**SANS Institute**
Information Security Reading Room

# IMPLEMENTING sudo TO REPLACE su

Robert Agnolo

# IMPLEMENTING sudo TO REPLACE su : (A Case Study Also Involving NIS and RDIST)

## INTRODUCTION:

Greetings, reader! I am about to relay to you the occurrence of actual events that transpired at my place of work sometime between September and October of last year (2002). It is essentially the beginning of a still on-going saga about a group of customers who are slowly being introduced to the concepts of network/system security in an environment where very little had existed before. Namely though, it involves the implementation of sudo to replace su access on two key Sun/Solaris servers used by a small group of scientists who do research and development for a major US manufacturer. Please note that while this story is true and accurate to the best of my recollection, the names of every participant involved, including the companies, employees, vendors, customers, administrators, computers, and even myself, have been changed to protect our confidentiality as well as to thoroughly "sanitize" the data contained herein.

Incidentally, my name is Trevor, and I am a Systems Administrator employed by an IT outsourcing firm called "System Operators At Large" (also known as SOAL). My current assignment is working on SOAL's "TEAM "project which stands for Technically Enhanced Administration & Management. TEAM is an enhanced service offering that SOAL provides above and beyond its usual plain-vanilla support packages. Specifically, TEAM is geared toward the support of highly technically users (usually engineers and scientists) in their hi-tech environments (usually laboratories and research facilities). In this service offering scenario, the customer pays a premium price and receives premium IT support in return. In short, whatever IT needs our customers require in order to complete their research and/or produce their prototypes, TEAM is contractually obligated to provide. It is important to note that, unlike typical IT support models, TEAM is based upon "best efforts" as opposed to "response times". Thus, TEAM's Systems Administrators are not penalized for failing to furnish pre-defined deliverables within agreed-upon timeframes. Rather, we are encouraged to provide the all-around best possible service to our customers in the most timely manner that is humanly possible. In short, our customers are used to getting things THEIR way. It is within this framework that sudo was to be introduced to a customer requiring (and in some cases even demanding) su access on key servers to do their work, and also accustomed to using it at will.

## BACKGROUND:

Some of the obvious questions you, the reader, may have are: Why did they need su access on their servers anyway? And did they truly need it? How in the world did they ever get it in the first place? How does a sophisticated group of corporate users exist with such a lax security stance? And finally, how does "the new guy" raise the consciousness level of security among those who've gone with so little of it for so long. I think the best way to answer these questions is to

provide a brief historical retrospective on my customer, The XYZ Corporation's "POISE" group.

BEFORE SNAPSHOT:

One of SOAL's prize customers is The XYZ Corporation. XYZ is a leading manufacturer of digital cameras & photographic printers. Most XYZ users subscribe to SOAL's plain vanilla IT support package, which is fine for its sales staff and clerical personnel. However, some XYZ users require a much more comprehensive support model. One such group is POISE, an acronym for Printing Operations & Imaging Science Engineers. The POISE group exists as one of several small but important R&D groups within the XYZ corporate structure. POISE is a close-knit group of 30 imaging scientists that once existed as a research firm-for-hire, until it was acquired by XYZ sometime in the late 90's. POISE specializes in developing high-quality image rendering software & firmware as it relates to the high-end printer and scanner market for the printing & publishing industry.

Until its acquisition by XYZ, POISE's IT support was provided by one of their in-house programmers named Steve Adams. Steve is still a major code-writer for the group and continues to play a key role in all of POISE's hardware and software procurement decisions. (Side Note: It was Steve who originally ordered POISE's Sun/Solaris servers and named them after characters from George Orwell's "Animal Farm".) When, Steve Adams stepped aside as SysOp, my immediate predecessor, Fab Cuttia took over as Sys Admin from early 2000 to mid 2002. Fab continued to run POISE's IT department similarly to how Steve had, with much user autonomy, minimal security, but with outstanding documentation! Fab still works at XYZ in a facility nearby, and is usually only a phone call away. Being so familiar with the POISE group's systems, Steve and Fab have been excellent mentors to me. And it is a rare opportunity when a newly assigned Systems Administrator gets to speak with the previous IT support staff. But let us get back to the POISE group.

As a historically semi-autonomous user community, conducting research on its own, mostly in isolated labs, and separate from the main production lines, the POISE group has enjoyed an usually high level of freedom. And among these freedoms are, not only Administrator privileges on their personal Windows/Intel workstations, but also su access on their Sun/Solaris servers! This is rather ironic, given that most XYZ users must observe very strict adherence to that corporation's security policies. Later, I will show you actual excerpts from those policies.

Furthermore, because of the potential danger associated with su access (more on this later), many companies, both large and small, strictly limit its use on ALL production servers. And on those rare occasions, when su access is allowed, its use is strictly limited to that of certain authorized/qualified members of the IT support staff, and then only at the systems' consoles! So, to say that granting su access to regular users is contrary to the security policies of most major US

corporations (not to mention the conventional wisdom among many security professionals) is indeed an understatement. It was for this reasons above all, that made me realize that the lunatics had been running the asylum long enough. It was high time that the POISE group joined the rest of the security-conscience world. Just like TEAM, SOAL, and the rest of The XYZ Corporation before them.

DEFINING su VS sudo - DIFFERENCES & SIMILARITIES:

For those of you who may not be familiar with either the su or sudo utilities, lets take time to briefly describe each. For each command (or capability) I will give a short, dictionary-type definition from a reputable source on the Internet. Then I will follow up with brief summaries in plain English. It is my hope that the combination of these definitions will give you, the reader, a thoroughly fleshed-out understanding of su & sudo, plus how they fit together in the big puzzle of UNIX Systems Administration.

The introductory terms section of the Red Hat Docs web site defines su like so: "The command su gives you access to the root account or other accounts on your system. When you type su to switch to your root account while still inside your user account shell, you have access to important system files that you can change, or damage, permanently. Logging in with the su – command makes you root within the root account shell. Use caution when you are logged in as root." (Referenced Source: "The Official Red Hat Linux 8.0 Getting Started Guide" found at http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/getting-started-guide/s1-starting-intro-terms.html.)

So su traditionally stands for "switch user" but more literally means "SuperUser", which in the UNIX world is tantamount to God. Basically, the SuperUser enjoys complete administrative control over the UNIX machine. (It may be helpful to MS-Windows users to think of su as the analog to the local machine Administrator on NT, 2K, or XP systems.) But along with this amazing control comes the awesome responsibility to wield it with great care. For, along with the capability to fix most UNIX problems, the SuperUser has the equivalent ability to break the system beyond repair. Simply entering the wrong command by a single character can have disastrous results! For example, typing the "rm <dir>" command when you meant "rm **–i** <dir>" is the difference between purposely deleting unwanted files when prompted –or– accidentally blowing away an entire directory you may never get back. (Actually, it was a similar case of a user who typed the wrong command as su that is one of the main scenarios on which this paper is based…But more on that later). Again, the importance of being vigilant while using su cannot be overstated.

Another major problem with su usage is its infamous lack of accountability. Meaning, a Systems Administrator can login directly to a UNIX system's console without having to provide his/her own username and/or password. I'm sure you can see the problem this potentially creates. For, if that same Systems Administrator mistakenly types the wrong command, thereby wreaking havoc on the system, there is no reliable audit trail to tell us who did what. Sure, there's

the sulog (located in a Solaris system's /var/adm/ directory). But sulog can only tell us that SOMEBODY logged into the system as root at a particular date & time, but that's all it tells you. Clearly, such a powerful tool as su needs to be tracked much more closely than that! So, wouldn't it be nice to have a utility that gives Administrators the power of su, plus auditing for greater accountability? Wouldn't it be helpful to track whenever the root account is misused, and by whom? How about something that already utilizes syslog? Enter sudo!

DEFINING su VS sudo - DIFFERENCES & SIMILARITIES (continued):

The "What You Need To Know About " web page has a "Focus on Unix" section that defines sudo like this: "sudo (Super User DO), a public domain program, provides a flexible solution to giving partial root privileges. It allows selective root access to be given by user, machine, or command, and keeps copious logs every time these privileges are used." (Referenced Source: "Focus On Unix: Managing Root Access With sudo" at http://unix.about.com/library/weekly/aa102500a.htm)

I like to think of sudo as a kinder, gentler version of su. sudo essentially gives Sys Admins (and sometimes regular users) the ability to wield the power of root in a far more prudent manner (one command at a time) and for limited time periods (5-minute sessions by default). This interposes significantly less risk associate with unattended consoles. While you would never want anyone walking away from a console without locking it; if they did forgot, the odds of them leaving the system completely wide-open and vulnerable are dramatically reduced.

Unlike su, which merely logs its successful and failed uses in the sulog file (mentioned earlier), sudo provides a highly-configurable auditing process that records its information into UNIX's built-in System Log, called the syslog file. (Note: On Sun/Solaris systems, the syslog file is located in the /var/adm/ directory). By default, sudo only logs failed attempts of its use, but it can be modified to track a great deal more. For starters, it has the capability to record where it was invoked (machine), when it was invoked (date & time), and who invoked it (user). How's that for accountability? With sudo, Systems Administrators can be reasonably certain that su access is NOT being granted to just anybody.

DURING SNAPSHOT:

This is where the trouble began: One Tuesday morning in September of 2002, I returned to work from a three-day weekend. As usual, I began by checking the /var/adm/messages logs on POISE's two main Solaris servers, Napoleon and Snowball. (Aside: Windows Server Administrators might find it handy to think of the UNIX messages file as the functional equivalent of the Microsoft Event Viewer logs found on NT, 2K, and XP machines.) Their purpose also being the recording of events that occur on the machine such as system messages, application errors, or security alerts. But on this day an "ls –la /var/adm/" command run on each box, revealed that both of their messages files were 0

bites large. And viewing the files via "more /var/adm/messages" confirmed they were indeed empty. No system, app, nor security messages of any kind had been logged since Friday of the previous week. Furthermore, "ps –elf | grep sys" commands revealed that syslogd, (The UNIX syslog daemon: the process responsible for recording messages into the messages file), was not running. Somebody, intentionally –or– unintentionally must have turned off the syslog facility. While this is a difficult enough event to trace, it was exasperated by the fact that all POISE users had su access to Napoleon and Snowball. In short, all 30 of my users were suspects, and my only clue was a one-line entry in sulog. All I knew was that SOMEBODY accessed su SOMETIME the previous Friday…that's all. Still, some kind of investigation was necessary.

THE INVESTIGATION:

I began by phoning my friend, colleague, predecessor, and backup, the previous Sys Admin for the POISE group, Fabrizio Cuttia. After a brief conversation with Fab, it was clear that he was not the culprit, nor did he know who was any more than I did. However, he was just as concerned as I was that this was a very serious problem. To paraphrase his thoughts: Turning off syslog is like disabling a bank's alarm. Whether or not it's accidental, it's always bad news. So, great pains must be taken to avoid it from ever happening.

Due to several reasons, I already knew it would be nearly impossible to track down the guilty party. Sure, I asked everyone who was working Friday if they had "su-ed" that day. But most couldn't honestly remember whether they had or not. And those that could recall couldn't really know if it was one of their commands that stopped syslogd. As stated earlier, with su, it takes only one mistyped command to cause a problem. My theory is that someone tried to halt a job with the pkill command by grep-ing for a process name containing the letters "sys", or "log". Anyway, the horse was already out of the barn. I decided to move on to the future prevention stage of the game. So, without further ado, let's walk through the nuts & bolts.

INSTALLING & CONFIGURING sudo ON SERVER #1 (NAPOLEON):

Obviously, I will need to go into detail as to how sudo was implemented from start to finish. But the entire installation and configuration process can be summarized as follows: "[1] Download the source code. [2] Prepare the source code for compilation. [3] Compile the source code and install. [4] Modify the search path. [5] Configure sudo. [6] Use sudo." (Referenced Source: "Solaris Resources at Kempston: Installing and Configuring sudo on Solaris 7 and Solaris 8", quoted from http://www.kempston.net/solaris/sudo.html). This delineates a generic method of installing sudo locally on a single UNIX box. Normally, that would be fine, but I needed something that would allow me to install sudo on one server (Napoleon), distribute it to another (Snowball), and allow all POISE users to run it from any directory on any Sun/Solaris box in our domain. We needed a solution that would not demand us to run the command's full path (/usr/local/bin/sudo) on specific machines (Napoleon or Snowball). To be sure, our primary focus was giving the Exceed users the freedom to run sudo on our

main servers (Napoleon & Snowball). But we also wanted our users to have the option of running it from older, low-end Sparc workstations (Boxer and Clover). And, of course, we wanted all this without having to install it on every Solaris box in the POISE domain. So it was decided to employ the RDIST facility, and utilize our NIS auto-mount maps to make it all come together. But first things first. Let's verify some of our prerequisites.

INSTALLATION REQUIREMENTS: It's important to note up-front, that before I began, I had to confirm that I had the following prerequisites available to me on the Napoleon server's console: root access, gzip & gzcat, a working C compiler (like gcc), and a functional make utility. Please Note: This is where the #which gzip and #which make commands came in handy. Once these confirmations were made; I was ready to get down to the nitty-gritty of downloading and installing.

sudo INSTALLATION DETAILS:

*(1) Download the source code:*

The latest version of sudo (1.6.6 at the time of this writing) can be obtained free via http or ftp at several sites on the Internet. However, it is always wise to obtain software from sites that are deemed both well-know and reputable among your security industry peers. This in itself, may not be an absolute guarantee that the software is completely virus-free, but it is one of you safest bets. Two such reputable and well-known sources of sudo for Sun and other popular platforms are ftp://ftp.sudo.ws/pub/sudo and http://www.sunfreeware.com. From either of these sites you would download the sudo source code in the form of a tape archive (or tar) file that has been compressed using GNU's zip (or gzip) utility. Thus, the file would be named "sudo-1.6.6.tar.gz". It is this file that I downloaded to the Napoleon server.

*(Extra NIS Step) Create directory & auto-mount map:*

All users in the POISE domain have access to a common auto-mount called "/software". It is here where many of POISE's commonly used programs are located. First, to create a directory to house the sudo program, I ran the command, *#mkdir /u01/software/sudo* on Napoleon as root. Then I logged into our NIS master (Squealer) to modify and push-out (as root) the NIS map named "auto.software".

```
[ROOT on SQUEALER] # vi /nis/maps/auto.software
(add entry: napoleon,snowball:/u01/software/sudo)
[ROOT on SQUEALER] # cd /var/yp
[ROOT on SQUEALER] # make
```

*(2) Prepare the source code for compilation:*

Then, as root on Napoleon, I copied the gzipped tar file to the /tmp directory via the *# cp sudo-1.6.6.tar.gz /tmp* command. Furthermore, I "sourced" my *.cshrc* & *.login* files, so that even as root, I would have my normal user environment variables. The following two commands made the path to GNU's C Compiler (gcc) visible to root:

[ROOT on NAPOLEON] # source ~tdaniels/.cshrc
[ROOT on NAPOELON] # source ~tdaniels/.login

Next, I used the GNU Zip *gzcat* command piped to the *tar* command. I did this in order to uncompress the "/tmp/sudo-1.6.6.tar.gz" archive while extracting it to the /tmp/sudo-1.6.6 directory at the same time:

[ROOT on SQUEALER] # cd /tmp
[ROOT on SQUEALER] # gzcat sudo-1.6.6.tar.gz | tar xvf –
[ROOT on SQUEALER] # cd sudo-1.6.6

sudo INSTALLATION DETAILS (cont.):

*(3) Compile the source code and install:*

The "configure" utility provides a daunting number of sudo program options that can be run at installation time. There is even one that berates you with "Hal 2001-like insults when an incorrect password is entered". (Source: "Sudo Installation Notes" from http://www.courtesan.com/sudo/install.html.) However, we're most concerned with the *--prefix=* and *--sysconfigdir=* arguments, which allow us to change the default locations of the installation and configuration directories respectively.

[ROOT on NAPOLEON] # pwd (to confirm that we are still in /tmp/sudo-1.6.6)
[ROOT on NAPOLEON] # **./configure --prefix=/software/sudo --sysconfigdir=/software/sudo**
[ROOT on NAPOLEON] # make
[ROOT on NAPOLEON] # make install

*(4) Modify the search path:*

I actually skipped this step because I had already addressed the path issue by creating an auto-mount directory (napoleon:/u01/software/sudo), then modifying and pushing the appropriate NIS map (squealer:/nis/map/auto.software). At this point, the *#sudo –V* command was entered to confirm that this was indeed the case. Note: "The -V (version) option causes sudo to print the version number and exit." (Referenced Source: #man sudo "The sudo man pages" by Todd Miller, et. al.)

*(5) Configure sudo:*

The /etc/sudoers file (or the /software/sudo/etc/sudoers file in our particular case) is the heart of the sudo configuration. Nearly all of its access controls and options are controlled via modifying this file. So it shouldn't surprise anyone that you must be root to edit it. Please note that editing this file must be accomplished using a vi-like editor called visudo. (That's /software/sudo/sbin/visudo in our case.)

Below is a glimpse of our /software/sudo/etc/sudoers file. Please note that it is not much larger or terribly different from the original "sample sudoers file" that comes with the program by default. Observe how ordinary users, like Jacques Moiré and Steve Adams are only permitted to run sudo from the four servers specified in the SERVERS1 group. Moreover, these regular users are prohibited from running any "shells" or so-called "danger" commands via sudo. While Systems Administrators like Fabrizio Cuttia and me (Trevor Daniels) are allowed to run ALL commands via sudo from ALL machines in the domain. Of course there are scores more options we can chose at a later date, but this is enough for now. The two important points I want to stress are: (a) The default options pre-set in the sample sudoers file allows us to get sudo up & running very quickly. (b) The power and flexibility of sudo is made possible by changing this incredibly small and easily editable sudoers file.

```
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.

Host_Alias     SERVERS1=napoleon, snowball, boxer, clover

Cmnd_Alias     SHELLS=/bin/sh,/bin/csh,/bin/ksh,/usr/bin/ksh,\
               /usr/bin/csh,/usr/bin/sh,/sbin/sh,/bin/tcsh,/usr/bin/tcsh

Cmnd_Alias     DANGER=/bin/su,/usr/bin/su,/bin/rsh,/usr/bin/rsh,\
               /bin/login,/usr/bin/login,/usr/sbin/snoop,/software/bin/visudo

jmoire         SERVERS1=ALL,!SHELLS,!DANGER
sadams         SERVERS1=ALL,!SHELLS,!DANGER
tdaniels       ALL=(ALL) ALL
fcuttia        ALL=(ALL) ALL
```

*(6) Use sudo:*

Two more factors in deciding to go with sudo were its cost (free BSD License), and its ease of use. XYZ is experiencing financial difficulties, so procurement of a costly software solution, however necessary, would have been impossible at this time. Plus, requests for time & money to send users to training would have

been flatly denied. But sudo is so simple to use that this brief e-mail memo was all that was needed to educate our customers on it uses. In these cost-conscience times, it's no wonder why great "open source software solutions" like sudo are gaining popularity.

From: Daniels, Trevor
Sent: Monday, October xx, 2002 7:59 AM
To: "USA POISE GROUP"
Subject: IMPORTANT: sudo REPLACING su

To improve security, POISE is implementing sudo to replace the functionality of su.
The sudo command will give you the same root access su does, but with less risk.

sudo is very easy to use & works on a command-by-command basis (see below).
NOTE: It requires you to enter your password before it runs your command.

```
% sudo <command> [options | arguments]
password: <my_unix_account_password>
```

This will greatly reduce the risk of dangerous typos related with su-ing to root.
Please let me know if you have any difficulties. Thank you for your cooperation.

INSTALLING sudo VIA rdist ON SERVER #2 (SNOWBALL):

For purposes of system redundancy and network load-balancing, POISE group users equally utilize both the Napoleon and Snowball servers. Therefore, since we've already installed sudo on Napoleon, we must have an identical installation of it on Snowball. But how do we accomplish this without performing the exact same keystroke-per-keystroke installation? ANSWER: rdist! "RDIST is an open source program to maintain identical copies of files over multiple hosts. It preserves the owner, group, mode, and mtime of files if possible and can update programs that are executing. Almost all versions of UNIX include rdist." (Referenced Source: "Introduction to RDIST Homepage" at http://www.magnicomp.com/rdist.)

Basically, our entire rdist mechanism works on the basis of three files: the executable rdist program (/usr/bin/rdist), the rdist configuration file (/etc/export/admin/rdist.conf), and a root cronjob that ties them all together (/var/spool/cron/crontabs/root). Please see the sample file fragments below:

***# napoleon:/etc/export/admin/rdist.conf:***

```
software:  /u01/software -> snowball
   install -R /u01/software/;
```

```
##########################
```

```
#     /var/spool/cron/crontabs/root     #
##############################
#ident "@(#)root       1.14    97/03/31 SMI"   /* SVr4.0 1.1.3.1       */
#
20 20 * * 6 /usr/bin/rdist -f /export/admin/rdist.conf | fgrep -v -e "no password" | fgrep -v -e "no name"
```

AFTER SNAPSHOT:

In terms of this case study, it would not be completely unfair to say that sudo was sprung upon the POISE group, that is to say; "implementation after the fact". But this was decided for several reasons. [1] Something had to be done. The continued use of su couldn't go on. It was contrary to both XYZ's security policies and common sense. Nowhere in my experience were regular users granted Systems Administrator access to key servers. [2] Although I didn't expect great resistance; I nevertheless prepared for it. There was no doubt in my mind that if push came to shove, and the POISE user community revolted against the implementation, I could rely on reinforcement from their superiors within the upper echelons of XYZ's IT Infrastructure. The fact was that, up until this point, POISE was the only group of UNIX users at XYZ who weren't using sudo instead of su. So, I knew I was on very solid ground. [3] Because sudo is so simple to use, there wouldn't be any training resources required. A short and simple mail note would provided sufficient user education, and downtime (if any) would be minimal. [4] It's free! Since sudo is a freely distributed, open source program, it was a solution a financially troubled company could afford. We were even spared the licensing & procurement hassles that go along with purchasing commercial software for a large corporation.

sudo's IMPACT (THE "GOOD" NEWS):

It didn't take long for the positive effects of sudo to manifest themselves. Almost immediately, I began noticing its alerts in the /var/adm/messages file. I also liked being able to test sudo's effectiveness by intentionally entering the wrong password. This way I could verify that it was working (by denying unauthorized users) and recording the errors via syslog whenever I wanted. I could even track occasional attempts of non-POISE XYZ users trying to access our systems from within the corporate firewall. Below, in bold, is one such recent example of sudo on the job.

**snowball:/var/adm/messages:**

```
Dec 17 09:23:52 snowball sudo: tdaniels : 3 incorrect password attempts ; TTY=pts/18 ;
PWD=/home/tdaniels/temp; USER=root ; COMMAND=/software/bin/top
Dec 17 09:23:56 snowball tdaniels : SA testing sudo logging on the line above

Dec 17 13:42:57 snowball sudo:  dbrown : command not allowed ; TTY=pts/31 ;
PWD=/u10/misc/fs2/images/finals ; USER=root ; COMMAND=/usr/bin/su sadams
```

Also in the "plus column" was my customers' acceptance of sudo. I had braced myself for a backlash that, fortunately, never came. Maybe it was Murphy's Law at work, but I prefer to think that it was sudo's "ticketing system" that played a part in customer satisfaction. You see, I had anticipated complaints like: "You mean I have to type my password for every su/root command?!?!" But, instead

they appreciated how the "ticketing system" gave them full five-minute sessions with each password. At any rate, I'd like to show you the e-mail note I was prepared to send in the event of a user revolt. Note my heavy reliance on excerpts, quoted chapter and verse, from the actual security policies posted on XYZ' internal web site.

From: Daniels, Trevor
Sent: **<DRAFT: MESSAGE NEVER SENT>**
To: "USA POISE GROUP"
Subject: Using sudo To Comply with Security Policy

Dear Customers,

I regret the inconvenience some of you have experienced resulting from our switch to sudo. It's never the intention of SOAL, TEAM, or myself to hamper your product- ivity in any way. Rather, the decision to replace su was based entirely on improving security, and complying with **XYZ security policies**. Please take a moment to read from *your company's security policies* below. I hope it helps you understand why we had to switch to sudo, and why we can never go back to using su. If then, you still have complaints, I will help you escalate them to XYZ Corporate Security.

Thank You. -Trevor

**(DRAFT: MESSAGE NEVER SENT continued)**

Section 3 - Electronic Information System Security (Subset 6: Access Control):
==========

"Security safeguards for Electronic Information Systems must be installed to ensure access control, identification, authentication, and authorization...Access to XYZ Confidential and XYZ Third Party Confidential information must be restricted to individually authenticated persons."

Section 3 - Electronic Information System Security (Subset 8: User Identification):
==========

"For tracking purposes; personal identifiers (IDs) must be used to identify people, data, and resources...All systems and information must be easily matched to an individual...IDs must be employed to restrict system privileges based on job function...IDs must be uniquely assigned to an individual and/or system...All Users are personally responsible for the usage of his or her User-IDs and their associated passwords, and must therefore never share them with others."

Section 5 - Authorized Use (Subset 10: Authentication):
==========
"Approved authentication techniques must be employed on XYZ Electronic Information Systems to prevent unauthorized access to XYZ Classified Information, or the unauthorized use, control, or administration of the system...Strong Authentication mechanisms must be employed on Electronic Information Systems especially when said business system has high dependencies upon valid identities."

Section 5 - Authorized Use (Subset 12: IS Responsibilities):
==========
"...bypassing Electronic Information Systems security measures are prohibited...A User shall not access or attempt to access an Electronic Information System unless he or she is authorized to do so...A User shall not represent himself or herself as another person...Information Users are required to maintain the Confidentiality, Integrity, and Availability of information accessed."

**<END OF MESSAGE>**

It was probably one of the most effective notes I never sent. Overall, the POISE users were very understanding, and had no great qualms about doing things a little differently than they'd been used to. I suppose I may have underestimated my customer's ability to adapt. But I also think it goes to show you that users can surprise you in more ways than one.

sudo's VULNERABILITIES (THE "BAD" NEWS):

Obviously my struggle to attain network security doesn't end here. Although sudo is very cool, it is by no means the answer to every network administrators' prayer. Like many free software programs (I did mention that it's fee? Didn't I?), sudo has some drawbacks that one simply cannot ignore. And I think it would be remiss of me to sing its praises for ten pages without mentioning a few of its vulnerabilities. So here is a short list of sudo's security shortcomings, along with a few ideas on how to combat them. (a.) sudo traverses passwords in clear text. Fortunately, we use switches at XYZ to minimize the risk of password sniffing and port snooping. But for those firms who are still utilizing hubs, sudo may be more of a curse than a blessing. Still, it is making me seriously consider the introduction of some form of password encryption mechanism, like Kerberos or SSH, as my next major project. (b.) Since sudo uses NIS for name resolution, and NIS has some notorious security problems, naturally it brings those problems along to the party. I will have to give much consideration as to how to effectively combat this issue. Should I use IP addresses or aliases instead of the actual hostnames? Or would that decrease security further? Would sudo still work if

we switched to the more-secure NIS+? Suffice to say, I will have to do a lot more research and consult more of my security colleagues on this subject before I react to it. (c.) There is a real danger that sudo users could alter the sudoers file and "help themselves" to even more privileges. Fortunately, you can safeguard against this, by configuring the sudoers file to make the visudo editor one of the so-called DANGER commands, so regular users can't run it. (Please note that we did this in our sudoers file; seen at the top of page 8.) (d.) From a hacker's perspective, the root passwords are no longer the only targets of temptation. Now we have to be concerned with vigilantly protecting the password of each and every sudo user. As usual, this is yet another case for having a strong password policy and enforcing it with login scripts and/or user account properties. Fortunately, all UNIX users at XYZ are forced to change their passwords every 30 days. Furthermore, those passwords must be at least 8 characters long and contain at least 2 non-alphanumeric characters. Plus, they are only acceptable if they are sufficiently different from the previously used password. Even so, we may still benefit from the addition of TCP Wrappers and PAM to our Solaris systems.

FINAL THOUGHTS:

So as you can see, we've only just begun to fight. A lot more has to be done before we call our systems truly secure (or at least sufficiently hardened). But as we were taught in the SANS-GSEC classes, the quest for attaining security is never-ending, and it must occur via a multi-layered approach. So I think it is only appropriate to close with the words of the legendary sixth century Chinese philosopher, and teacher of Confucius, Lao Tzu, who said: "A journey of a thousand miles begins with a single step". So, if you can think of this quest for security as the thousand mile journey; then implementing sudo to replace su is an excellent first step.

## BIBLIOGRAPHY / REFERENCES:

THE BOOKS:

Ambro, Darrell. Solaris 8 System Admin Exam Cram. Scottsdale,AZ: Coriolis, 2001.

Cook, Randy. Sun Certified Systems Administrator for Solaris 8.0 Study Guide (Exams 310-011 & 310-012). Berkeley, CA: McGraw-Hill, 2001.

Cooper, Michael. Overhauling Rdist for the '90s. Long Beach, CA: U of SC, 1992.

Stern, Hal. Managing NFS and NIS. Sebastopol, CA: O'Reilly & Associates, 2001.

Winsor, Janice. Solaris System Administrator's Guide. Palo Alto, CA: Sun, 2000.

Winsor, Janice. Solaris 8 Advanced System Administrator's Guide. Palo Alto, CA: Sun/Prentice Hall, 2001.

THE WEB:

"Introduction to Rdist Homepage." http://www.magnicomp.com/rdist (January 2003).

Miller, Todd. "Sudo Installation Notes." http://www.courtesan.com/sudo/install.html

Russell, Kathy. "Focus On Unix: Managing Root Access with Sudo." (October 2000) http://unix.about.com/library/weekly/aa102500a.htm (January 2003)

"The Compaq Tru64 UNIX rdist Man Page." http://www.tru64unix.compaq.com/docs/base_doc/DOCUMENTATION/V40G_HTML/MAN/MAN1/0305____.HTM

"The Official Red Hat Linux 8.0 Getting Started Guide." (November, 2003) http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/getting-started-guide/s1-starting-intro-terms.html

"Solaris Resources at Kempston: Installing and Configuring sudo on Solaris 7 and Solaris 8." (April 2000) http://www.kempston.net/solaris/sudo.html (January 2003)