100 XP

Introduction

2 minutes

If you want to understand what can go wrong with managing an application's configuration secrets, look no further than the story of Steve the senior developer.

Steve had been in his job at a pet food delivery company for a few weeks. He was exploring the details of the company's web app — a .NET Core web application that used an Azure SQL database for storing order information and third-party APIs for credit card billing and mapping customer addresses — when he accidentally pasted the connection string for the orders database into a public forum.

Days later, accounting noticed that the company was delivering a lot of pet food that nobody had paid for. Someone had used the connection string to access the database and created orders by updating the database directly.

After realizing his mistake, Steve hurriedly changed the database password to lock out the attacker. After changing the password, the website started returning errors to users: the application server needed an updated configuration with the new password. Steve logged directly into the application server and changed the app configuration instead of redeploying, but the server was still showing failed requests.

Steve had forgotten that multiple instances of the app ran on different servers, and he had only changed the configuration for one. A full redeployment was needed, causing another 30 minutes of downtime.

Fortunately for Steve, the accounting department was able to correct the errors quickly, and only one day's worth of orders were impacted. He might not be so lucky in the future, though, and needs to find a way to improve the security and maintainability of the app.

Leaking a database connection string, API key, or service password can be catastrophic. Stolen or deleted data, financial harm, application downtime, and irreparable damage to business assets and reputation are all potential results. Unfortunately, secret values often need to be deployed in multiple places simultaneously and changed at inopportune times. And you have to store them *somewhere*! Let's see how Steve can reduce risk and improve the security and maintainability of his app with Azure Key Vault.

Learning objectives

In this module, you will:

- Explore what types of information can be stored in Azure Key Vault
- Create an Azure Key Vault and use it to store secret configuration values
- Enable secure access to the vault from an Azure App Service web app with managed identities for Azure resources
- Implement a web application that retrieves secrets from the vault

---

Next unit: What is Azure Key Vault?

Continue >