Guide de dépannage de sécurité Passerelle à Contrôleur d'accès (H.235) et Contrôleur d'accès à Contrôleur d'accès (IZCT)

Contenu

Introduction

H.323 les réseaux ont différents genres de configurations et l'appel circule. Ce document discute la plupart des problèmes de sécurité avec H.323 les réseaux qui font participer des garde-portes. Ce document récapitule la manière des travaux de chaque caractéristique et comment la dépanner avec une explication sur les la plupart de met au point. Ce document n'adresse pas la Sécurité globale du VoIP.

Ce document couvre ces caractéristiques :

- **Passerelle d'Intradomain à la Sécurité de garde-porte** — Cette Sécurité est basée sur H.235, dans lequel des appels sont authentifiés, autorisés, et H.323 conduits par un garde-porte. Le garde-porte est considéré une entité dans une certaine mesure connue et de confiance que la passerelle ne l'authentifie pas quand les essais de passerelle pour s'inscrire à elle.
- **Garde-porte d'Interdomain à la Sécurité de garde-porte** — Cette Sécurité couvre authentifier et autoriser H.323 des appels entre les domaines administratifs des fournisseurs de services

de téléphone d'Internet (ITSPs) utilisant le **jeton d'espace libre d'InterZone (IZCT)**. Ce document couvre seulement la partie où le garde-porte de terminaison envoie un jeton dans son message de la confirmation d'emplacement (LCF) de sorte qu'il authentifie la demande d'admission d'answerCall (ARQ). La validation de la demande d'emplacement (LRQ) n'est pas incluse dans cette caractéristique. La validation LRQ est une caractéristique programmée pour une future version logicielle de Cisco IOS®.

Définitions

Acronyme	Définition
ARQ	Demande d'admission — Un enregistrement, une admission, et un message de Protocol d'état (RAS) envoyé H.323 d'un point final à un garde-porte qui invite une admission pour établir un appel.
ACF	L'admission confirment — Un message RAS envoyé du garde-porte au point final qui confirme l'acceptation d'un appel.
ARJ	Rejet d'admission — Un message RAS du garde-porte au point final qui rejette la demande d'admission.
CAT	Cisco accèdent au jeton — Les H.235 effacent le jeton.
CHAP	Authentification Protocol à échanges confirmés — Un protocole d'authentification où un défi est utilisé.
GCF	Le garde-porte confirment — Un message RAS envoyé d'un garde-porte H.323 au point final qui confirme la détection du garde-porte.
GRQ	Demande de garde-porte — Un message RAS envoyé H.323 d'un point final pour découvrir le garde-porte.
H.235	Recommandation ITU pour la Sécurité et cryptage pour des terminaux de multimédia de H-gamme (H.323 et l'autre H.245-based).
IZCT	Jeton d'espace libre d'InterZone — Un IZCT est généré dans le garde-porte de commencement quand un LRQ est initié ou un ACF est sur le point d'être envoyé pour un appel d'intrazone dans le domaine administratif ITSP.
LRQ	Demande d'emplacement — Un message RAS envoyé d'un garde-porte au prochain garde-porte de saut ou tronçon d'appel pour tracer et conduire l'appel.
RAS	Enregistrement, admission, et état — Un protocole qui permet à un garde-porte pour exécuter l'enregistrement, l'admission, et les contrôles d'état du point final.
RCF	L'enregistrement confirment — Un message RAS envoyé du garde-porte au point final qui confirme

	l'enregistrement.
RRJ	Anomalie d'enregistrement — Un message RAS envoyé du garde-porte qui rejette la demande d'enregistrement.
RRQ	Demande d'enregistrement — Un message RAS envoyé du point final au garde-porte qui demande de s'inscrire à lui.
RIP	Demande en cours — Un message RAS envoyé d'un garde-porte à l'expéditeur qui énonce l'appel est en cours.

Passerelle d'Intradomain à la Sécurité de garde-porte

Est H.323 une recommandation ITU qui adresse sécuriser la communication en temps réel au-dessus des réseaux non sécurisés. Ceci implique deux sujets de préoccupation importants : authentification et intimité. Il y a deux types d'authentification, selon H.235 :

- Authentification basée sur cryptage symétrique qui n'exige aucun contact antérieur entre les entités de communication.
- Basé sur la capacité d'avoir un certain secret partagé antérieur (encore référencé comme abonnement basé), deux formes d'authentification payante sont fournies :mot de passecertificat

Groupe date/heure passé dans les jetons

Un groupe date/heure est utilisé pour empêcher des attaques par relecture. Par conséquent, il est nécessaire pour la référence acceptable a mutuellement - au temps (de quel pour dériver des groupes date/heure). La quantité de distorsion acceptable de temps est une question d'implémentation locale.

Comment Cisco implémente la recommandation H.235

Cisco utilise un modèle d'authentification comme de protocole d'authentification CHAP (Challenge Handshake Authentication Protocol) comme base pour sa passerelle à son implémentation du garde-porte H.235. Ceci te permet pour accroître l'Authentification, autorisation et comptabilité (AAA), utilisant la fonctionnalité existante pour exécuter l'authentification réelle. Il signifie également que le garde-porte n'est pas requis d'avoir accès à une base de données des id de passerelle, compte utilisateur numérote, des mots de passe, et des broches. Le schéma est basé sur H.235, la section 10.3.3. Il est décrit en tant que mot de passe payant avec le hachage.

Cependant, au lieu d'utiliser les cryptoTokens H.225, cette méthode utilise les clearTokens H.235 avec des champs remplis convenablement pour l'usage avec le RAYON. Ce jeton désigné sous le nom du jeton de Cisco Access (CAT). Vous pouvez toujours exécuter l'authentification localement sur le garde-porte au lieu d'utiliser un serveur de RAYON.

L'utilisation des cryptoTokens exige que le garde-porte met à jour ou a une certaine manière de saisir des mots de passe pour tous les utilisateurs et passerelles. C'est parce que le champ symbolique du cryptoToken est spécifié tels que l'entité authentifiante a besoin du mot de passe pour générer son propre jeton contre lequel pour comparer reçu.

Les garde-portes de Cisco ignorent complètement des cryptoTokens. Cependant, garde-portes de vocalTek et d'autres qui prennent en charge H.235 l'utilisation de la section 10.3.3 les cryptoTokens d'authentifier la passerelle. Les garde-portes de Cisco emploient le CAT pour authentifier la passerelle. Puisque la passerelle ne connaît pas à quel type de garde-porte il parle, il envoie chacun des deux dans le RRQ. L'authenticationCapability dans le GRQ sont pour le cryptoToken et indiquent que le hachage de MD5 est le mécanisme d'authentification (bien que le CAT utilise également le MD5).

Référez-vous au pour en savoir plus d'améliorations de Sécurité et de comptabilité de passerelle H.323 de Cisco.

Comment configurer les niveaux de Sécurité

- Sécurité de point final ou niveau de l'enregistrementAvec la sécurité activée d'enregistrement sur le garde-porte, la passerelle est exigée pour inclure un CAT dans tous les messages du poids lourd RRQ. Le CAT, dans ce cas, contient les informations qui authentifient la passerelle elle-même au garde-porte. Le garde-porte formate un message à un serveur de RAYON qui authentifie les informations contenues dans le jeton. Il répond de nouveau au garde-porte avec un Access-recevoir ou l'Access-anomalie. Ceci, consécutivement, répond à la passerelle avec un RCF ou un RRJ.Si un appel est alors placé d'une passerelle avec succès authentifiée, cette passerelle génère un nouveau CAT dès réception d'un ACF du garde-porte utilisant le mot de passe de la passerelle. Ce CAT est identique à celui généré pendant l'enregistrement excepté le groupe date/heure. Il est placé dans le message de configuration sortant. La passerelle de destination extrait le jeton du message de configuration et le place dans le côté ARQ de destination. Le garde-porte emploie le RAYON pour authentifier la passerelle d'origine avant qu'elle envoie le côté ACF de destination. Ceci empêche un point final non-authentifié qui connaît l'adresse d'une passerelle de l'employer pour éviter le modèle de sécurité et d'accéder au réseau téléphonique public commuté (PSTN).Par conséquent, dans ce niveau, il n'y a aucun besoin de n'inclure aucun jeton dans l'ARQs d'origine.Tapez **[non] le jeton de Sécurité exiger-pour l'enregistrement de l'**interface de ligne de commande de garde-porte (CLI) pour configurer le garde-porte. *L'aucune* option de la commande ne fait ne vérifier plus le garde-porte des jetons dans des messages RAS.Tapez **[non] le point final de niveau du mot de passe <PASSWORD> de Sécurité de la** passerelle CLI pour configurer la passerelle. *L'aucune* option de la commande ne fait ne générer plus la passerelle des jetons pour des messages RAS.
- Sécurité de niveau de Par-appelconstructions de Sécurité de Par-appel sur la Sécurité niveau de l'enregistrement. En plus des exigences de sécurité d'enregistrement de téléconférence, une passerelle est également exigée pour inclure des jetons d'Access dans tous les messages du côté d'origine ARQ quand la Sécurité de Par-appel est activée sur le garde-porte. Le jeton contient dans ce cas les informations qui identifient l'utilisateur de la passerelle au garde-porte. Ces informations sont obtenues utilisant un script de la réponse vocale interactive (RVI) sur la passerelle. Ceci incite des utilisateurs à écrire leur user-id et PIN du pavé numérique avant qu'ils placent un appel.Le CAT contenu dans l'ARQ d'origine est authentifié par le RAYON de la même manière que décrit plus tôt dans la Sécurité de point final ou niveau de l'enregistrement. Après qu'il reçoive l'ACF, la passerelle génère un nouveau CAT utilisant son mot de passe et l'envoie dans le message de configuration H.225 à la dernière passerelle.Tapez **[non] le jeton de Sécurité exiger-pour tous du** garde-porte CLI pour configurer le garde-porte. *L'aucune* option de la commande ne fait ne vérifier plus le garde-

porte des jetons dans des messages RAS.Tapez **[non] le par-appel de niveau du mot de passe <PASSWORD> de Sécurité de la** passerelle CLI pour configurer la passerelle. *L'aucune* option de la commande ne fait ne générer plus la passerelle des jetons pour des messages RAS.

- Toute la Sécurité de niveauCeci permet à la passerelle pour inclure un CAT dans tous les messages RAS requis pour l'enregistrement et pour des appels. Par conséquent, c'est une combinaison des deux niveaux ci-dessus. Avec cette option, la validation du CAT dans des messages ARQ est basée sur le numéro de compte et le PIN de l'utilisateur qui fait un appel. La validation du CAT introduit tous les autres messages RAS est basée sur le mot de passe configuré pour la passerelle. Par conséquent, il est semblable au niveau de Par-appel.Tapez **[non] le jeton de Sécurité exiger-pour tous du** garde-porte CLI pour configurer le garde-porte. *L'aucune* option de la commande ne fait ne vérifier plus le garde-porte des jetons dans des messages RAS.Tapez **[non] le niveau tout du mot de passe <PASSWORD> de Sécurité de la** passerelle CLI pour configurer la passerelle. *L'aucune* option de la commande ne fait ne générer plus la passerelle des jetons pour des messages RAS.

Utilisation H.235 à un niveau de Par-appel sans RVI

H.235 ne peut pas être utilisé à un niveau de par-appel sans RVI. S'il n'y a aucun RVI pour collecter un compte et un PIN, la passerelle doit envoyer l'ARQ sans jeton clair (mais avec un crypto jeton). Puisque le garde-porte de Cisco reçoit seulement les jetons clairs, l'appel est rejeté par le garde-porte avec une raison de refus de Sécurité.

Cet exemple affiche que le **h225 asn1** met au point collecté d'une **passerelle d'origine (OGW)** qui n'est pas configurée pour qu'un RVI collecte le compte et le PIN. Le message RRQ a un jeton clair, mais l'ARQ ne fait pas. Un message ARJ est renvoyé à la passerelle.

```
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
```

Référez-vous à la section de tâches de configuration de comptabilité et d'améliorations de la sécurité de Cisco H.235 pour des passerelles Cisco pour plus d'informations sur la configuration RVI.

Problèmes cruciaux

Les problèmes cruciaux que vous devez être concerné par incluent :

- Configuration de la passerelle et du garde-porte
- Configuration RADIUS sur le garde-porte et le serveur de RAYON
- Protocole NTP (Network Time Protocol) — Vous devez avoir le même temps sur tous les passerelles et garde-portes. Puisque les informations d'authentification incluent un horodateur, il est important que toutes les Passerelles H.323 de Cisco et garde-portes (ou toute autre entité qui exécute l'authentification) soient synchronisés. Les Passerelles H.323 de Cisco doivent être synchronisées utilisant le NTP.
- Panne de logiciel due à une bogue

Debugs et écoulement d'appel pour les différents niveaux

Puisque toute la Sécurité de niveau couvre des cas d'enregistrement et de par-appel, le laboratoire est installé avec ce niveau de sécurité pour cet exercice. Les écoulements d'appel pour la partie enregistrement et un appel normal VoIP sont expliqués dans la configuration ici.

Remarque: La configuration ici n'est pas complète. Plus de commandes suivent entre les sorties de débogage. On le conçoit pour afficher quel problème peut se produire si vous ne vérifiez pas

toutes les choses telles que la configuration, le NTP, et le RAYON. En outre, la passerelle est authentifiée sur le garde-porte localement de sorte que vous puissiez voir que ce qui évalue sont placés pour l'ID et le mot de passe de passerelle. Cette configuration est coupée d'un coup de ciseaux de sorte que seulement la configuration relative soit affichée.

```
!
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
```

Ceux-ci met au point sont activés dans cet exemple :

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

La première chose qui se produit est que la passerelle envoie un GRQ au garde-porte et au garde-porte envoie un GCF à la passerelle. La passerelle alors envoie un RRQ et attend un RCF ou RRJ.

Dans la configuration précédente, la passerelle n'est placée pour aucun niveau de sécurité de sorte que son GRQ ne porte aucun **authenticationCapability** qui est nécessaire pour les jetons. Cependant, le garde-porte renvoie toujours un GCF pendant que cette sortie affiche :

```
*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 } } }
```

Ceci explique certains des messages dans le GRQ :

- **authenticationCapability** — Ce champ a seulement la valeur du pwdHash. Il indique que le hachage de MD5 est le mécanisme d'authentification.
- **algorithmOIDs** — L'identification d'objet d'algorithme dans ce cas il porte la valeur (1 2 840 113549 2 5) qui est l'object id pour le hachage de Message Digest 5.(1 2 840 113549 2 5) se traduisent iso(1) au membre-body(2) US(840) rsadsi(113549) digestAlgorithm(2) md5(5). Par conséquent Cisco fait le hachage de mot de passe de MD5.Rsadsi signifie l'identifiant d'objet de l'interconnexion de systèmes ouverts (OSI) d'Inc. RSA Data Security, Inc.'s de protection des données RSA a 1.2.840.113549 ans (0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d dans l'hexa), comme enregistré par l'American National Standards Institute (ANSI).

Le garde-porte envoie de nouveau un GCF comme message précédent. La passerelle envoie alors un RRQ avec certains champs pour décrire les jetons qu'il l'utilise pour être authentifié. Le message **asn1 RRQ** qui est envoyé est affiché dans cet exemple.

```
*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 }
```

Avant que la réponse soit discutée, certains des champs relatifs dans le message ci-dessus RRQ sont expliqués ici. Il y a deux types de jetons : un jeton ou un CAT clair) et un crypto jeton.

Comme indiqué précédemment, les garde-portes de Cisco ignorent les cryptos jetons. Cependant, la passerelle envoie toujours chacun des deux parce qu'elle ne sait pas à quel type de garde-porte elle parle. Puisque d'autres constructeurs pourraient utiliser de cryptos jetons, la passerelle envoie chacun des deux.

Ceci explique la syntaxe de ClearToken.

- **tokenOID** — L'object id pour identifier le jeton.
- **horodateur** — Le courant a coordonné le temps du temps universel (UTC) de la passerelle. Secondes depuis 00:00 1/1/1970 UTC.Il est utilisé comme CHAP-défi implicite comme si il était au commencement provenu le garde-porte.
- **défi** — Un condensé de message du MD5 16-byte généré par la passerelle utilisant ces champs :défi = [aléatoire + mot de passe + horodateur GW/User] informations parasites de MD5Le RAYON exécute ce calcul (puisqu'il connaît le nombre aléatoire, le mot de passe de la passerelle, et le défi de CHAP) pour déterminer ce qu'être le défi devrait : Réponse de CHAP = [ID de CHAP + UserPassword + défi de CHAP] informations parasites de MD5
- **aléatoire** — Une valeur de à un octet utilisée par le RAYON pour identifier cette demande particulière.La passerelle incrémente un module variable de 256 pour que chaque demande d'authentification réponde à cette exigence de RAYON.
- **generalID** — La passerelle H323-ID ou le numéro en fonction de compte utilisateur sur le niveau de sécurité et le type de message RAS.

Remarque: Tous ces champs sont importants. Cependant, plus d'attention est accordée au groupe date/heure et au generalID. Dans ce cas, le groupe date/heure est **731030839** et le generalID est **gwa-1@cisco.com**.

Le cryptoToken dans le RRQ contient des informations sur la passerelle qui génère le jeton. Ceci inclut l'ID de passerelle (qui est H.323 l'ID configuré sur la passerelle) et le mot de passe de la passerelle.

Ce champ contient un des types de cryptoToken définis pour le champ CryptoH323Token spécifié dans H.225. Actuellement, le seul type de cryptoToken pris en charge est le cryptoEPPwdHash.

Ces champs sont contenus dans le champ de cryptoEPPwdHash :

- **alias** — La passerelle alias, qui est H.323 l'ID de la passerelle.
- **horodateur** — Le groupe date/heure en cours.
- **jeton** — Le Message Digest 5 (MD5)-encoded PwdCertToken. Ce champ contient ces éléments :**horodateur** — Les mêmes que l'horodateur du cryptoEPPwdHash.**mot de passe** — Le mot de passe de la passerelle.**generalID** — La même passerelle alias que celle incluse dans le cryptoEPPwdHash.**tokenID** — L'object id.

Visualisez la réponse du garde-porte dans cet exemple.

```
*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:

Le RRQ est rejeté par le garde-porte. La raison pour ceci est parce qu'il n'y avait aucun NTP réglé dans la configuration de la passerelle. Le garde-porte vérifie le groupe date/heure du jeton pour voir s'il est dans une fenêtre acceptable relativement à son propre temps. Actuellement cette fenêtre est de +/- 30 secondes autour de la période UTC du garde-porte.

Un extérieur symbolique de cette fenêtre fait jeter le garde-porte ce message. Ceci empêche des attaques par relecture de quelqu'un qui essaye de réutiliser un jeton pillé. Dans la pratique, tous les passerelles et garde-portes doivent employer le NTP pour éviter ce problème oblique de fois. Les découvertes de garde-porte que le groupe date/heure dans le jeton est dans la fenêtre acceptable de son temps. Par conséquent, il ne vérifie pas avec le RAYON pour authentifier la passerelle.

La passerelle est alors configurée pour le NTP indiquant le garde-porte comme ntp master, de sorte que la passerelle et le garde-porte aient le même temps. Quand ceci se produit, la passerelle envoie un nouveau RRQ et cette fois le garde-porte répond de nouveau au nouveau RRQ avec un RRJ.

Ceux-ci met au point sont du garde-porte. Débugge le passage pour voir si le garde-porte va à la phase d'authentification.

```
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"} }
```

Après avoir configuré le NTP, le garde-porte rejette toujours le RRQ. Cette fois, cependant, il passe par la procédure d'authentification pour cette passerelle. Le garde-porte rejette le RRQ parce que l'utilisateur (ici l'ID de passerelle) n'est pas trouvé sur la liste d'utilisateurs valides sur le

RAYON. La passerelle est authentifiée localement dans la configuration du garde-porte. Sur la liste des utilisateurs vous voyez gwa-1. Cependant, ce n'est pas l'utilisateur droit puisque l'utilisateur dans le RRQ est gwa-1@cisco.com.

En outre, une fois que la commande 2222 du mot de passe 0 de gwa-1@cisco.com de nom d'utilisateur est configurée sur le garde-porte, le garde-porte confirme le RRQ et la passerelle est enregistrée.

Dans ce laboratoire, une autre passerelle (gwa-2) est enregistrée au même garde-porte (gka-1). Un appel est fait à partir de gwa-1@cisco.com à gwa-2 pour voir à quoi les ARQ, les ACF, et les messages de configuration ressemblent.

Ceux-ci met au point collecté sont du commencement et de la dernière passerelle (gwa-2).

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

Une explication de certains des messages de débogage est incluse.

Avant que vous imprimiez mette au point du commencement/dernière passerelle, ce texte explique l'écoulement d'appel :

1. Quand un message de configuration est reçu du PSTN, la passerelle envoie un ARQ à et reçoit un ACF du garde-porte.
2. Quand la passerelle reçoit l'ACF, la passerelle génère un CAT utilisant le mot de passe de la passerelle, le H323-ID alias, et le temps en cours. Le jeton est placé dans le bloc de Contrôle d'appel (CCB).
3. Quand la passerelle envoie le message de configuration à la dernière passerelle, elle récupère le jeton d'accès du CCB et le place dans le domaine de nonStandardParameter d'un clearToken dans le message de configuration.
4. La dernière passerelle retire le jeton du message de configuration, le convertit d'un nonStandardParameter en CAT, et le place dans l'ARQ.
5. Le garde-porte vérifie le groupe date/heure du jeton pour voir s'il est dans une fenêtre acceptable relativement à son propre temps. Actuellement cette fenêtre est de +/- 30 secondes autour de la période UTC du garde-porte. Un extérieur symbolique de cette fenêtre fait jeter le garde-porte ce message. Ceci cause l'appel d'être rejeté.
6. Si le jeton est acceptable, le garde-porte formate un paquet de demandes d'accès de RAYON, complète les attributs appropriés pour vérifier un défi de CHAP et les envoie à un serveur de RAYON.
7. Fondé sur l'hypothèse que la passerelle alias est connue au serveur, le serveur localise le mot de passe associé avec ce pseudonyme et génère sa propre réponse de CHAP utilisant le pseudonyme, le mot de passe, et le défi de CHAP du garde-porte. Si sa réponse de CHAP apparie celui reçu du garde-porte, le serveur envoie Access reçoivent le paquet au garde-porte. S'ils ne s'assortissent pas, ou si la passerelle alias n'est pas dans la base de données de serveur, le serveur envoie un paquet d'anomalie d'Access de nouveau au garde-porte.
8. Le garde-porte répond à la passerelle avec un ACF s'il reçoit Access reçoivent, ou un ARJ avec code de cause **securityDenial** s'il reçoit une anomalie d'Access. Si la passerelle reçoit un ACF, l'appel est connecté.

Cet exemple affiche le débogage de la passerelle d'origine.

Remarque: Le h225 asn1 met au point pour l'installation n'est pas dans cet exemple puisque c'est pareil que vu dans l'exemple de dernière passerelle qui suit l'exemple de passerelle d'origine.

```
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
```

Cet exemple affiche que met au point de la **dernière passerelle (TGW)**. Notez que le TGW a installé le deuxième tronçon depuis qu'il a obtenu ACF, et l'appel est connecté.

```
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,
'4000000604010040C6013801114000100AC100D0F...'H } mediaWaitForConnect FALSE canOverlapSend FALSE
} h245Tunneling TRUE nonStandardControl { { nonStandardIdentifier h221NonStandard : {
```

t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data
'E0010200010415040390 90A31803A983811E0285...'H } } } } RAW_BUFFER::= E0 01020001 04150403
9090A318 03A98381 1E028583 70058133 36353302 80060004 00000003 Mar 2 19:39:07.509: PDU DATA =
6147F378 value H323_UU_NonStdInfo ::= { version 2 protoParam qsigNonStdInfo : { iei 4 rawMesg
'04039090A31803A983811E028583700581333635...'H } progIndParam progIndIEinfo : { progIndIE
'00000003'H } } PDU DATA = 6147F378 value ARQnonStandardInfo ::= { sourceAlias { }
sourceExtAlias { } } RAW_BUFFER::= 00 0000 Mar 2 19:39:07.517: RAW_BUFFER::= 61 000100C0
2B93B7DA 08003200 32003200 3200000A 00670077 0061002D 00320000 Mar 2 19:39:07.517: PDU DATA =
6147C2BC value RasMessage ::= **admissionRequest** : *!--- An answer ARQ is sent to the gatekeeper to authenticate the caller.* **{ requestSeqNum 22** callType pointToPoint : NULL callModel direct : NULL
endpointIdentifier {"81F5989C00000002"} destinationInfo { e164 : "2#3653" } srcInfo { e164 :
"1#5336" } srcCallSignalAddress ipAddress : { ip 'AC100D0F'H port 11032 } bandWidth 640
callReferenceValue 2 nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode
181 t35Extension 0 manufacturerCode 18 } data '000000'H } conferenceID
'6AEF3A87165C11CC8040D661B74F9390'H activeMC FALSE answerCall TRUE canMapAlias FALSE
callIdentifier { guid '6AEF3A87165C11CC8041D661B74F9390'H } **tokens** *!--- CAT is included.* **{ { tokenOID { 0 4 0 1321 1 2 } timeStamp 731101147 challenge 'AFBAAFDF79446B9D8CE164DB8C111A87'H random 247 generalID {"gwa-1@cisco.com"} } } cryptoTokens { cryptoEPPwdHash : { alias h323-ID : {"gwa-2"} timeStamp 731101147 token { algorithmOID { 1 2 840 113549 2 5 } paramS { } hash "8479E7DE63AC17C6A46E9E19659568" } }** } willSupplyUUIEs FALSE } RAW_BUFFER::= 27 98001500
F0003800 31004600 35003900 38003900 43003000 30003000 30003000 30003000 32010280 50698601
02804086 6900AC10 0D0F2B18 40028000 0240B500 00120300 00006AEF 3A87165C 11CC8040 D661B74F
939044E3 20010011 006AEF3A 87165C11 CC8041D6 61B74F93 9044014D 00060400 8A290102 C02B93B7
DA10AFBA AFDF7944 6B9D8CE1 64DB8C11 1A870200 F71E0067 00770061 002D0031 00400063 00690073
0063006F 002E0063 006F006D 00002E01 04040067 00770061 002D0032 C02B93B7 DA082A86 4886F70D
02050080 808479E7 0DE63AC1 7C6A46E9 E1965905 680100 Mar 2 19:39:07.533: h323chan_dgram_send:Sent
UDP msg. Bytes sent: 228 to 172.16.13.35:1719 Mar 2 19:39:07.533: RASLib::GW_RASSendARQ: ARQ
(seq# 22) sent to 172.16.13.35 Mar 2 19:39:07.549: h323chan_dgram_recvdata:rcvd from
[172.16.13.35:1719] on sock[1] RAW_BUFFER::= 2B 00001540 028000AC 100D1706 B800EF1A 00C00100
020000 Mar 2 19:39:07.549: PDU DATA = 6147C2BC value RasMessage ::= **admissionConfirm** : *!--- ACF is received from the gatekeeper.* **{ requestSeqNum 22** bandWidth 640 callModel direct : NULL
destCallSignalAddress ipAddress : { ip 'AC100D17'H port 1720 } irrFrequency 240 willRespondToIRR
FALSE uuiesRequested { setup FALSE callProceeding FALSE connect FALSE alerting FALSE information
FALSE releaseComplete FALSE facility FALSE progress FALSE empty FALSE } } Mar 2 19:39:07.553:
ACF (seq# 22) rcvd Mar 2 19:39:07.553: **cc_api_call_setup_ind** (vdbPtr=0x61BC92EC,
callInfo={called=2#3653,called_oct3=0x81,calling=1#5336,calling_oct3=0x81,
calling_oct3a=0x0,subscriber_type_str=Unknown, fdest=1 peer_tag=5336,
prog_ind=3},callID=0x6217CC64) Mar 2 19:39:07.553: cc_api_call_setup_ind type 0 , prot 1 Mar 2
19:39:07.553: cc_api_call_setup_ind (vdbPtr=0x61BC92EC, callInfo={called=2#3653, calling=1#5336,
fdest=1 peer_tag=5336}, callID=0x6217CC64) Mar 2 19:39:07.553: cc_process_call_setup_ind
(event=0x61E1EAFC) Mar 2 19:39:07.553: >>>>CCAPI handed cid 9 with tag 5336 to app "DEFAULT" Mar
2 19:39:07.553: sess_appl: ev(25=CC_EV_CALL_SETUP_IND), cid(9), disp(0) Mar 2 19:39:07.553:
sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(9), disp(0) Mar 2 19:39:07.553: ssaCallSetupInd Mar 2
19:39:07.553: ccCallSetContext (callID=0x9, context=0x62447A28) Mar 2 19:39:07.553:
ssaCallSetupInd cid(9), st(SSA_CS_MAPPING),oldst(0), ev(25)ev-
>e.evCallSetupInd.nCallInfo.finalDestFlag = 1 Mar 2 19:39:07.553: ssaCallSetupInd finalDest
cllng(1#5336), clled(2#3653) Mar 2 19:39:07.553: ssaCallSetupInd cid(9),
st(SSA_CS_CALL_SETTING),oldst(0), ev(25)dpMatchPeersMoreArg result= 0 Mar 2 19:39:07.557:
ssaSetupPeer cid(9) peer list: tag(3653) called number (2#3653) Mar 2 19:39:07.557: ssaSetupPeer
cid(9), destPat(2#3653), matched(5), prefix(21), peer(620F1EF0), peer->encapType (1) Mar 2
19:39:07.557: ccCallProceeding (callID=0x9, prog_ind=0x0) Mar 2 19:39:07.557: ccCallSetupRequest
(Inbound call = 0x9, outbound peer =3653, dest=, params=0x61E296C0 mode=0, *callID=0x61E299D0,
prog_ind = 3) Mar 2 19:39:07.557: ccCallSetupRequest numbering_type 0x81 Mar 2 19:39:07.557:
dest pattern 2#3653, called 2#3653, digit_strip 1 Mar 2 19:39:07.557: callingNumber=1#5336,
calledNumber=2#3653, redirectNumber=display_info= calling_oct3a=0 Mar 2 19:39:07.557:
accountNumber=, finalDestFlag=1, guid=6aef.3a87.165c.11cc.8040.d661.b74f.9390 Mar 2
19:39:07.557: peer_tag=3653 Mar 2 19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C,
dest=, callParams={called=2#3653,called_oct3=0x81, calling=1#5336,calling_oct3=0x81,
subscriber_type_str=Unknown, fdest=1, voice_peer_tag=3653},mode=0x0) vdbPtr type = 6 Mar 2
19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C, dest=, callParams={called=2#3653,
called_oct3 0x81, calling=1#5336,calling_oct3 0x81, fdest=1, voice_peer_tag=3653}, mode=0x0,
xltrc=-4) Mar 2 19:39:07.557: ccSaveDialpeerTag (callID=0x9, dialpeer_tag= Mar 2 19:39:07.557:
ccCallSetContext (callID=0xA, context=0x6244D9EC) Mar 2 19:39:07.557: ccCallReportDigits
(callID=0x9, enable=0x0)

Problème IOS de passerelle

Dans le même laboratoire, l'image IOS 12.2(6a) est chargée sur l'OGW. Quand un appel est fait, on le note que l'OGW envoie toujours un jeton clair basé sur son mot de passe quoique la passerelle ne soit pas configurée pour que le RVI collecte l'Account/PIN. En outre, le garde-porte configuré pour le tout le niveau reçoit cet appel. Ceci est documenté dans l'ID de bogue Cisco CSCdw43224 (clients enregistrés seulement).

Sécurité avec des points finaux alternatifs

Comme mentionné plus tôt dans ce document, la Sécurité d'appel bout à bout est équipée d'utilisation des jetons d'accès qui sont introduits le champ de clearTokens dans les messages RAS/H.225. En activant une telle Sécurité, la passerelle de source en avant le jeton d'accès reçu du garde-porte dans un ACF au point final de destination H.323 dans le message de configuration H.225. Ce de destination point final H.323 puis en avant le jeton d'accès reçu dans le message de configuration au garde-porte dans sa demande d'admission. Ce faisant, il donne au garde-porte distant la capacité d'admettre des appels basés sur la validité du jeton d'accès. Le contenu du jeton d'accès est jusqu'à l'entité qui le génère. Afin de réduire des failles de sécurité et garder contre des attaques homme-dans-le-moyennes, les garde-portes peuvent encoder les informations spécifiques de destination dans le jeton d'accès. Ceci signifie que quand des alternateEndpoints sont fournis dans un ACF, le garde-porte peut fournir un jeton distinct d'accès pour chaque alternateEndpoint spécifié.

Quand il des premiers essais d'établir une connexion, la passerelle Cisco envoie le jeton d'accès il a reçu dans le domaine de clearToken de l'ACF avec l'adresse dans le domaine de destCallSignalAddress. Si cette tentative est infructueuse et la passerelle Cisco se poursuit aux connexions de tentative par un point final alternatif, elle utilise le jeton associé d'accès (si elle est disponible) de la liste d'alternateEndpoints. Si les alternateEndpoints les répertorient reçu dans l'ACF n'inclut pas des jetons d'accès, mais l'ACF inclut un jeton d'accès, la passerelle Cisco inclut ce jeton d'accès dans toutes les tentatives de se connecter à un point final alternatif.

Support symbolique OSP

Actuellement le Protocole OSP (Open Settlement Protocol) et ses jetons sont seulement pris en charge sur des passerelles Cisco. Il n'y a aucun support sur le garde-porte. La passerelle identifie des jetons OSP reçus d'un serveur de règlement et les insère dans le message de configuration Q.931 à une dernière passerelle.

Différents niveaux de Sécurité pour chaque point final ou zone

Actuellement vous ne pouvez pas configurer des différents niveaux de Sécurité pour chaque point final ou zone. Le niveau de Sécurité est pour toutes les zones gérées par ce garde-porte. Une demande de caractéristique peut être ouverte pour une telle question.

Garde-porte d'Interdomain à la Sécurité de garde-porte

Le garde-porte d'Interdomain à la Sécurité de garde-porte fournit la capacité de valider des demandes de garde-porte-à-garde-porte d'intradomain et d'interdomain sur une base de par-saut. Ceci signifie que le garde-porte de destination termine le CAT et génère un neuf si le garde-porte décide d'expédier le LRQ en avant. Si le garde-porte détecte une signature non valide LRQ elle

répond en envoyant une anomalie d'emplacement (LRJ).

Garde-porte de mise en place à la Sécurité de garde-porte

Le garde-porte de commencement génère un IZCT quand un LRQ est initié ou un ACF est sur le point d'être envoyé en cas d'appel d'intra-zone. Ce jeton est traversé par son chemin de routage. Le long du chemin, chaque garde-porte met à jour l'ID de garde-porte de destination et/ou l'ID de garde-porte de source s'il y a lieu, pour refléter les informations de zone. Le garde-porte de terminaison génère un jeton avec son mot de passe. Ce jeton est rapporté dans les messages de la confirmation d'emplacement (LCF) et passé à OGW. L'OGW inclut ce jeton dans le message de configuration H.225. Quand le TGW reçoit le jeton, il est expédié dans l'answerCall ARQ et validé par le garde-porte de terminaison (TGK) sans n'importe quel besoin de serveur de RAYON.

Le type d'authentification est basé sur le mot de passe avec le hachage comme décrit dans ITU H.235. Spécifiquement, la méthode de cryptage est MD5 avec le hachage de mot de passe.

Le but de l'IZCT est de savoir si le LRQ est arrivé d'un domaine étranger, duquel zone, et de quel transporteur. Il est également utilisé pour passer un jeton à l'OGW dans le LCF du TGK. Dans le format IZCT, ces informations sont exigées :

- srcCarrierID — Identification de transporteur de source
- dstCarrierID — Identification de transporteur de destination
- intCarrierID — Identification intermédiaire de transporteur
- srcZone — Zone de source
- dstZone — Zone de destination
- type d'interzoneINTRA_DOMAIN_CISCOINTER_DOMAIN_CISCOINTRA_DOMAIN_TERM_NOT_CISCOINTER_DOMAIN_ORIG_NOT_CISCO

Cette caractéristique fonctionne bien sans n'importe quel besoin d'ID de transporteur de la passerelle ou d'un serveur sensible du routage de transporteur (CSR). Dans un tel cas, les champs au sujet de l'ID de transporteur sont vides. Les exemples ici n'incluent aucun ID de transporteur. Pour un écoulement d'appel détaillé, le support de release et de plate-forme, et les configurations, se rapportent au Fonction Inter-Domain Gatekeeper Security Enhancement.

Configuration du contrôleur d'accès

La caractéristique IZCT exige cette configuration sur le garde-porte.

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

Le mot de passe doit être six à huit caractères. Identifiez quelle zone est dans un domaine étranger comme ceci :

```
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]
```

Écoulement d'appel IZCT

Ce diagramme affiche que les IZCT circulent.

Dans cette configuration, les noms des passerelles et les garde-portes sont identiques que ceux utilisés dans l'organigramme d'appel IZCT mais avec la lettre minuscule. L'écoulement d'appel est expliqué après la configuration, avec mettent au point des explications.

Pour expliquer l'IZCT comportez et appelez l'écoulement, le premier exemple n'a pas la passerelle d'intradomain à la Sécurité de garde-porte. Après cela, il y a des exemples où le TGW ne peut pas générer l'IZCT de sorte que le TGK1 rejette l'appel. C'est de prouver que la caractéristique fonctionne comme conçu. Toutes ces installations sont basées sur la topologie dans l'organigramme d'appel IZCT.

Exemple 1 : Appelez l'écoulement pour le garde-porte à la Sécurité de garde-porte seulement

Cet exemple affiche les configurations relatives de tous les passerelles et garde-portes.

Configuration OGW	Configuration TGW
! hostname ogw !controller E1 3/0 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	hostname tgw ! controller E1 0 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

Configuration OGK1	Configuration TGK1
! hostname ogk1 ! interface Ethernet0/0 ip address 172.16.13.35	! hostname tgk1 ! interface Ethernet0/0 ip address 172.16.13.41

```
255.255.255.224
 half-duplex
!
gatekeeper
 zone local ogk1
domainA.com
172.16.13.35
 zone remote ogk2
domainA.com
172.16.13.14 1719
 zone prefix ogk2 36*
 zone prefix ogk1 53*
 security izct password
111222
 gw-type-prefix 1#*
default-
technology
no shutdown
!
!
no scheduler max-task-time
no scheduler allocate
ntp master
!
end
```

```
255.255.255.224
 ip directed-broadcast
 half-duplex
!
gatekeeper
 zone local tgk1 domainB.com
172.16.13.41
 zone remote tgk2
domainB.com
172.16.13.16 1719
 zone prefix tgk1 36*
 zone prefix tgk2 53*
 security izct password
111222
 gw-type-prefix 2#* default-
technology
 no shutdown
!
ntp clock-period 17179797
ntp server 172.16.13.35
!
end
```

Configuration OGK2

```
!
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
```

Configuration TGK2

```
!
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
```

Écoulement d'appel avec des debugs

Ces exemples utilisent met au point pour expliquer l'écoulement d'appel.

1. Un utilisateur sur le transporteur E appelle un utilisateur sur le transporteur 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. Puisque le dialpeer de la passerelle d'origine (tag=3653) est configuré pour RAS, il envoie un ARQ à 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. Quand OGK1 reçoit l'ARQ, il détermine que la destination est entretenue par la zone distante OGK2. Il l'identifie alors qu'un IZCT est nécessaire (par le CLI : **<pwd> de mot de passe de security izct**). OGK1 poursuit pour créer l'IZCT avant que le LRQ soit envoyé. Il envoie alors l'IZCT et le LRQ à OGK2 et envoie un message RIP de nouveau à 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. Quand OGK2 reçoit le LRQ, il vérifie l'IZCT. De la configuration il constate que les neets LRQ pour contenir également un IZCT. OGK2 crée alors un nouvel IZCT en changeant l'izctSrcZone et l'izctDstZone pour être ogk2 et en avant le LRQ à TGK2. Après qu'il envoie le LRQ à TGK2, il renvoie un message RIP à OGK1.Si les garde-portes font partie d'une batterie, le nom du cluster est utilisé pour le SrcZone ou le DstZone.

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

value RasMessage ::= requestInProgress :
!--- RIP message is sent back to OGK1. { **requestSeqNum 2048** delay 6000 } Mar 4 15:31:20.055: RAS OUTGOING ENCODE BUFFER::= 80 050007FF 176F Mar 4 15:31:20.055: Mar 4 15:31:20.055: IPSOCK_RAS_sendto: msg length 7 from 172.16.13.14:1719 to 172.16.13.35: 1719 Mar 4 15:31:20.059: RASLib::RASSendRIP: RIP (seq# 2048) sent to 172.16.13.35 Mar 4 15:31:20.059: H225 NONSTD OUTGOING PDU ::= value LRQnonStandardInfo ::= { ttl 5 nonstd-callIdentifier { guid '221B686C17CC11CC8011A049E0524766'H } callingOctet3a 128 gatewaySrcInfo { e164 : "4085272923", h323-ID : {"ogw"} } } Mar 4 15:31:20.063: H225 NONSTD OUTGOING ENCODE BUFFER::= 82 06B01100 221B686C 17CC11CC 8011A049 E0524766 01801002 048073B8 5A5C5640 02006F00 670077 Mar 4 15:31:20.072: Mar 4 15:31:20.072: PDU ::= **value IZCToken ::= { izctInterZoneType intraDomainCisco : NULL** *!--- This is still intraDomain since message OGK1 is !--- not a foreign domain.* **izctSrcZone "ogk2"** *!--- ScrZone and DstZone become ogk2.* **izctDstZone "ogk2" }** Mar 4 15:31:20.076: ENCODE BUFFER::= 47 00C06F67 6B32066F 676B3204 73726304 64737404 696E74 Mar 4 15:31:20.080: Mar 4 15:31:20.080: RAS OUTGOING PDU ::= value RasMessage ::= **locationRequest :** *!--- The LRQ is forwarded to TGK2.* { requestSeqNum 2048 destinationInfo { e164 : "3653" } nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '8206B01100221B686C17CC11CC8011A049E05247...'H } replyAddress ipAddress : { ip 'AC100D23'H port 1719 } sourceInfo { h323-ID : {"ogk1"} } canMapAlias TRUE **tokens** *!--- IZCT is included.* **{ { tokenOID { 1 2 840 113548 10 1 0 } nonStandard { nonStandardIdentifier { 1 2 840 113548 10 1 0 } data '4700C06F676B32066F676B320473726304647374...'H } } }** } Mar 4 15:31:20.104: RAS OUTGOING ENCODE BUFFER::= 4A 8007FF01 01806986 40B50000 12288206 B0110022 1B686C17 CC11CC80 11A049E0 52476601 80100204 8073B85A 5C564002 006F0067 007700AC 100D2306 B70BA00B 01400300 6F006700 6B003101 80300100 80092A86 4886F70C 0A010009 2A864886 F70C0A01 00184700 C06F676B 32066F67 6B320473 72630464 73740469 6E74 Mar 4 15:31:20.120: Mar 4 15:31:20.120: IPSOCK_RAS_sendto: msg length 127 from 172.16.13.14:1719 to 172.16.13.16: 1719 Mar 4 15:31:20.124: RASLib::RASSendLRQ: **LRQ (seq# 2048) sent to 172.16.13.16**

5. TGK2 détermine que le LRQ provient un domaine étranger. Il met à jour le dstZone de l'IZCT avec son propre ID et l'interZoneType comme INTER_DOMAIN_CISCO. Il alors crée un nouveau CAT et passe l'IZCT mis à jour et le LRQ à TGK1.TGK2 traite la zone dont un LRQ est reçu comme zone d'étranger-domaine dans l'un ou l'autre de ces deux scénarios :La liste distante de zone TGK2 ne contient pas la zone dont un LRQ est reçu.La liste distante de zone TGK2 contient la zone dont un LRQ est reçu. La zone est identifiée par un indicateur d'étranger-domaine.Il envoie alors à une demande le message en cours de nouveau à 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. Normalement TGK1 met à jour le dstCarrierID de l'IZCT au transporteur E, qui est déterminé par le processus de routage. Cependant, puisqu'aucun transporteur n'est utilisé, vous ne voyez pas cela. TGK1 génère un jeton d'informations parasites avec le mot de passe de l'IZCT. Il envoie un LCF avec l'IZCT mis à jour dans lui à OGK1. Cet izctHash est utilisé pour authentifier l'answerCall ARQ que TGK1 reçoit du TGW quand le plus tard reçoit le message de configuration VoIP d'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 '0001400200740067007706007400670 06B003101...'H } destinationType { gateway { protocol { voice : { supportedPrefixes { } } } } } mc FALSE undefinedNode FALSE } **tokens** *!--- The IZCT is included.* **{ { tokenOID { 1 2 840 113548 10 1 0 } nonStandard { nonStandardIdentifier { 1 2 840 113548 10 1 0 } data '7F20C06F676B320674676B32C02B9620C7010310...'H } }** } } Mar 4 15:31:20.367: RAS OUTGOING ENCODE BUFFER::= 4F 07FF00AC 100D1706 B800AC10 0D17D9D2 40B50000 122F0001 40020074 00670077 06007400 67006B00 31011001 40020074 00670077 00AC100D 1706B800 00000000 00000000 00104808 0880013C 05010000 48010080 092A8648 86F70C0A 0100092A 864886F7 0C0A0100 307F20C0 6F676B32 0674676B 32C02B96 20C70103 105A7D5E 18AA658A 6A4B4709 BA5ABEF2 B9047372 63046473 7404696E 74 Mar 4 15:31:20.371: Mar 4 15:31:20.371: IPSOCK_RAS_sendto: msg length 154 from 172.16.13.41:1719 to 172.16.13.35: 1719 Mar 4 15:31:20.371: RASLib::RASSendLCF: **LCF (seq# 2048) sent to 172.16.13.35**

7. OGK1 extrait l'IZCT du LCF et l'envoie dans un ACF à l'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. L'OGW envoie l'IZCT au TGW dans le message de configuration 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. Le TGW passe l'IZCT au TGK1 dans un answerCall 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. TGK1 authentifie la destination IZCT avec succès. C'est parce que TGK1 génère les informations parasites dans l'IZCT et il renvoie un ACF au 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. Le TGW établit l'appel vers le transporteur D après qu'il reçoive l'ACF.**Exemple 2 : L'appel a manqué parce que le TGW ne peut pas extraire l'IZCT du message SETUP reçu.**Cet exemple est basé sur la mêmes topologie et configuration que l'exemple 1. Dans cet exemple, le logiciel du TGW est changé à une version où l'IZCT n'est pas pris en charge. En pareil cas, le TGW ne peut pas extraire l'IZCT du message de configuration. Ceci fait rejeter le TGK1 l'appel avec une raison de débranchement de refus de Sécurité.Cet exemple affiche seulement le message de configuration, ARQ, et l'ARJ sur le TGW puisque l'écoulement d'appel est identique que l'exemple 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
```

Informations connexes

- Fonction Inter-Domain Gatekeeper Security Enhancement
- Comptabilité et améliorations de la sécurité de Cisco H.235 pour des passerelles Cisco
- Cisco IOS référence de débogage des commandes, version 12.3
- Assistance technique concernant la technologie vocale
- Assistance concernant les produits vocaux et de communications unifiées
- Dépannage des problèmes de téléphonie IP Cisco
- Support et documentation techniques - Cisco Systems