

# AWDRAT: A Cognitive Middleware System for Information Survivability \*

Howard Shrobe and Robert Laddaga  
MIT CSAIL  
32 Vassar Street  
Cambridge, MA 02139

Bob Balzer and Neil Goldman  
and Dave Wile and Marcelo Tallis  
and Tim Hollebeek and Alexander Egyed  
Teknowledge  
4640 Admiralty Way, Suite 1010  
Marina del Rey, CA 90292

## Abstract

The Infrastructure of modern society is controlled by software systems that are vulnerable to attacks. Many such attacks, launched by "recreational hackers" have already led to severe disruptions and significant cost. It, therefore, is critical that we find ways to protect such systems and to enable them to continue functioning even after a successful attack.

This paper describes AWDRAT, a middleware system for providing survivability to both new and legacy applications. AWDRAT stands for Architectural-differencing, Wrappers, Diagnosis, Recovery, Adaptive software, and Trust-modeling. AWDRAT uses these techniques to gain visibility into the execution of an application system and to compare the application's actual behavior to that which is expected. In the case of a deviation, AWDRAT conducts a diagnosis that figures out which computational resources are likely to have been compromised and then adds these assessments to its trust-model. The trust model in turn guides the recovery process, particularly by guiding the system in its choice among functionally equivalent methods and resources.

AWDRAT has been used on an example application system, a graphical editor for constructing mission plans. We present data showing the effectiveness of AWDRAT in detecting a variety of compromises to the application system.

## Overview

To the extent that traditional systems provide for immunity against attack, they rely either on detecting known patterns of attack or on detecting statistically anomalous behavior. Neither of these approaches is satisfactory: The first approach fails in the face of novel attacks, producing an unacceptably high false negative rate. The second approach confounds unusual behavior with illegal behavior; this produces unacceptably high false positive rates and lacks diagnostic resolution even when an intrusion is correctly flagged.

In this paper, we present AWDRAT, a middleware system to which an existing application software may be retrofitted

and that provides immunity to compromises of the system by making it appear to be self-aware and capable of actively checking that its behavior corresponds to that intended by its designers. "AWDRAT" stands for Architectural-differencing, Wrappers, Diagnosis, Recovery, Adaptivity and Trust-modeling; it provides a variety of services that are normally taken care of in an *ad hoc* manner in each individual application, if at all. These services include fault containment, execution monitoring, diagnosis, recovery from failure and adaption to variations in the trustworthiness of the available resources. Software systems tethered within the AWDRAT environment behave adaptively, and with the aid of AWDRAT these system regenerate themselves when attacks cause serious damage.

## The AWDRAT Architecture

The AWDRAT architecture is shown in Figure 1. AWDRAT is provided with models of the intended behavior of the applications it is intended to protect. These models are based on a "plan level" decomposition that provides pre- and post- and invariant conditions for each module. AWDRAT actively enforces these declarative models of intended behavior using "wrapper" technology. Non-bypassable wrappers check the model's conditions at runtime, allowing execution to proceed only if the observed behavior is consistent with the model's constraints. We call this technique "Architectural Differencing". In the event that unanticipated behavior is detected, AWDRAT uses Model-Based Diagnosis to determine the possible ways in which the system could have been compromised so as to produce the observed discrepancy. AWDRAT proceeds to use the results of the diagnosis to update a "trust model" indicating the likelihood and types of compromise that may have been effected to each computational resource. Finally, AWDRAT helps the application recover from failure, using this trust model to guide its selection of computational techniques (assuming that the application has more than one method for carrying out its intended tasks) and in its selection of computational resources to be used in completing the task.

AWDRAT uses its model of an application's intended behavior to recognize the critical data that must be preserved in case of failure. AWDRAT generates wrappers that dynamically provision backup copies and redundant encodings of this critical data. During recovery efforts, AWDRAT installs these backup copies in place of compromised data resources.

Using this combination of technologies, AWDRAT provides "cognitive immunity" to both intentional and accidental compromises. An application that runs within the AW-

---

\*This article describe research conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided by the Information Processing Technology Office of the Defense Advanced Research Projects Agency (DARPA) under AFRL Contract Number FA8750-04-2-0240. The views presented are those of the author alone and do not represent the view of DARPA or AFRL.  
Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.













