# Protecting configuration files using AIDE and checking for file integrity.

**Project files used:**

Apache pilot project: [DELETED FOR PRIVACY PURPOSES]

AIDE check: [DELETED FOR PRIVACY PURPOSES]

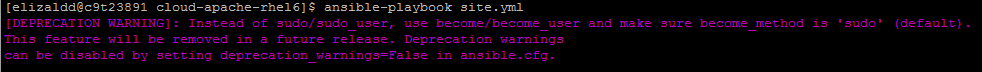
Tools needed:   
  
Ansible (ITG server: [DELETED FOR PRIVACY PURPOSES]), at least one redhat client, and your ssh keys properly configured

(If you’re not able to download these files please contact AIS engineering: [DELETED FOR PRIVACY PURPOSES])

**Summary:**This project is meant to act as a test for future provision of Apache middleware and protecting the original configuration files using a combination of “chattr” attribute command and “AIDE” (http://www.cs.tut.fi/~rammer/aide/manual.html) which is a free tool that comes with RHEL and also providing the application teams with a new “custom” folder for them to use their own custom configuration files for both http and https.

**Example:   
Part 1. Using “cloud-apache-rhel6” project.**Download both projects into your Ansible server. Configure your hosts file, for this example this is what my “/etc/ansible/hosts” file looks like:  
  
[webservers]  
c9t24179.itcs.hpecorp.net  
[webservers:vars]  
ansible\_user=ansibleusr  
ansible\_ssh\_private\_key\_file=/home/ansibleusr/.ssh/id\_rsa  
owner=paasweb  
group=paaswebgrp  
http\_port=80  
https\_port=443  
apache\_content\_dir={{base\_dir}}/apache/www  
apache\_logs\_dir={{base\_dir}}/logs/apache  
ssl\_enabled=yes  
public\_interface=eth1  
siteminder\_home=/opt/ca/siteminder12

Run “**site.yml**” from **cloud-apache-rhel6 project**:

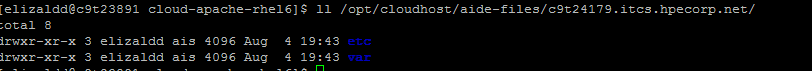


This playbook is in charge of provisioning the target node with the old-fashioned way of installing Apache and all its dependencies. The only new features would be the following:   
  
1) Creates **“/opt/cloudhost/apache/custom**” folder. Within this directory we will find “**conf**” and “**conf.d**” which will server for the application teams to enter their own configuration without modifying the original httpd.conf and ssl.conf files in “/etc/httpd”

2) Locks down /etc/httpd/conf/httpd.conf and /etc/httpd/conf.d/ssl.conf for editing using “chattr” as in “chattr +i /etc/httpd/conf/httpd.conf”. If we go to the target node and use “lsattr” on the httpd.conf file, the output would be similar to the screenshot below. “i” stands for immutable which means not even a root user will be able to modify the file, unless this attribute is removed.



3) Installs AIDE, initializes the AIDE database in the target node.  
  
4) Retrieves the AIDE database, the httpd.conf and ssl.conf from the target node and saves it in the ansible server. It also deletes the AIDE database file from the target node. Files retrieved from the node will be stored in **“/opt/cloudhost/aide-files/$nodefqdn**”.

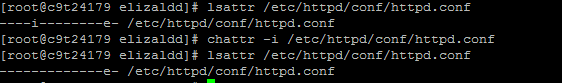


This step is very important, as these files are required by the “**cloud-apache-aide-check**” project to check for file integrity. As explained next.

**Part 2. Using “cloud-apache-aide-check” to verify file integrity.**

Now that we have the aide database and the original configuration files, we can simply run “aide-check.yml” playbook from the “cloud-apache-aide-check ”project against the target node to verify the configurations file have not been compromised/changed.  
If the playbook detects the files have been altered, it will delete the httpd.conf and ssl.conf from the target node, and the original configuration files we saved in our ansible server will be transferred, then a new aide database will be initialized and retrieved. If file check is successful, the playbook will do nothing.

Let’s try for example altering the “httpd.conf” file in our target node. Remember this file is locked by “chattr” so to edit it we just need to remove this attribute as root:



And this is all it is needed for the file to be considered “compromised” even if we don’t edit the file itself. Aide will detect the change and then the playbook will replace the file with the original Config files, then reinitialize the Aide database again and retrieve it. Have a look at the following screenshot showing the output of our playbook:  
  
