# CWE Detail – CWE-1249

## Description

The product provides an application for administrators to manage parts of the underlying operating system, but the application does not accurately identify all of the relevant entities or resources that exist in the OS; that is, the application's model of the OS's state is inconsistent with the OS's actual state.

## Extended Description

Many products provide web-based applications or other interfaces for managing the underlying operating system. This is common with cloud, network access devices, home networking, and other systems. When the management tool does not accurately represent what is in the OS - such as user accounts - then the administrator might not see suspicious activities that would be noticed otherwise. For example, numerous systems utilize a web  
 front-end for administrative control. They also offer  
 the ability to add, alter, and drop users with various  
 privileges as it relates to the functionality of the  
 system. A potential architectural weakness may exist  
 where the user information reflected in the web  
 interface does not mirror the users in the underlying  
 operating system. Many web UI or REST APIs use the  
 underlying operating system for authentication; the  
 system's logic may also track an additional set of  
 user capabilities within configuration files  
 and datasets for authorization capabilities. When  
 there is a discrepancy between the user information in  
 the UI or REST API's interface system and the  
 underlying operating system's user listing, this may  
 introduce a weakness into the system. For example, if an  
 attacker compromises the OS and adds a new user  
 account - a "ghost" account - then the attacker could escape detection if  
 the management tool does not list the newly-added  
 account. This discrepancy could be exploited in several ways: A rogue admin could insert a new account into a system that will  
persist if they are terminated or wish to take action on a system that  
cannot be directly associated with them. An attacker can leverage a separate command injection attack available through the web interface to insert a ghost account with shell privileges such as ssh. An attacker can leverage existing web interface APIs, manipulated in such a way that a new user is inserted into the operating system, and the user web account is either partially created or not at all. An attacker could create an admin  
 account which is viewable by an administrator,  
 use this account to create the ghost account,  
 delete logs and delete the first created admin  
 account. Many of these attacker scenarios can be  
 realized by leveraging separate vulnerabilities  
 related to XSS, command injection, authentication  
 bypass, or logic flaws on the various systems.

## Threat-Mapped Scoring

Score: 3.25

Priority: P2 - Serious (High)

## Modes of Introduction

**•** Architecture and Design: The design might assume that the underlying OS does not change.

**•** Implementation: Assumptions about the underlying OS might be hard-coded into the application or otherwise in external data stores in a way that is not updated when the OS's state changes.

## Common Consequences

**•** Impact: Varies by Context — Notes:

**•** Impact: Hide Activities — Notes:

**•** Impact: Unexpected State — Notes:

## Potential Mitigations

**•** Architecture and Design: Ensure that the admin tool refreshes its model of the underlying OS on a regular basis, and note any inconsistencies with configuration files or other data sources that are expected to have the same data. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** This new user2 account would not be noticed on the web interface, if the interface does not refresh its data of available users.