Using opConfig to Detect Unwanted Software

At Opmantek we needed to find all the servers which were running some software and then uninstall it. Between our product, development and test servers we have about 50 Linux servers to check, checking manually was not an option, so we needed a quick automated way to identify the servers in question.

- 5 minutes to read.
- 15-30 minutes to put into production.

Table of Contents

- Methodology to create an Automation
    - Detection
        - Linux_Software_Installed Command Set
        - Linux_Software_Installed.nmis
        - Running the command set
    - Diagnose
        - Access the Commands Overview
        - Advanced Search
    - Actionable Information
    - Remediation
- Change Detection and Regression
- Conclusion

Methodology to create an Automation

What do we want to automate, how do we detect the condition we want to detect. A simple analogy would be that if the doctors suspects you have a broken bone, they send you to get an x-ray, which confirms the injury or shows that the bone is not broken. This could be referred to as a diagnostic or test.

Detection

In this case I wanted to confirm if ActiveState Perl software was installed on the server, unfortunately, the software does not use a Linux package manager, so we can not use RPM and APT commands. There were two simple ways to verify if the software was installed, firstly run perl -v to see which Perl was being used and to look in /usr/local to see if there were any directories starting with active.

The Linux commands I needed were:

- perl -v
- ls -ld /usr/local/active* /usr/local/Active*

Now I want to run those commands quickly and easily on 50 Linux servers and I want to make sure that no one installs the software again later. A new command set was needed which I called "Linux_ActiveState", I created a new command set file for this and similar things called "Linux_Software_Installed.nmis".

Linux_Software_Installed Command Set

Command sets in opConfig are stored in /usr/local/omk/conf/command_sets.d by default. I copied an existing one and edited it to make it reflect what I needed, importantly this needed to have os_info matching Linux only and I needed to change the two commands, in the most recent version of opConfig for NMIS9 these files are JSON.

To understand the contents it is quite straightforward, os_info means, only run these commands when these os_info conditions are met. Each of the command sections are simple and the tagging system is powerful:

- privileged: means does this require elevated privileges to run, e.g. sudo access
- command: the command you want to run, which is also how the data is saved into the system
- exec: optional if you want to save the command as some other name, use the exec as the command which is actually executed.
- tags: HOURLY means this will automatically run every hour, Linux and operations are handy for finding the command, detect-change and report-change means that opConfig will monitor this command output for change and if a change is found raise an event.

Change detection with change reporting is incredibly powerful, automated change detection to ensure compliance.

Linux_Software_Installed.nmis

The final command set looks like this:

```
%hash = (
  'Linux_ActiveState' => {
    'os_info' => {
      'os' => '/(Linux|CentOS|Ubuntu)/'
    },
    'scheduling_info' => {
      'run_commands_on_separate_connection' => 'false',
    },
    'commands' => [
      {
        'privileged' => 'true',
        'command' => 'perl version',
        'exec' => 'perl -v',
        'tags' => ['HOURLY','Linux','operations'],
      },
      {
        'privileged' => 'true',
        'command' => 'activestate in usr-local',
        'exec' => 'ls -ld /usr/local/active* /usr/local/Active*',
        'tags' => ['HOURLY','Linux','operations','detect-change','report-change'],
      },
    ],
  },
);
```

Running the command set

Because it is tagged with "HOURLY" the command set will run automatically every hour. If you want to run it manually for testing, you run the following command:

```
sudo /usr/local/omk/bin/opconfig-cli.pl quiet=1 nodes=NODE-TO-TEST-WITH act=run_command_sets tags=HOURLY
debug=true
```

Check for any errors, if all good, run manually for all nodes or wait an hour or so.

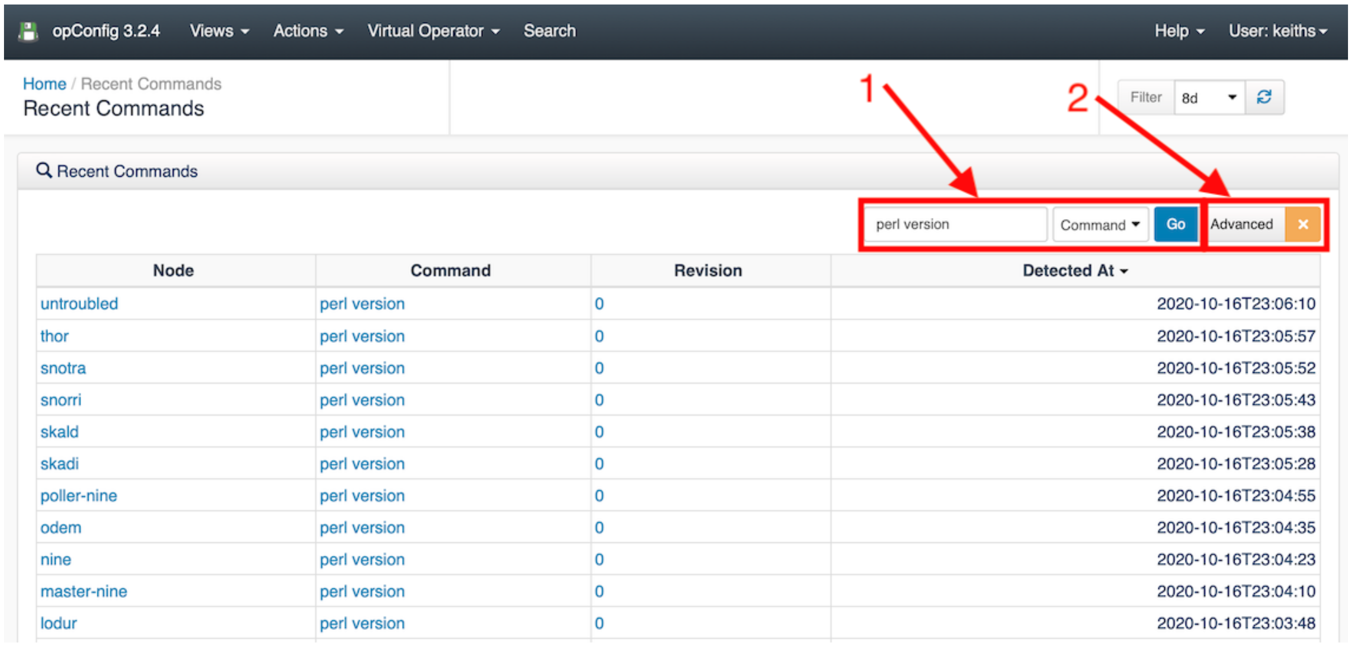
Diagnose

Now I can go to the opConfig GUI and find the matching nodes. The criteria were quite simple, any command of the "perl version" commands which contained the word "ActiveState" would indicate that ActiveState Perl was installed and being used.

Access the Commands Overview

From the opConfig menu, select "Views Commands Overview" and you should be seeing a screen which looks like the one below, first we can see how many instances of "perl version" we have collected.

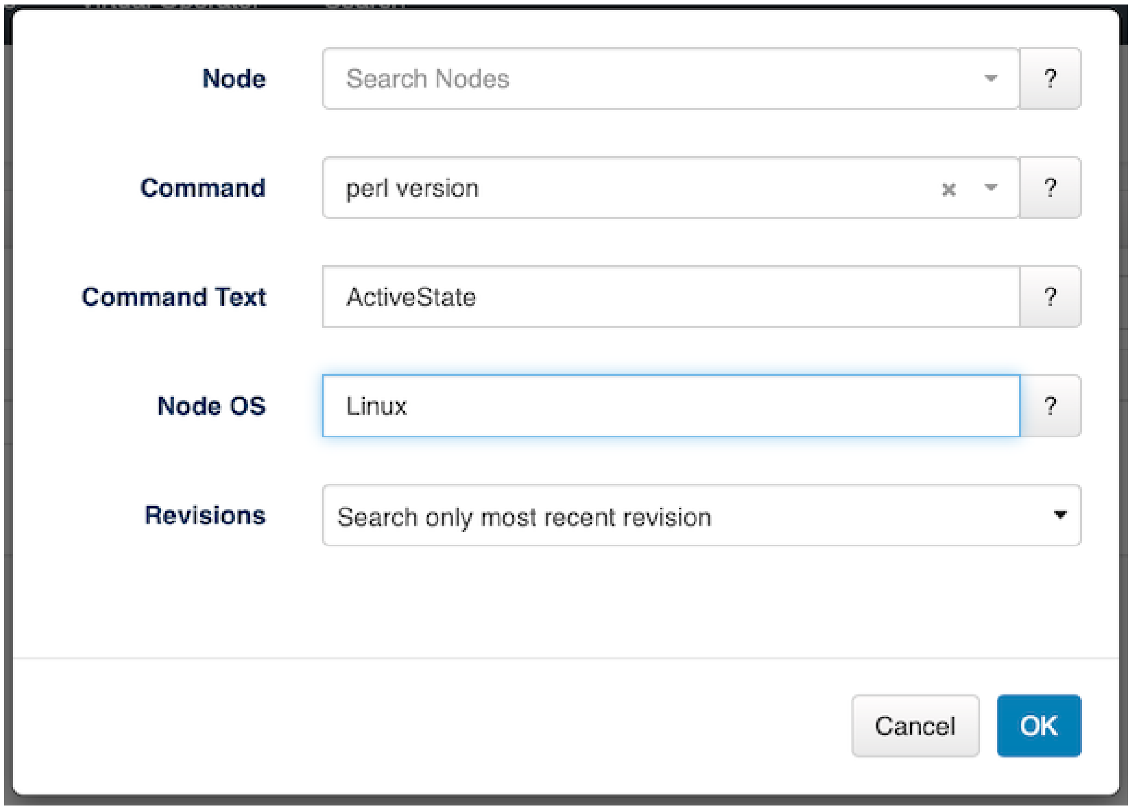
In the box enter "perl version" change the select to "Command" and click "Go", you will have a list of nodes and the command name, all of these are samples we can not check for. Step 2 is to click on the "Advanced" button on the right.



Node	Command	Revision	Detected At
untroubled	perl version	0	2020-10-16T23:06:10
thor	perl version	0	2020-10-16T23:05:57
snotra	perl version	0	2020-10-16T23:05:52
snorri	perl version	0	2020-10-16T23:05:43
skald	perl version	0	2020-10-16T23:05:38
skadi	perl version	0	2020-10-16T23:05:28
poller-nine	perl version	0	2020-10-16T23:04:55
odem	perl version	0	2020-10-16T23:04:35
nine	perl version	0	2020-10-16T23:04:23
master-nine	perl version	0	2020-10-16T23:04:10
lodur	perl version	0	2020-10-16T23:03:48

Advanced Search

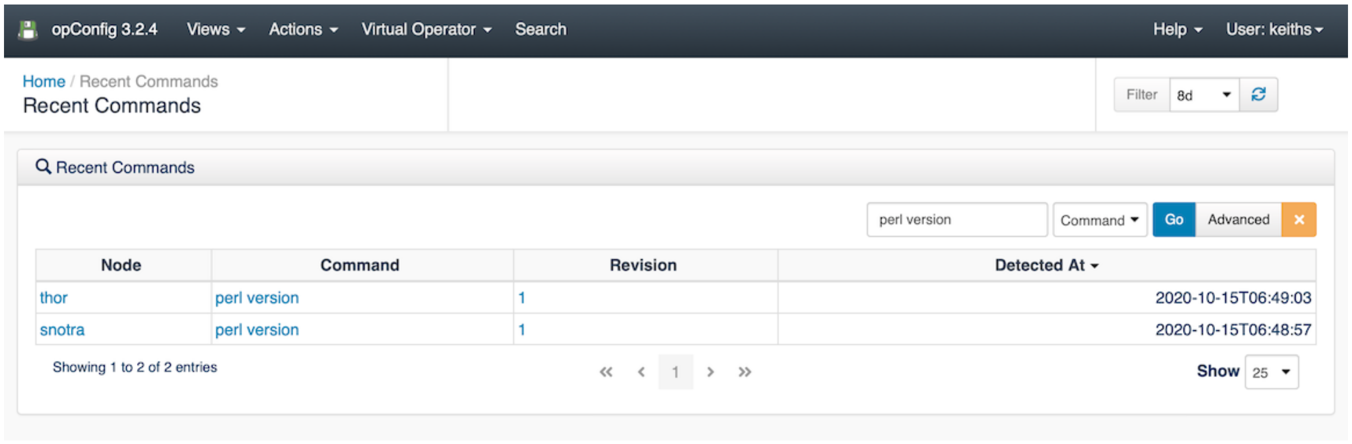
Complete for form, "perl version" should be there already, if not add it, the the Command Text you want to find and select the Node OS to limit the search and change Revisions to "Search only most recent version". Click OK to get the results.



Node	Search Nodes
Command	perl version
Command Text	ActiveState
Node OS	Linux
Revisions	Search only most recent revision

Actionable Information

From the search results, you see a list of nodes that matched "ActiveState" in the command output.



Recent Commands

Recent Commands

perl version

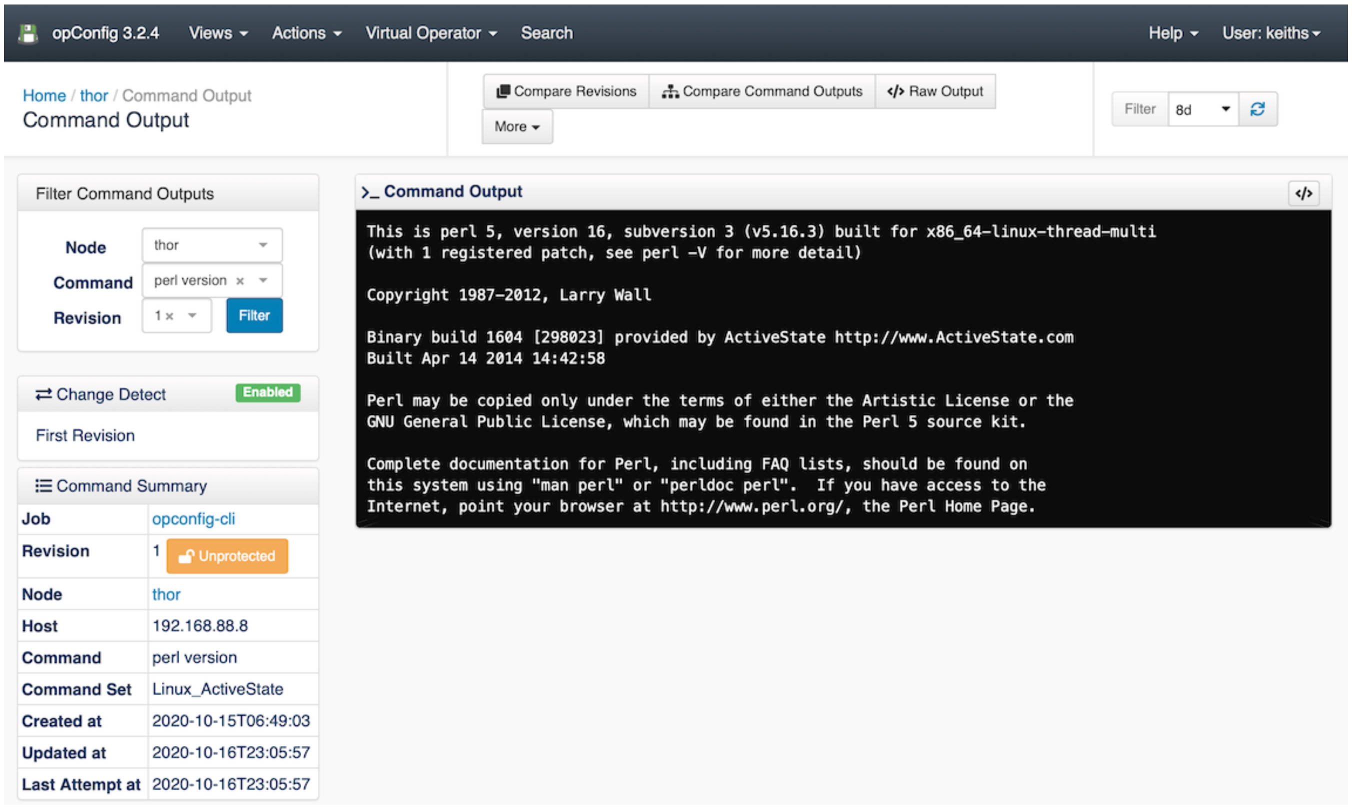
Node	Command	Revision	Detected At
thor	perl version	1	2020-10-15T06:49:03
snotra	perl version	1	2020-10-15T06:48:57

Showing 1 to 2 of 2 entries

Show 25

Quickly checked one by clicking on the command name, we can clearly see the text here:

```
Binary build 1604 [298023] provided by ActiveState http://www.ActiveState.com
Built Apr 14 2014 14:42:58
```



Command Output

Filter Command Outputs

Node	thor
Command	perl version
Revision	1

Change Detect

First Revision

Command Summary

Job	opconfig-cli
Revision	1
Node	thor
Host	192.168.88.8
Command	perl version
Command Set	Linux_ActiveState
Created at	2020-10-15T06:49:03
Updated at	2020-10-16T23:05:57
Last Attempt at	2020-10-16T23:05:57

Command Output

```
This is perl 5, version 16, subversion 3 (v5.16.3) built for x86_64-linux-thread-multi
(with 1 registered patch, see perl -V for more detail)

Copyright 1987-2012, Larry Wall

Binary build 1604 [298023] provided by ActiveState http://www.ActiveState.com
Built Apr 14 2014 14:42:58

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl".  If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.
```

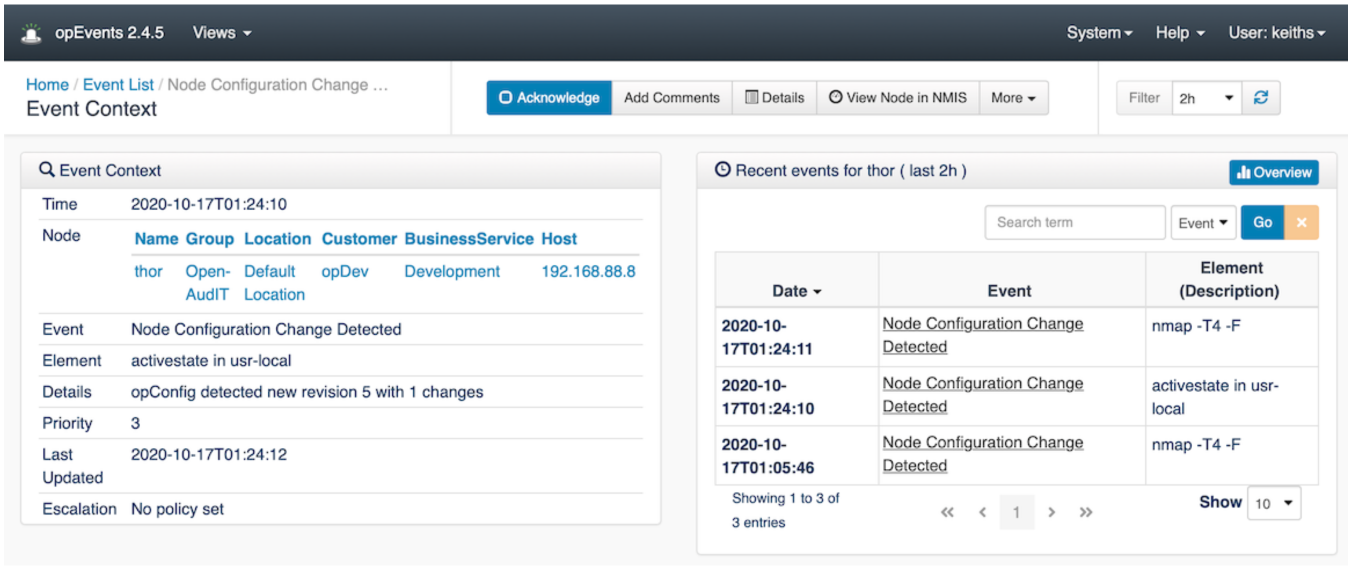
Remediation

In this case remediation requires one of the development team to install PerlBrew on each server and installed the related packages, the Opmantek development team use Vagrant to automate this kind of activity and this issue will be resolve quickly.

Change Detection and Regression

The next problem is how do I make sure no one installs ActiveState ever again and how will I be notified if they do. The second command we added earlier will provide that. The first time it runs it will detect a change, and when the developers remove ActiveState it will change, and then we should never see this event in opEvents again.

Change detection provides timely notification of unwanted software being installed. In opEvents your event policies can make this a critical event requiring urgent attention, sending notifications directly to your compliance manager.



Time	2020-10-17T01:24:10
Node	Name: thor; Group: Open-AudIT; Location: Default Location; Customer: opDev; BusinessService: Development; Host: 192.168.88.8
Event	Node Configuration Change Detected
Element	activestate in usr-local
Details	opConfig detected new revision 5 with 1 changes
Priority	3
Last Updated	2020-10-17T01:24:12
Escalation	No policy set

Date	Event	Element (Description)
2020-10-17T01:24:11	Node Configuration Change Detected	nmap -T4 -F
2020-10-17T01:24:10	Node Configuration Change Detected	activestate in usr-local
2020-10-17T01:05:46	Node Configuration Change Detected	nmap -T4 -F

Showing 1 to 3 of 3 entries

Show 10

Conclusion

Using Operational Process Automation methodology of detect, diagnose and action, Opmantek was able to identify the servers requiring the change quickly (about 15 minutes) and then complete the remediation. To ensure compliance, change detection will stay active ensuring that we will be notified if someone installs this software again.