



Information System Security Operation

Secure Protected Development Repository

Supporting Trust in Development Environments & Protecting Critical Software Assets

Problem Description

A continuing risk encountered in the development of critical software is the porous and vulnerable nature of current software development environments. These environments provide a broad suite of tools to aid in developing a software design and architecture, developing the software, and performing version management among other tools that serve to accelerate the software development process. Whereas the currently available software development tools improve software developer productivity, and serve to accelerate the code-compile-test cycle, they do not provide the any meaningful aid in protecting software during development.

The range of required protection tools is broad. Examples include tools to handle automatic encryption and protection of software from access while on disk, protection of software during compilation and testing (potentially using virtual machines), documentation of authorized access to the software, and inspection tools to automatically scan developed software for a variety of viruses, Trojan horses, or worms that could be embedded within the software being developed. Insuring the security of a software application when it is assembled from components and/or objects is an additional consideration for the software development environment. This issue is especially salient as object oriented software composition is increasingly viewed as the most promising means for controlling development costs and enabling the assembly of complex software applications. However, this same technique also permits one component/object to compromise the protection of an application during

development or later during execution. Finally, it is most important to develop tools that assist programmers in establishing, tracking, and maintaining the “pedigree” of the software under development. A pedigree documents who has touched the software over the course of its development. It is also an estimation of the extent that the software has “leaked” prior to the addition of protection technology. A “good” pedigree for a specific piece of software can greatly improve the robustness of the applied protection technology. A “bad” pedigree can leave both the software and the protection technologies at risk.

Objective

The objective of the Secure Protected Development Repository project is to enhance the infrastructure for developing software within a development environment that:

- Protects software from theft or piracy;
- Preserves and documents its pedigree throughout the entire development process; and
- Insures that the protection of an application is not compromised through the inadvertent use of a component that alters the security and protection of the application.

The repository will also provide software protection and security for compositional software development practices. We have built a prototype of a development repository designed with these security goals as the primary objective to make software substantially less prone to theft, piracy, and sabotage than existing methods.

SPDR Overview

SPDR is designed to provide support for a secure development environment that protects

This work sponsored by Air Force Research Laboratory (AFRL), Contract Number F33615-02-C-1304, with McAfee Research, which is now the Security Research Division of SPARTA.



Secure Protected Development Repository

Supporting Trust in Development Environments & Protecting Critical Software Assets

software from theft and corruption, preserves and documents its pedigree, and prevents the introduction of malicious code. A key component to protecting software and its associated artifacts is the version control system that serves as a point of control for the development effort. Amoroso developed the Trusted Software Methodology (TSM) based on research in developing trustworthy software. SPDR will provide support for several key Trust Principles outlined in the TSM; in particular, it supports:

- **Access Control:** Identified software lifecycle activity shall be automatically controlled by the software engineering environment in conformance with an explicitly defined security policy.
- **Auditing:** A record of identified software lifecycle activity shall be automatically logged and stored by the software engineering environment in a protected repository.
- **Configuration Management:** A configuration management system shall be established...All configuration items shall be stored in a protected repository that maintains all software versions, software modification requests, and software changes.
- **Identification and Authentication:** The initiation of an identified software lifecycle activity shall be done only by an individual identified and authenticated by the software engineering environment.
- **Least Privilege:** Privileges to perform identified software lifecycle activity shall be allocated and maintained so that a privilege is only given to individuals who require that privilege.
- **Multi-Person Control:** Identified software development activity shall not be completed without the active endorsement and involvement of at least two qualified software developers.

By addressing these principles, the SPDR system enhances the application development infrastructure to maintain software integrity and

pedigree, protect against software theft, and reduce vulnerabilities in developed software.

In addition, SPDR mitigates the risk that a developer will be able to insert malicious code into the system by incorporating the McAfee[®] VirusScan[™] engine into the process of checking-in code to the repository. Since a development repository often stores scripts, documents and binaries, as well as source code, there are ample opportunities to deposit viral code into the repository.

Security Posture

SPDR combines both prevention and detection techniques to protect critical software resources. We can divide the possible attackers into two spaces: those with access to the system, and those without. We cannot prevent insiders from retrieving information for which they are authorized. However, the system can securely track their actions, preventing them from retrieving data without attribution. Additionally, the system can require multi-party cooperation for some actions on the repository, thus increasing the risk insiders face when accessing the system.

By completely protecting the repository, unauthorized attackers will not be able to retrieve valuable assets from the repository, nor will they be able to corrupt the integrity of the system without detection.

SPDR pushes security vulnerabilities to the client developer machines realizing that the developers must have access to code in the clear, and that a properly administered policy set will prevent all code from appearing on a single platform in the clear. The workstations were already a potential vulnerability in existing systems for the reasons given about needing to access plaintext code. The SPDR system dramatically reduces the vulnerability of the central repository with no change to the vulnerabilities presented by the workstations, shifting risk from the central repository to key management solutions for distributing and protecting user credentials.