Руководство по решению проблем безопасности между шлюзом и привратником (H.235), а также привратником и привратником (IZCT)

Содержание

Введение

Сети H.323 имеют различные виды конфигураций и диаграмм вызовов. Этот документ обсуждает большинство проблем безопасности с сетями H.323, которые включают сторожевые устройства. Этот документ суммирует способ, которым работает каждая функция и как устранить неполадки его с пояснением на большинстве отладок. Этот документ не обращается к общей безопасности VoIP.

Этот документ покрывает эти функции:

- **Безопасность соединения шлюза со сторожевым устройством внутри домена** — Эта безопасность основывается на H.235, в котором вызовы H.323 аутентифицируются, авторизуются и направляются сторожевым устройством. Сторожевое устройство

считают известным и надежным объектом в некотором смысле, что шлюз не аутентифицирует его, когда шлюз пытается зарегистрироваться в нем.

- **Внутридоменное сторожевое устройство к Безопасности сторожевого устройства** — Эта аутентификация покрытий безопасности и авторизация H.323 звонит между административными доменами интернет-Поставщиков услуг телефонной связи (ITSP) с помощью **InterZone Clear Token (IZCT)**. Этот документ покрывает только часть, куда завершающееся сторожевое устройство передает маркер в своем Location Confirmation (LCF) сообщение так, чтобы это аутентифицировало Запрос на доступ answerCall (ARQ). Запрос местонахождения (LRQ) проверка не включен в эту функцию. Проверка LRQ является функцией, планируемой для будущего выпуска программного обеспечения Cisco IOS.

Определения

Акроним	**Определение**
ARQ	Запрос на доступ — Протокол регистрации, входа и состояния (RAS), который сообщение передало от оконечной точки H.323 до сторожевого устройства, которое запрашивает разрешение установить вызов.
ACF	Подтверждение допуска — сообщение RAS передало от сторожевого устройства до оконечной точки, которая подтверждает принятие вызова.
ARJ	Отказ от приема — сообщение RAS от сторожевого устройства до оконечной точки, которая отклоняет запрос на доступ.
CAT	Cisco Access Token — удаление маркера H.235.
CHAP	Протокол аутентификации по квитированию вызова — протокол аутентификации, где используется проблема.
GCF	Gatekeeper Confirm — Сообщение RAS передало от сторожевого устройства до оконечной точки H.323, которая подтверждает обнаружение сторожевого устройства.
GRQ	Запрос сторожевого устройства — сообщение RAS, передаваемое от оконечной точки H.323 для обнаружения сторожевого устройства.
H.235	Рекомендации по безопасности ITU и шифрование для Серии H (H.323 и другое основанное на H.245) мультимедийные терминалы.
IZCT	InterZone Clear Token — IZCT генерируется в исходном сторожевом устройстве, когда LRQ инициируется, или ACF собирается быть переданным за вызовом внутренней зоны в административном домене ITSP.

LRQ	Запрос местонахождения — сообщение RAS, передаваемое от сторожевого устройства до сторожевого устройства следующего перехода или ветви вызовов, чтобы отследить и направить вызов.
RAS	Регистрация, Разрешение и Статус — протокол, который позволяет сторожевому устройству выполнять регистрацию, разрешение и проверки состояния оконечной точки.
RCF	Регистрация Подтверждает — сообщение RAS, передаваемое от сторожевого устройства до оконечной точки, которая подтверждает регистрацию.
RRJ	Отказ в регистрации — сообщение RAS передало от сторожевого устройства, которое отклоняет запрос регистрации.
RRQ	Запрос регистрации — сообщение RAS передало от оконечной точки до сторожевого устройства, которое запрашивает зарегистрироваться в нем.
RIP	Происходящий запрос — сообщение RAS передало от сторожевого устройства до отправителя, который сообщает, что вызов происходит.

Внутридоменный шлюз для защиты привратника

H.323 является рекомендацией ITU, которая обращается к связи в режиме реального времени обеспечения по ненадежным сетям. Это включает две обширных области беспокойства: аутентификация и конфиденциальность. Существует два типа аутентификации, согласно H.235:

- Проверка подлинности на основе симметричного шифрования, которая не требует никакого предшествующего контакта между взаимодействующими объектами.
- На основе способности иметь некоторых предшествующих общий secret (далее ссылаемый как основанная подписка), предоставлены две формы основанной на подписке аутентификации:passwordсертификат

Временная метка, переданная в маркерах

Штамп времени используется для предотвращения атак с повторением пакетов. Поэтому это необходимо для взаимоприемлемой ссылки на время (из которого можно получить штампы времени). Сумма приемлемого перекоса времени является вопросом локального внедрения.

Реализация рекомендаций H.235 компанией Cisco

Cisco использует Протокол аутентификации по квитированию вызова (CHAP) - как схема проверки подлинности как основание для его шлюза к его сторожевому устройству

реализация H.235. Это позволяет вам усиливать Аутентификацию, авторизацию и учет (AAA), с помощью существующей функциональности для выполнения реальной аутентификации. Это также означает, что сторожевое устройство не требуется, чтобы иметь доступ к базе данных шлюза IDS, номерам учетной записи пользователя, паролям и PIN. Схема основывается на H.235, разделите 10.3.3. Это описано как основанный на подписке пароль с хешированием.

Однако вместо того, чтобы использовать cryptoTokens H.225, этот метод использует H.235 clearTokens с полями, заполненными соответственно для использования с RADIUS. Этот маркер упоминается как Cisco Access Token (CAT). Можно всегда выполнять аутентификацию локально на сторожевом устройстве вместо того, чтобы использовать сервер RADIUS.

Использование cryptoTokens требует, чтобы сторожевое устройство или поддержало или имело некоторый способ получить пароли за всех пользователей и шлюзы. Это вызвано тем, что маркерное поле cryptoToken задано таким образом, что аутентифицирующийся объект нуждается в пароле для генерации его собственного маркера, с которым можно сравнить полученный.

Сторожевые устройства Cisco полностью игнорируют cryptoTokens. Однако привратники vocalTek и другие, которые поддерживают H.235, разделяют 10.3.3, используют cryptoTokens для аутентификации шлюза. Сторожевые устройства Cisco используют CAT для аутентификации шлюза. Так как шлюз не знает, с каким сторожевым устройством он говорит, он передает обоим в RRQ. authenticationCapability в GRQ для cryptoToken и указывают, что хеширование MD5 является механизмом аутентификации (невзирая на то, что CAT также использует MD5).

См. Безопасность Cisco H.323 Gateway и Улучшения учета для получения дополнительной информации.

Как конфигурировать уровни безопасности

- Оконечная точка - или безопасность на уровне регистрацииС Безопасностью регистрации, включенной на сторожевом устройстве, шлюз требуется, чтобы включать CAT во все сообщения RRQ большого объема. CAT, в этом случае, содержит информацию, которая аутентифицирует сам шлюз на сторожевом устройстве. Сторожевое устройство форматирует сообщение к серверу RADIUS, который аутентифицирует информацию, содержавшуюся в маркере. Это назад отвечает на сторожевое устройство или с Access-Accept или с Access-Reject. Это, в свою очередь, отвечает на шлюз или с RCF или с RRJ.Если вызов тогда размещен от успешно аутентифицируемого шлюза, тот шлюз генерирует новый CAT по получении ACF от сторожевого устройства с помощью пароля шлюза. Этот CAT идентичен тому, генерируемому во время регистрации за исключением штампа времени. Это размещено в исходящее Сообщение SETUP. Шлюз назначения извлекает маркер из Сообщения SETUP и размещает его в ARQ стороны - получателя. Сторожевое устройство использует RADIUS для аутентификации исходного шлюза, прежде чем это передаст ACF стороны - получателя. Это предотвращает неаутентифицируемую оконечную точку, которая знает, что адрес шлюза от использования его обходит схему безопасности и доступ к открытой коммутируемой телефонной сети (PSTN).Поэтому в этом уровне, нет никакой потребности включать любые маркеры в инициирующие ARQ.Не введите

security token required-for registration от интерфейса командной строки (CLI) сторожевого устройства для настройки сторожевого устройства. *Никакая* опция команды не заставляет сторожевое устройство больше не проверять для маркеров в сообщениях RAS.Не введите оконечную точку уровня <PASSWORD> надежного пароля от CLI шлюза для настройки шлюза. *Никакая* опция команды не заставляет шлюз больше не генерировать маркеры для сообщений RAS.

- Для каждого вызова безопасность уровняБезопасность каждого вызова полагается на безопасность на уровне регистрации. Когда Безопасность каждого вызова включена на сторожевом устройстве, в дополнение к соответствию требованиям безопасности регистрации шлюз также требуется, чтобы включать Маркеры доступа во все сообщения ARQ вызывающей стороны. Маркер в этом случае содержит информацию, которая определяет пользователя шлюза к сторожевому устройству. Эта информация получена с помощью сценария Интерактивного голосового ответа (IVR) на шлюзе. Это побуждает пользователей вводить свой Идентификатор пользователя и PIN от клавиатуры, прежде чем они закажут телефонный разговор.CAT, содержавшийся в инициирующем ARQ, аутентифицируется RADIUS таким же образом, как описано ранее в Оконечной точке - или Безопасность на уровне регистрации. После того, как это получит ACF, шлюз генерирует новый CAT с помощью своего пароля и передает его в рамках Сообщения SETUP H.225 к конечному шлюзу.Не введите **требуемый маркер безопасности - для всех** от CLI сторожевого устройства для настройки сторожевого устройства. *Никакая* опция команды не заставляет сторожевое устройство больше не проверять для маркеров в сообщениях RAS.Не введите **уровень <PASSWORD> надежного пароля для каждого вызова** от CLI шлюза для настройки шлюза. *Никакая* опция команды не заставляет шлюз больше не генерировать маркеры для сообщений RAS.
- Вся безопасность уровняЭто позволяет шлюзу включать CAT во все сообщения RAS, необходимые для регистрации и для вызовов. Поэтому это - комбинация вышеупомянутых двух уровней. При использовании этой опции проверка CAT в сообщениях ARQ основывается на номере учетной записи и PIN пользователя, который звонит. Проверка CAT, передаваемого во всех других сообщениях RAS, основывается на пароле, настроенном для шлюза. Поэтому это подобно Для каждого вызова уровень.Не введите **требуемый маркер безопасности - для всех** от CLI сторожевого устройства для настройки сторожевого устройства. *Никакая* опция команды не заставляет сторожевое устройство больше не проверять для маркеров в сообщениях RAS.Не введите **уровень <PASSWORD> надежного пароля все** от CLI шлюза для настройки шлюза. *Никакая* опция команды не заставляет шлюз больше не генерировать маркеры для сообщений RAS.

Использование H.235 на уровне отдельных звонков без системы интерактивного речевого ответа (IVR)

H.235 не может использоваться на для каждого вызова уровень без IVR. Если нет никакого IVR для сбора учетной записи и PIN, шлюз должен передать ARQ без Удаления маркера (но с криптографическим маркером). Так как Сторожевое устройство Cisco только принимает Удаления маркера, требование отклонено сторожевым устройством с причиной отказа в доступе из соображений безопасности.

Данный пример показывает отладки **h225 asn1**, собранные от **исходного шлюза (OGW)**,

который не настроен для IVR для сбора учетной записи и PIN. Сообщение RRQ имеет Удаление маркера, но ARQ не делает. Сообщение ARJ передают обратно в шлюз.

```
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
```

См. раздел Задач конфигурации Учета Cisco H.235 и Улучшений безопасности для шлюзов Cisco для получения дополнительной информации о конфигурации IVR.

Главные проблемы

Основные проблемы, в которых вы должны быть обеспокоены, включают:

- Конфигурация шлюза и сторожевого устройства
- Конфигурация RADIUS на сторожевом устройстве и сервере RADIUS
- Протокол NTP — у Вас должно быть то же время на всех шлюзах и сторожевых устройствах. Поскольку информация для аутентификации включает метку времени, важно, что вся Cisco H.323 Gateways и Сторожевые устройства (или другой объект, который выполняет аутентификацию) синхронизироваться. Cisco H.323 Gateways должна синхронизироваться с помощью NTP.
- Сбой программного обеспечения из-за дефекта

Отладка и маршрут вызова для разных уровней

Так как вся безопасность уровня покрывает и регистрацию и для каждого вызова случаи, лабораторная работа установлена с тем уровнем безопасности для этого осуществления. Диаграммы вызовов для регистрации и обычного вызова VoIP объяснены в конфигурации здесь.

Примечание: Конфигурация здесь не завершена. Больше команд придерживается промежуточный выходные данные отладки. Это разработано для показа, какая проблема может произойти, если вы не проверяете все вещи, такие как конфигурация, NTP и RADIUS. Кроме того, шлюз аутентифицируется на сторожевом устройстве локально так, чтобы вы были в состоянии видеть, какие значения установлены для идентификатора шлюза и пароля. Эта конфигурация отрезана так, чтобы только показали связанную конфигурацию.

```
!
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
```

Эти отладки включены в данном примере:

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

- debug aaa authorization

Первая вещь, которая происходит, состоит в том, что шлюз передает GRQ к сторожевому устройству, и сторожевое устройство передает GCF к шлюзу. Шлюз тогда передает RRQ и ждет или RCF или RRJ.

В предыдущей конфигурации шлюз является "not set" для любого уровня безопасности так, чтобы его GRQ не нес **authenticationCapability**, который необходим для маркеров. Однако сторожевое устройство все еще передает GCF обратно как показано в выходных данных ниже:

```
*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 } } }
```

Это объясняет некоторые сообщения в GRQ:

- **authenticationCapability** — Это поле только имеет значение pwdHash. Это указывает, что хеширование MD5 является механизмом аутентификации.

- **algorithmOIDs** — Идентификатор объекта алгоритма. В этом случае это несет значение (1 2 840 113549 2 5), который является идентификатором объекта для Профиля сообщения 5 хеширований.(1 2 840 113549 2 5), преобразовывает в iso (1) комитет-член (2) US (840) rsadsi (113549) digestAlgorithm (2) md5 (5). Следовательно Cisco делает хеширование пароля MD5.Rsadsi выдерживает за безопасность данных RSA Inc. Идентификатор объекта Взаимодействия открытых систем (OSI) RSA Data Security, Inc. 1.2.840.113549 (0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d в hex), как зарегистрировано Американским национальным институтом стандартов (ANSI).

Сторожевое устройство снова передает GCF как предыдущее сообщение. Шлюз тогда передает RRQ с определенными полями для описания маркеров, которые он использует, чтобы аутентифицироваться. **Сообщение RRQ asn1**, которое передается, показывают в данном примере.

```
*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 }
```

Прежде чем ответ обсужден, некоторые смежные области в вышеупомянутом сообщении RRQ объяснены здесь. Существует два типа маркеров: удаление маркера или CAT) и криптографический маркер.

Как упомянуто прежде, Сторожевые устройства Cisco игнорируют криптографические маркеры. Однако шлюз все еще передает обоим, потому что он не знает к тому, какое сторожевое устройство он говорит. Так как другие поставщики могли бы использовать криптографические маркеры, шлюз передает обоим.

Это объясняет синтаксис ClearToken.

- **tokenOID** — Идентификатор объекта для определения маркера.
- **timeStamp** — Текущее Согласованное текущее время (UTC) время шлюза. Секунды с тех пор 00:00 01.01.1970 UTC.Это используется в качестве подразумеваемого запроса вызова CHAP, как будто это первоначально прибыло из сторожевого устройства.
- **проблема** — 16-байтовый профиль сообщения MD5, генерируемый шлюзом с помощью этих полей:бросьте вызов = [случайный + GW/Пароль пользователя + метка времени] Хэш MD5RADIUS выполняет это вычисление (так как это знает случайное число, пароль шлюза и запрос вызова CHAP) определить, какова проблема должна быть: ответ CHAP

= [ID CHAP + UserPassword + запрос вызова CHAP] Хэш MD5
- **случайный** — однобайтовое значение, используемое RADIUS для определения этого определенного запроса.Шлюз инкрементно увеличивает переменный модуль 256 для каждого запроса аутентификации для удовлетворения этого Требования RADIUS.
- **generalID** — H323-ID шлюза или номер учетной записи пользователя на основе уровня безопасности и типа сообщения RAS.

Примечание: Все эти поля важны. Однако большее внимание уделяют и штампу времени и generalID. В этом случае штамп времени **731030839**, и generalID **gwa-1@cisco.com**.

CryptoToken в RRQ содержит информацию о шлюзе, который генерирует маркер. Это включает идентификатор шлюза (который является ID H.323, настроенным на шлюзе), и пароль шлюза.

Это поле содержит один из типов cryptoToken, определенных для поля CryptoH323Token, заданного в H.225. В настоящее время единственный тип поддерживаемого cryptoToken является cryptoEPPwdHash.

Эти поля содержатся в поле cryptoEPPwdHash:

- **alias** — Псевдоним шлюза, который является ID H.323 шлюза.
- **timestamp** — Штамп текущего времени.
- **маркер** — алгоритм представления сообщения в краткой форме 5 (MD5) - закодировал PwdCertToken. Это поле содержит эти элементы:**timestamp** — То же как метка времени cryptoEPPwdHash.**пароль** — пароль шлюза.**generalID** — Тот же шлюз искажает как тот, включенный в cryptoEPPwdHash.**tokenID** — Идентификатор объекта.

Просмотрите ответ от сторожевого устройства в данном примере.

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

value RasMessage ::= registrationReject : { requestSeqNum 4 protocolIdentifier { 0 0 8 2250 0 3
} rejectReason securityDenial : NULL !--- The gatekeeper rejects the RRQ with securityDenial
reason. gatekeeperIdentifier {"gka-1"} } *Mar 2 13:33:15.727: RAS OUTGOING ENCODE BUFFER::= 14
80000306 0008914A 00038301 00080067 006B0061 002D0031 *Mar 2 13:33:15.731:
```

RRQ отклонен сторожевым устройством. Причина для это вызвано тем, что не была никаким набором NTP в конфигурации шлюза. Сторожевое устройство проверяет штамп времени маркера, чтобы видеть, ли это в приемлемом интервале времени относительно его собственного времени. В настоящее время это окно +/-30 секунд около времени UTC сторожевого устройства.

Маркер за пределами этого окна заставляет сторожевое устройство сбрасывать от этого сообщения. Это предотвращает атаки с повторением пакетов от кого-то, кто пытается снова использовать snooping маркер. На практике все шлюзы и сторожевые устройства должны использовать NTP для предотвращения на этот раз перекоса временной диаграммы. Сторожевое устройство находит, что штамп времени в маркере в приемлемом интервале времени его времени. Поэтому это не сверяется с RADIUS для аутентификации шлюза.

Шлюз тогда настроен для NTP, указывающего на сторожевое устройство как NTP master, так, чтобы и шлюз и сторожевое устройство имели то же время. Когда это происходит, шлюз передает новый RRQ и на этот раз ответы сторожевого устройства назад на новый RRQ с RRJ.

Эти отладки от сторожевого устройства. Отлаживает выполненный, чтобы видеть,

переходит ли сторожевое устройство к фазе проверки подлинности.

```
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"} }
```

После настройки NTP сторожевое устройство все еще отклоняет RRQ. На этот раз, однако, это проходит процесс проверки подлинности для того шлюза. Сторожевое устройство отклоняет RRQ, потому что пользователь (здесь идентификатор шлюза) не найден в списке допустимых пользователей на RADIUS. Шлюз аутентифицируется локально в конфигурации сторожевого устройства. На списке пользователей вы видите Гва 1. Однако это не подходящий пользователь, так как пользователь в RRQ gwa-1@cisco.com.

Кроме того, однажды имя пользователя gwa-1@cisco.com команда пароля 0 2222 настроена на сторожевом устройстве, сторожевое устройство подтверждает RRQ, и шлюз зарегистрирован.

В этой лабораторной работе другой шлюз (Гва 2) зарегистрирован к тому же сторожевому устройству (gka-1). Вызов выполнен от gwa-1@cisco.com до Гва 2, чтобы видеть, как смотрят ARQ, ACF и сообщения SETUP.

Эти собранные отладки от начального и терминального шлюза (Гва 2).

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

Пояснение некоторых сообщений отладки включено.

Перед печатью отладок от возникновения/конечного шлюза этот текст объясняет поток вызовов:

1. Когда Сообщение SETUP получено от PSTN, шлюз передает ARQ к и получает ACF от сторожевого устройства.
2. Когда шлюз получает ACF, шлюз генерирует CAT с помощью пароля шлюза, псевдонима H323-ID, и текущее время. Маркер размещен в Блок управления вызовом (CCB).
3. Когда шлюз передает Сообщение SETUP к конечному шлюзу, это получает маркер доступа из CCB и размещает его в nonStandardParameter поле clearToken в рамках Сообщения SETUP.
4. Конечный шлюз удаляет маркер из Сообщения SETUP, преобразовывает его из nonStandardParameter в CAT и размещает его в ARQ.
5. Сторожевое устройство проверяет штамп времени маркера, чтобы видеть, ли это в приемлемом интервале времени относительно его собственного времени. В настоящее время это окно +/-30 секунд около времени UTC сторожевого устройства. Маркер за пределами этого окна заставляет сторожевое устройство сбрасывать от этого сообщения. Это заставляет вызов быть отклоненным.
6. Если маркер приемлем, сторожевое устройство форматирует пакет запроса Доступа к серверу RADIUS, заполняет соответствующие атрибуты для проверки запроса вызова CHAP и передает его к серверу RADIUS.
7. Основанный на предположении, что псевдоним шлюза известен в сервере, сервер определяет местоположение пароля, привязанного к этому псевдониму, и генерирует его собственный ответ CHAP с помощью псевдонима, пароля и запроса вызова CHAP от сторожевого устройства. Если его ответ CHAP совпадает с тем, полученным от сторожевого устройства, сервер передает Доступ, Принимают пакет к сторожевому устройству. Если они не совпадают, или если псевдоним шлюза не находится в базе данных сервера, сервер передает пакет Отклонения доступа обратно в сторожевое устройство.
8. Сторожевое устройство отвечает на шлюз с ACF, если это получает Доступ, Принимают, или ARJ с кодом причины **securityDenial**, если это получает Отклонение доступа. Если шлюз получает ACF, вызов связан.

Данный пример показывает отладку от исходного шлюза.

Примечание: Отладки **h225 asn1** для настройки не находятся в данном примере, так как это совпадает с замеченный в примере конечного шлюза, который придерживается примера вызывающего шлюза.

```
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

Данный пример показывает отладки от **конечного шлюза (TGW)**. Заметьте, что TGW установил ответный матч, так как это получило ACF, и вызов связан.

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


value H323_UserInformation ::=
  {
   h323-uu-pdu
   {
    h323-message-body setup :
    {
     protocolIdentifier { 0 0 8 2250 0 2 }
     sourceAddress { h323-ID : {"gwa-1@cisco.com"} !--- Setup is sent from gwa-1@cisco.com
gateway. } sourceInfo { gateway { protocol { voice : { supportedPrefixes { { prefix e164 : "1#"
} } } } } mc FALSE undefinedNode FALSE } activeMC FALSE conferenceID
'6AEF3A87165C11CC8040D661B74F9390'H conferenceGoal create : NULL callType pointToPoint : NULL
sourceCallSignalAddress ipAddress : { ip 'AC100D0F'H port 11032 } callIdentifier { guid
'6AEF3A87165C11CC8041D661B74F9390'H } tokens !--- Setup includes the Clear Token (CAT). { {
tokenOID { 1 2 840 113548 10 1 2 1 } timeStamp 731101147 challenge
'AFBAAFDF79446B9D8CE164DB8C111A87'H random 247 generalID {"gwa-1@cisco.com"} nonStandard {
nonStandardIdentifier { 0 1 2 4 } data '2B93B7DBAFBAAFDF79446B9D8CE164DB8C111A87...'H } }
fastStart { '0000000C6013800A04000100AC100D0F4673'H,
'400000060401004C6013801114000100AC100D0F...'H } mediaWaitForConnect FALSE canOverlapSend FALSE
} h245Tunneling TRUE nonStandardControl { { nonStandardIdentifier h221NonStandard : {
t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data
'E0010200010415040390 90A31803A983811E0285...'H } } } } RAW_BUFFER::= E0 01020001 04150403
9090A318 03A98381 1E028583 70058133 36353302 80060004 00000003 Mar 2 19:39:07.509: PDU DATA =
6147F378 value H323_UU_NonStdInfo ::= { version 2 protoParam qsigNonStdInfo : { iei 4 rawMesg
'04039090A31803A983811E028583700581333635...'H } progIndParam progIndIEinfo : { progIndIE
'00000003'H } } PDU DATA = 6147F378 value ARQnonStandardInfo ::= { sourceAlias { }
sourceExtAlias { } } RAW_BUFFER::= 00 0000 Mar 2 19:39:07.517: RAW_BUFFER::= 61 000100C0
2B93B7DA 08003200 32003200 3200000A 00670077 0061002D 00320000 Mar 2 19:39:07.517: PDU DATA =
6147C2BC value RasMessage ::= admissionRequest : !--- An answer ARQ is sent to the gatekeeper to
authenticate the caller. { requestSeqNum 22 callType pointToPoint : NULL callModel direct : NULL
endpointIdentifier {"81F5989C00000002"} destinationInfo { e164 : "2#3653" } srcInfo { e164 :
"1#5336" } srcCallSignalAddress ipAddress : { ip 'AC100D0F'H port 11032 } bandWidth 640
callReferenceValue 2 nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode
181 t35Extension 0 manufacturerCode 18 } data '000000'H } conferenceID
'6AEF3A87165C11CC8040D661B74F9390'H activeMC FALSE answerCall TRUE canMapAlias FALSE
callIdentifier { guid '6AEF3A87165C11CC8041D661B74F9390'H } tokens !--- CAT is included. { {
tokenOID { 0 4 0 1321 1 2 } timeStamp 731101147 challenge 'AFBAAFDF79446B9D8CE164DB8C111A87'H
random 247 generalID {"gwa-1@cisco.com"} } } cryptoTokens { cryptoEPPwdHash : { alias h323-ID :
```

{"gwa-2"} timeStamp 731101147 token { algorithmOID { 1 2 840 113549 2 5 } paramS { } hash "8479E7DE63AC17C6A46E9E19659568" } } } willSupplyUUIEs FALSE } RAW_BUFFER::= 27 98001500 F0003800 31004600 35003900 38003900 43003000 30003000 30003000 30003000 32010280 50698601 02804086 6900AC10 0D0F2B18 40028000 0240B500 00120300 00006AEF 3A87165C 11CC8040 D661B74F 939044E3 20010011 006AEF3A 87165C11 CC8041D6 61B74F93 9044014D 00060400 8A290102 C02B93B7 DA10AFBA AFDF7944 6B9D8CE1 64DB8C11 1A870200 F71E0067 00770061 002D0031 00400063 00690073 0063006F 002E0063 006F006D 00002E01 04040067 00770061 002D0032 C02B93B7 DA082A86 4886F70D 02050080 808479E7 0DE63AC1 7C6A46E9 E1965905 680100 Mar 2 19:39:07.533: h323chan_dgram_send:Sent UDP msg. Bytes sent: 228 to 172.16.13.35:1719 Mar 2 19:39:07.533: RASLib::GW_RASSendARQ: ARQ (seq# 22) sent to 172.16.13.35 Mar 2 19:39:07.549: h323chan_dgram_recvdata:rcvd from [172.16.13.35:1719] on sock[1] RAW_BUFFER::= 2B 00001540 028000AC 100D1706 B800EF1A 00C00100 020000 Mar 2 19:39:07.549: PDU DATA = 6147C2BC value RasMessage ::= **admissionConfirm :** *!--- ACF is received from the gatekeeper.* **{ requestSeqNum 22** bandWidth 640 callModel direct : NULL destCallSignalAddress ipAddress : { ip 'AC100D17'H port 1720 } irrFrequency 240 willRespondToIRR FALSE uuiesRequested { setup FALSE callProceeding FALSE connect FALSE alerting FALSE information FALSE releaseComplete FALSE facility FALSE progress FALSE empty FALSE } } Mar 2 19:39:07.553: **ACF (seq# 22) rcvd** Mar 2 19:39:07.553: **cc_api_call_setup_ind** (vdbPtr=0x61BC92EC, callInfo={called=2#3653,called_oct3=0x81,calling=1#5336,calling_oct3=0x81, calling_oct3a=0x0,subscriber_type_str=Unknown, fdest=1 peer_tag=5336, prog_ind=3},callID=0x6217CC64) Mar 2 19:39:07.553: cc_api_call_setup_ind type 0 , prot 1 Mar 2 19:39:07.553: cc_api_call_setup_ind (vdbPtr=0x61BC92EC, callInfo={called=2#3653, calling=1#5336, fdest=1 peer_tag=5336}, callID=0x6217CC64) Mar 2 19:39:07.553: cc_process_call_setup_ind (event=0x61E1EAFC) Mar 2 19:39:07.553: >>>>CCAPI handed cid 9 with tag 5336 to app "DEFAULT" Mar 2 19:39:07.553: sess_appl: ev(25=CC_EV_CALL_SETUP_IND), cid(9), disp(0) Mar 2 19:39:07.553: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(9), disp(0) Mar 2 19:39:07.553: ssaCallSetupInd Mar 2 19:39:07.553: ccCallSetContext (callID=0x9, context=0x62447A28) Mar 2 19:39:07.553: ssaCallSetupInd cid(9), st(SSA_CS_MAPPING),oldst(0), ev(25)ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1 Mar 2 19:39:07.553: ssaCallSetupInd finalDest cllng(1#5336), clled(2#3653) Mar 2 19:39:07.553: ssaCallSetupInd cid(9), st(SSA_CS_CALL_SETTING),oldst(0), ev(25)dpMatchPeersMoreArg result= 0 Mar 2 19:39:07.557: ssaSetupPeer cid(9) peer list: tag(3653) called number (2#3653) Mar 2 19:39:07.557: ssaSetupPeer cid(9), destPat(2#3653), matched(5), prefix(21), peer(620F1EF0), peer->encapType (1) Mar 2 19:39:07.557: ccCallProceeding (callID=0x9, prog_ind=0x0) Mar 2 19:39:07.557: ccCallSetupRequest (Inbound call = 0x9, outbound peer =3653, dest=, params=0x61E296C0 mode=0, *callID=0x61E299D0, prog_ind = 3) Mar 2 19:39:07.557: ccCallSetupRequest numbering_type 0x81 Mar 2 19:39:07.557: dest pattern 2#3653, called 2#3653, digit_strip 1 Mar 2 19:39:07.557: callingNumber=1#5336, calledNumber=2#3653, redirectNumber=display_info= calling_oct3a=0 Mar 2 19:39:07.557: accountNumber=, finalDestFlag=1, guid=6aef.3a87.165c.11cc.8040.d661.b74f.9390 Mar 2 19:39:07.557: peer_tag=3653 Mar 2 19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C, dest=, callParams={called=2#3653,called_oct3=0x81, calling=1#5336,calling_oct3=0x81, subscriber_type_str=Unknown, fdest=1, voice_peer_tag=3653},mode=0x0) vdbPtr type = 6 Mar 2 19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C, dest=, callParams={called=2#3653, called_oct3 0x81, calling=1#5336,calling_oct3 0x81, fdest=1, voice_peer_tag=3653}, mode=0x0, xltrc=-4) Mar 2 19:39:07.557: ccSaveDialpeerTag (callID=0x9, dialpeer_tag= Mar 2 19:39:07.557: ccCallSetContext (callID=0xA, context=0x6244D9EC) Mar 2 19:39:07.557: ccCallReportDigits (callID=0x9, enable=0x0)

Неполадки IOS в шлюзе

В той же лабораторной работе Образ IOS 12.2 (6a) загружен на OGW. Когда вызов выполнен, замечено, что OGW все еще передает Удаление маркера на основе своего пароля даже при том, что шлюз не настроен для IVR для сбора Учетной записи/PIN. Кроме того, сторожевое устройство, настроенное для всего уровня, принимает тот вызов. Это задокументировано в идентификатор ошибки Cisco CSCdw43224 (только зарегистрированные клиенты).

Безопасность при чередовании конечных точек

Как отмечалось ранее, в этом документе, безопасности сквозного вызова предоставляют использование маркеров доступа, которые передаются в clearTokens поле в сообщениях RAS/H.225. При включении такой безопасности, исходный шлюз вперед маркер доступа,

полученный от сторожевого устройства в ACF к целевой оконечной точке H.323 в Сообщении SETUP H.225. Эта целевая оконечная точка H.323 тогда вперед маркер доступа, полученный в Сообщении SETUP к сторожевому устройству в его запросе на доступ. Путем выполнения этого это дает удаленному сторожевому устройству способность допустить вызовы на основе законности маркера доступа. Содержание маркера доступа до объекта, который генерирует его. Для уменьшения брешей системы безопасности и принять меры против атак по перехвату и возможному изменению передаваемых данных, сторожевые устройства могут закодировать целевую определенную информацию в маркере доступа. Это означает, что, когда alternateEndpoints предоставлены в ACF, сторожевое устройство может предоставить отдельный маркер доступа для каждого заданного alternateEndpoint.

Когда это сначала пытается установить соединение, шлюз Cisco передает маркер доступа, который это получило в clearToken поле ACF с адресом в destCallSignalAddress поле. Если эта попытка неуспешна, и шлюз Cisco продолжает делать попытку соединений с альтернативной оконечной точкой, это использует связанный маркер доступа (если это доступно) от списка alternateEndpoints. Если список alternateEndpoints, полученный в ACF, не включает маркеры доступа, но ACF включает маркер доступа, шлюз Cisco включает этот маркер доступа во все попытки соединиться с альтернативной оконечной точкой.

Поддержка OSP Token

В настоящее время Протокол OSP и его маркеры только поддерживаются на шлюзах Cisco. На сторожевом устройстве нет никакой поддержки. Шлюз распознает маркеры OSP, полученные от сервера группы, и вставляет их в сообщение SETUP Q.931 к конечному шлюзу.

Разные уровни безопасности для каждого конечного пункта или зоны

В настоящее время вы неспособны настроить разные уровни безопасности для каждой оконечной точки или зоны. Уровень безопасности для всех зон, которыми управляет то сторожевое устройство. Запрос новых функций может быть открыт для такой проблемы.

Interdomain Gatekeeper to Gatekeeper Security

Внутридоменное сторожевое устройство к безопасности сторожевого устройства предоставляет способность проверить внутридомен и внутридоменное сторожевое устройство к запросам сторожевого устройства на основе на переход. Это означает, что сторожевое устройство назначения завершает CAT и генерирует новый, если сторожевое устройство решает передать продвижения LRQ. Если сторожевое устройство обнаруживает недопустимую подпись LRQ, оно отвечает путем передачи Отказа в местонахождении (LRJ).

Сторожевое устройство внедрения к безопасности сторожевого устройства

Исходное сторожевое устройство генерирует IZCT, когда LRQ инициируется, или ACF собирается быть переданным в случае вызова внутренней зоны. Этот маркер пересечен через его путь маршрутизации. Вдоль пути каждое сторожевое устройство обновляет ID сторожевого устройства назначения и/или исходный идентификатор сторожевого устройства, при необходимости, для отражения зональной информации. Завершающееся сторожевое устройство генерирует маркер со своим паролем. Этот маркер приносят в

Location Confirmation (LCF) сообщения и передают к OGW. OGW включает этот маркер в Сообщение SETUP H.225. Когда TGW получает маркер, он передан в ARQ answerCall и проверен завершающимся сторожевым устройством (TGK) без любой потребности в сервере RADIUS.

Тип проверки подлинности основывается на пароле с хешированием, как описано в ITU H.235. В частности методом шифрования является MD5 с хешированием пароля.

Цель IZCT состоит в том, чтобы знать, поступил ли LRQ от внешнего домена, от который зона, и от который носитель. Это также используется для передачи маркера к OGW в LCF от TGK. В формате IZCT запрошена эта информация:

- srcCarrierID — Идентификация несущей источника
- dstCarrierID — Идентификация оператора назначения
- intCarrierID — Идентификация промежуточной несущей
- srcZone — Зона источника
- dstZone — Зона назначения
- межзональный типINTRA_DOMAIN_CISCOINTER_DOMAIN_CISCOINTRA_DOMAIN_TERM_NOT_CISCOINTER_DOMAIN_ORIG_NOT_CISCO

Эта функция хорошо работает без любой потребности в идентификаторе поставщика от шлюза или сервера Carrier Sensitive Routing (CSR). В таком случае поля об идентификаторе поставщика пусты. Примеры здесь не включают идентификатора поставщика. Для подробного потока вызовов выпуск и поддержка платформ и конфигурации, обращаются к Inter-Domain Gatekeeper Security Enhancement.

Конфигурация привратника

Функция IZCT требует этой конфигурации на сторожевом устройстве.

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

Пароль должен составить шесть - восемь символов. Определите, какая зона находится во внешнем домене как это:

```
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]
```

Поток вызова IZCT

Эта схема показывает поток IZCT.

В этой конфигурации названия шлюзов и сторожевых устройств совпадают с используемыми в диаграмме потока вызовов IZCT, но с нижним регистром. Поток вызовов объяснен после конфигурации с пояснениями отладки.

Для объяснения функции IZCT и потока вызовов первый пример не имеет безопасности соединения шлюза со сторожевым устройством внутри домена. После этого существуют примеры, где TGW не в состоянии генерировать IZCT так, чтобы TGK1 отклонил требование. Это должно показать, что функция работает, как разработано. Все эти настройки основываются на топологии в диаграмме потока вызовов IZCT.

Пример 1: Поток вызовов для сторожевого устройства к безопасности сторожевого

устройства только

Данный пример показывает связанные конфигурации всех шлюзов и сторожевых устройств.

Конфигурация OGW	Конфигурация 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

Конфигурация OGK1	Конфигурация TGK1
! hostname ogk1 ! interface Ethernet0/0 ip address 172.16.13.35 255.255.255.224 half-duplex ! gatekeeper zone local ogk1 domainA.com 172.16.13.35	! hostname tgk1 ! interface Ethernet0/0 ip address 172.16.13.41 255.255.255.224 ip directed-broadcast half-duplex ! gatekeeper zone local tgk1 domainB.com 172.16.13.41

```
 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
```

```
 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
```

Конфигурация 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
```

Конфигурация 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
```

Поток вызова с отладкой

Эти примеры используют отладки для объяснения потока вызовов.

1. Пользователь на носителе E вызывает пользователя на носителе 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. Начиная с dialpeer исходного шлюза (tag=3653) настроен для RAS, это передает ARQ к 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. Когда OGK1 получает ARQ, он решает, что назначение обслуживается удаленным

зональным OGK2. Это тогда определяет это, IZCT необходим (через CLI: **пароль security izct <pwd>**). OGK1 продолжает создавать IZCT, прежде чем будет передан LRQ. Это тогда отсылает IZCT и LRQ к OGK2 и передает сообщение RIP обратно в 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** L *!--- The destination is in the same domain, it is intraDomainCisco type.* **izctSrcZone "ogk1"** *!--- The source zone is ogk1.* **)** Mar 4 15:31:19.943: ENCODE BUFFER::= 07 00C06F67 6B310473 72630464 73740469 6E74 Mar 4 15:31:19.947: Mar 4 15:31:19.947: RAS OUTGOING PDU ::= value RasMessage ::= **locationRequest :** *!--- LRQ is sent out to ogk2.* { **requestSeqNum 2048** destinationInfo { e164 : "3653" } nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '8286B01100221B686C17CC11CC8011A049E05247...'H } replyAddress ipAddress : { ip 'AC100D23'H port 1719 } sourceInfo { h323-ID : {"ogk1"} } canMapAlias TRUE **tokens** *!--- The IZCT is included.* **{ { tokenOID { 1 2 840 113548 10 1 0 } nonStandard { nonStandardIdentifier { 1 2 840 113548 10 1 0 } data '0700C06F676B3104737263046473740469 6E74'H } } }** } Mar 4 15:31:19.967: RAS OUTGOING ENCODE BUFFER::= 4A 8007FF01 01806986 40B50000 12288286 B0110022 1B686C17 CC11CC80 11A049E0 52476601 80100204 8073B85A 5C56400 2 006F0067 007700AC 100D2306 B70BA00B 01400300 6F006700 6B003101 802B0100 80092A 86 4886F70C 0A010009 2A864886 F70C0A01 00130700 C06F676B 31047372 63046473 74046 96E 74 Mar 4 15:31:19.983: Mar 4 15:31:19.987: IPSOCK_RAS_sendto: msg length 122 from 172.16.13.35:1719 to 172.16.13.14: 1719 Mar 4 15:31:19.987: RASLib::RASSendLRQ: **LRQ (seq# 2048) sent to 172.16.13.14** Mar 4 15:31:19.987: RAS OUTGOING PDU ::= value RasMessage ::= **requestInProgress :** *!--- RIP message is sent back to OGW.* **{ requestSeqNum 1109** delay 9000 } Mar 4 15:31:19.991: RAS OUTGOING ENCODE BUFFER::= 80 05000454 2327 Mar 4 15:31:19.991: Mar 4 15:31:19.991: IPSOCK_RAS_sendto: msg length 7 from 172.16.13.35:1719 to 172.16.13.15: 57076 Mar 4 15:31:19.991: RASLib::RASSendRIP: RIP (seq# 1109) sent to 172.16.13.15

4. Когда OGK2 получает LRQ, он проверяет IZCT. От конфигурации это находит что LRQ neet также содержать IZCT. OGK2 тогда создает новый IZCT путем изменения izctSrcZone и izctDstZone, чтобы быть ogk2 и вперед LRQ к TGK2. После того, как это отошлет LRQ в TGK2, это передает сообщение RIP обратно в OGK1.Если сторожевые устройства являются частью кластера, имя кластера используется для SrcZone или 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 решает, что LRQ прибывает из внешнего домена. Это обновляет dstZone IZCT со своим собственным ID и interZoneType как INTER_DOMAIN_CISCO. Это тогда создает новый CAT и передает обновленный IZCT и LRQ к TGK1.TGK2 рассматривает зону, от которой LRQ получен как зона foreign-domain в любом из этих двух сценариев:Удаленный список зон TGK2 не содержит зону, от которой получен LRQ.Удаленный список зон TGK2 содержит зону, от которой получен LRQ. Зона отмечена флагом foreign-domain.Это тогда отправляет Запрос В Сообщении о прогрессе назад к 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. Обычно TGK1 обновляет dstCarrierID IZCT к Носителю E, который определен

процессом маршрутизации. Однако, так как никакие носители не используются, вы не видите это. TGK1 генерирует маркер хэша с паролем IZCT. Это передает LCF с обновленным IZCT в нем к OGK1. Этот izctHash используется для аутентификации ARQ answerCall, который TGK1 получает от TGW, когда позже получает сообщение SETUP VoIP от 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 '0001400200740067007706007400670006B003101...'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 извлекает IZCT из LCF и передает его в ACF к 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 t**okens** *!--- The IZCT is included.* **{ { tokenOID { 1 2 840 113548 10 1 0 } nonStandard { nonStandardIdentifier { 1 2 840 113548 10 1 0 } data '7F20C06F676B320674676B32C02B9620C7010310...'H } } }** willRespondToIRR FALSE uuiesRequested { setup FALSE callProceeding FALSE connect FALSE alerting FALSE information FALSE releaseComplete FALSE facility FALSE progress FALSE empty FALSE } } Mar 4 15:31:20.352: RAS OUTGOING ENCODE BUFFER::= 2B 00045440 028000AC 100D1706 B800EF1A 08C04801 0080092A 864886F7 0C0A0100 092A8648 86F70C0A 0100307F 20C06F67 6B320674 676B32C0 2B9620C7 0103105A 7D5E18AA 658A6A4B 4709BA5A BEF2B904 73726304 64737404 696E7401 00020000 Mar 4 15:31:20.364: Mar 4 15:31:20.364: IPSOCK_RAS_sendto: msg length 97 from 172.16.13.35:1719 to 172.16.13.15: 57076 Mar 4 15:31:20.368: RASLib::RASSendACF: **ACF (seq# 1109) sent to 172.16.13.15**

8. OGW передает IZCT к TGW в Сообщении 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. TGW передает IZCT к TGK1 в 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. TGK1 аутентифицирует целевой IZCT успешно. Это вызвано тем, что TGK1 генерирует хэш в IZCT, и это передает ACF обратно в 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. TGW устанавливает вызов к Носителю D после того, как это получит ACF.**Пример 2: Назовите подведенными, потому что TGW не в состоянии извлечь IZCT из полученного сообщения установки.**Данный пример основывается на той же топологии и конфигурации как Пример 1. В данном примере программное обеспечение TGW изменено на версию, где не поддерживается IZCT. В таком случае TGW не в состоянии извлечь IZCT из сообщения SETUP. Это заставляет TGK1 отклонять требование с причиной разъединения отказа в доступе из соображений безопасности.Данный пример показывает только сообщение SETUP, ARQ и ARJ на TGW, так как поток вызовов совпадает с Примером 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
```

Дополнительные сведения

- Inter-Domain Gatekeeper Security Enhancement
- Учет Cisco H.235 и улучшения безопасности для шлюзов Cisco
- Ссылка команды отладки Cisco IOS, релиз 12.3
- Поддержка голосовых технологий
- Поддержка продуктов Голосовой и Унифицированной связи
- Устранение неполадок в системах IP-телефонии Cisco
- Cisco Systems – техническая поддержка и документация