IAX2 Security

Document Version 1.0
09/03/09

Asterisk Development Team <asteriskteam@digium.com>

Table of Contents

1. Introduction

1.1. Overview

A change has been made to the IAX2 protocol to help mitigate denial of service attacks. This change is referred to as call token validation. This change affects how messages are exchanged and is not backwards compatible for an older client connecting to an updated server, so a number of options have been provided to disable call token validation as needed for compatibility purposes.

In addition to call token validation, Asterisk can now also limit the number of connections allowed per IP address to disallow one host from preventing other hosts from making successful connections. These options are referred to as call number limits.

For additional details about the configuration options referenced in this document, see the sample configuration file, `iax.conf.sample`. For information regarding the details of the call token validation protocol modification, see section 3 (Protocol Modification) of this document.

2. User Guide

2.1. Configuration

2.1.1. Quick Start

We strongly recommend that administrators leave the IAX2 security enhancements in place where possible. However, to bypass the security enhancements completely and have Asterisk work exactly as it did before, the following options can be specified in the `[general]` section of `iax.conf:`

```
[general]
…
calltokenoptional = 0.0.0.0/0.0.0.0
maxcallnumbers = 16382
…
```

2.1.2. Controlled Networks

This section discusses what needs to be done for an Asterisk server on a network where no unsolicited traffic will reach the IAX2 service.

2.1.2.1. Full Upgrade

If all IAX2 endpoints have been upgraded, then no changes to configuration need to be made.

2.1.2.2. Partial Upgrade

If only some of the IAX2 endpoints have been upgraded, then some configuration changes will need to be made for interoperability. Since this is for a controlled network, the easiest thing to do is to disable call token validation completely, as described in section 2.1.1.

2.1.3. Public Networks

This section discusses the use of the IAX2 security functionality on public networks where it is possible to receive unsolicited IAX2 traffic.

2.1.3.1. Full Upgrade

If all IAX2 endpoints have been upgraded to support call token validation, then no changes need to be made. However, for enhanced security, the administrator may adjust call number limits to further reduce the potential impact of malicious call number consumption. The following configuration will allow known peers to consume more call numbers than unknown source IP addresses:

```
[general]
; By default, restrict call number usage to a low number.
maxcallnumbers = 16
…

[callnumberlimits]
; For peers with known IP addresses, call number limits can
; be set in this section.  This limit is per IP address for
; addresses that fall in the specified range.
;    <IP>/<mask> = <limit>
192.168.1.0/255.255.255.0 = 1024
…

[peerA]
```

```
; Since we won't know the IP address of a dynamic peer until
; they register, a max call number limit can be set in a
; dynamic peer configuration section.
Type = peer
host = dynamic
maxcallnumbers = 1024
…
```

2.1.3.2. Partial Upgrade

If only some IAX2 endpoints have been upgraded, or the status of an IAX2 endpoint is unknown, then call token validation must be disabled to ensure interoperability. To reduce the potential impact of disabling call token validation, it should only be disabled for a specific peer or user as needed. By using the `auto` option, call token validation will be changed to required as soon as we determine that the peer supports it.

```
[friendA]
requirecalltoken = auto
…
```

Note that there are some cases where `auto` should not be used. For example, if multiple peers use the same authentication details, and they have not all upgraded to support call token validation, then the ones that do not support it will get locked out. Once an upgraded client successfully completes an authenticated call setup using call token validation, Asterisk will require it from then on. In that case, it would be better to set the `requirecalltoken` option to `no`.

2.1.3.3. Guest Access

Guest access via IAX2 requires special attention. Given the number of existing IAX2 endpoints that do not support call token validation, most systems that allow guest access should do so without requiring call token validation.

```
[guest]
; Note that the name “guest” is special here.  When the code
; tries to determine if call token validation is required, it
; will look for a user by the username specified in the
; request.  Guest calls can be sent without a username.  In
; that case, we will look for a defined user called “guest” to
; determine if call token validation is required or not.
type = user
```

```
requirecalltoken = no
…
```

Since disabling call token validation for the guest account allows a huge hole for malicious call number consumption, an option has been provided to segregate the call numbers consumed by connections not using call token validation from those that do. That way, there are resources dedicated to the more secure connections to ensure that service is not interrupted for them.

```
[general]
maxcallnumbers_nonvalidated = 2048
```

2.2. CLI Commands

2.2.1. iax2 show callnumber usage

```
Usage: iax2 show callnumber usage [IP address]
       Show current IP addresses which are consuming IAX2 call
numbers.
```

2.2.2. iax2 show peer

This command will now also show the configured call number limit and whether or not call token validation is required for this peer.

3. Protocol Modification

This section discusses the modification that has been made to the IAX2 protocol. This information would be most useful to implementors of IAX2.

3.1. Overview

The IAX2 protocol uses a call number to associate messages with which call they belong to. The available amount of call numbers is finite as defined by the protocol. Because of this, it is important to prevent attackers from maliciously consuming call numbers. To achieve this, an enhancement to the IAX2 protocol has been made which is referred to as call token validation.

Call token validation ensures that an IAX2 connection is not coming from a spoofed IP address. In addition to using call token validation, Asterisk will also limit how many call numbers may be consumed by a given remote IP address. These limits have defaults that will usually not need to be changed, but can be modified for a specific need.

The combination of call token validation and call number limits is used to mitigate a denial of service attack to consume all available IAX2 call numbers. An alternative approach to securing IAX2 would be to use a security layer on top of IAX2, such as DTLS [RFC4347] or IPsec [RFC4301].

3.2. Call Token Validation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

For this section, when the word "request" is used, it is referring to the command that starts an IAX2 dialog.

This modification adds a new IAX2 frame type, and a new information element be defined.

```
Frame Type: CALLTOKEN --- 0x28 (40)
IE:         CALLTOKEN --- 0x36 (54)
```

When a request is initially sent, it SHOULD include the `CALLTOKEN` IE with a zero-length payload to indicate that this client supports the `CALLTOKEN` exchange. When a server receives this request, it MUST respond with the IAX2 message `CALLTOKEN`. The `CALLTOKEN` message MUST be sent with a source call number of 0, as a call number will not yet be allocated for this call.

For the sake of backwards compatibility with clients that do not support token validation, server implementations MAY process requests that do not indicate `CALLTOKEN` support in their initial request. However, this SHOULD NOT be the default behavior, as it gives up the security benefits gained by `CALLTOKEN` validation.

After a client sends a request with an empty `CALLTOKEN` IE, it MUST be prepared to receive a `CALLTOKEN` response, or to receive a response that would be given in the case of a valid `CALLTOKEN`. This is how a client must behave to inter operate with IAX2 server implementations that do not yet support `CALLTOKEN` validation.

When an IAX2 client receives a `CALLTOKEN` response, it MUST send its initial request again. This request MUST include the `CALLTOKEN` IE with a copy of the value of the `CALLTOKEN` IE received in the `CALLTOKEN` response. The IE value is an opaque value. Clients MUST be able to accept a `CALLTOKEN` payload of any length, up to the maximum length allowed in an IAX2 IE.

The value of the payload in the `CALLTOKEN` IE is an implementation detail. It is left to the implementor to decide how sophisticated it should be. However, it MUST be enough such that when the `CALLTOKEN` IE is sent back, it can be used to verify that the source IP address and port number has not been spoofed.

If a server receives a request with an invalid `CALLTOKEN` IE value, then it MUST drop it and not respond.

3.3. Example Message Exchanges

3.3.1. Call Setup

```
 Client                              Server
--------                            --------
   ------------- NEW ----------->
      (with empty CALLTOKEN IE)

   <---------- CALLTOKEN ------------
         (client must ignore
          source call number
          from this message)

   ------------- NEW ----------->
         (with received token)

   <---------- AUTHREQ ----------

      ... continue as normal …
```

3.3.2. Call Setup, client does not support CALLTOKEN

```
 Client                              Server
--------                            --------
```

```
   ------------- NEW ----------->
       (with no CALLTOKEN IE)

   <----------- REJECT ----------
      (sent without allocating
       a call number)

   ------------- ACK ----------->
```

3.3.3. Call Setup, client supports CALLTOKEN, server does not

```
 Client                        Server
--------                      --------
   ------------- NEW ----------->
      (with empty CALLTOKEN IE)

   <----------- AUTHREQ ---------
      (sent without allocating
       a call number)

      ... continue as normal …
```

3.3.4. Call Setup from client that sends invalid token

```
 Client                        Server
--------                      --------
   ------------- NEW ----------->
    (with invalid CALLTOKEN IE)

           ... dropped …
```

4. Asterisk Implementation

This section includes some additional details on the implementation of these changes in Asterisk.

4.1. CALLTOKEN IE Payload

For Asterisk, we will encode the payload of the `CALLTOKEN` IE such that the server is able to validate a received token without having to store any information after transmitting the `CALLTOKEN` response. The `CALLTOKEN` IE payload will contain:

- A timestamp (epoch based)
- SHA1 hash of the remote IP address and port, the timestamp, as well some random data generated when Asterisk starts.

When a `CALLTOKEN` IE is received, its validity will be determined by recalculating the SHA1 hash. If it is a valid token, the timestamp is checked to determine if the token is expired. The token timeout will be hard coded at 10 seconds for now. However, it may be made configurable at some point if it seems to be a useful addition. If the server determines that a received token is expired, it will treat it as an invalid token and not respond to the request.

By using this method, we require no additional memory to be allocated for a dialog, other than what is on the stack for processing the initial request, until token validation is complete.

However, one thing to note with this `CALLTOKEN` IE encoding is that a token would be considered valid by Asterisk every time a client sent it until we considered it an expired token. However, with use of the "`maxcallnumbers`" option, this is not actually a problem. It just means that an attacker could hit their call number limit a bit quicker since they would only have to acquire a single token per timeout period, instead of a token per request.