Cron - the Linux task scheduler for incident responders

Taz Wake | Halkyn Consulting Ltd | Originally published 2024-02-13

Introduction

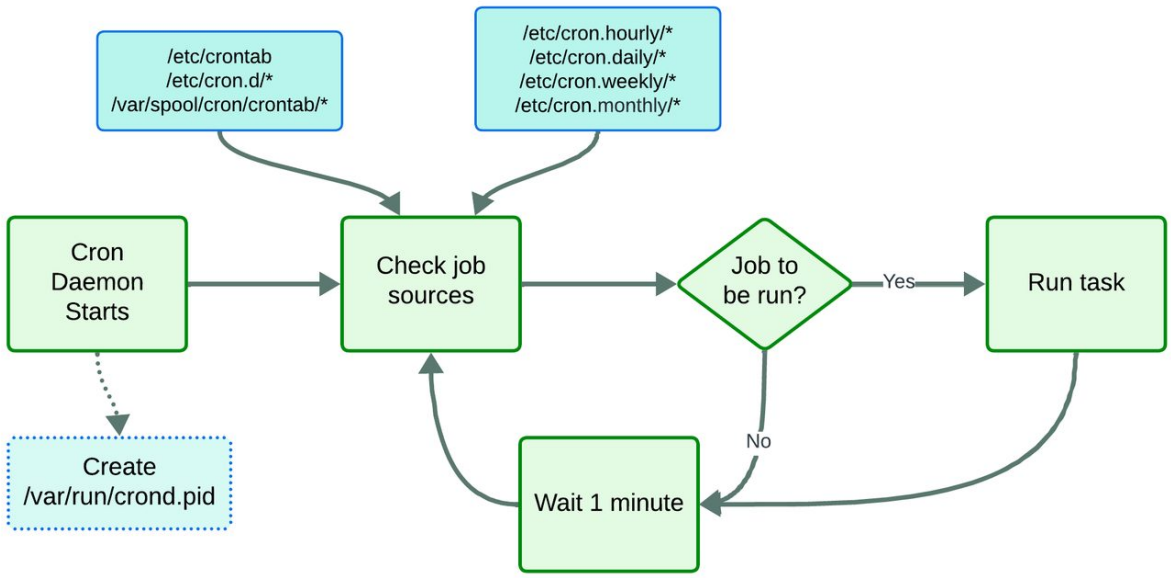
Cron is a time-based job scheduler in Unix-like operating systems, such as Linux, used for automating system maintenance and administration tasks. It has been around since pretty much the start of Unix and is a fundamental tool for sysadmins. The name comes from the Greek word "chronos", meaning "time" which is also often associated with figures from Greco-Roman mythology.

Cron allows tasks to be automatically performed at a specific time (such as 0900 every day) or at intervals (such as every 15 minutes). The tasks themselves can range from simple scripts to complex programs and binaries.

Its nature as the task scheduler means Cron is extensively abused by attackers looking to maintain persistence when they get access to Linux devices. Whenever you are responding to any intrusion involving Unix-like Operating Systems, you absolutely need to check Cron!

Note: *Linux has another task scheduler - the at daemon. This is normally used to set a one-time task which needs to run at a given time. How at works is outside the scope of this article but will be covered in a future one.*

How does Cron work?



A high-level overview of how Cron works.

The task scheduler daemon, Cron, generally starts as part of the boot cycle, This is managed by Systemd/Initd so it is possible to have it start separately, but it is very rare to find any changes here.

Once initialised, the Cron daemon checks to see if exists on the filesystem and if it does, it checks to see if it contains the Process ID for a running process. If it doesn't, the daemon continues to start up and writes its Process ID to this file. This is used to prevent multiple instances of Cron trying to run and help protect system resources.

Once the daemon has loaded into memory, it starts a timer and checks the "*crontabs*" and other task sources.

Crontabs

These are files that contain configuration information detailing what tasks should run and when. There are multiple crontab locations in most Linux systems:

- **/etc/crontab** - This is the system-wide crontab file.
- **/etc/cron.d/*** - This is a directory holding system-wide crontabs for installed applications and is normally modified or updated by Sysadmins or package management tools.
- **/var/spool/cron/crontabs/*** - This is where user crontabs live. There is normally a folder here for each user account that has created scheduled tasks.

The crontabs are plain text files with a set structure breaking down when the events will run.

```
root@siftworkstation:/cases# icat -o 227328 starkskunk5.E01 378
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Example /etc/crontab contents

Each job has its own line with the entry determining when it will run. The fields dictate the timing based on *minute - hour - day of the month - month - day of the week*. In the system crontabs, there is also the option to specify the user account the task runs as.

For example, take the following line in :

This would mean the file at would run, under the root account context, at 10 minutes past the hour, every hour but only on the 3rd day of each month.

In reality, the configuration of crontabs can become incredibly complicated involving ranges and words.

This will create a task that runs at 22 and 23 minutes past the hour, between 0400 and 0559 on the 12-18th of August each year.

Another, complex, example - this time running a task at 15 minutes past the hour, every 3 hours, every day:

Some important points about the Crontab:

- If you specify values other than * for the Day of the Month (field 4) **and** Day of the Week (field 6), the result can be unexpected. Cron will normally trigger when *either* condition matches.
- Using the "star slash" format to specify every "period" only really works on minutes and hours. Cron doesn't support nth-day syntax and if you use something like the string below, it will actually go for "every day divisible by 3" rather than "every 3rd day."

Regular tasks

In addition to the task data stored in the various crontabs, the Cron daemon also uses four directories to trigger activities at set intervals. These generally hold scripts or executables and launch at specific intervals.

They do not have the same concept as the crontab, in that there is no direct configuration file. They are just run at the specified intervals, although the exact interval is specified by the file. Sometimes this can be configured by . The locations are:

- /etc/cron.hourly
- /etc/cron.daily
- /etc/cron.weekly
- /etc/cron.monthly
- (some systems may have an /etc/cron.yearly but this is not standard)

The filename indicates the interval the files are expected to run - however, modifying the crontab can trigger the contents of to fire once a month or every minute if the administrator wishes it.

Incident Response

The cron system is heavily abused by attackers looking to maintain persistence on Linux-like platforms. As a result, it is imperative that the related artifacts on the filesystem, and in memory, are checked when assessing systems.

Parent-Child Processes

Anything running under the Cron scheduler will show up as a child process of one or more Cron processes. This is often down to the parent process being the Cron daemon, which then launches a cron child process at the specified interval, that calls the application.

This is an example of how a script running under Cron might look when viewed with the command.

```
root@siftworkstation:/cases# pstree -h 1275
cron---cron---sh---m.sh
```

Evidence collection

As well as memory, you should collect all the available Cron data during your evidence collection. Tools such as UAC have preconfigured options but you can also build your own. On Github I have provided an example that could be used to build something which works in your environment. (*Note: the example on Github will produce some errors based on the path and user entries in a crontab*)

Ultimately it doesn't matter which approach you use, but you do need to analyse the contents!

The good news is that most crontabs aren't very noisy. You shouldn't expect to see more than 3-4 entries per user. Things to look for include:

- Entries pointing to unusual locations (such as)
- Entries running under unusual accounts
- Entries with unusual names or typos
- Anything that is calling out over the network
- Scripts calling other scripts or signs of obfuscation

With the cron.[hourly|daily|weekly|monthly] folders, you may need to check the individual files for signs of malicious activity.

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858