Software Security Analysis Steps

* Gather the security policy of the system. The security policy tells us what is malicious behavior that is anything not allowed by the policy.
  + Who are the authorized users? Who controls this list?
  + What are the authorized user actions? Which privileges are given? Which privileges are given to software services/components?
  + List all the roles and the privileges given to them. E.g a regular user should not modify configuration file but a daemon user can.
* High-level Security goals
  + What are the security goals in the system? For example, a goal of the VOMS-Admin is to protect the integrity of membership data. Another goal is to provide appropriate attributes to members.
  + Are there any confidential data in the system? E.g. user credentials?
  + Are there any data its integrity is important
  + What are the system’s availability requirements?
* Perform architectural analysis
  + Identify software components
  + Draw a process flow diagram that shows interactions between the software components. The interactions should show the data transferred between the components as well as the actions taken over the data
  + Identify the resources accessed by components
  + Label the privileges necessary to access these resources.
  + Re-draw the process flow diagram to show how resources are accessed during the component interactions
* Trust analysis. List all assumptions (implicit or explicit) made during the process flow diagram. To identify the trust relationships:
  + For any interaction that involves data transfer, consider the security properties of the data: Confidentiality, Integrity, Availability, Authorization and Authentication. Are there any assumptions over any of these properties. For example, a user sending his password to an email server trusts the server protects the confidentiality of his password. The email server assumes network layer protects the integrity of the password data.
  + To understand the authorization assumptions, list the actions expected on the data and the actions not allowed on the data.
  + To understand authentication assumptions, list what actions are required to prove one’s identity. For example, a web server assume that only the authorized user has access to his web cookie. Sending his web cookie to the server proves the user’s identity.
* Identify high value assets

VOMS-Admin Trust Architecture

Security Policy:

* There are four roles: root, daemon, VO admin, and user
* A user with a valid certificate can only make requests for himself. More specifically user can make a request for the person identified by the DN that is included in the request. A user cannot make requests on behalf of another user, nor remove or modify other users account. A user may request for registry or removal of his own membership. A user can update their account information although this is not explained in the architectural analysis.
* A VO Admin can approve a user’s registration or removal request. Admin has read and write access to all membership data. It is not clear who manages the Admins. There is a group of individuals working as admins. It is not clear if one admin has super privileges to manage all other admins.

Trust relationship types:

What are the universal security policies in across all of the packages.

A logic of authentication

Lamport protocol anaylsis. What are the formal trust notations. Needham’s papers. What are the trust description. What are the properties in protocol analysis.

Different trust types. Define trust in human terms.

End of july is the meeting for the security team here. Defense and nato person will be here.

**Universal rules:**

Software should not allow any arbitrary code to run under its own account.

Compartmentalization between the accounts. If there are multiple users, software should have multiple accounts, and there must be authN authZ framework and software must prevnt users from accessing one another’s files.

Separate the system accounts (root, db, os accounts) from user accounts. Attack models are different. Attacking condor accounts is different from attacking system accounts. software accounts act as the second os.

Map privileges of users to objects. Are operations allowed to perform on another user’s objects?

All data passed from user should all together propagate.

User input mistakes. Assumptions of the privilege model is not what it should be. Condor user write a log file and root reads and executes the log file. This is fundamentally same as user input case.

Vulnerability Summaries and Security Policy of Condor

1. 2005-0001 Condor check\_point server runs under condor account. It allows any user to read/write condor-owned files including /etc/passwd. Policy Rule Violated (1): No user should have read/write access to condor owned files.

Policy Rule Violated (2): Condor user should not have read/write access to files that are owned by root, such as /etc/passwd or condor\_configuration files which is executed with root privileges to start condor daemon.

1. 2005-0003 Shadow and gridmanager daemons do not sanitize/check user inputs properly. This allows a user to execute arbitrary code under condor account privileges. Due to a violation of Rule (2), a user can execute arbitrary commands with root privileges.

Policy Rule Violated (2): Condor user should not have read/write access to files that are owned by root

Policy Rule Violated (3): A user cannot execute arbitrary commands under any privileged accounts such as condor.

Policy Rule Violated (4): A user can only execute jobs under his own authorized account.

1. 2005-0004 Condor daemons have ownership over condor configuration files. Condor daemons are started as root processes. Daemons are only allowed to change the configuration parameters to certain authorized values. However, there are no checks enforced. If the condor account is compromised, this can lead to compromise of root account since condor daemon is started as root.

Policy Rule Violated (2): Condor should not have read/write access to files that are executed with root privileges.

Policy Rule Violated (5): Configuration values should be checked against an authorized list.

1. 2005-0006 Startd authenticates the schedd only based on host-ip. Startd runs a user job under any account name sent by the schedd. A user can spoof the ip address of an authorized schedd and submit a job under condor account. Due to a violation of Rule (2), a user can execute arbitrary commands with root privileges.

Policy Rule Violated (2): Condor user should not have read/write access to files that are owned by root

Policy Rule Violated (3): A user cannot execute arbitrary commands under any privileged accounts such as condor.

Policy Rule Violated (4): A user can only execute jobs under his own authorized account.

1. 2006-0001 User can inject arbitrary commands into stork server. This allows executing arbitrary code under user’s account on the stork server. If the stork server is configured to run under condor account, then user can execute arbitrary code under condor user. Due to a violation of Rule (2), a user can execute arbitrary commands with root privileges.

Policy Rule Violated (2): Condor user should not have read/write access to files that are owned by root

Policy Rule Violated (3): A user cannot execute arbitrary commands under any privileged accounts such as condor.

1. 2006-0003 and 2006-0004 User is allowed to execute some external scripts under a special directory (/condor/scripts) if an execution host allows it. User input to ClassAds is not sanitized/checked. User can execute arbitrary code under condor privileges. Due to a violation of Rule (2), a user can execute arbitrary commands with root privileges.

Policy Rule Violated (2): Condor user should not have read/write access to files that are owned by root

Policy Rule Violated (3): A user cannot execute arbitrary commands under any privileged accounts such as condor.

Policy Rule Violated (6): A user can only execute external scripts defined in the /condor/scripts directory.

1. 2006-0005 A user can modify the condor\_schedd log file. When schedd restarts, it uses the log file to start jobs. User can execute arbitrary code with condor privileges or any other user privileges. Due to a violation of Rule (2), a user can execute arbitrary commands with root privileges.

Policy Rule Violated (1): No user should have read/write access to condor owned files

Policy Rule Violated (2): Condor user should not have read/write access to files that are owned by root

Policy Rule Violated (3): A user cannot execute arbitrary commands under any privileged accounts such as condor.

Policy Rule Violated (4): A user can only execute jobs under his own account.

Policy Rule Violated (7): A user cannot start/stop/restart any daemon with a higher privilege such as schedd.

Policy Rule Violated (8): A user is not permitted to execute condor queue management commands.

1. 2006-0006 A user can spoof identity of any user in the system due to condor’s file system based authentication. The user cannot spoof root user because condor startd does not run any user job with root privileges. However, the user can submit jobs with condor privileges which then can run as root.

Policy Rule Violated (2).

Policy Rule Violated (3): A user cannot execute arbitrary commands under any privileged accounts such as condor

Policy Rule Violated (4): A user can only execute jobs under his own account.

1. 2006-0008 On some platforms, a temporary configuration file is created under /tmp. An attacker can take advantage of a race condition and modify the configuration file. Attacker can execute arbitrary code as privileged user. (Rule: users should not run under privileged accounts)

Policy Rules Violated: 1, 2, 3, 4

1. 2006-0009 Condor negotiator does not check/sanitize the input values sent by startd that can lead to buffer overflow attacks. Any user can invoke the vulnerable function call because negotiator only performs host-based authentication of startd and schedd hosts.

Policy Rule Violated (9): A user cannot directly invoke any function calls to the condor negotiator. Only authorized startd and schedd daemons can send input to negotiator.