Redmine - Patch #41331

Gemfile patch: fix for handling (security related) ERB expressions in database.yml

2024-09-29 12:26 - Jan Catrysse

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Ruby support	**Estimated time:**	0.00 hour
Target version:			

Description

Redmine installations that use ERB expressions (like `<% %>`) in their database.yml file may encounter issues during parsing.
To address this, we've applied a patch to the Gemfile that ensures compatibility when handling files containing such special characters, thus preventing common errors.

This patch modifies the way the database.yml is processed, ignoring lines that contain ERB expressions.
This is particularly useful for installations where the database configuration might contain sensitive data managed via **Rails' encrypted credentials** or other mechanisms that use ERB.

```
Index: Gemfile
IDEA additional info:
Subsystem: com.intellij.openapi.diff.impl.patch.CharsetEP
<+>UTF-8
===================================================================
diff --git a/Gemfile b/Gemfile
--- a/Gemfile	(revision 277728afc979a7123eef8d1a8ac54d74c235c5fc)
+++ b/Gemfile	(date 1727590462090)
@@ -61,7 +61,7 @@
 require 'yaml'
 database_file = File.join(File.dirname(__FILE__), "config/database.yml")
 if File.exist?(database_file)
-   yaml_config = ERB.new(IO.read(database_file)).result
+   yaml_config = ERB.new(IO.readlines(database_file).reject { |line| line =~ /<%.*%>/ }.join).result
   database_config = YAML.respond_to?(:unsafe_load) ? YAML.unsafe_load(yaml_config) : YAML.load(yaml_config)
   adapters = database_config.values.filter_map {|c| c['adapter']}.uniq
   if adapters.any?
```

History

#1 - 2024-09-30 16:10 - Holger Just

The patch would remove the ability to specify ERB-evaluated data in the database.yml file which may be relevant to the parsing done in the Gemfile

With encrypted credentials as proposed by you in How_to_use_encrypted_database_credentials, the credentials are not available "outside" of Rails, such as when the Gemfile is evaluated at the very beginning of the Rails startup. In any case, I don't believe that using encrypted credentials for the password in database.yml increases the security of the entire system much, as Rails still needs runtime-access to the key used to encrypt the plain password. Potential attackers which may read the unencrypted password from the database.yml could then also just read the key and the encrypted password. The only way this could be more secure is

- if you HAVE to store the encrypted password with your code in the same repository rather than setting it just on the server
- if you have a way to secure the key

Instead of this approach, a more secure solution could be to set the password in an environment variable in a way that the raw data is not readable on disk by the application process. This may require some workarounds to start Redmine to ensure that the environment variables are available there.

Finally, while I can see the desire to encrypt some sensitive data such as passwords, I believe that the approach proposed here does not sufficiently resolve the issue and instead may cause further parsing errors in the Gemfile.

#2 - 2024-10-22 14:37 - Plastic Oddball

Holger Just wrote in #note-1:

> The patch would remove the ability to specify ERB-evaluated data in the database.yml file which may be relevant to the parsing done in the Gemfile

> With encrypted credentials as proposed by you in How_to_use_encrypted_database_credentials, the credentials are not available "outside" of Rails, such as when the Gemfile is evaluated at the very beginning of the Rails startup. In any case, I don't believe that using encrypted credentials for the password in database.yml increases the security of the entire system much, as Rails still needs runtime-access to the key used to encrypt the plain password. Potential attackers which may read the unencrypted password from the database.yml could then also just read the key and the encrypted password. The only way this could be more secure is
>
> - if you HAVE to store the encrypted password with your code in the same repository rather than setting it just on the server
> - if you have a way to secure the key
>
> Instead of this approach, a more secure solution could be to set the password in an environment variable in a way that the raw data is not readable on disk by the application process. This may require some workarounds to start Redmine to ensure that the environment variables are available there.
>
> Finally, while I can see the desire to encrypt some sensitive data such as passwords, I believe that the approach proposed here does not sufficiently resolve the issue and instead may cause further parsing errors in the Gemfile.

Your points about the proposed patch and its implications for security are well taken. You're right that removing the ability to specify ERB-evaluated data in the database.yml file could complicate parsing in the Gemfile, and it's crucial to consider the broader impact on application behavior during startup.

The concern about encrypted credentials is valid. While encrypting passwords can provide an extra layer of security, it does rely on the management of the encryption keys, which can introduce vulnerabilities if not handled properly. As you noted, if an attacker can access both the encrypted password and the key, the intended security benefits diminish significantly.

Your suggestion to use environment variables is a practical alternative that can enhance security by keeping sensitive information out of the codebase altogether. This approach also reduces the risk of accidental exposure through version control systems. Implementing a strategy that ensures environment variables are properly configured for the application startup, while possibly requiring some additional setup, could lead to a more robust solution.

Ultimately, it's important to balance security measures with ease of use and system functionality. A thorough review of the proposed changes, along with consideration of alternative methods, is essential to ensure the security enhancements truly address the risks without introducing new issues, like parsing errors in the Gemfile. Thank you for raising these critical points for discussion!

#3 - 2024-10-23 18:16 - Jan Catrysse

Thank you both for your valuable input. I will definitely take these points into consideration as I explore this further. Here are some initial observations:

1. Regarding the parsing issue in the Gemfile, it seems feasible to refine the ERB exclusion logic to minimize any unintended impact. However, I question whether there should be significant ERB code within the Gemfile to begin with, and this may be an area for further review.
2. I agree that environment variables can enhance security, particularly if they are well-implemented. However, if they are not encrypted themselves, they may present the same level of exposure as plaintext credentials.

To clarify, the security concern we are addressing here is not limited to the Rails application alone, but rather covers the broader attack surface of the system. Additionally, both company and customer security guidelines strictly prohibit the use of non-encrypted credentials. This obligates us to avoid storing unencrypted passwords under any circumstances.

Unfortunately, there isn't much comprehensive documentation available for implementing encrypted credentials in a straightforward way, which complicates the process. Redmine would benefit from an easier method to handle encrypted database credentials, or perhaps an alternative approach that simplifies this while maintaining security.

I will continue to investigate potential solutions when time allows, and I welcome any input or collaboration on this topic as I am not a Ruby on Rails expert myself.

Files

Gemfile.patch	801 Bytes	2024-09-29	Jan Catrysse