CIS-481: Introduction to Information Security

In-Class Exercise #7

IQ Team: 4
Names of team members: Daniel Kearl, Mohammed Al Madhi, Yuxuan Chen, Joseph Baxter

Logistics

A. Get together with other students on your assigned team in person and virtually.
B. Discuss and complete this assignment in a collaborative manner. Don't just assign different problems to each teammate as that defeats the purpose of team-based learning.
C. Choose a scribe to prepare a final document to submit via Blackboard for grading, changing the file name provided to denote the number of your assigned **IQ Team**.

Problem 1
Consider the logical access control needs for joint software development teams using a typical Linux environment. Roles must include Developers (that can commit changes made in the code), Testers, and Code Reviewers. The technical access control mechanisms that you design must reflect these organizational roles. Your access control solution must:

1. Protect the software being developed from outsiders stealing it
2. Protect against unauthorized changes (including from internal actors)
3. Ensure that we can trace *who* made each change

Situation 1: A small team on a single machine *(5 points)*

The machine would control access via groups, using three different groups (named appropriately) with access levels appropriate to their role. Developers would have read, write and execute (rwx) permissions, testers read and execute (rx), code reviewers, assuming their role is only advisory, would have read permission (r). The Other group would not have permission to access files, since users are members of their respective groups. The computer itself would obviously also enforce typical username and password login. Tracing who made changes to files would be done with built-in Linux audit tools (e.g. auditctl).

Situation 2: A medium-to-large team on a LAN [Hint: Use of a version control system like Subversion is highly recommended] *(10 points)*

Access levels would be the same, each group mentioned in solution one would have the same permissions in this solution. The version control system would ideally integrate with Unix users and groups to gate access to repositories. Workstations would enforce user/pass login, and for an added layer of (potentially unnecessary) security, the version control system would also require login before committing any changes for the first time in the user session. For tracing changes, almost the entire point of version control is tracking changes to files, including who made them, so it would be reasonable to assume that we would leave it up to whatever VC software is used by the group.

Situation 3: A large, distributed team, including outsourced contractors *(10 points)*

This situation is a bit more complicated than the others, but similar nonetheless. The team, assuming everyone is part of the three aforementioned groups, would remain the same. However, there would now be an additional group for each contractor (potentially more if only some members of one contractor need different permissions), and each group would have only the minimum necessary permissions to do their job. If an individual contractor is removed from a project, their group of users would simply need to be removed as well.

[Inspired by https://www.cs.columbia.edu/~smb/classes/f09/l08.pdf - many thanks to Columbia University for providing under Creative Commons!]