frente o LRQ ao TGK2. Depois que manda o LRQ ao TGK2, envia para trás uma mensagem do RASGO ao OGK1.Se os porteiros são parte de um conjunto, o nome de grânulos está usado para o SrcZone ou o 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. O TGK2 determina que o LRQ vem de um domínio internacional. Atualiza o dstZone do IZCT com seu próprio ID e o interZoneType como INTER_DOMAIN_CISCO. Então cria um CAT novo e passa o IZCT actualizado e o LRQ ao TGK1.O TGK2 trata a zona de que um LRQ é recebido como uma zona do domínio internacional em qualquer uma destas duas encenações:A lista de zona remota do TGK2 não contém a zona de que um LRQ é recebido.A lista de zona remota do TGK2 contém a zona de que um LRQ é recebido. A zona é identificada por meio de uma bandeira do domínio internacional.Envia então a um pedido a mensagem em andamento de volta ao 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 172.16.13.4**1

6. Normalmente o TGK1 atualiza o dstCarrierID do IZCT ao portador E, que é determinado pelo processo de roteamento. Contudo, desde que nenhum portador é usado, você não vê aquele. O TGK1 gerencie um token da mistura com a senha do IZCT. Envia um LCF com o IZCT actualizado nele ao OGK1. Este izctHash é usado para autenticar a chamada de resposta ARQ que o TGK1 recebe do TGW quando o mais atrasado recebe o mensagem setup de VoIP do 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 '0001400200740067007706007400670068003101...'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. O OGK1 extrai o IZCT do LCF e envia-o em um ACF ao 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. O OGW envia o IZCT ao TGW no mensagem 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, '4000000604010040C6013801114000100AC100D0F...'H } mediaWaitForConnect FALSE canOverlapSend FALSE } h245Tunneling TRUE nonStandardControl { { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '6001020001041F04038090A31803A983816C0C21...'H } } } }
9. O TGW passa o IZCT ao TGK1 em um ARQ answerCall.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. O TGK1 autentica o destino IZCT com sucesso. Isto é porque o TGK1 gerencie a mistura no IZCT e envia para trás um ACF ao 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. O TGW estabelece o atendimento para o portador D depois que recebe o ACF.**Exemplo 2: O atendimento falhou porque o TGW não pode extrair o IZCT do mensagem de configuração recebida.**Este exemplo é baseado na mesmas topologia e configuração que o exemplo 1. Neste exemplo, o software do TGW é mudado a uma versão onde o IZCT não seja apoiado. Em tal caso, o TGW não pode extrair o IZCT do mensagem setup. Isto faz com que o TGK1 rejeite o atendimento com um motivo de desconexão da recusa de segurança.Este exemplo mostra somente o mensagem setup, o ARQ, e o ARJ no TGW desde que o fluxo de chamadas é o mesmo que o exemplo 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
```

Informações Relacionadas

- Aprimoramentos do gatekeeper para segurança interdomínios
- Contabilidade e aprimoramentos de segurança de Cisco H.235 para Cisco gateway
- Referência do comando Debug do Cisco IOS, Versão 12.3
- Suporte à Tecnologia de Voz
- Suporte ao Produto de Voz e Comunicações Unificadas
- Troubleshooting da Telefonia IP Cisco
- Suporte Técnico e Documentação - Cisco Systems