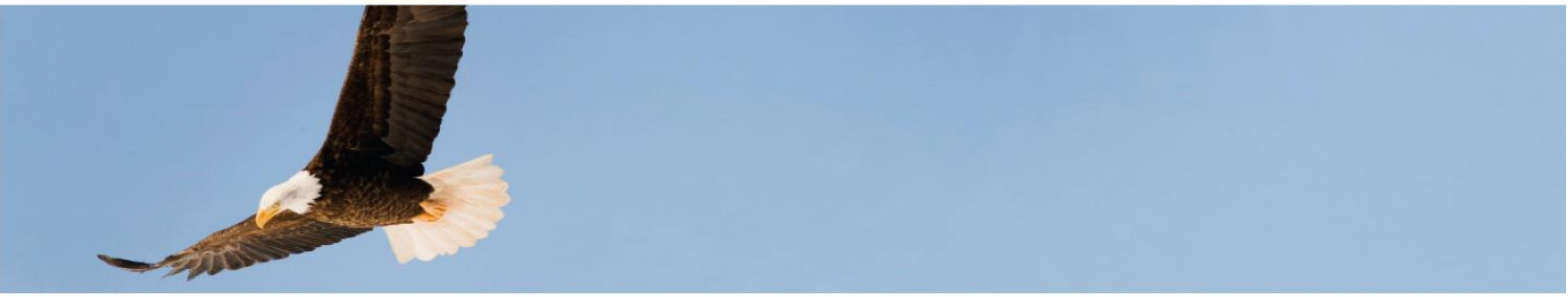
Salt Group



Echidna Concepts Guide

Version 15.1

May 2015

Contents

Chapter 1: Before starting

1.1 About this book

This *Concepts Guide* is part of a set of books about Echidna. It describes Echidna and how it can be used.

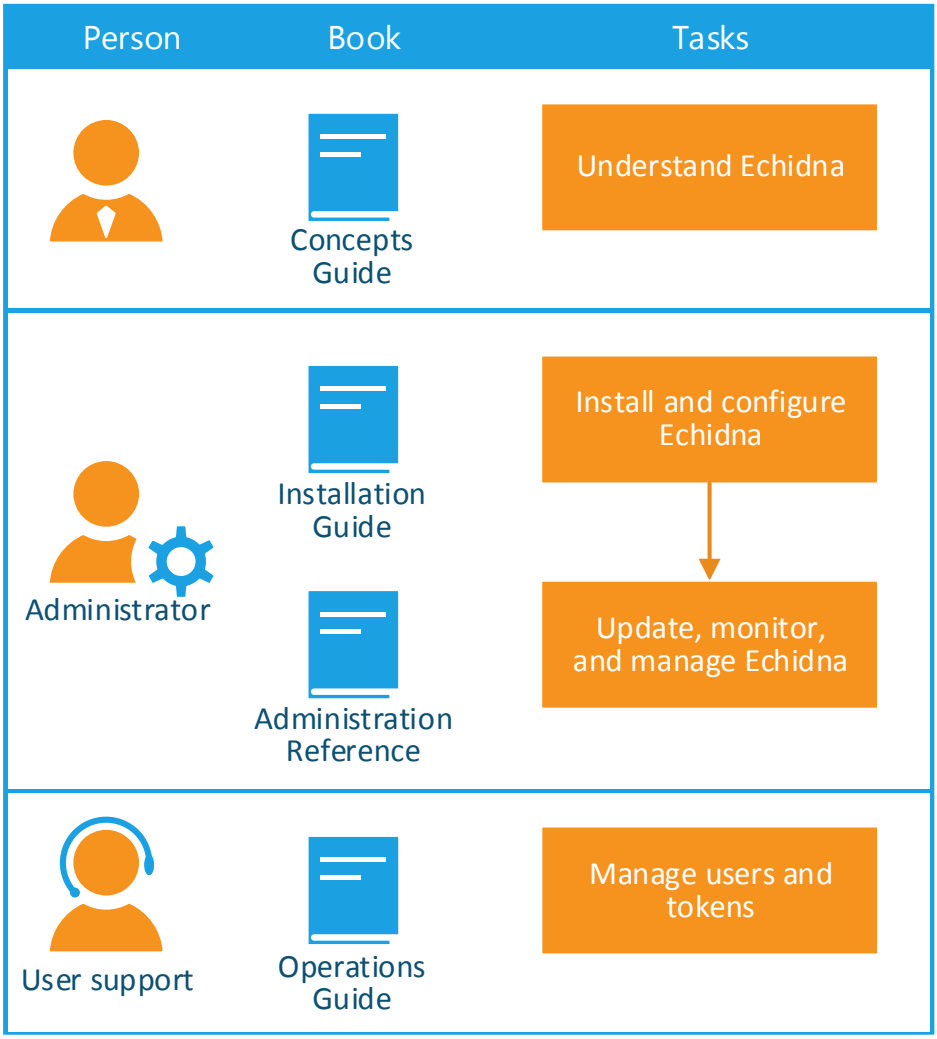


Figure 1: When to use each book in the Echidna documentation set

1.2 About Echidna

Echidna is a multi-factor authentication server, designed to let staff and users not only access applications securely, but also add other security features to existing applications. For example, Echidna can add transaction-signing capabilities to an Internet banking application.

Echidna supports a range of authentication devices, allowing fit-for-purpose and no lock-in, irrespective of legacy systems.

Other benefits include:

- Echidna's pricing model is below 40% of the costs of legacy authentication servers.
- Echidna can easily be skinned to allow branded deployments.
- Echidna can be fully customised on-the-fly without code changes.

Echidna is suitable for financial institutions, government departments, and enterprises.

Echidna accepts both RADIUS and HTTP/S requests, and it connects to your existing user store:

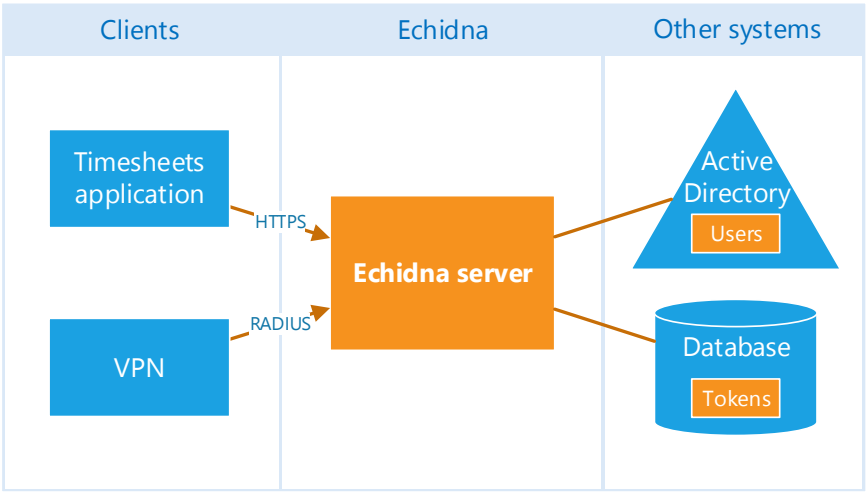


Figure 2: How Echidna relates to other systems

Chapter 2: Benefits of Echidna

2.1 Reduces token costs

Authentication servers are usually restrictive on the various forms of authentication types that they support, leading to customers being locked in to having to continue to use them, due to their proprietary nature.

Echidna supports a range of standards-based authentication devices, as well as proprietary authentication devices through its brokering services.

When Echidna is used with free Salt mobile tokens, costs are further reduced because token replacement costs (which can be more than 30%[1] of total token cost per year) are eliminated.

2.2 Seamless migration from legacy solution

Echidna allows users to gradually migrate from the legacy solution, without requiring a massive single change. Echidna can continue to support legacy hardware tokens, by brokering authentication requests to the legacy server. As the legacy tokens expire, users can be given newer, more cost-effective tokens.

For more information, see Chapter 6: Brokering authentication in the *Administration Reference*.

2.3 Efficient deployment and operation

Two-factor authentication servers are usually expensive to deploy and operate, due to a combination of user licensing costs, token costs (including mobile), replacement token costs, and token operational management, particularly when provisioning tokens to users.

Not only can Echidna be deployed in under 20 minutes, but its operation model allows for users to be either self-managed or remotely managed. When combined with over-the-air provisioning, this provides a very easy-to-manage an efficient operational environment.

For a full description of how to install and configure Echidna, see the *Installation Guide*.

[1] The industry-standard churn rate on tokens that are lost, broken, or never returned.

2.4 Easy to customise

Echidna can be fully customised without code changes, allowing organisations to use stronger authentication more easily across applications and work force.

Echidna can better adapt to the various business authentication requirements and flows. For example, Echidna can support timed temporary passwords as a backup authentication method for users who have hardware tokens, but not for users with mobile tokens, or vice versa.

2.5 Perpetual licensing

Echidna is sold using a perpetual licensing model based on active tokens/users.

This means that two-factor authentication does not have to be restricted to a sub-set of your work force: you can bring the security, efficiency, and productivity benefits to everyone.

2.6 Supports many token types

Echidna supports multiple authentication tokens, including but not limited to the following:

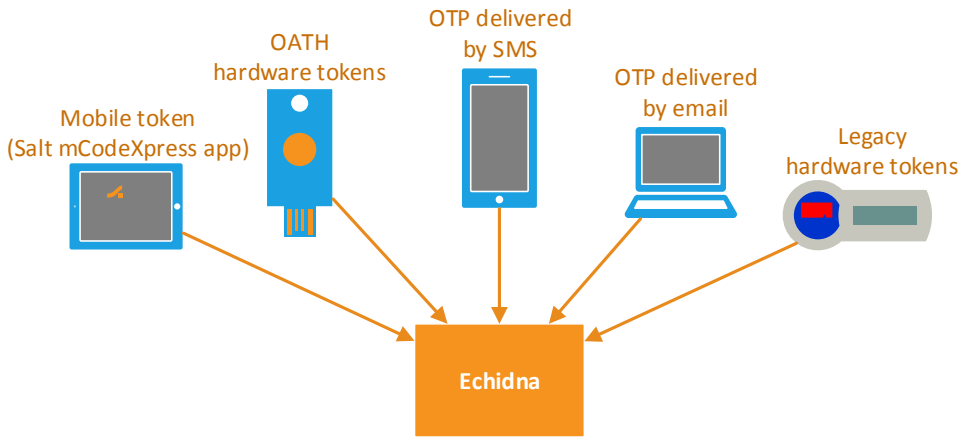


Figure 3: Some of the tokens that Echidna supports

Salt mCodeXpress is an app that securely generates one-time passcodes (OTPs). Users can install Salt mCodeXpress on their phone or tablet and then register it with their organisation's Echidna server. After registration, Salt mCodeXpress generates OTPs that let the user sign in to their organization's VPN or other remote access system.

OATH hardware tokens comply with the OATH HOTP, OATH TOTP or OATH OCRA standards.

One-time passcodes (OTPs) are delivered to a user by SMS or email.

Legacy hardware tokens such as RSA SecurID and Vasco tokens are supported by brokering authentication requests to a legacy server.

For more information, see Chapter 5: Configure authentication in the *Administration Reference*.

Chapter 3: Deployment models

Echidna is flexible in the deployment models that it can support. The following diagrams show some of the ways it can be deployed within an organisation.

3.1 Two-factor authentication for an access point

Echidna can receive RADIUS requests for authentication.

The following diagram shows an example system. In this example, a remote staff member uses a VPN gateway to access protected resources behind the company firewall.

In this example, Echidna works with the mCodeXpress token. This is a Salt mobile app, downloadable for free from your favourite app store. When the user tries to access a protected application, they see a login page that requires them to enter a one-time passcode (OTP). The user generates the OTP using mCodeXpress on their phone and types the new OTP into the Password box on the login page.

When the user signs in, the VPN gateway delegates authentication to Echidna by sending the credentials via RADIUS.

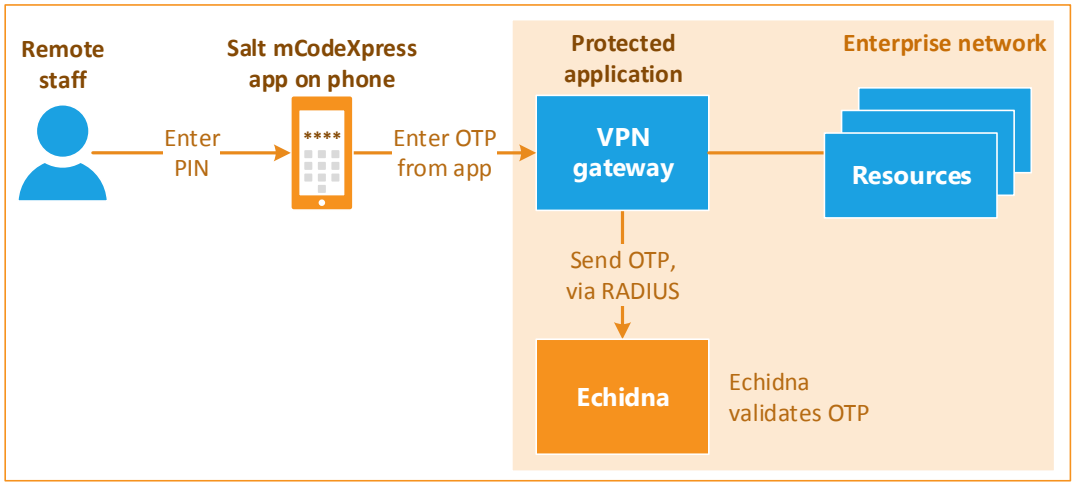


Figure 4: Example of a system in which Echidna provides two-factor authentication to a RADIUS client

Echidna checks the credentials and returns a response to the VPN gateway, also via RADIUS. The gateway then grants or denies access to the user, depending on Echidna's response.

3.2 Replace legacy tokens

Echidna can broker authentication requests to a legacy authentication server, such as RSA Authentication Manager and Vasco IDENTIKEY. Echidna proxies the authentication requests to the third-party authentication server, allowing this third-party server to validate legacy or proprietary tokens.

The organisation retains its existing authentication mechanisms and at the same time seamlessly introduces new lower-cost and convenient tokens.

Users with legacy tokens continue to use the legacy server. As the legacy tokens expire, users receive their new lower-cost standards-based tokens. Because the legacy authentication server continues to respond to authentication requests, there is minimal user or technology impact. User with the new tokens are unaware that their tokens are now validated by the Echidna server, as brokering for the user will no longer be required.

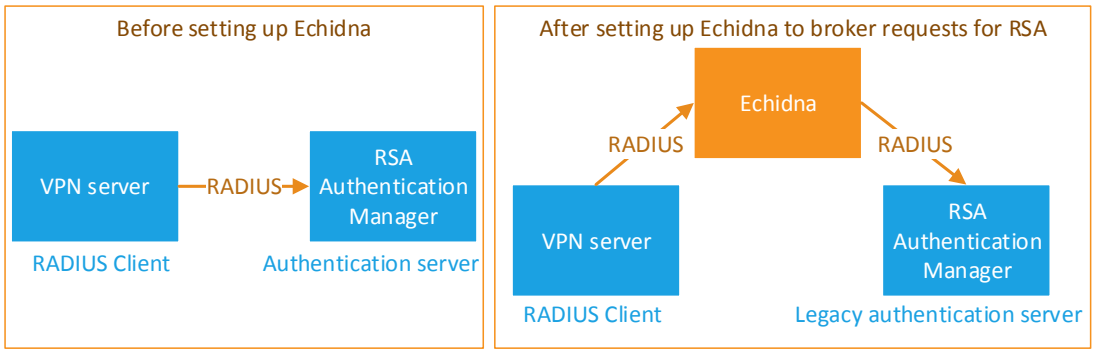


Figure 5: Echidna can be placed in front of a legacy authentication server

This diagram shows how Echidna provides a single target for all authentication traffic for the enterprise. Echidna handles some authentication requests itself, and it forwards other requests to RSA Authentication Manager and then waits for the response.

The following example shows a legacy RSA system. In this situation, Echidna receives all authentication requests, and delegates the requests that contain a SecurID OTP to RSA Authentication Manager.

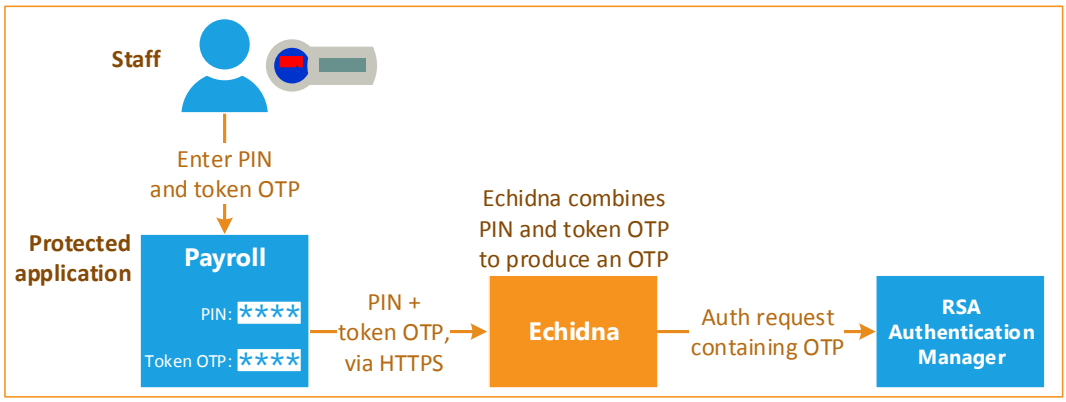


Figure 6: Example of how Echidna can broker authentication requests to RSA Authentication Manager

With this model, after the users have all been transparently migrated, the legacy authentication server can be either re-purposed or retired.

For more information, see Chapter 6: Brokering authentication in the *Administration Reference*.

3.3 Protect an Internet banking application

Echidna can receive authentication requests via its web services API, over HTTP/S.

The following diagram shows an example of how Echidna can protect an Internet banking application.

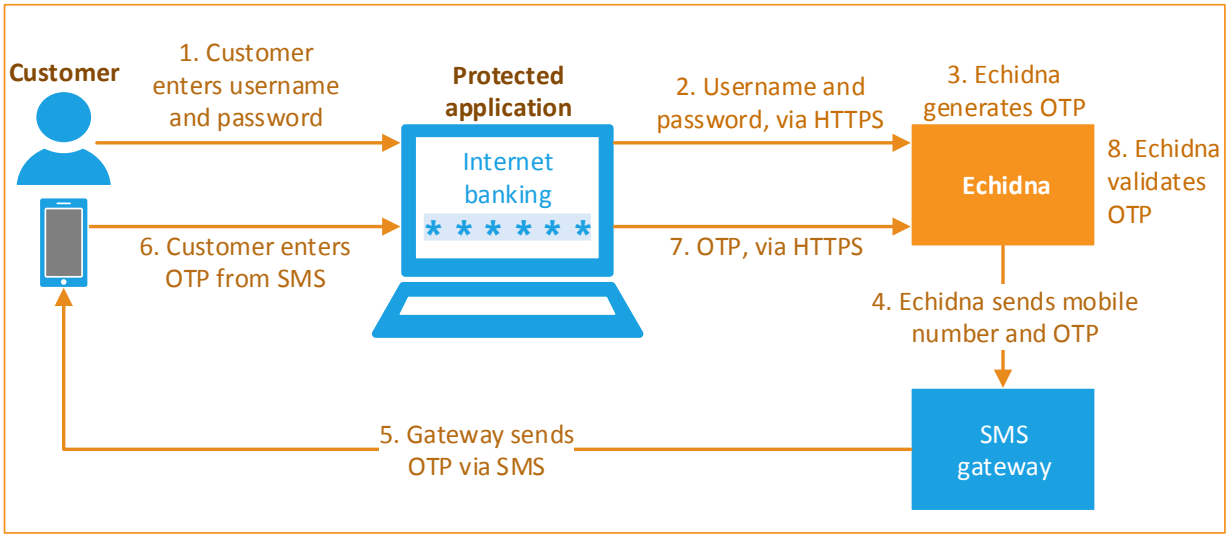


Figure 7: Example system shows Echidna working with a messaging gateway to provide OTP tokens via SMS

In this diagram, Echidna works with an SMS OTP token, which is an OTP produced and validated by Echidna. The user wants to sign into a secure banking application, which requires a one-time passcode (OTP). To get this OTP, the user signs in with their usual password, and is then prompted to check their phone for an SMS. The application sends the credentials to Echidna via HTTPS. Echidna generates an OTP and gives it to an SMS gateway to be sent to the user's mobile number.

The user enters the OTP in the banking application, which sends the OTP via HTTPS to Echidna for validation. If the OTP is correct, the user is granted access to the Internet banking application.

Chapter 4: Integration with existing infrastructure

4.1 Echidna runs in a virtual appliance

Echidna is supplied as a virtual appliance[2], which is a virtual machine that already contains Echidna. Echidna is served by Apache Tomcat. In addition, the virtual machine automatically includes OpenJDK and a selection of JDBC drivers.

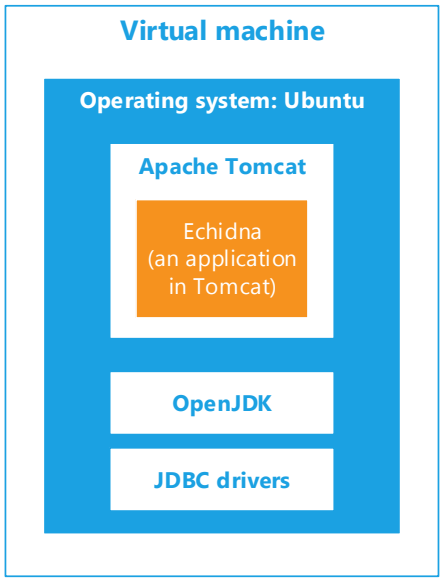


Figure 8: Echidna is an application within Tomcat, which is already set up in a virtual appliance

Salt Group supplies the virtual appliance in Open Visualization Format (OVF) format, which most hypervisors can use. To set up Echidna, the administrator deploys a new VM, using the OVF file as a template. In the new VM, the administrator then configures the new Echidna server, and connects it to other systems in the organisation.

The Ubuntu operating system is configured to restrict access to only the services provided by Echidna, the Virtual Appliance System Management Interface, and SSH.

For a full description of how to set up Echidna, see the *Installation Guide*.

[2] Echidna can be supplied as an installation package, upon request.

4.2 Components of Echidna

In this architecture diagram, the orange items are the Echidna components and the blue items are the other systems in the organisation that Echidna connects to.

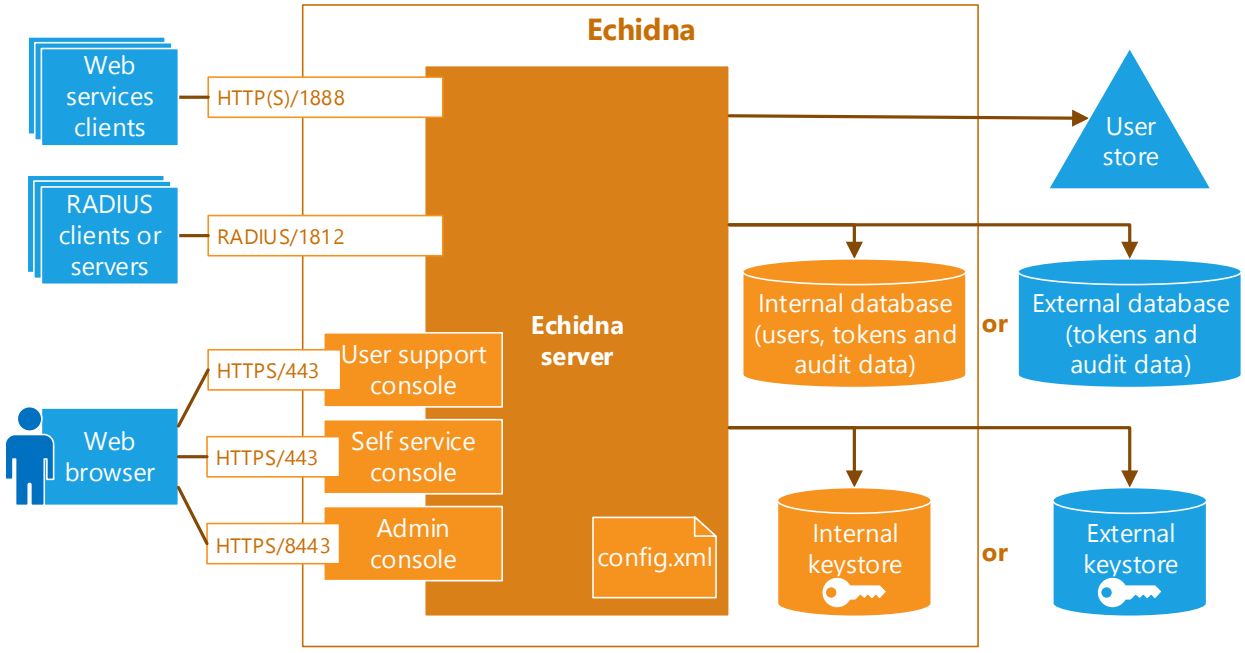


Figure 9: Architecture of Echidna and the systems it connects to

4.2.1 Clients

When a user attempts to log in to an application, the application sends the request to Echidna for validation. Each application is a client of Echidna. In this diagram, they are shown in the top left corner. Each client must be configured to send authentication requests to the Echidna server.

Echidna can receive requests via RADIUS and via web services (HTTP and HTTPS).

4.2.2 Consoles

Staff and end users use a web browser to access a console, which lets them do relevant tasks with Echidna. For more information, see Chapter 6: Configure and manage Echidna on page 16.

4.2.3 User store

Echidna needs user records to authenticate against.

If an organisation already has a user store (such as a database or Active Directory), Echidna can connect to that user store. If there is no existing user store, use Echidna's internal database to store user records.

For more information, see Chapter 4: Configure the user data in the *Administration Reference*.

4.2.4 Database

Echidna stores token records and audit data in a database. If the organisation already has a database, Echidna can connect to these. Alternatively, Echidna can use its own internal database and keystore if these do not already exist.

In addition, this internal database can store user records if required.

For more information, see Chapter 3: Configure the Echidna database in the *Administration Reference*.

4.2.5 Keystore

Echidna uses cryptographic keys and certificates to encrypt sensitive data such as administrator passwords, client credentials, and token seed material. These keys and certificates are also used to establish SSL connections to and from Echidna. Echidna stores its keys and certificates in standard Java cryptographic keystores, which can be hardware-based if required.

If the organisation already has a keystore management process, Echidna can make use of it. Alternatively, Echidna can use its own internal keystores.

For more information, see Chapter 9: Configure passwords, keystores, and SSL in the *Administration Reference*.

4.2.6 Configuration

Echidna stores its configuration in XML format. The configuration file can be scrutinised by auditors, and managed under change control. Administrators can export this configuration and then import all or some of it.

To permit failover and high availability, two Echidna servers can share the same XML file if it includes licenses for both Echidna servers.

For more information, see Chapter 12: Manage Echidna Configuration in the *Administration Reference*.

Chapter 5: High availability and disaster recovery

When Echidna provides authentication services in a mission-critical system, you need enterprise-grade disaster recovery and high availability. Echidna is ready to be deployed in a high-availability environment.

Echidna stores its essential information outside its own server, which lets multiple instances of Echidna share the same user and token information. This allows for both failover and load balancing. It also supports backup and recovery.

Echidna uses information in the following external locations:

- **Configuration** - Stored in an external XML file
- **User data** - Stored in a user store or in Echidna's database
- **Token state data** - Stored in a database
- **Cryptographic key data** - Stored in Java cryptographic keystores

5.1 Load balancing

In a high-traffic system, you can use a load balancer to distribute requests across multiple instances of Echidna. Configure each instance of Echidna to use the same sources of user and system information, as shown in this example:

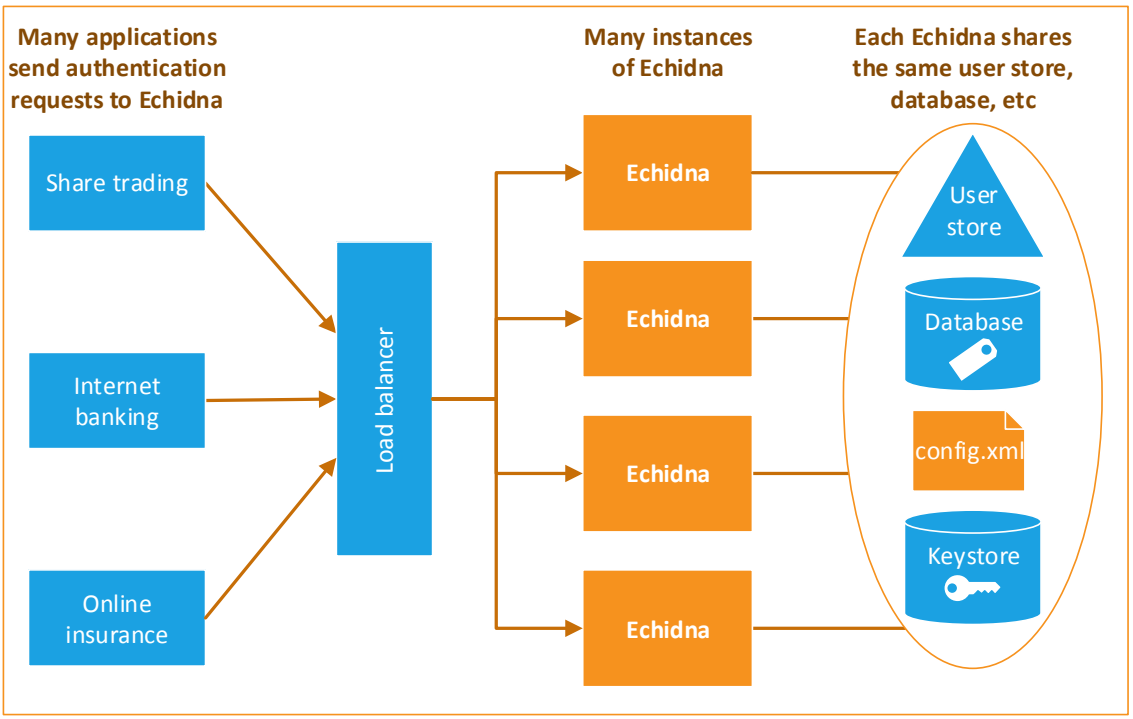


Figure 10: Example load-balancing setup for Echidna

To keep your service capacity level high, the load balancer spreads the requests across all instances of Echidna.

5.2 Failover

In a high-SLA environment, set up failover to prevent system outage and allow for maintenance downtime.

Configure Echidna clients to fail over to a second Echidna server. In addition, ensure that the user, token, configuration, and SSL data is all replicated.

5.3 Disaster recovery

Ensure that the Echidna configuration file, the user store, the token database, and the keystores are backed up using your usual backup procedures. In the event of a disaster requiring recovery of Echidna, restore each of the four components first, then restore Echidna.

Alternatively, because Echidna runs in a virtual machine, administrators can take snapshots of the entire machine.

Chapter 6: Configure and manage Echidna

Echidna comes with three console applications, which users access with a web browser. This diagram shows the three consoles, and the people who use them:

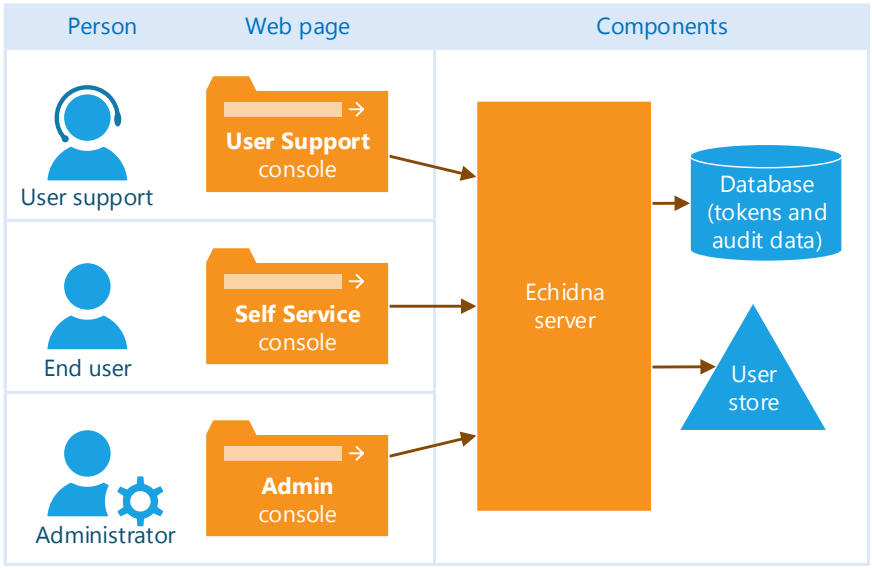


Figure 11: Consoles that come with Echidna

Administrators install, configure, maintain, and monitor Echidna, using the **Administration console**. Administrators have the highest level of access to Echidna's configuration. Administrators are responsible for keeping Echidna working smoothly.

Support staff manage and monitor tokens and users, using the **User Support console**. They take calls from users about login problems and they troubleshoot these problems. If support staff cannot fix an end user's problem, they escalate the problem to an administrator.

End users log in to applications with their tokens. If the organisation permits it, end users can register tokens and change their preferred authentication method using the **Self Service console**. If the organisation does not let end users manage their own tokens, support staff do these tasks on behalf of users.

Chapter 7: Licensing

Echidna is installed with an **evaluation license**. This is valid for one month from the time the server is configured.

The evaluation license limits Echidna to only two tokens for each authentication method, and no connection to an SMS gateway. To remove these limitations, register Echidna by applying for a permanent license.

To receive a **permanent license**, use the Administration console to create a license request, and then email the request to your supplier. You will receive an email with an attached license file, which is an XML file that updates the Echidna configuration. The administrator imports the license and then saves the Echidna configuration.

Each permanent license permits a certain number of tokens to be used. Each license can include an overall token limit, plus a limit for each authentication method. For example, if an organisation's license has an overall token limit of 100 plus limits of 100 mCodeXpress tokens and 10 OATH TOTP tokens, Echidna allows up to 100 mCodeXpress tokens and up to 10 OATH TOTP tokens, provided the total does not exceed 100.

The Salt mCodeXpress mobile app is free and has no expiry date. An organisation's users can install Salt mCodeXpress on as many mobile devices as they like. When a user registers their Salt mCodeXpress app with their organisation, Echidna counts it as a token, for licensing purposes.

For more information, see 12.9 Licensing for Echidna in the *Administration Reference*.