Access Control List (ACL)

1. Overview

An **access control list** (ACL) is a list of permissions (or **rules**) associated with an object where the list defines what network entities are allowed to access the object.

1.1 Rules

Rules specifically **allow** or **deny** access based on the provided parameters.

Their priority depends on how specific they are (i.e., more specific rules enjoy higher priority than less specific ones).

```
192.168.1.10        most specific
192.168.1.0/24        less specific
10.1.1.0/16         least specific
```

1.2 Main ACL configuration file

Additional lists can be defined in `${conf_dir}/autoload_configs/acl.conf.xml`.

The default configuration file from the vanilla configuration:

TODO Link to a configuration page for `${conf_dir}` (or how to find it, i.e., `:`)

acl.conf.xml

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <!--
         These ACLs are automatically created on startup.

         rfc1918.auto  - RFC1918 Space
         nat.auto      - RFC1918 Excluding your local lan.
         localnet.auto - ACL for your local lan.
         loopback.auto - ACL for your local lan.
    -->

    <list name="lan" default="allow">
      <node type="deny" cidr="192.168.42.0/24"/>
      <node type="allow" cidr="192.168.42.42/32"/>
    </list>

    <!--
        This will traverse the directory adding all users
        with the cidr= tag to this ACL, when this ACL matches
        the users variables and params apply as if they
        digest authenticated.
    -->
    <list name="domains" default="deny">
      <!-- domain= is special it scans the domain from the directory to build the ACL -->
      <node type="allow" domain="$${domain}"/>
      <!-- use cidr= if you wish to allow ip ranges to this domains acl. -->
      <!-- <node type="allow" cidr="192.168.0.0/24"/> -->
    </list>

  </network-lists>
</configuration>
```

1.3 Pre-defined ACLs

There are some ACLs automatically created on startup:

ACL name	Description
`rfc1918.auto`	RFC 1918 Space
`nat.auto`	RFC 1918, excluding your local LAN
`localnet.auto`	ACL for your local LAN
`loopback.auto`	ACL for your local LAN

TODO Link sources of pre-defined ACLs.

⚠ `localnet.auto` only defines a local network, and doesn't interfere or authenticate any calls by default as other ACLs. If you use the internal profile on a public IP which accepts calls from other servers then it doesn't hurt leaving it at `localnet.auto`, but **the best way to prevent unauthorized calls is using a firewall**.

✓ One way to use these auto generated ACLs is by

1. activating them in `${conf_dir}/sip_profiles/` (see Sofia SIP Stack):

```
<param name="local-network-acl" value="localnet.auto"/>
<param name="apply-inbound-acl" value="localnet.auto"/>
```

2. then using them, for example in `${conf_dir}/autoload_configs/acl.conf.xml`:

```
<list name="localnet.auto" default="allow">
  <node type="allow" cidr="41.XXX.XXX.XXX/29"/>
</list>
```

2. General structure

2.1 `network-lists` tag

TODO How are individual ACLs processed? E.g., if none of the node rules match, the ACLs default action gets executed.

ACL rules are usually defined in `${conf_dir}/autoload_configs/acl.conf.xml` in a `network-lists` element, that is a container for a list of ACLs.

network-lists tag

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <!-- ACL 1 -->
    <!-- ...   -->
    <!-- ACL n -->
  </network-lists>
</configuration>
```

2.2 `list` tag

The `list` element is where an ACL is declared, listing all rules that belong to it.

list tag

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <list name="<ACL_name>" default="[allow|deny]">
      <!-- rule 1 -->
          <!-- ...    -->
      <!-- rule n -->
    </list>
  </network-lists>
</configuration>
```

Attribute	Description	Accepted values	Example
`name`	The arbitrary name given to the ACL.	String	`name="test"`
`default`	The default ACL action. It specifies whether incoming actions should be allowed or denied access when none of the ACL's rules match. ⚠ This implies that a rule's action (see **2.3 node tag** section) always overrides the ACL's default action.	`[ allow | deny ]`	`default="allow"`

2.3 node tag

The individual rules that an ACL contains are declared in `node` tags.

node tag

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <list name="<ACL_name>" default="<action>">
      <node type="<action>" cidr="<IP_address_in_CIDR_notation>"/>
      <!-- rule 2 -->
      <!-- ...    -->
      <!-- rule n -->
    </list>
  </network-lists>
</configuration>
```

Attribute	Description	Accepted values	Example
`type`	Specifies the action to be taken when this rule matches the IP address under test. ⚠ The rule's action always overrides the ACL's default actions (see **2.2 `list` tag** section).	`[ allow | deny ]`	`type="allow"`

<table>
<tr>
<td><code>cidr</code></td>
<td>Match an incoming connection by their IP address. Multiple ranges need to be separated by a comma.

Overlapping IP address ranges

In the case of overlapping IP addresses, the rule with the more specific range will take precedence.

For example, <code>NODE A</code> will win over <code>NODE B</code> in the same list below.

acl.conf.xml
<pre><list name="sample" default="allow">
  <!-- NODE A -->
  <node type="allow" cidr="192.168.42.42/32"/>
  <!-- NODE B -->
  <node type="deny" cidr="192.168.42.0/24"/>
</list></pre>
IPv6 ACL definitions are only supported in FreeSWITCH vesion 1.0.7 and later.</td>
<td>list of IP address ranges using CIDR notation</td>
<td><code>cidr="1.2.3.0/24"</code>

<code>cidr="12.34.56.78/32,20.0.0.0/8"</code></td>
</tr>
<tr>
<td><code>domain</code></td>
<td>Scans the user definitions of the specified domain from the directory, and if your domain's users have <code>cidr</code> attributes, the ACL will be automatically built.

Example usage:

<pre><node type="allow" domain="$${domain}"/></pre>
See section 3.3 Domain user example for more.

Directory

User definitions are usually found in <code>${conf_dir}/directory/default/*.xml</code> (see XML User Directory).

Channel variables

For <code>$${domain}</code>, see Channel Variables and vars.xml. (The <code>domain</code> variable is an alias to the <code>domain_name</code> variable.)</td>
<td>Any domain name acceptable by FreeSWITCH.</td>
<td><code>domain="$${domain}"</code></td>
</tr>
</table>

Beware that applying the domain attribute to users changes the behavior of the sofia state machine. If you find users in the specified domain starting in the public context with an empty user_context variable, check here first.

3. ACL examples

3.1 Sample allow

allows access from anyone on 1.2.3.*

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <list name="test1" default="deny">
      <node type="allow" cidr="1.2.3.0/24"/>
    </list>
  </network-lists>
</configuration>
```

3.2 Sample deny

allows access from anyone except 4.3.2.*

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <list name="test2" default="allow">
      <node type="deny" host="4.3.2.0" mask="255.255.255.0"/>
    </list>
  </network-lists>
</configuration>
```

3.3 Domain user example

It is possible to automatically add users with a CIDR attribute to an ACL list. This is particularly useful for authenticating people by static IP address instead of using challenge authentication.

1. First of all, make sure you have the following in `${conf_dir}/autoload_configs/acl.conf.xml`
   (the Vanilla config already does; see **1.2 Main ACL configuration file** section)

   Automatically add users with CIDR= attribute

   ```
   <list name="domains" default="deny">
     <node type="allow" domain="$${domain}"/>
   </list>
   ```

   The node element with the `domain` attribute tells the ACL module to look into that FreeSWITCH domain to insert ACL entries.

   ⚠ If you have a multi-domain (multi-tenant) FreeSWITCH configuration, make sure you add `node` elements for all your domains.

2. The next step is creating a user with the CIDR attribute.

   ✓ You can separate multiple CIDRs with a comma.

User directory entry with CIDR

```
<include>
  <user id="1000" cidr="12.34.56.78/32,20.0.0.0/8">
    <params>
      <param name="password" value="1234"/>
      <param name="vm-password" value="1000"/>
    </params>
    <variables>
      <variable name="accountcode" value="1000"/>
      <variable name="user_context" value="default"/>
      <variable name="effective_caller_id_name" value="Extension 1000"/>
      <variable name="effective_caller_id_number" value="1000"/>
    </variables>
  </user>
</include>
```

3. The last step is to verify that your channel driver has been instructed to use this ACL. For Sofia, you should see the following line in your `${conf_dir}/sip_profiles/` (as noted above):

SIP profile definition

```
 <param name="apply-inbound-acl" value="domains"/>
```

4. Additionally, you can restrict a user to a predefined CIDR without allowing the whole CIDR block. Users in the directory can have `auth-acl` parameters applied to them so as to restrict that user's access to a predefined ACL or a CIDR.

User directory auth-acl restriction

```
<param name="auth-acl" value="1.2.3.0/8"/>
```

⚠ This will require `auth-acl` to be set to true in your Sofia SIP profile.

TODO `auth-acl` is only semi-documented in Sofia Configuration Files.

Example:

```
<include>
  <user id="1000" number-alias="1000">
    <params>
      <param name="password" value="1234"/>
      <param name="vm-password" value="1000"/>
      <param name="auth-acl" value="1.2.3.0/8"/>
    </params>
    <variables>
      <variable name="accountcode" value="1000"/>
      <variable name="user_context" value="default"/>
      <variable name="effective_caller_id_name" value="Extension 1000"/>
      <variable name="effective_caller_id_number" value="1000"/>
    </variables>
  </user>
</include>
```

TODO Figure out how to incorporate the rest of this page from this point.

4. Application

Access control lists may be applied in

- SIP profiles, via the Event Socket Layer from a script, or in a dialplan application
- the `mod_event_socket` configuration file, `event_socket.conf.xml`

TODO Where else?

4.1 Sample usage in mod_event_socket

event_socket.conf.xml

```
<configuration name="event_socket.conf" description="Socket Client">
  <settings>
    <param name="nat-map" value="false"/>

    <!-- ::1 is the IPv6 version of 127.0.0.0/8 in IPv4 -->
    <param name="listen-ip" value="::1"/>
    <param name="listen-port" value="8021"/>
    <param name="password" value="ClueCon"/>

    <!-- Using the predefined `loopback.auto` ACL -->
    <param name="apply-inbound-acl" value="loopback.auto"/>
    <!--<param name="stop-on-bind-error" value="true"/>-->

  </settings>
</configuration>
```

sip_profile settings (see mod_sofia)

These Access Control Lists are named in `${conf_dir}`/autoload_configs/acl.conf.xml and applied in sip_profiles/internal.xml and sip_profiles/external.xml

apply-inbound-acl

Allow users to make calls from a particular CIDR without authenticating

`Usage: <param name="apply-inbound-acl" value="<list name>"/>`

<list name> is set in acl.conf.xml and defines the subnet that will be processed by the ACL bearing this name. The default name is "domains".

apply-register-acl

Allow users to register from a particular CIDR without authenticating.

apply-proxy-acl

Use the IP specified in X-AUTH-IP header sent from proxy for apply-inbound-acl

> You'll need to configure your proxy to add this header.

apply-candidate-acl

ICE candidates for RTP transport are checked against this list. It defaults to wan.auto if unset, which excludes the LAN.

auth-calls

Can be set to true/false forcing users to authenticate or no on the profile. Only allow users from a specific CIDR to register/make calls. Note: Currently auth-calls does not work with registrations/invites through a proxy. You'll need to do this inside your xml_curl directory scripts or on your proxy.

Directory settings:

`<user id="1000" number-alias="1000" cidr="12.34.56.78/32,20.0.0.0/8">`

Used with in conjunction with apply-inbound-acl and apply-register-acl.

```
<param name="auth-acl" value="1.2.3.0/8"/>
```

Used in conjunction with auth-calls.

`event_socket.conf.xml` parameters (for `mod_event_socket`)

apply-inbound-acl

See above.

stop-on-bind-error

TODO From the vanilla event_socket.conf.xml. Where is it documented? `stop-on-bind-error` for example seems specific only to `mod_event_socket` and `mod_erlang_event` (at least, the name only pops up in their source), and the same goes to `auth-calls` that is seemingly specific to `mod_sofia`.

Services

Event Socket

See Event Socket

Sofia

See Sofia

Sofia SIP profiles

In your SIP (Sofia) profiles, you can use the following lines to apply the ACL setting to incoming requests for either REGISTERs or INVITEs (or both).

```
  <param name="apply-inbound-acl" value="<acl_list|cidr>"/>
  <param name="apply-register-acl" value="<acl_list|cidr>"/>
```

More than one ACL can be defined, in that case all the ACLs will be tested and the message will be rejected if any of the ACLs fail (within an acl_list the test is an OR, with multiple params the test is an AND of all the ACLs)

Phones having IPs within these ACLs will be able to perform calls (apply-inbound-acl) or register (apply-register-acl) without having to provide a password (i.e. without getting a "401 Unauthorized" challenge message).

Those ACLs do not block any traffic. Should you want to protect your FreeSWITCH installation from being contacted by some IP addresses, you will need to setup some firewall rules. To protect your installation, you can look at QoS

Should you want to allow everyone to call your FreeSWITCH installation but restrict outgoing calls, this should be done in the dialplan see mod_dptools: respond.

The ACL behavior is modified by `auth-calls`, `accept-blind-reg`, and `accept-blind-auth`.

You can also specify a C-style ternary test <list name>:<pass context>:<fail context> for `apply-inbound-acl`.

Dialplan Apps

check_acl

See mod_dptools: check_acl

API Commands

reloadacl

reloadacl [<reloadxml>]

```
freeswitch@internal> reloadacl reloadxml
```

If you've made a change to an **existing list** in acl.conf.xml, you can run 'reloadacl reloadxml' in that sequence to avoid restarting FreeSWITCH and your new change will be effective.

Commands reloadxml and reloadacl do not load new access control lists. You must restart FreeSWITCH to recognize the newly added ACL name.

acl

```
acl <ip> <list|net>
```

This command will allow you to test an IP address against one of your ACLs. Will return true or false. Use it to validate that your ACL behaves as expected. This test can also be a part of a dialplan <condition> test.

```
  freeswitch@mybox> acl 192.168.42.42 192.168.42.0/24
  freeswitch@mybox> acl 192.168.42.42 list_foo
```

For the second line, 'list_foo' refers to the <list name=> that you specified in acl.conf.xml. When you change acl.conf.xml you must restart the FreeSWITCH process. **Commands reloadxml and reloadacl do not load new lists**.

Routing using ACL can be accomplished using the acl command. For example, if you want to pass calls for hosts in list_foo ACL:

Dialplan test condition using ACL

```
<extension name="foo-hosts-calls">
  <condition field="${acl(${network_addr} list_foo)}" expression="true"/>
  <condition field="destination_number" expression="(.*)">
    <action application="bridge" data="sofia/switchbox/$1@myapp.signalwire.com:5060"/>
  </condition>
</extension>
```