“Unix” certification tool

October 8, 2008

Synopsis

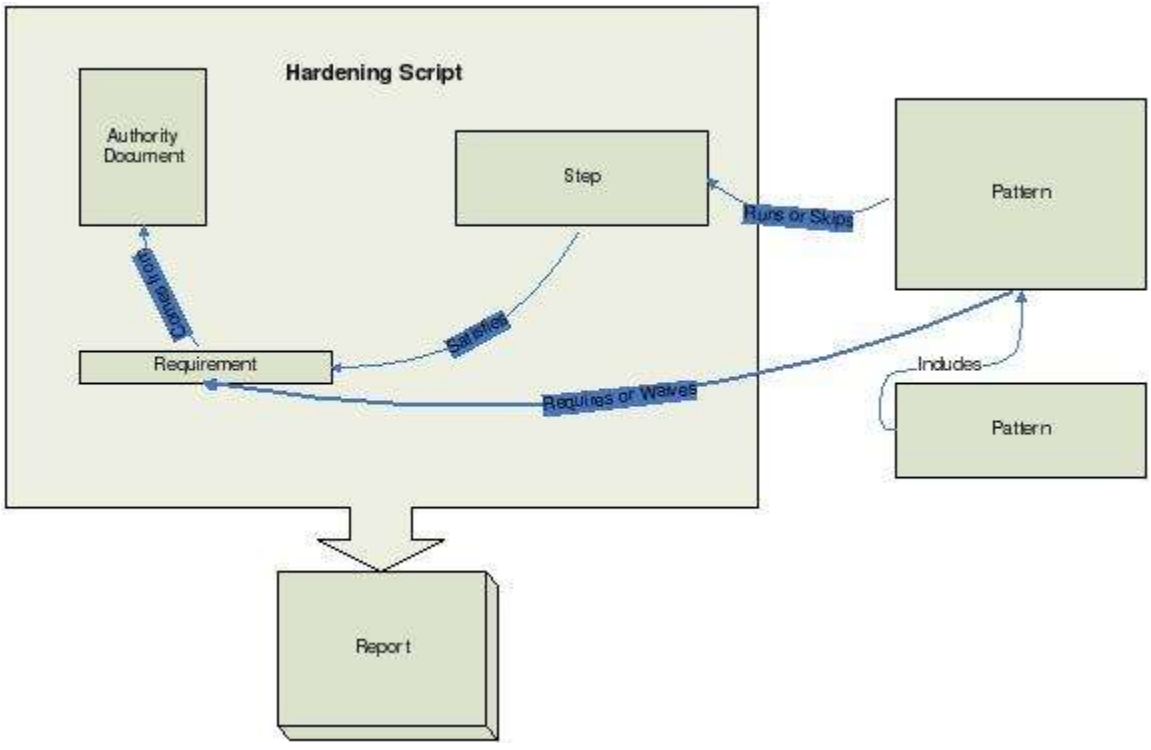
The tool reports a workstation's compliance with a named subset (such as “PublicServer”) of the requirements from various authorities, such as HHS, CIS, and LHC.

```
# lhc_harden/package/scripts/check PublicServer

# lhc_harden/package/scripts/check /path/to/clinftpp.criteria
```

Solution Architecture

The solution involves **Authorities**, **Requirements**, **Steps** (scriptlets), **Patterns**, and a master **script** that makes use of all-of-the-above.



Example 1: The “Public Server” *pattern* specifies 75 *requirements* from the HHS, CIS, and LHC *authority documents*. To satisfy those requirements, the hardening script will run 64 *steps* (scriptlets).

Example 2: The “Internal Workstation” pattern is based on the “Public Server” pattern, but waives certain requirements and, furthermore, skips certain steps.

Authority documents, requirements, and steps (scriptlets) are “built in” to the hardening solution. They are modular, so a programmer can add or modify authorities, requirements, and steps, without upsetting the apple cart.

Patterns are matters of configuration. Patterns may be added or adjusted by the people who are competent to execute the script and evaluate its results.

Steps are small, independent shell (or Perl) scripts. These are expected to be “obvious” – to have low algorithmic complexity. They could be maintained and adjusted by system administrators familiar with shell scripting.

The “master” script that evaluates the patterns, chooses the necessary steps to run or skip to satisfy the requirements, executes the steps, and prepares the report will be in Perl.

Customization

Authorities and Requirements

Organize requirements by their source (the “authority”), distill each requirement down to an identifying short label or number and one line of text, and list the requirements in a file per authority.

The authority files live in lib/resources/Authority. The current files are HHS, CIS, and LHC.

Here are fragments of the HHS and LHC authority files, as samples:

```
title=HHS Minimum Security Configuration Standards for Departmental Operating
   Systems and Applications 8/28/2006
AC-2,1   No "." (current working directory) or group/world writable files
   exist in root's $PATH.
AC-2,2   Install TCP Wrappers.
AC-2,3   Remove user .netrc files.
AC-2,4   Set "mesg n" as default for all users.
```

```
title=LHC Unix System Hardening Guidelines
by-hand!1.1 Disable all services that are not absolutely necessary.
by-hand!1.2 Create non-root users to perform non-root tasks.
by-hand!1.3 Create separate user accounts for each person required to access
   the machine.
1.4 Do not allow "root" to login remotely.
1.5 Do not allow service accounts to log in.
```

Scriptlets

The scriptlets are where the rubber meets the road. The scriptlets contain the shell commands or Perl operations that attack the requirements.

Each scriptlet begins with a header that identifies the requirements it probes.

```
# - - -
# authority: CIS/1.2
# description: Checks "Protocol" in ssh config
# - - -
```

A scriptlet reports a failure by writing to standard output:

```
    diff -b $ssh_config_real $ssh_config_scratch || :
```

A scriptlet may label failures by emitting notes flagged with "))":

```
    echo "))= Deviant (<) and expected (>) contents of ${ssh_config_real}:"
    diff -b $ssh_config_real $ssh_config_scratch || :
    echo "))  A modified file for your review is at ${ssh_config_scratch}."
```

The report suppresses labels that don't flank failures.

The scriptlets live in scripts/scriptlets.

Here is a fragment of a scriptlet as an example:

```
#!/bin/bash
#
# - - -
# authority: CIS/1.2
# description: Checks "Protocol" in ssh config
# - - -
set -e

. `dirname $0`/../../lib/functions

check()
{
    # Anything written to stdout or stderr indicates a compliance problem.

    ssh_config_real=/etc/ssh/ssh_config
    ssh_config_scratch=`scratchfile $ssh_config_real`

    # Copy ssh_config_real to ssh_config_scratch, tweaking on the way:

    awk '($1=="Protocol") { print "Protocol 2"; next };
         { print }' $ssh_config_real >$ssh_config_scratch

    if [ "`egrep -l ^Protocol $ssh_config_scratch`" == "" ]; then
        echo 'Protocol 2' >>$ssh_config_scratch
    fi

    # If we had to make changes, note it as noncompliance:
    echo "))= Deviant (<) and expected (>) contents of ${ssh_config_real}:"
    diff -b $ssh_config_real $ssh_config_scratch || :
    echo "))  A modified file for your review is at ${ssh_config_scratch}."

    # Permissions:
    echo "))= Deviant permissions:"
    find $ssh_config_real -maxdepth 0 \
        \( \! -perm 644 -o \! -user root -o \! -group root \) -ls
}

# ...
```

Patterns

A pattern indicates a subset of requirements, e.g., those that apply to public servers or to internal workstations.

Patterns can be expressed as lists of requirements, or adjustments to other patterns, or both. For example,

Patterns live in lib/resources/Pattern.

Here is the PublicServer pattern as an example:

```
option uid_threshold=200

waive CIS/8.1

skip services-off-portmap

require HHS/*
require CIS/*
require LHC/*
```

The InternalWorkstation pattern is an adjustment (a relaxation) of the PublicServer pattern. InternalWorkstation establishes certain waivers, then delegates further definition to the PublicServer pattern:

```
waive CIS5/4.4
waive HHS/CM-7,5
waive HHS/CM-7,6
waive LHC/1.25

skip services-off-smb

include PublicServer
```

Scriptlets, Requirements, and Patterns

A scriptlet satisfies all or part of at least one requirement:

- A scriptlet may name several requirements. Usually, they are equivalent recommendations from different authorities.
- Several scriptlets may name the same requirement. Usually, each scriptlet probes one part of a multi-part requirement. (Taken together, the scriptlets must probe the entire requirement.) Dividing a compound or complex requirement into multiple scriptlets allows patterns to selectively skip certain parts.

In short, there is a many-to-many relation between scriptlets and requirements.

If a scriptlet reports a failure, then all of the requirements with which the scriptlet is associated are deemed to be unmet.

A scriptlet can be parameterized to check on various similar requirements. For example,

the services-off scriptlet has 74 variants, one per service. This is simply a shortcut equivalent to copying the same script into 74 different files and varying each one of them slightly.

Synthesis

The user provides two inputs: the computer (!) and the name of a pattern file.

The script reads all of the scriptlets, making a mental note of the requirements that are satisfied by each scriptlet.

Then the script reads the pattern file. When the pattern file names a certain requirement, the script immediately knows which scriptlets it must run. When the pattern file names "all requirements from an authority" as in "require HHS/*", the script reads the authority file and gets a list of those requirements, then derives the list of scriptlets that must run.

The script runs the scriptlets.

The script finally assembles a report, organized by authority and requirement, that includes the output of all the scriptlets that indicated an error.