SOURCE CODE MANAGEMENT TOOL

VCS

Version Control System

Purpose

This document is intended to identify the requirements of the Source Code Management (SCM) Tool. It contains a collection of various requirements test analysis for the Version Control System (VCS) software. The associated risks and their mitigation plan is covered to lay the framework that the team will follow. In order to adapt the usage of Source Code Management Tool (SCM) the context will be referred to.

Document Convention

This document uses the following convention to refer certain words

SCM	Source Code Management
VCS	Version Control System
CI/CD	Continuous Integration/ Continuous Development
IDE	Integrated Development Environment

Core Functional Requirements

1. **Versioning**
   The Source Code Management system should be able to maintain the versions and track changes in the code. Versioning will be used to mark the release and help with rolling back breaking changes if required.

2. **Tagging**
   Users should be able to tag important commits/checkpoints. This will help carve out releases properly.

3. **Version trail**
   The source code management should be able to maintain a complete history of the code changes. The system should be able to display all the versions for a particular piece of code. Users should also be able to compare different versions.

4. **Effortless Context Switching**
   The source code management should provide users with effortless context switching so that they should be able to move from one piece work to another without losing changes from either work.

5. **Track Changes**
   The source code management system should be able to track changes in the code and highlight them. This will be useful to understand what changes have been made to the code since the beginning or since the first release. Users should also be able to leave a message as for why the change(s) was done.

6. **Simultaneous code change**
   The source code management system should enable multiple users to make changes to the same file without interfering with each other's work.

7. **Micro commits**
   The source code management should be able to mark any change that happened without worrying if the change is a small change or big change. It should also be able to detect and mark line break changes, character change or even a spacing change.

8. **Ownership**
   The source code management should provide users to control the CRUD(Create, Read, Update, Delete) and merge access for their own code bases.

9. **Backups**
   The source code management should maintain backups of code and the history of updates/changes (versions)

10. **Restore and revert**
    The source management system should provide the ability to restore previous changes. It should also provide the ability to revert specific changes.

11. **Integration with various IDE’s**
    The source code management system should integrate with major IDE’s like IntelliJ, Eclipse and Visual Source Code.

12. **Integration with Orchestration/CI/CD systems**
    The source code management system should integrate with the orchestration tools like Jenkins, Spinnaker and TravisCI for automated code delivery and continuous code integration.

13. **Integration with monitoring tools**
    The source code management system should integrate with monitoring tools like Nagios, NewRelic and Prometheus to monitor the health of the system for availability and reliability.

14. **Data Storage**
    The source code management system should be able to store the data on the trusted on-premise servers or on the major cloud providers like AWS, GCP and Azure.

Security Requirements

Source code management tool should be able to fulfill below mentioned security requirements.

1. **Firewall**
   The source code management system should only be accessible behind the firewall network of the server.

2. **Network restrictions - only allow access from within company's network**
   The source code management system should only be accessible to internal employees of the company.

3. **Two factor authentication**
   The source code management system should have a two factor authentication as an added security measure.

4. **Single sign-on**
   The source code management system should provide authentication via single sign on for ease of use and network security.

5. **Encryption**
   The source code management system should encrypt all the incoming and outgoing data as well as stored data.

Testing Requirements Analysis

Source code management tool should be able to work and perform against below mentioned testing requirements.

1. **Integration Testing**
   a. Testing for integrations with various IDEs like Eclipse, IntelliJ and Visual Source code, Atom etc.
   b. Integration with Orchestration/CI/CD platforms like Jenkins, Travis CI and Spinnaker.
   c. Integration with monitoring tools for support and incident mitigation purposes.

2. **Performance Testing**
   a. Load Testing - Ability to perform test with large amounts of data and concurrent users able to log-in to the system.
   b. Stress Testing - Ability to perform in high traffic situations like major holidays, catastrophic events etc.

3. **Functional Testing**
   a. Test the features to make sure they are functioning as per the requirements.
   b. Automation testing is preferred over manual testing.

4. **User Interface Testing**
    a. Test the user interfaces to make sure UI flows are working as required.

Risk Analysis

The Source code management tool has the below mentioned risks associated with it.

1. **Risk of credentials leak**
   Sometimes the developer stores the credentials in the code in order to call or access some other application. If the credentials to access some application gets leaked from the code it could become a threat to the company.

2. **Threat to sensitive data in the code**
   The data stored in code could be a big threat to the company and its customers. This can have a huge impact on trust. The application that saves the sensitive data in encrypted format could also possess a danger of the information being misused or a decryption key stored in code to automatically decrypt the data.

3. **Risk of security attacks**
   Anytime the data/code is saved or stored on the external servers over the network it is prone to various types of security attacks. Cyber threats can originate from various actors like hackers, spies, criminals or terrorist groups. It can lead to various types of attacks like phishing, password attacks to the code or data.

4. **Lack of security awareness for the users/team**
   The team members lack information about security. Lack of education about the security policies and practices could lead to unawareness in developers which poses a risk to the company.

5. **Threats from third party integrations**
   Third party tools which can be used for integration could have risk of data leaks. A Softwares for untrusted vendors that can be integrated with the code online can be an inefficient way to use the integration tools.

Mitigation Steps

The source code management system could mitigate the above mentioned risks by following the below mentioned guidelines.

1. Measures to avoid transferring credentials in code/config on the server by creating tools that should statically analyse your commits and ensure users are not trying to transfer any passwords or sensitive information into the repository. Saved data will be

rejected if the tool matches any configured pattern that is created to find sensitive information.

2. Delete sensitive data from the files and history which include the trails of the change logs that will list your sensitive information online on the server. Make sure to provide the training to the developers to not to store sensitive data from the server repository and delete the history of comments. Purging the file with sensitive data from your repository's history.

3. Strict control access should be achieved by inheriting few measures:-
    a. Enable 2-factor authentication on every contributor's account.
    b. Make sure user's and contributor's don't share username and passwords with anyone.
    c. Secure all the devices used to access the repository
    d. Provide restricted access to the contributors of the repository or access control on the basis of roles should be provided.
    e. Delete the accounts once the user leaves the team and company.

4. Create a security metadata file that instructs users about how and when to report security vulnerabilities to the repository maintenance team. It will educate the users on how they should deal with security disclosures ,updates and other security practices.The security details should be mentioned in the security.MD file on the server.

5. When integrating with the other applications or the third party tools, ensure that the other applications should not be provided with more rights than required. Discuss within the team, why the application requires a certain level of access and what damage it can cause. Also, ensure to validate the author and company behind the application is legitimate and credible.

References

- https://git-scm.com/
- https://git-scm.com/about/branching-and-merging
- https://www.atlassian.com/git/tutorials/what-is-version-control
- https://www.linuxnix.com/what-is-source-code-management-or-version-control/