CIS-481: Introduction to Information Security

InfoSec Chapter Exercise #7

Team: Seven
Participants: Jackson Dillingham, Matthew Jackson, Hilton Siaffa, Tabor Payne, Emily Wantland

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 **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)*

- Discretionary access controls would be the most appropriate. DACs are implemented at the discretion or option of the data user. They allow users to control and possibly provide access to information/resources at their disposal. Since this is a small team, they know who needs access. Unauthorized changes would not be a smaller team's primary concern. Smaller teams hold each other accountable and can implement push requests. A simple CRM can assign tickets and schedule code review meetings. An IDPS can alert the team if the data is breached. If they are committing their code to a site such as Github, they can create repositories in the same organization. This will allow everyone access and keep track of all commits.

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

- Discretionary access controls give certain users specific access. This means no one will have to micromanage the projects. Once again, using a private repository on Github will secure the code and prevent people outside the organization from accessing the software in development. An IDPS can alert the team if unusual activity occurs. A medium-to-large team isn't as close as a smaller one, so activity outside of the baseline can be quite telling. If they are pushing their code to a platform like AWS' Lambda, they will be able to publish versions and assign them to development or production environments. This will allow them to revert to a previous version.

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

- A large team will need several access controls. Non-discretionary access controls are the best fit. They are implemented by a central authority and are strictly enforced. A vulnerability scanner may be necessary. Since there are outside contractors and the team is distributed, it is a good to place protections that will expose risk areas. Finding system weaknesses and patching them will help avoid breaches. Since this is a large team, not every member needs full access to everything. Using a Github extension with Visual Studio will prevent team members from overwriting. A merge request will be required and the user will need to choose whether or not to keep their local code, or what is received on a pull request.

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