Addressing Routine Security Tasks with COPS

The topic of security of UNIX systems has risen considerable interest not only since the announcement of a C2 validated CONVEX-OS. But all too often this discusson just added quite a few new entries to a system administrator's list of things 'never to be done'. To aid the plagued system manager, Dan Farmer from Purdue University wrote and collected several shell scripts and C programs, each of which tackles a different security problem. He distributes them in a package called COPS ('Computer Oracle and Password System'), which is available on the US internet. Furthermore he added Kuang, a rule-based system written by Bob Baldwin, which tries to simulate the strategies of a possible intruder.

1 Installation

COPS comes in the form of five shell scripts, which contain all files inline. They create a subdirectory ./cops for all executables and data files, and the subdirectories ./cops/src with the C sources, ./cops/docs with a lot of documentation (man pages, articles and other texts) and ./cops/extensions with some unrelated informations, mainly on future plans. For compiling the C sources and formatting the manual pages a Makefile is provided.

2 Running COPS

The whole package is started from the shell-script cops. It first changes the access mode of the cops directory to 700 to prevent curious eyes from reading the report it is going to produce. After that it starts all the COPS programs - except for suid.chk, see below -, and the kuang system, collects all their output in a report file and, optionally, mails it to selected users. The produced report contains a concise list of all found security problems. Further information about the warnings and how to fix the problems can be found in cops/docs/warnings.

3 The COPS programs in detail

root.chk: checks path and umask of root, root entry in ftpusers, non-root entries in /.rhosts, +-entry in hosts.equiv .

dev.chk: checks files that should not be world readable, especially in /dev; looks for unrestricted NFS file systems in /etc/exports.

dir.chk: checks, whether the directories listed in dir.chklst are world-writable. The supplied list contains most of the critical directories like /, /etc, /usr, /usr/bin etc.

file.chk: checks, whether the files listed in file.chklst are world-writable. The list can contain wildcards.

rc.chk: scans /etc/rc*-Files for pathnames and files, checks , if they are world writable.

cron.chk: same as 5. for crontab-files

group.chk: checks /etc/group file for inconsistencies and syntax .

home.chk checks whether the home directories are world writable.

passwd.chk: checks passwd file for syntax and inconsistencies.

pass.chk: tries to guess passwords; lots of possible extra options for set of guesses (username, GECOS information, dictionary, all upper case, lower, backwards ...).

user.chk: checks, whether 'critical' files in home directories (.cshrc, .login, ..) are world writable.

suid.chk: reports changes in the list of setuid or setgid files it produced in the previous run. Since it is CPU consumptive ('find' over the whole file system), it is not started from the cops shell.

4 The Kuang system

This collection of shell scripts and C programs tries to simulate a hacker attack. Given a starting point (e.g. "logged in as user foo") and a goal ("try to become root") it uses a set of rules like

> write access to /etc/passwd → become root
> write access to bar/.cshrc → become bar
> (being bozo) & (bozo is in group weirdos) → become member of group weirdos

to try to achieve the goal. It does this by applying the rules in a backward manner (like: if I could write to passwd, I could become root. Let's look at passwd ...), thereby spanning a growing tree of possible attacking points, which stays manageable (more or less), as some branches 'die out'. The following example of a security hole it found at the MIT gives an impression of its strategy:

> start: member of the group MIT (standard group)
> → write access to tom/.cshrc
> → become member of group a-staff
> → write access to dick/.login
> → become member of group staff
> → write access to /etc
> → replace /etc/passwd
> → become root (goal)

(The example is cited from Bob Baldwin's article on kuang, which is included in the cops package.) Though the rules used here are rather simple, the complicated path to the goal made it almost impossible to find the hole - except for the 'bad guy', who surely uses his own kuang! A week point of kuang is that it does not scan through critical files to find further objects to attack, e.g. programs started from /.crontab.

5 Remarks

1. In a first run on our system - without any adaptation to the CONVEX environment - cops reported 36 problems, among them two users with open .cshrc files, 19 guessed passwords and some world writable files.

2. The places, where to include system specific information, are scattered around several places: special files, variables in shell scripts, in the C code. This makes adaptation a tedious task (notice 1., however).

3. The sources show all signs of their history: Bad translations into C (with gotos and continues everywhere), insertions to cope with yet another system - but not with ours, similar tasks coded differently in different programs.

6 Conclusion

Cops is a useful tool for security management - even without adaptations to a CONVEX environment. It will find at least some of the more obvious holes in a system and most likely make the system manager aware of security topics. Making some modifications to file tables (e.g. including /usr/convex) and minor changes to the sources will further increase its usefulness.

One of the 'readme' files of cops lists things to be done. This is obviously not meant as an announcement, but as an invitation. So come up with your solution to a security problem, make it an article in this news letter or mail it to Dan Farmer so that it can be included in a future release of cops!

If you have problems to get the sources: We plan to include a slightly adapted version to the user tape of the german Convex user group.

Peter Junglas
Rechenzentrum der TU Hamburg-Harburg
Eissendorfer Str.38
2100 Hamburg 90

junglas@tu-harburg.dbp.de
junglas@tuhhco.rz.tu-harburg.de
junglas@tuhhco.uucp