RSA Token Server and SDI Protocol Usage for ASA and ACS

Contents

Introduction

This document describes troubleshooting procedures for the RSA Authentication Manager, which can be integrated with the Cisco Adaptive Security Appliance (ASA) and the Cisco Secure Access Control Server (ACS).

The RSA Authentication Manager is a solution that provides the One Time Password (OTP) for authentication. That password is changed every 60 seconds and can be used only once. It supports both hardware and software tokens.

Prerequisites

Requirements

Cisco recommends that you have basic knowledge of these topics:

- Cisco ASA CLI configuration
- Cisco ACS configuration

Components Used

The information in this document is based on these software versions:

- Cisco ASA software, Version 8.4 and later
- Cisco Secure ACS, Version 5.3 and later

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Theory

The RSA server can be accessed with RADIUS or the proprietary RSA protocol: SDI. Both the ASA and the ACS can use both protocols (RADIUS, SDI) in order to access the RSA.

Remember that the RSA can be integrated with the Cisco AnyConnect Secure Mobility Client when a software token is used. This document focuses solely on ASA and ACS integration. For more information about AnyConnect, refer to the Using SDI Authentication section of the Cisco AnyConnect Secure Mobility Client Administrator Guide, Release 3.1.

RSA via RADIUS

RADIUS has one big advantage over SDI. On the RSA, it is possible to assign specific profiles (called groups on ACS) to users. Those profiles have specific RADIUS attributes defined. After successful authentication, the RADIUS-Accept message returned from the RSA contains those attributes. Based on those attributes, the ACS makes additional decisions. The most common scenario is the decision to use ACS Group Mapping in order to map specific RADIUS-attributes, related to the profile on the RSA, to a specific group on the ACS. With this logic, it is possible to move the whole authorization process from the RSA to the ACS and still maintain granular logic, as on the RSA.

RSA via SDI

SDI has two main advantages over RADIUS. The first is that the whole session is encrypted. The second is the interesting options that the SDI agent provides: it is able to determine if the failure is created because authentication or authorization failed or because the user was not found.

This information is used by the ACS in action for identity. For example, it could continue for "user not found" but reject for "authentication failed."

There is one more difference between RADIUS and SDI. When a Network Access Device like ASA uses SDI, the ACS performs only authentication. When it uses RADIUS, the ACS performs authentication, authorization, accounting (AAA). However, this is not a big difference. It is possible to configure SDI for authentication and RADIUS for accounting for the same sessions.

SDI Protocol

By default, SDI uses User Datagram Protocol (UDP) 5500. SDI uses a symmetric encryption key, similar to the RADIUS key, in order to encrypt sessions. That key is saved in a node secret file and is different for every SDI client. That file is deployed manually or automatically.

Note: ACS/ASA does not support manual deployment.

For the automatic deployment node, the secret file is downloaded automatically after the first successful authentication. The node secret is encrypted with a key derived from the user's passcode and other information. This creates some possible security issues, so the first authentication must be performed locally and use encrypted protocol (Secure Shell [SSH], not telnet) in order to ensure that the attacker cannot intercept and decrypt that file.

Configuration

Notes:

Use the Command Lookup Tool (registered customers only) in order to obtain more information on the commands used in this section.

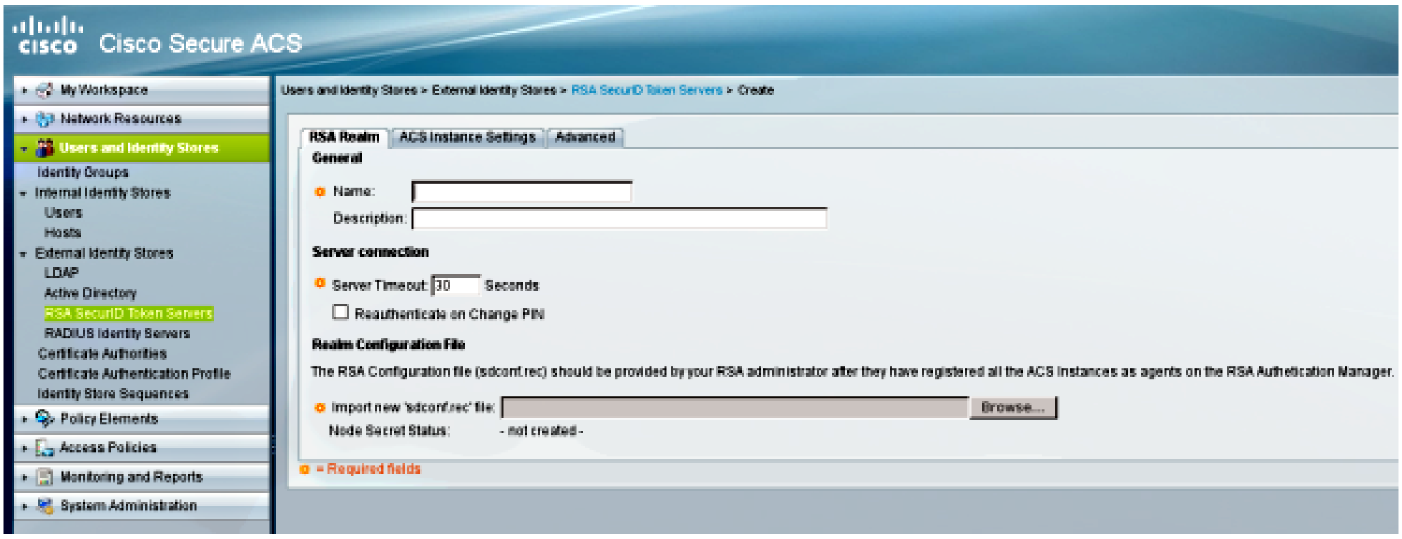
The Output Interpreter Tool (registered customers only) supports certain **show** commands. Use the Output Interpreter Tool in order to view an analysis of **show** command output.

Refer to Important Information on Debug Commands before you use **debug** commands.

SDI on ACS

It is configured in **Users and Identity Stores > External Identity Store > RSA Secure ID Token Servers**.

The RSA has multiple replica servers, such as the secondary servers for the ACS. There is no need to put all the addresses there, just the **sdconf.rec** file provided by the RSA administrator. This file includes the IP address of the primary RSA server. After the first successful authentication node, the secret file is downloaded along with the IP addresses of all RSA replicas.



Users and Identity Stores > External Identity Stores > RSA SecurID Token Servers > Create

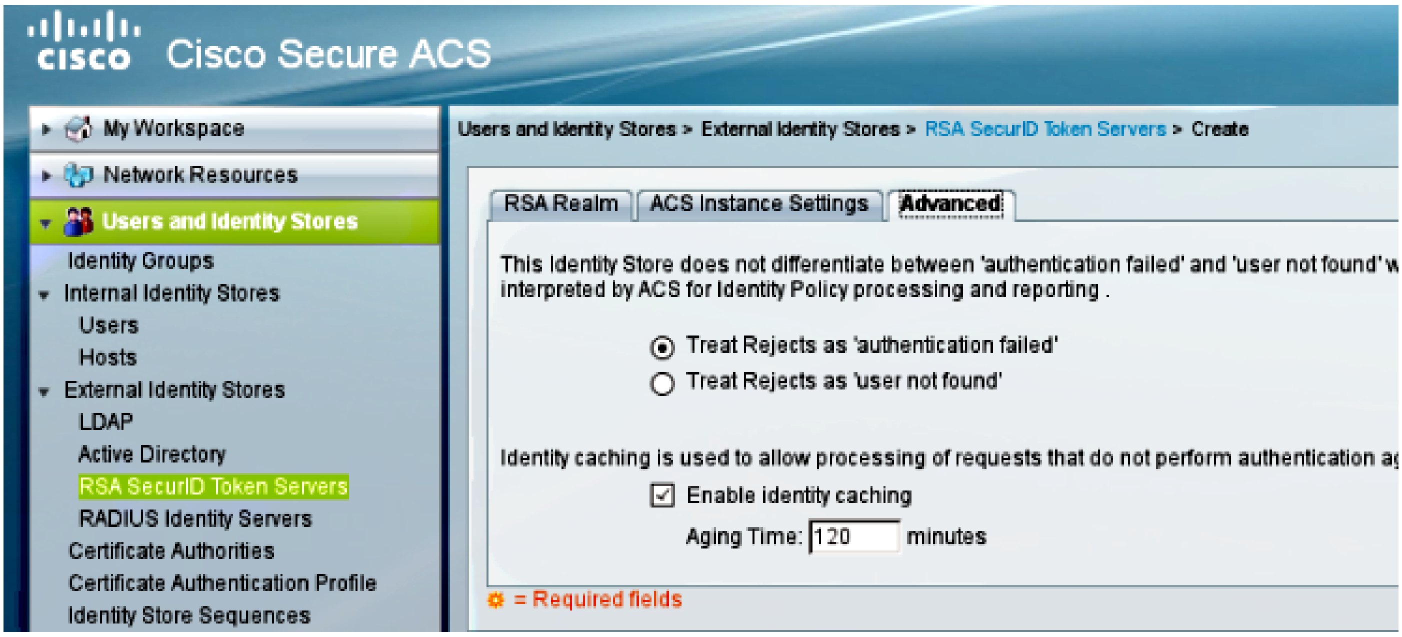
General
Name:
Description:

Server connection
Server Timeout: 30 Seconds
☐ Reauthenticate on Change PIN

Realm Configuration File
The RSA Configuration file (sdconf.rec) should be provided by your RSA administrator after they have registered all the ACS Instances as agents on the RSA Authetication Manager.
Import new 'sdconf.rec' file:
Node Secret Status: - not created -

= Required fields

In order to differentiate "user not found" from "authentication failure," choose settings in the **Advanced** tab:



Users and Identity Stores > External Identity Stores > RSA SecurID Token Servers > Create

This Identity Store does not differentiate between 'authentication failed' and 'user not found' w
interpreted by ACS for Identity Policy processing and reporting .

[✓ Treat Rejects as 'authentication failed']
Treat Rejects as 'user not found'

Identity caching is used to allow processing of requests that do not perform authentication ag

[✓] Enable identity caching

Aging Time: 120 minutes

= Required fields

It is also possible to change the default routing (load balancing) mechanisms between multiple RSA servers (primary and replicas). Change it with the **sdopts.rec** file provided by the RSA administrator. In ACS, it is uploaded in **Users and Identity Stores > External Identity Store > RSA Secure ID Token Servers > ACS Instance Settings**.

For cluster deployment, the configuration must be replicated. After the first successful authentication, each ACS node uses its own node secret downloaded from the primary RSA server. It is important to remember to configure the RSA for all the ACS nodes in the cluster.

SDI on ASA

The ASA does not allow upload of the **sdconf.rec** file. And, like the ACS, it allows for automatic deployment only. The ASA needs to be configured manually in order to point to the primary RSA server. A password is not needed. After the first successful authentication node, the secret file is installed (.sdi file on flash) and further authentication sessions are protected. Also the IP address of other RSA servers are downloaded.

Here is an example:

```
aaa-server SDI protocol sdi
aaa-server SDI (backbone) host 1.1.1.1
debug sdi 255
test aaa auth SDI host 1.1.1.1 user test pass 321321321
```

After successful authentication, the **show aaa-server protocol sdi** or **show aaa-server <aaa-server-group>** command shows all RSA servers (if there are more than one), while the **show run** command shows only the primary IP address:

<#root>

```
bsns-asa5510-17#

show aaa-server RSA
```

```
Server Group:    RSA
Server Protocol:
```

sdi

Server Address: 10.0.0.101

```
Server port:     5500
Server status:   ACTIVE (admin initiated), Last transaction at
10:13:55 UTC Sat Jul 27 2013
Number of pending requests              0
Average round trip time                 706ms
Number of authentication requests       4
Number of authorization requests        0
Number of accounting requests           0
Number of retransmissions               0
Number of accepts                       1
Number of rejects                       3
Number of challenges                    0
Number of malformed responses           0
Number of bad authenticators            0
Number of timeouts                      0
Number of unrecognized responses        0


SDI Server List:
```

Active Address: 10.0.0.101

```
        Server Address:          10.0.0.101
        Server port:             5500
        Priority:                0
        Proximity:               2
```

Status: OK

```
        Number of accepts                       0
        Number of rejects                       0
        Number of bad next token codes          0
        Number of bad new pins sent             0
        Number of retries                       0
        Number of timeouts                      0
```

Active Address: 10.0.0.102

```
        Server Address:          10.0.0.102
        Server port:             5500
        Priority:                8
        Proximity:               2
```

Status: OK

```
        Number of accepts                      1
        Number of rejects                      0
        Number of bad next token codes         0
        Number of bad new pins sent            0
        Number of retries                      0
        Number of timeouts                     0
```

Troubleshoot

This section provides information you can use in order to troubleshoot your configuration.

No Agent Configuration on RSA

In many cases after you install a new ASA or change the ASA IP address, it is easy to forget to make the same changes on the RSA. The Agent IP address on the RSA needs to be updated for all clients that access the RSA. Then, the new node secret is generated. The same applies to the ACS, especially to secondary nodes because they have different IP addresses and the RSA needs to trust them.

Corrupted Secret Node

Sometimes the secret node file on the ASA or the RSA becomes corrupted. Then, it is best to remove the agent configuration on the RSA and add it again. You also need to do the same process on the ASA/ACS - remove and add configuration again. Also, delete the .sdi file on the flash, so that in the next authentication, a new .sdi file is installed. Automatic node secret deployment must occur once this is complete.

Node in Suspended Mode

Sometimes one of the nodes is in suspended mode, which is caused by no response from that server:

<#root>

```
asa#
```

```
show aaa-server RSA
```

```
<.....output ommited"
SDI Server List:
        Active Address:         10.0.0.101
        Server Address:         10.0.0.101
        Server port:            5500
        Priority:               0
        Proximity:              2
```

```
  Status:                  SUSPENDED
```

In suspended mode, the ASA does not try to send any packets to that node; it needs to have an **OK** status for that. The failed server is put in active mode again after the dead timer. For more information, refer to the reactivation-mode command section in the Cisco ASA Series Command Reference, 9.1 guide.

In such scenarios, it is best to remove and add the AAA-server configuration for that group in order to trigger that server into active mode again.

Account Locked

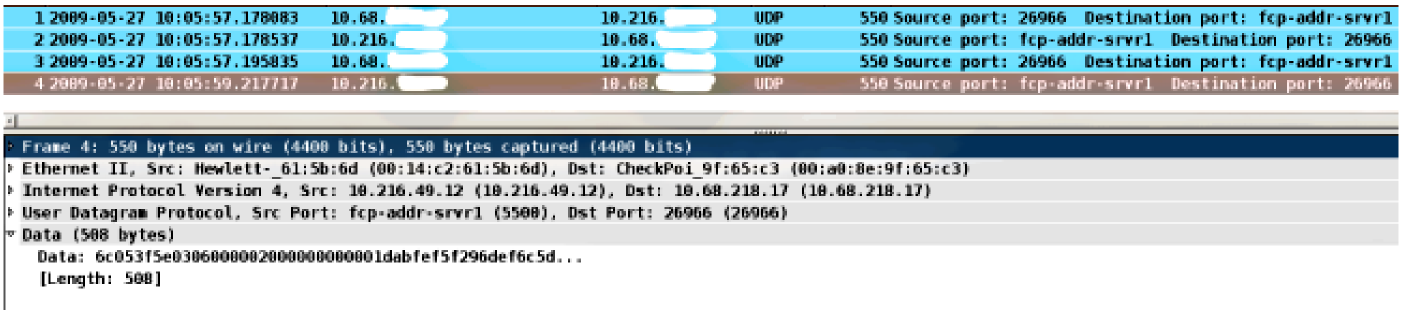
After multiple retries, the RSA can lock out of the account. It is easily checked on the RSA with reports. On the ASA/ACS, reports only show "failed authentication."

Maximum Transition Unit (MTU) Issues and Fragmentation

SDI uses UDP as transport, not MTU path discovery. Also UDP traffic does not have the Don't Fragment (DF) bit set by default. Sometimes for larger packets, there can be fragmentation problems. It is easy to sniff traffic on the RSA (both the appliance and Virtual Machine [VM] use Windows and use Wireshark). Complete the same process on the ASA/ACS and compare. Also, test RADIUS or WebAuthentication on the RSA in order to compare it to SDI (in order to narrow down the problem).

Packets and Debugs for ACS

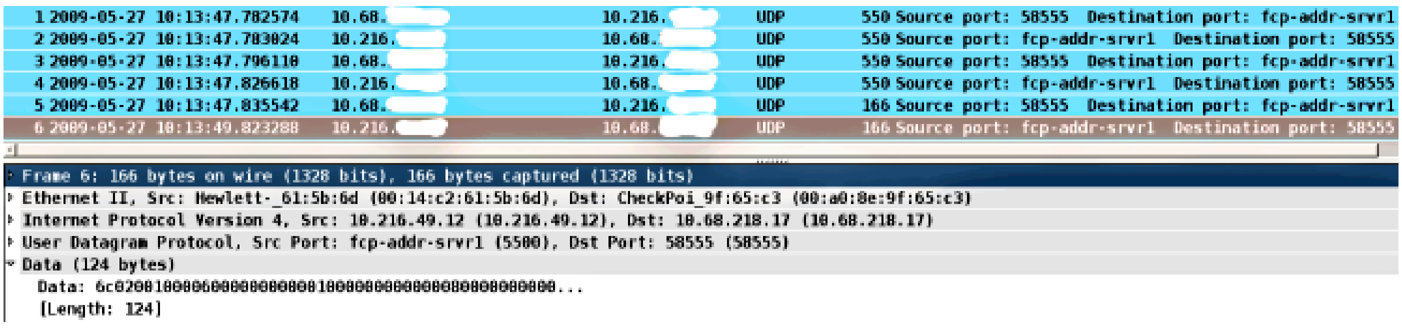
Because SDI payload is encrypted, the only way to troubleshoot the captures is to compare the size of the response. If it is smaller than 200 bytes, there can be a problem. A typical SDI exchange involves four packets, each of which is 550 bytes, but that can change with the RSA server version:



```
  1 2009-05-27 10:05:57.178083   10.68.[redacted]    10.216.[redacted]   UDP   550 Source port: 26966  Destination port: fcp-addr-srvr1
  2 2009-05-27 10:05:57.178537   10.216.[redacted]   10.68.[redacted]    UDP   550 Source port: fcp-addr-srvr1  Destination port: 26966
  3 2009-05-27 10:05:57.195835   10.68.[redacted]    10.216.[redacted]   UDP   550 Source port: 26966  Destination port: fcp-addr-srvr1
  4 2009-05-27 10:05:59.217717   10.216.[redacted]   10.68.[redacted]    UDP   550 Source port: fcp-addr-srvr1  Destination port: 26966

▸ Frame 4: 550 bytes on wire (4400 bits), 550 bytes captured (4400 bits)
▸ Ethernet II, Src: Hewlett-_61:5b:6d (00:14:c2:61:5b:6d), Dst: CheckPoi_9f:65:c3 (00:a0:8e:9f:65:c3)
▸ Internet Protocol Version 4, Src: 10.216.49.12 (10.216.49.12), Dst: 10.68.218.17 (10.68.218.17)
▸ User Datagram Protocol, Src Port: fcp-addr-srvr1 (5500), Dst Port: 26966 (26966)
▾ Data (508 bytes)
    Data: 6c053f5e030600000200000000001dabfef5f296def6c5d...
    [Length: 508]
```

In case of problems, it is usually more than four packets exchanged and smaller sizes:



```
  1 2009-05-27 10:13:47.782574   10.68.[redacted]    10.216.[redacted]   UDP   550 Source port: 58555  Destination port: fcp-addr-srvr1
  2 2009-05-27 10:13:47.783024   10.216.[redacted]   10.68.[redacted]    UDP   550 Source port: fcp-addr-srvr1  Destination port: 58555
  3 2009-05-27 10:13:47.796110   10.68.[redacted]    10.216.[redacted]   UDP   550 Source port: 58555  Destination port: fcp-addr-srvr1
  4 2009-05-27 10:13:47.826618   10.216.[redacted]   10.68.[redacted]    UDP   550 Source port: fcp-addr-srvr1  Destination port: 58555
  5 2009-05-27 10:13:47.835542   10.68.[redacted]    10.216.[redacted]   UDP   166 Source port: 58555  Destination port: fcp-addr-srvr1
  6 2009-05-27 10:13:49.823288   10.216.[redacted]   10.68.[redacted]    UDP   166 Source port: fcp-addr-srvr1  Destination port: 58555

▸ Frame 6: 166 bytes on wire (1328 bits), 166 bytes captured (1328 bits)
▸ Ethernet II, Src: Hewlett-_61:5b:6d (00:14:c2:61:5b:6d), Dst: CheckPoi_9f:65:c3 (00:a0:8e:9f:65:c3)
▸ Internet Protocol Version 4, Src: 10.216.49.12 (10.216.49.12), Dst: 10.68.218.17 (10.68.218.17)
▸ User Datagram Protocol, Src Port: fcp-addr-srvr1 (5500), Dst Port: 58555 (58555)
▾ Data (124 bytes)
    Data: 6c0200100006000000000000001000000000000000000000000...
    [Length: 124]
```

Also, the ACS logs are quite clear. Here are typical SDI logs on the ACS:

<#root>

```
EventHandler,11/03/2013,13:47:58:416,DEBUG,3050957712,Stack: 0xa3de560
```

`Calling backRSAIDStore`

```
: Method MethodCaller<RSAIDStore, RSAAgentEvent> in
thread:3050957712,EventStack.cpp:242
```

```
AuthenSessionState,11/03/2013,13:47:58:416,DEBUG,3050957712,cntx=0000146144,
sesn=acs-01

/150591921/1587,

user=mickey.mouse

,[

RSACheckPasscodeState


::onEnterState],RSACheckPasscodeState.cpp:23

EventHandler,11/03/2013,13:47:58:416,DEBUG,3002137488,Stack: 0xa3de560


Calling RSAAgent

:Method MethodCaller<RSAAgent, RSAAgentEvent> in thread:
3002137488,EventStack.cpp:204

RSAAgent,11/03/2013,13:47:58:416,DEBUG,3002137488,cntx=0000146144,sesn=

acs-01

/150591921/1587,

user=mickey.mouse

,[RSAAgent::handleCheckPasscode],
RSAAgent.cpp:319

RSASessionHandler,11/03/2013,13:47:58:416,DEBUG,3002137488,[RSASessionHandler::

checkPasscode

] call AceCheck,RSASessionHandler.cpp:251

EventHandler,11/03/2013,13:48:00:417,DEBUG,2965347216,Stack: 0xc14bba0
Create newstack, EventStack.cpp:27

EventHandler,11/03/2013,13:48:00:417,DEBUG,3002137488,Stack: 0xc14bba0 Calling
RSAAgent: Method MethodCaller<RSAAgent,

RSAServerResponseEvent

> in
 thread:3002137488,EventStack.cpp:204

RSAAgent,11/03/2013,13:48:00:417,DEBUG,3002137488,cntx=0000146144,sesn=

acs-01


/150591921/1587,

user=mickey.mouse

,[RSAAgent::handleResponse]

operation completed
with ACM_OKstatus

,RSAAgent.cpp:237

EventHandler,11/03/2013,13:48:00:417,DEBUG,3002137488,Stack: 0xc14bba0
EventStack.cpp:37
```

```
EventHandler,11/03/2013,13:48:00:417,DEBUG,3049905040,Stack: 0xa3de560 Calling
back RSAIDStore: Method MethodCaller<RSAIDStore, RSAAgentEvent> in thread:
3049905040,EventStack.cpp:242

AuthenSessionState,11/03/2013,13:48:00:417,DEBUG,3049905040,cntx=0000146144,sesn=
acs-01/150591921/1587,
```

`user=mickey.mouse`

```
,[RSACheckPasscodeState::onRSAAgentResponse]
```

`Checkpasscode succeeded`

```
,
```

`Authentication passed`

```
,RSACheckPasscodeState.cpp:55
```

Related Information

- **RSA Authentication Manager Resources**
- **RSA/SDI Server Support section of the Cisco ASA 5500 Series Configuration Guide using the CLI, 8.4 and 8.6**
- **RSA SecurID Server section of the User Guide for Cisco Secure Access Control System 5.4**
- **Technical Support & Documentation - Cisco Systems**