Attachment 3 COPS

The package, which will henceforth be referred to as COPS (Computer Oracle and Password System), can be broken down into three key parts. The first is the actual set of programs that attempt to automate security checks that are often performed manually (or perhaps with self-written short shell scripts or programs) by a systems administrator. The second part is the documentation, which details how to set up, operate, and interpret the results of the programs. It also includes a paper or two on COPS itself. Third, COPS is an evolving beast, so it includes a list of possible extensions that might appear in future releases. In addition, it includes some short papers on various topics in UNIX security and pointers to other works in UNIX security that could not be included at this time, due to space or other restrictions.

This document contains four sections:

1) What is COPS?
2) What is COPS _not_?
3) Installation, Execution, and Continuing Use of COPS
4) Disclaimer and End Notes

1) What is COPS?

The heart of COPS is a collection of about a dozen (actually, a few more, but a dozen sounds so good) programs that each attempt to tackle a different problem area of UNIX security. Here is what the programs currently check, more or less (they might check more, but never less, actually):

- file, directory, and device permissions/modes.
- poor passwords.
- content, format, and security of password and group files.
- the programs and files run in /etc/rc* and cron(tab) files.
- existance of root-SUID files, their writeability, and whether or not they are shell scripts.
- a CRC check against important binaries or key files to report any changes therein.
- writability of users home directories and startup files (.profile, .cshrc, etc.)
- anonymous ftp setup.
- unrestricted tftp, decode alias in sendmail, SUID uudecode problems, hidden shells inside inetd.conf, rexd running in inetd.conf.
- miscellaneous root checks -- current directory in the search path, a "+" in /etc/host.equiv, unrestricted NFS mounts, ensuring root is in /etc/ftpusers, etc.
- dates of CERT advisories vs. key files. This checks the dates that various bugs and security holes were reported by CERT against the actual date on the file in question. A positive result doesn't always mean that a bug was found, but it is a good indication that you should look at the advisory and file for further clues. A negative result, obviously, does not mean that your software has no holes, merely that it has been modified in SOME way (perhaps merely "touch"'ed) since the advisory was sent out.

o the Kuang expert system. This takes a set of rules and tries to determine if your system can be compromised (for a more complete list of all of the checks, look at the file "release.notes" or "cops.report"; for more on Kuang, look at at "kuang.man".)

All of the programs merely warn the user of a potential problem -- COPS DOES NOT ATTEMPT TO CORRECT OR EXPLOIT ANY OF THE POTENTIAL PROBLEMS IT FINDS! COPS either mails or creates a file (user selectable) of any of the problems it finds while running on your system. Because COPS does not correct potential hazards it finds, it does _not_ have to be run by a privileged account (i.e. root or whomever.) The only security check that should be run by root to get maximum results is the SUID checker: although it can be run as an unprivileged user, it should be run as root so that it can find all the SUID files in a system. In addition, if key binaries are not world-readable, only executable, the CRC checking program ("crc.chk") needs to be run as a privileged user to read the files in question to get the result.) Also note that COPS cannot used to probe a host remotely; all the tests and checks made require a shell that is on the host being tested.

The programs that make up COPS were originally written primarily in Bourne shell (using awk, sed, grep, etc.) for (hopefully) maximum portability, with a few written in C for speed (most notably parts of the Kuang expert system and the implementation of fast user home directory searching), but the entire system should run on most BSD and System V machines with a minimum of tweaking. In addition, a perl version is included that, while perhaps not as portable as the shell/C version, has some advantages.

COPS includes various support programs as well. The primary one is CARP (COPS Analysis and Report Program). CARP is a results interpreter that is designed to analyze and generate a summary on various COPS reports from a complete network or set of hosts.

2) What is COPS _not_?

COPS mostly provides a method of checking for common procedural errors. It is not meant to be used as a replacement for common sense or user/operator/administrative alertness! Think of it as an aid, a first line of defense, not as an impenetrable shield against security woes. An experienced wrong-doer could easily circumvent *any* protection that COPS can give. However, COPS *can* aid a system in protecting its users from (their own?) ignorance, carelessness, and the occasional malcontent user.

Once again, COPS does not correct any errors found. There are several reasons for this: first and foremost, computer security is a slippery beast. What is a major breach in security at one site may be a standard policy of openness at another site. Additionally, in order to correct all problems it finds, it would have to be run as a privileged user; I'm not going to go into the myriad problems of running SUID shell scripts (see the bibliography at the end of the technical report "cops.report" for pointer to a good paper on this subject by Matt Bishop; look at the included paper "SU" for pointers on how to write a SUID program) -- suffice to say it's a bad idea that can give an attacker privileges equal to whatever account the shell is SUID to.

3) Installation, Execution, and Continuing Use of COPS

There are two versions of COPS that can be run. The original ("COPS classic"?) needs nothing more than a C compiler and the standard shell tools that any (or most any) UNIX system should have: awk, sed, grep, etc. For information on how to configure and run this version, look at

the file "README.2.sh". The most important thing to do is to run the shell program "reconfig" if you have a system V or a non-standard Berkeley UNIX system -- the paths to the programs that COPS uses are hard-coded, and this will reconfigure the paths so that COPS can find these programs.

If you have installed perl on your system (I think it works with perl versions > 3.18) and would like to try the perl version, look at the file "README.2.pl" for details on how to use that. There are several advantages and disadvantages to using the perl version, so if you have perl, I would advise trying both packages to see which one better suits your environment.

If you need help to interpret the results of COPS, look in the file "warnings", in the "doc" directory. All of the individual programs in the COPS package have a man page there as well.

For continuing use, multiple architecture sites, or other advanced COPS topics, check out "README.3".

There are additional "readme" files for the following topics: Apollo and Xenix machines, C2 and other shadow passord files, NIS/Yellow Pages, and the COPS filter. Look at the corresponding readme (note lower case) file for these in the "docs" directory -- e.g. "docs/readme.apollo."

4) Disclaimer and End Notes

COPS is meant to be a tool to aid in the tightening of security, not as a weapon to be used by an enemy to find security flaws in a system. It may be argued that allowing anyone to have access to such a tool may be dangerous, but hopefully the overall benefit for systems that use this package will outweigh any negative impact. To me it is akin to a law enforcement problem -- although telling the public how to break into a house may foster a slight rise in break-in attempts, the overall rise in public awareness of what to defend themselves against would actually result in a drop in break-ins. The crackers with black hats already know how to crush system defenses and have similar tools, I'm sure. It's time we fought back.

COPS is not the final answer to anyone's security woes. You can use the system as long as you realize that COPS has no warranty, implied or otherwise, and that any problems that you may have with it are not my or any of the other authors' fault. I will certainly attempt to help you solve them, if I am able. If you have ideas for additional programs or a better implementation of any of the programs here, I would be very interested in seeing them. COPS was the work of a LOT of people, both in writing code and in the testing phase (thanks, beta testers!). For a complete list of contributors, look at the file "XTRA_CREDIT".

So, good luck, and I hope you find COPS useful as we plunge into UNIX of the 1990's.

dan farmer
January 31, 1989
(Now January 31, 1990)
(Now November 17, 1991... how time goes on...)

```
# include "./disclaimer"
```

p.s. Just for snix, here are some of the machine/OS's I know this sucker works on; far and away the most common problem was getting that stupid password cracking program to compile, followed by systems without the -ms package to nroff. Some minor problems with config files -- I *think* these are all ok:

DECstation 2100, 3100, 5000, Ultrix 2.x, 3.x, 4.x (Ultrix is braindead.)

Sun 3's, 4's (incl. Solbourne and clones) -- 3.x, 4.x
Gould 9080 Powernode, hacked up Gould OS (whatever it is)
sequent S-87 symmetry, dynix V3.x (both att & bsd universes; att required
                    "BRAINDEADFLAGS = -lcrypt" to be uncommented.
ETA-10P, Sys V R3 based
Convex boxes, all types, OS's (up to 9.x, the most recent)
Apollo dn3000 & dsp90, Domain SR 9.7, 10.x (see "readme.apollo")
Vax 11/780, 4.x BSD (Mt. Xinu, tahoe and stock)
Vaxstation, MicroVax, Vax 6320 & 8800, Ultrix 2.x, 3.x, 4.x
HP900/370, HP-UX 6.x, 7.x
Cray 2 & Y-MP, UNICOS 5.x, 6.x
Amdahl 5880, UTS 580-1.2.3
SGI 2500's, IRIX GL 3.6
SGI 4D's, IRIX System V Release 3.x
'286 & '386 Boxes, running Xenix (see "readme.xenix")
AT&T 3B2 & 3B1, SysVR[3-4]
CADMUS box (R3000 & 68020 cpu), SysVR3.2
Pyramid, running 4.4c and 5.1a

Apple Mac IIci, running AUX 2.x. The "test -z" seemed broken on this,
but I only had a brief chance to test it out, but kuang didn't like it
as a result. I'll get a working version soon; everything seemed ok
(change the /etc/servers line in "misc.chk").

NeXT, 1.x
(password stuff is different on this machine, though; cracking is
strange. Diffs anyone? Also, /bin/test vs. shell builtin "test" is
weird.)

Multimax 320, 12 Processors, 64Mb Memory, Encore Mach Version B1.0c (Beta)
(no crypt(3) on this machine. Sigh.)

IBM rs6000, AIX 3.1 (DEADBEEF about sums it up.)

  I've lost track of the others. If you have some bizzare piece of
hardware that you've run it on, I'd like to hear about it...