Job Scheduler Security Guide

A component of Mark Dickinsons Unix Job Scheduler

This manual covers recomeded security settings for Marks Job Scheduler.
This manual is for version V1.13 of Marks Job Scheduler

Other reference material available
jobsched_cmd User guide
Job Scheduler Messages Manual
Job Scheduler Daemon Configuration Guide
Job Scheduler Utilities Manual (jobsched_take_snapshot/jobutil)
GPLV2 license is at http://www.gnu.org/licenses/gpl-2.0.html

Tested under Linux (Fedora, CentOS), and Solaris (2.8, 10 and OpenSolaris)

Table of Contents

1. Who should read this

This document is intended for system administrators and security administrators who may have concerns over the use of this program, and how to secure it more tightly than provided by the default installation.

It covers the default settings that the program will run with as shipped, and what you can do to tighten up on some of the settings to further limit access, and why you might want to.

You ***must read the entire document before changing anything***, as some configuration changes identified here may prevent the job scheduler application from accepting requests, unless they are done in conjunction with other changes.

2. The jobsched_daemon network settings and implications

2.1 Port and listening IP addresses by default

The default settings for the job scheduler server is for it to listen on port 9002 on every available tcp-ip interface on the server. This is fairly standard for server applications.

The implication of this is however that the default configuration is not suitable for any server that has an interface facing the external world.

2.2 Changing the default network settings

The jobsched_daemon accepts two parameters. The first changes the default port address. The second overrides the default option of binding to all tcp-ip interfaces and forces the scheduler server to bind to only one ip-address.

If you wish to override the tcp-ip interface to force it to bind (listen on) only one ip address you must also provide the port number parameter.

If the server is running on a firewall or other server with a network interface to the external world then you should force it to bind only to one of the addresses on the internal network; it should not be allowed to default to accept requests from the external network.

Update the /etc/init.d script your site uses to start the job scheduler server to include the port number and tcpip address to listen on as per the instructions in the manual for the jobsched_daemon server

> Basically the syntax is...
> ./jobsched_daemon port-num ip-address
>
> example: ./jobsched_daemon 9002 169.20.5.8

2.3 Considerations to note with changing the defaults

If you change either the port number or the interface address the server will bind to the job scheduler server application is only accessable via that ip address (ie: localhost 127.0.0.1 is not usable) so you must provide the port number and ip-address every time you run the jobsched_cmd program in order for it to be able to connect to the server. I would recommend setting up an alias for users to use in this case.

Also if your site uses the web browser interface you must update the cgi scripts used by the supplied sample browser pages to reflect the changes you have made.

Likewise the J2EE Tomcat configurion files will need to be updated if you have implemented that.

3. The provided Security Reporting program should be run regularly

On a periodic basis you should run the supplied program security_checks. This will identify
- any job command files secured dangerously
- any jobs that are configured to run under a unix id that is no longer on the system
- any autoauth users that no longet have unix user records
- all users that have not logged on in over 60 days

The first check is to ensure the command files to be run by the job scheduler are tightly secured. All the command files will be checked to ensure they are owned by, and only able to be edited by, the unix user the command is to be run under; it is no good tightly securing access to the job scheduler if any user can update the programs that are being run.
This is a check that is often overlooked by most people, on my servers I also have a script that performs the same checks on files that are run by cron; it is amasing how many badly secured cron scripts can be found being executed from the root crontab.

The second and third checks are pretty self-explainatory.

The last check is simply a maintenance check rather than a security check. If a user has not logged on in over 60 days that user may no longer require access and could possible be deleted from the job scheduler user database.

Syntax: security_checks directory-path
where directory-path is the directory the job scheduler databases are in.

Example: security_checks /opt/dickinson/job/scheduler

Notes: returns 0 if no probs, 1 if any issues were found.
Ideally this should run from a script daily that mails results if alerts are found.

4. Audit trails

The job scheduler server records all activity to a datestamped log file.
As well as the sndard job stop/start information all user login/logoff activity is recorded plus all changes made to job definition databases are logged against the user record that performed the change.

The log files should be reviewed regularly.

5. The auto-login user

5.1 What is it and why is it needed

The job scheduler will always have a user record for a auto-login user, this userid is authorised for use only by the root user and has full administration authority.

This record allows the root user to 'autologin' to administration authority without a password. ***This is required !***. It permits the root user to login to the scheduler to cleanly shut it down from the init.d script without the need to hard code a password in the init.d scipt; which is obviously a good idea.

This is not considered a security risk as only the root user has access to this userid, and you should have access to the unix root userid tightly locked down in the first place.

This access is also what you will use initially to add the first real adminsitration user.

5.2 Security implications for auto-login from remote access

The check for whether it is the root user requesting access to this userid is done by the jobsched_cmd program itself, which checks the effective userid of the user running the program to determine if the user is root.

It does not neccesarily follow that the jobsched_cmd program is being run on the same physical server as the job scheduler server being connected to. This means that to secure the job scheduler environment you must tightly control root access even on development servers, to avoid the possibility of a user on a losely controlled development server obtaining adminsitrator access on any remote production scheduler.

This issue is limited to the auto-login userid which must exist on every server. For normal users this is not an issue as normal users are authorised against individual user records, if a user is not authorised they simply won't have a job scheduler user record on the production scheduler so can't login remotely; but the auto-login userid will always exist on every job scheduler instance.

5.3 Recommendation

Regularly run reports against the audit logs to monitor if the auto-login user is being used. Ideally any user should match a system shutown/restart period; if not your root user is being used far too often :-).

6. General User Security Considerations

6.1 Overview

Every user connection to the job scheduler automatically connects with a session limited to browse (guest) only access. This browse access is granted to all connections without a login being required. This is usefull for support, you don't have to add specific users with read-only access as it's the default.

At this point there is no intention to make browse access require authentication.

As no actual operations, other than display commands, are permitted from a browse only connection you need to add individual user records for those users that require update access to any functions.

Where possible the job scheduler userid should match the users unix userid if they have one, as that will optionally allow the use of the autologin for the user. It is not required that the userid match a unix userid however.

6.2 Access Levels available to a user

It is recomended that each individual user that is expected to be able to perform scheduler tasks be allocated their own personal job scheduler user record that grants them only the access needed to perform their jobs.

The access levels that are available for a user are...

- ADMIN - all access, is effectively root
- SECURITY - user management only. Is not able to update anything except user records (of course they could always create a temporary user with admin and do whatever they want that way, but any security team anywhere could do that with any app, so you just have to trust them)
- JOB - job and calendar definition management only. Can create and delete job and calendar definitions, but has no access to submit jobs for execution or manage the active schedule
- OPERATOR - active scheduler queue management only. Can submit new jobs if needed, restart jobs from a failed state, hold/release jobs etc. Has no access to update job, calendar or user entries
- BROWSE - display only access (default). Can issue display commands. Can update nothing.

6.3 SubsetAuth Feature

An individual user may be granted the ability to create other users with the same or less priviledges than that user without needing a security access level.

This function is intended to allow a functional administrator, for example a manager or team leader, to create user records for other members of that team, with the same or less functionality.

This can be used in environments whete the security team do not want to be involved in managing every user add request.

However, at this point only security authority group members can delete user records.

6.4 Userid authority list

This is relevant only to userids in the JOB authority group. A user can be defined with a list of up to ten (10) unix userids for which the job scheduler user is permitted to create job definitions for.

This can be further used to control access to unix system resources. For example a development user may have “test1,test2” wheras a systems administrator could be set with “root,test1,test”; so development users can create jobs to run under multiple test userids but only a systems administrator could create job definitions to run as the root user. Which admitedly is not a great example as a systems administrator would normaly have admin authority and be able to do whatever they want anyway.

Refer to the jobsched_cmd manual for further information.

7. A note on development systems

While it is required that the job scheduler run under the root userid in order to submit jobs under multiple userids that is not necessarily required for a development system.

Theoretically you could run the job scheduler under a development userid and on all JOB authority users records only allow them to create job definitions to be scheduled under that one userid. If the job scheduler does not not need the ability to run jobs under multiple userids but just under the userid the job scheduler is started as that should work.

I haven't tested that however.