Cisco − Lock−and−Key (Dynamic Access Lists)

Table of Contents

Lock−and−Key: Dynamic Access Lists 1
- Contents 1
- Introduction 1
- Spoofing Considerations 1
- Performance 1
- When to Use Lock−and−Key Access 2
- Lock−and−Key Access Operation 2
- Sample Configuration and Troubleshooting 2
  - Using TACACS+ 4
  - Using RADIUS 5
- Tools Information 5
- Related Information 5

Lock−and−Key: Dynamic Access Lists

Contents

Introduction
Spoofing Considerations
Performance
When to Use Lock−and−Key Access
Lock−and−Key Access Operation
Sample Configuration and Troubleshooting

Using TACACS+
Using RADIUS
Tools Information
Related Information

Introduction

Lock−and−key access allows you to set up dynamic access lists that grant access per user to a specific source/destination host through a user authentication process. You can allow user access through a firewall dynamically, without compromising security restrictions.

Spoofing Considerations

Lock−and−key access allows an external event to place an opening in the firewall. After this opening exists, the router is susceptible to source address spoofing. To prevent this, you need to provide encryption support using IP encryption with authentication or encryption.

Spoofing is a problem with all existing access lists. Lock−and−key access does not address this problem.

Because lock−and−key access introduces a potential pathway through your network firewall, you need to consider dynamic access. Another host, spoofing your authenticated address, might gain access behind the firewall. With dynamic access, there is the possibility that an unauthorized host, spoofing your authenticated address, can gain access behind the firewall. Lock−and−key access does not cause the address spoofing problem. The problem is only identified here as a concern to the user.

Performance

Performance is affected in the following two situations:

- Each dynamic access list forces an access list rebuild on the silicon switching engine (SSE). This causes the SSE switching path to slow down momentarily.
- Dynamic access lists require the idle timeout facility (even if the timeout is left to default), so

dynamic access lists cannot be SSE switched. These entries must be handled in the protocol fast−switching path.

Pay close attention to the border router configurations. Remote users create access list entries on the border router. The access list will grow and shrink dynamically. Entries are dynamically removed from the list after either the idle−timeout or max−timeout period expires. Large access lists degrade packet switching performance.

When to Use Lock−and−Key Access

Two examples of when you might use lock−and−key access are listed below:

- When you want a remote host to be able to access a host in your internetwork via the Internet. Lock−and−key access limits access beyond your firewall on an individual host or net basis.
- When you want a subset of hosts on a network to access a host on a remote network protected by a firewall. With lock−and−key access, you can enable only a desired set of hosts to gain access by having them authenticate through a TACACS+ or RADIUS server.

Lock−and−Key Access Operation

The following process describes the lock−and−key access operation:

1. A user opens a Telnet session to a border router configured for lock−and−key access.
2. The Cisco IOS software receives the Telnet packet and performs a user authentication process. The user must pass authentication before access is allowed. The authentication process can be done by the router or a central access server such as a TACACS+ or RADIUS server.

Sample Configuration and Troubleshooting

It is highly recommended that you use a TACACS+ server for your authentication query process. TACACS+ provides authentication, authorization, and accounting services. It also provides protocol support, protocol specification, and a centralized security database.

You can authenticate the user on the router or with a TACACS+ or RADIUS server. First, we'll describe configuring this from the router and later add the commands needed to do this with TACACS+ or RADIUS.

Note: The following commands are global unless otherwise indicated.

On the router, you will need to have a **username** for the user for local authentication.

username test password test

Having **login local** on the vty lines causes this username to be used.

line vty 0 4
login local

Because you may not trust the user to issue the **access−enable** command, you can do one of two things, as indicated below.

- Associate the timeout with the user on a per−user basis.:

  username test autocommand access−enable host timeout 10

  or

- Force all users Telnetting in to have the same timeout.

line vty 0 4
login local
autocommand access−enable host timeout 10

Note: The **10** in the syntax above is the *idle* timeout of the access list and can be overriden by the absolute timeout in the dynamic access list.

You then define an extended access list that will be applied when a user (any user) logs into the router and the **access−enable** command is issued. The maximum absolute time for this "hole" in the filter is being set to 15 minutes. After 15 minutes, the hole will close whether or not anyone is using it. The name **testlist** needs to exist but is not significant. You can limit the networks to which the user has access by configuring the source or destination address (here, we have not limited the user):

access−list 120 dynamic testlist timeout 15 permit ip any any

Then, you define the access list needed to block everything except the ability to Telnet into the router (in order to open a hole, the user needs to Telnet to the router). The IP address here is the Ethernet IP address of the router:

access−list 120 permit tcp any host 171.68.117.189 eq telnet

Keep in mind that there is an implicit "deny all" at the end (even though we have not entered it here).

Now, apply this access list to the interface on which users are coming in:

```
interface ethernet1
    ip access−group 120 in
```

You're done.

Here is what the filter looks like on the router right now:

```
Router# show ip access−lists 120
Extended IP access list 120
    Dynamic testlist Max. 15 mins. permit ip any any timeout 15 min.
    permit tcp any host 171.68.117.189 eq telnet (243 matches)
```

Now, users getting access to your internal network will not be able to see anything until they Telnet to the router.
Note: The "10" here is the *idle* timeout of the access list and can be overriden by the absolute timeout in the dynamic access list.

```
% telnet 2514A
Trying 171.68.117.189 ...
Connected to 2514A.network.com.
Escape character is '^]'.


User Access Verification

Username: test
Password: test

Connection closed by foreign host.
```

Now the filter looks like this:

```
Router# show ip access-lists 120
Extended IP access list 120
    Dynamic testlist Max. 15 mins. permit ip any any timeout 15 min.
       permit ip host 171.68.109.158 any idle-time 10 min. (590 matches)
    permit tcp any host 171.68.117.189 eq telnet (421 matches)
```

You can see that there is a hole in the filter for this one user based on the source IP address.

When someone else does this, you should see *two holes*:

```
Router# show ip access-lists 120
Extended IP access list 120
    Dynamic testlist Max. 15 mins. permit ip any any timeout 15 min.
       permit ip host 171.68.109.64 any idle-time 10 min. (10 matches)
       permit ip host 171.68.109.158 any idle-time 10 min. (23 matches)
    permit tcp any host 171.68.117.189 eq telnet (725 matches)
```

And these users should be able to have complete IP access to any destination IP address from their *source* IP address.

Using TACACS+

In order to use TACACS+, you need to configure a TACACS+ server to force authentication and authorization to be done on the TACACS+ server, as shown below:

```
tacacs-server host 111.111.111.111
tacacs-server key aaaaaaa123

aaa authentication login default tacacs+ local
aaa authorization exec default tacacs+
```

The user's configuration on the TACACS+ server should look something like this:
Note: This configuration is for the TACACS+ freeware daemon; CiscoSecure software would be set up similarly with EXEC privileges and autocommand access-enable.

```
# This user is a Lock and Key user. Lock and Key must also
# be configured on the NAS. Timeout for the user is 10 minutes.
user = test {
        login = cleartext "test"
        service = exec {
                autocmd = "access-enable host timeout 10"
        }
```

```
}
```

Using RADIUS

In order to use RADIUS, you need to configure a RADIUS server to force authentication to be done on the RADIUS server with authorization parameters (the autocommand) to be sent down in vendor−specific attribute 26), as shown below:

```
radius−server host 111.111.111.111
radius−server key aaaaaaa123
```

aaa authentication login default radius local aaa authorization exec default radius

The user's configuration on the RADIUS server should look something like this:
Note: This configuration is from a RADIUS freeware daemon; CiscoSecure software would be sent up similarly with Service−type = Shell−user and shell autocommand access−enable:

```
LK      Password = "LK"
        User−Service−Type = Shell−User,
        cisco−avpair = "shell:autocmd=access−enable host timeout 10"
```

Tools Information

For additional resources, refer to Cisco TAC Tools for Security Technologies.

Related Information

- **Cisco IOS Software documentation**
- More Technical Tips for **TACACS+**, **RADIUS**, and **Access Lists**
- Support Pages for **TACACS+** and **RADIUS**
- **Lock−and−Key Security (Dynamic Access Lists) White Paper**

Updated: Mar 07, 2002 Document ID: 7604