Jacob Cannamela

CSD-380

6/14/2025

**Module 3 Assignment**

In today's software and configuration management, especially in critical environments where reliability and traceability is key, version control is essential. Simply put, version control is the process of managing and tracking changes in files, documents or code. It assists groups in collaborating, reverting to older versions in case of malfunctions, and maintaining a structured chronological record of changes. In my role as a detection engineer in an enterprise organization who oversees more than 250 Splunk correlation searches and 150,000 identities, I have learned to appreciate the importance of version control. It is no longer a matter of convenience; it is now an operational necessity. Our team follows a Detection-as-Code (DaC) model, which means every SPL search is written, updated, or deleted under strict version control and peer-review processes. This eliminates instability and unmonitored changes that could lead to disruption of critical detection rules.

For my research, I investigated three sets of version control guidelines: Ernst’s practices version control, workflows on Git-based platforms like GitLab, and version control paradigms from Perforce, which focus on enterprise-scale solutions. All three approaches focus on atomic commits, one change per commit, as well as writing clear, informative commit messages. They also emphasize branches for features or bug fixes, where changes must be tested and reviewed before merging. Something unique about Ernst’s model is the focus on file locking and code ownership which, while beneficial to larger teams, may be considered archaic in today’s agile world. Git and GitLab workflows better accommodate distributed development and modern DevOps practices like continuous integration (CI) and automated testing.

In examining these guidelines, it’s clear to me that most of the foundational practices are still relevant. That said, practices like file locking or avoiding binaries are less relevant due to modern tools like Git LFS and cloud-native development workflows. In my normal routines, atomic commits are imperative. If any of our SPL correlation searches starts to trigger false positives or worse, fails silently, the ability to pinpoint the last specific alteration made is key. We ensure that each commit is a single logical change, and we enforce that the developer provides a thorough explanation of the “what” and “why” related to the change. If that discipline is not followed, our investigations would take longer and our detection pipeline may become unreliable.

We have also integrated branch-based development as one of the core practices. Each individual change is stored in its own branch and direct pushes to the mainline branch are strictly prohibited. After the work is done, we initiate a merge request which goes through the review process. Automated checks for SPL syntax as well as running test data through the updated search are done by our CI pipeline during this process ensuring that broken searches cannot make it into production. In the past, we have experienced an incident where a filter in a correlation search was modified which caused thousands of alerts to spam our security analysts yielding alert fatigue and ultimately breaking the SIEM tool and SOAR platform. Thanks to version control and CI, we are now able to prevent this type of thing in the future.

I believe the most critical version control practices are: adopting atomic commits, composing clear commit messages, enforcing branch protections with mandatory peer reviews, validating integration through Continuous Integration (CI), and ensuring a full audit log is preserved. The audit trail captures every single action within the system, preserving accountability. These aren’t just nice to have policies. They are foundational for handling detection logic. With more than 250 splunk correlation searches and constant contributions from our engineers, version control helps maintain order within our workflows, keep our detections sharp, and foster team accountability.

As we can see, modern practices and adaptation of these practices, through Detection-as-Code, has allowed us to create a version-controlled environment that ensures full trust, testing, and tracking for every SPL search. In the field of detection engineering, operational security relies on detection and response systems that are trustworthy and reliable, which is why modern version control systems are so essential for all teams. Having a controlled environment not only improves detection and response accuracy, but also enables rapid response when things go wrong. Control, ultimately, enhances trust in the code, trust in the team, and trust in the operational process.

**References:**

*Version control concepts and best practices*. (n.d.). <https://homes.cs.washington.edu/~mernst/advice/version-control.html>

Atlassian. (n.d.-b). *Git Workflow | Atlassian Git Tutorial*. <https://www.atlassian.com/git/tutorials/comparing-workflows>

Hajirajabi, M. (2022, January 7). A brief history of Version Control Systems - Mehran Hajirajabi - Medium. *Medium*. <https://medium.com/@mehran.hrajabi98/a-brief-history-of-version-control-systems-vcss-5881f07ba0e1>