Coping with the Threat of Computer Security Incidents
A Primer from Prevention through Recovery

Russell L. Brand *

June 8, 1990

Abstract

As computer security becomes a more important issue in modern society, it begins to warrant a systematic approach. The vast majority of the computer security problems and the costs associated with them can be prevented with simple inexpensive measures. The most important and cost effective of these measures are available in the prevention and planning phases. These methods are presented followed by a simplified guide to incident handling and recovery.

*Copyright © Russell L. Brand 1989, 1990 Permission to copy and redistribute this information in whole or in part is granted provided each copy includes attribution and the version information. This permission extends for one year minus one day from June 8, 1990; past that point, the reader should obtain a newer copy of the article as the information will be out of date.

Contents

1 Overview **4**

2 Incident Avoidance **5**

2.1 Passwords 5
2.1.1 Joe's 5
2.1.2 Same Passwords on Different Machines 6
2.1.3 Readable Password Files 7
2.1.4 Many faces of a person 8
2.1.5 Automated Checks for Dumb Passwords 8
2.1.6 Machine Generated Passwords 9
2.1.7 The Sorrows of Special Purpose Hardware 11
2.1.8 Is Writing Passwords Down that Bad? 11
2.1.9 The Truth about Password Aging 12
2.1.10 How do you change a password 12
2.2 Old Password Files 12
2.3 Dormant Accounts 12
2.3.1 VMS 12
2.4 Default Accounts and Objects 13
2.4.1 Unix 14
2.4.2 VMS 15
2.4.3 CMS 15
2.5 File Protections 16
2.6 Well Known Security Holes 17
2.7 New Security Holes 18
2.7.1 CERT 18
2.7.2 ZARDOZ 18
2.7.3 CIAC 18
2.8 Excess Services 18
2.9 Search Paths 18
2.10 Routing 19
2.11 Humans 19
2.11.1 Managers 19
2.11.2 Secretaries 19
2.11.3 Trojan Horses 20
2.11.4 Wizards 20
2.11.5 Funders 20
2.12 Group Accounts 20
2.13 .rhosts and proxy logins 21
2.14 Debugging 21
2.15 Getting People Mad at You 21

3 Pre-Planning your Incident Handling **22**

3.1 Goals 22
3.1.1 Maintaining and restoring data 22
3.1.2 Maintaining and restoring service 23
3.1.3 Figuring how it happenned 23

3.1.4 Avoiding the Future Incidents and Escalation 23
3.1.5 Avoiding looking foolish 23
3.1.6 Finding out who did it 24
3.1.7 Punishing the attackers 24
3.2 Backups 24
3.2.1 Why We Need Back Ups 24
3.2.2 How to form a Back Up Strategy that Works 25
3.3 Forming a Plan 26
3.4 Tools to have on hand 27
3.5 Sample Scenarios to Work on in Groups 27

4 Incident Handling **29**
4.1 Basic Hints 29
4.1.1 Panic Level 29
4.1.2 Call Logs and Time Lines 29
4.1.3 Accountability and Authority 29
4.1.4 Audit Logs 29
4.1.5 Timestamps 29
4.2 Basic Techniques 30
4.2.1 Differencing 30
4.2.2 Finding 30
4.2.3 Snooping 30
4.2.4 Tracking 30
4.2.5 Psychology 30
4.3 Prosecution 30
4.4 Exercise 31

5 Recovering From Disasters **32**

A Micro Computers **32**

B VMS Script **35**

C Highly Sensitive Environments **37**

D Handling the Press **38**
D.1 Spin Control 38
D.2 Time Control 38
D.3 Hero Making 38
D.4 Discouraging or Encouraging a Next Incident 38
D.5 Prosecution 39
D.6 No Comment 39
D.7 Honesty 39

E Object Code Protection **40**

F The Joy of Broadcast **41**

G Guest Accounts **42**
G.1 Attack Difficulty Ratios 42
G.2 Individual Sponsors 42
G.3 The No Guest Policy 42

H Orange Book **43**

I Acknowledgements **44**

1 Overview

Since 1984, I have been periodically distracted from my education, my research and from my personal life to help *handle* computer emergencies. After presenting dozens of papers, tutorials talks on computer security, Roger Anderson and George Michale arranged for me to lead a one day intensive seminar on the practical aspects of computer security in an unclassified networked environment for IEEE Compcon. This primer was written as a basic text for this type seminar and has been used for about 2 dozen of them in the past year , and is still in draft form.

The text is divided into four main sections with a number of appendices. The first two major sections of this document contain the material for the morning lecture. The two following sections contain the afternoon lecture contain the afternoon's material. The remaining appendices include material that is of interest to those people who have to deal with other computer security issues.

Since this primer is a direct and simple "how to guide" for cost-effective solutions to computer security problems, it does not contain as many stories and examples as my other tutorials. Those readers interested in these stories or who are having difficulty convincing people in their organization of the need for computer security are referred to *Attack of the Tiger Team*, when it becomes available. and those readers interested in comprehensive list of computer security vulnerabilities should contact the author regarding the *Hackman* project.

Suggestions, questions and other comments are always welcome. Please send comments to `primer@cert.sei.cmu.edu`. I hope to publish a this set of notes in a more complete form in the future. When sending comments or questions, please mention that you were reading version CERT 0.6 of June 8, 1990.

Russell L. Brand
brand@lll-crg.llnl.gov
1862 Euclid Ave, Suite 136
Berkeley, CA 94709

2 Incident Avoidance

"An ounce of prevention is worth a pound of cure." In computer security this is an understatement by a greater factor than can be easily be believed. Very little has historically been done to prevent computer break-ins and I have been told by a number of the country's top computer scientists that "Computer Security is a waste of time." The belief that security measures or preventive medicine is a waste has led to giant expenditures to repair damage to both computers and people respectively. Must of my surprise, several system managers reviewing this document were sure that even basic preventative measures would not be cost effective as compared to repairing disasters after they occurred.

The vast majority of the security incidents are caused by one of about a dozen well understood problems. By not making these mistakes, you can prevent most of the problems from happening to your systems and avoid untold hassles and losses. Almost every site that I survey and almost every incident that did not involve insiders was caused by one of these problems. In the most of the insider cases, no amount of computer security would have helped and these are in many ways demonstrated problems with physical security or personnel policy rather than with computer security per se.

Most of the security incidents are caused by "attackers" of limited ability and resources. Because of this and because there are so many easy targets, if you provide the most basic level of protection, most of the attackers will break into some other site instead of bothering yours. There are of course exceptional cases. If you are believed to have highly sensitive information or are on a "hit list" of one type or another, you may encounter more dedicated attackers. Readers interested in more comprehensive defensive strategies should consult the appendices.

Over all, prevention of a problem is about four orders of magnitude cheaper than having to handling it in the average case. Proper planning can reduce the cost of incident handling and recovery and is discussed in the section on planning. In addition to whatever other measures are taken, the greatest incremental security improvement will be obtained be implementing the simple measures described below.

2.1 Passwords

While "good passwords" is not a hot and sexy topic and will never command the prestige of exploitable bugs in the operating system itself, it is the single *most important* topic in incident prevention. Doing everything else entirely correctly is almost of *no value* unless you get this right!

2.1.1 Joe's

A "Joe" is an account where the username is the same as the password. This makes the password both easy to remember and easy to guess. It is the single most common cause of password problems in the modern world.

In 1986, there was popular conjecture that every machine had a Joe. There was fair amount of random testing done and in fact a Joe was found on each

and every machine tested. These included machines that had password systems designed to prevent usernames from being used as passwords.

This summer, while I was testing a series of sensitive systems, where hundred of thousands of dollars were spent to remove security holes including re-writing a fair fraction of the operating system, there were Joes.

It is worthwhile to include a process in your system batching file (`cron` on unix) to check for Joes explicitly. The most common occurrences of Joes is the initial password that the system administrators set for an account which has never been changed. Often this initial password is set by the administrator with the expectation the user will change it promptly. Often the user doesn't know how to change it or in fact never logs in at all. In the latter case a dormant account lies on the system accomplishing nothing except wasting system resources and increasing vulnerabilities.

2.1.2 Same Passwords on Different Machines

Many years ago when a computing center had a single mainframe the issue of a user having the same password on multiple machines was moot. As long the number of machines that a user accessed was very small, it was reasonable to request that a person to use a different password on each machine or set of machines. With a modern workstation environment, it is no longer practical to expect this from a user and a user is unlikely to comply if asked. There are a number of simple compromise measures that can and should be taken.

Among these measures is requesting that privileged users have different passwords for their privileged accounts than for their normal use account and for their accounts on machines at other centers. If the latter is not the case, then anyone who gains control of one of these "other" machines which you have no control over, has gained privileged access to yours as well.

The basic question of when passwords should be the same is actually a simple one. Passwords should be the same when the two machines are (1) logically equivalent (as in a pool of workstations), (2) "trust each other" to the extent that compromising one would compromise the others in other ways, or (3) are run by the same center with the same security measures. Passwords should be different when the computers are (1) run by different organizations, (2) have different levels of security or (3) have different operating systems.

Lest this seems too strict, be assured that I have on several occasions broken into machines by giving privileged users on the target machines accounts on one of my own and exploiting their use of the same password on both. Further, machines with different operating systems are inherently vulnerable to different "programming bugs" and hence by having the same passwords on the two machines, each machine is open to the all the bugs that could exist on either system.

It is interesting (but of little practical value) to note that an attacker can gain a cryptographic advantage by having two different encrypted strings for the same password. This would happen when the user has the same password on two machines but it has been encrypted with different salts. In principle, this makes hostile decryption much easier. In practice, the attack methods that are most often used do not exploit this.

The worst offenders of the "shared password problem" are network maintenance people and teams. Often they want an account on every local area net that they service, each with the same password. That way they can examine network problems and such without having to look up hundreds of passwords.

While the network maintainers are generally (but not always) good about picking reasonable passwords and keeping them secret, if any one machine that they are using has a readable password file (discussed below) or is ever compromised, this password is itself compromised and an attacker can gain unauthorized access to hundreds or thousands of machines.

2.1.3 Readable Password Files

A readable password file is an accident waiting to happen. With access to the encrypted password an attacker can guess passwords at his leisure without you being able to tell that he is doing so. Once he has a correct password, he can then access your machine as that user. In the case of certain operating systems, including older versions of VMS, there is a well know inversion for the password encryption algorithm and hence the attacker doesn't need to guess at all once he can read the password file.

Changing the encryption method to some other method that is also publically known doesn't help this set of problems, even if the crypto-system itself is much stronger. The weakness here is not in the crypto-system but rather in the ease of making guesses.

It is vital to protect your password file from being read. There are two parts to this. First you should prevent anonymous file transfers from be able to remove a copy of the password file. While this is generally very easy to do correctly, there is a common mistake worth avoiding. Most file transfer facilities allow you to restrict the part of the file system from which unauthenticated transfers can be made. It is necessary to put a partial password file in this subsection so that an anonymous agent knows "who it (itself) is". Many sites have put complete password files here defeating one of the most important purposes of the restrictions. (Of course without this restriction "World Readable" takes on a very literal meaning...)

The second part of the solution is somewhat harder. This is to prevent unprivileged users who are using the system from reading the encrypted password from the password file. The reason that this is difficult is that the password file has a great deal of information that people and programs need in it other than the passwords themselves. Some version of some operating systems have privileged calls to handle the details of all this and hence their utilities have already been written to allow protection of the encrypted passwords.

Most of the current versions of Unix are *not* among of these systems. Berkeley has distributed a set of patches to incorporate this separation (called shadow passwords) and the latest version of the SunOS has facilities for it. For those who are using an operating system that does not yet have shadow passwords and cannot use one of the new releases, a number of ad hoc shadowing systems have been developed. One can install shadow passwords by editing the binaries of `/bin/login`, `/bin/passwd` and similar programs that actually need to use the password fields and then modify `/etc/vipw` to work with both the diminished

and shadow password files.

Of course, since most of us use broadcast nets, there is a real danger of passwords being seen as they go over the wire. This class of problems is discussed in the the Joys of Broadcast appendix and the Guests appendix.

Kerberos, developed at MIT's Athena project has an alternative means of handling passwords. It allows one to remove all the passwords from the normal use machines and to never have them broadcasted in clear text. While Kerberos is vulnerable to a number of interesting password guessing and cryptographic attacks and currently has problems with multi-home machines (Hosts with more than one IP address), it does provide the first practical attempt and network security for a university environment.

An often overlooked issue is that of passwords for games. Many multiplayer computer games, such as "Xtrek" and "Empire" require the user to supply a password to prevent users from impersonating one another during the game. Generally these passwords are stored by the game itself and are in principle unrelated to the passwords that the operating system itself uses. Unfortunately, these passwords are generally stored unencrypted and some users use the same password as they do for logging into the machine itself. Some games now explicitly warn the users not use his login passwords. Perhaps these games will eventually check that the password is indeed not the same as the login password.

2.1.4 Many faces of a person

A single individual can have many different relationships to a computer at different times. The system programmers are acting as "just users" when they read their mail or play a computer game. In many operating systems, a person gets all of his privileges all of the time. While this is not true in Multics, it is true in the default configuration of almost every other operating system. Fortunately a computer doesn't know anything about "people" and hence is perfectly happy to allow a single person have several accounts with different passwords at different privilege levels. This helps to prevent the accidentally disclosure of a privileged password. In the case where the privileged user has his unprivileged account having the same password as his unprivileged account on other machines it will at least be the case that his privileges are not compromised when and if this other machine is compromised.

The one case where it is especially important to have separate accounts or passwords for a single individual is for someone who travels to give demos. One can be assured that his password will be lost when he is giving a demo and something breaks. The most common form of "breakage" is a problem with duplex of of delay. It would nice if all that was lost was the demo password and for the demo password to be of no use to an attacker.

2.1.5 Automated Checks for Dumb Passwords

Automated checks for dumb passwords come in three varieties. The first is to routinely run a password cracker against the encrypted passwords and notice what is caught. While this is a good idea, it is currently used without either of the other two mechanisms we will describe. Since it is computationally less

efficient than the others by about a factor of 50,000, it should be used to supplement the others rather than be used exclusively. Among its many virtues is that an automated checking system that reads the encrypted passwords does not require having source for the operating system or making modification an system modifications.

The second method of preventing dumb password is to alter the password changing facility so that it doesn't accept dumb passwords. This has two big advantages over the first method. The first of these is computational. The second is more important. By preventing the user from selecting the poor password to begin with, one doesn't need an administrative procedure to get him to change it later. It can all happen directly with no human intervention and no apparent accountability. As a general rule, people are not happy about passwords and really don't want to hear from another person that they need to change their password yet again.

While this change does require a system modification, it can often be done without source code by writing a pre-processor to screen the passwords before the new password is passed to the existing utilities. The weakness in this approach lies with the users who are not required to use the new style of password facility. As a result, one finds that facilities that use only this method have good passwords for everyone except the system staff and new users who have had their initial passwords set by the system staff.

The third method is designed primarily to catch the bad passwords that are entered in despite the use of the second method. Once could check the "dumbness" of a password with each attempted use. While this is computationally more expensive than the second method, it generally catches everyone. Even the system programmers tend to use the standard login utility. It has the nice feature of locking out anyone that finds a way to circumvent the second method. This generally requires a small amount of system source and risks causing embarrassment to "too clever" system staff members.

In terms of dumb passwords, there are a number of "attack lists". An attack list is a list of common passwords that an attacker could use to try to login with. Several of these have been published and more are constantly being formed. These lists are used for the automated password guesser and they may also be used directly in the second and third method described above. With the second and third method one may also use criteria including minimum length, use of non-alphabetic characters, etc. Finally, information about the individual user found in standard system files can be scanned to see if the user has incorporated this information into his password.

2.1.6 Machine Generated Passwords

Most users hate machine generated passwords. Often they are unrememberable and accompanied by a warning to "*Never* write them down" which is a frustrating combination. (We will discuss the the writing down of passwords later.) Machine generated passwords come in four basic types

Gibberish. This is the most obvious approach to randomness. Independently selected several characters from the set of all printable characters. For a

six character password, this gives about 40 bits of randomness. It is very hard to guess and perhaps even harder to remember.

Often a little bit of post processing is done on these passwords as well as on the random syllables discussed below. This post processing removes passwords that might prove offensive to the user. When a potentially offensive password is generated, the program simply tries again. The user often behaves the same way and runs the randomizer over and over again until a password that seems less random and more memorable to him is selected. In principle, the clever user could write a program that kept requesting new random passwords until an English word was chosen for him; this would take much too long to be practical.

Numbers. Numbers are a lot like letters. People don't try to pronounce them and there are very few numbers that are "offensive" per se. An eight digit random number has about 26 bits of randomness in it and is of comparable strength to a 4 character random password chosen from the unrestricted set of printable characters. (The amount of randomness in a password is the log (base 2) of the number of possible passwords if they were all equally likely to occur.)

Eight digit numbers are hard to remember. Fortunately "chunking" them into groups (as 184—25—7546) makes this less difficult than it would otherwise be.

Syllables. This is by far the most common method currently used. The idea is to make non-words that are easy to remember because they sound like words. A three syllable, eight letter non-word often has about 24 bits of randomness in it making it not quite as strong as an 8 bit number but hopefully a little bit more memorable.

The principle here is good. In fact, this pseudo-word idea should work very well. In practice it fails miserably because the standard programs for generating these pseudo-syllables are very poor. Eventually we may find a good implementation of this and see a higher level of user acceptance.

Pass Phrases. Pass phrases are the least common way to implement machine generated passwords. The idea here is very simple. Take 100 nouns, 100 verbs, 100 adjective and 100 adverbs. Generate an eight digit random number. Consider it as four 2 digit random numbers and use that to pick one of each of the above parts of speech. The user is then given a phrase like "Orange Cars Sleep Quickly." The words within each list are uniquely determined by their first two characters. The user may then type the phrase, the first few letters of each word or the eight digit number.

The phrases are easy to remember, the system remains just as secure if you publish the list of words and has about 26 bits of randomness. One can adapt the system down to three words with 20 bits of randomness and still be sufficiently safe for most applications.

I believe that machine generated passwords are generally a bad solution to the password problem. If you must use them, I strongly urge the use of pass-phrases over the other methods. In any event, if your center is using machine

generated passwords, you should consider running an occasional sweep over the entire user file system looking for scripts containing these passwords. Proper selection of your password generation algorithm can make this much easier than it sounds.

As with almost all password issues, the user of a single computer center which gives him one machine generated password for access to all the machines he will use will not have nearly the level of difficulty as the user who uses computers at many centers and might have to remember dozens or even hundreds of such passwords.

2.1.7 The Sorrows of Special Purpose Hardware

With the problems of broadcast networks and user selecting bad passwords or rebelling at machine generated password, some facilities have turned to special purpose hardware that generates keys dynamically. Generally these devices look like small calculators (or smart card) and when a user enters a short password (often four digits) they give him a password that is good for a single use. If the person wants to login again, he must get a new password from his key-generator.

With a few exceptions, the technology of these devices works very well. The exceptions include systems with bad time synchronization, unreliable or fragile hardware or very short generated keys. In at least one case the generated keys were so short that it was faster to attack the machine by guessing the password "1111" than by guessing at the user generated passwords it replaced.

Despite the technology of these devices working well and the installation generally being almost painless, there are two serious problems with their use. The first is cost. Buying a device for a user of large center can easily cost more than an additional mainframe. The second problem is more serious. This is one of user reluctance. Most users are unwilling to carry an extra device and the people who are users of many centers are even less willing to hold a dozen such devices and remember which is which.

In one center, these devices were used only for privileged accesses initiated from insecure locations. Only a handful of them had to be made. (Being innovative, the center staff built them from old programmable calculators.) They were used only by the "on call" system programmer when handling emergencies and provided some security without being to obtrusive.

2.1.8 Is Writing Passwords Down that Bad?

One of the first things that we were all told when we began using timesharing is that one should never write down passwords. I agree that the users should not record their passwords on-line. There have been a large number of break-ins enable by a user having a batch script that would include a clear-text password to let them login to another machine.

On the other hand, how often has your wallet been stolen? I believe that a password written down in wallet is probably not a serious risk in comparison to other the problems including the selection of "dumb" password that are easier to remember. In classified systems, this is, of course, not permitted.

2.1.9 The Truth about Password Aging

Some facilities force users to change their passwords on a regular basis. This has the beneficial side effect of removing dormant accounts. It is also the case that it limits the utility of a stolen password.

While these are good and worthwhile effects, most system administrators believe that changing passwords on a regular basis makes it harder for an attacker to guess them. In practice, for an attacker that has gotten the crypt text of the password file, he generally only needs a few hours to find the passwords of interest and hence frequent changes do not increase the difficulty of his task. For the attacker who is guessing without a copy of the encrypt password, even changing the password every minute would at most double the effort he would be required to expend.

2.1.10 How do you change a password

Users should be told to change their passwords whenever they have reason to expect that another person has learned their passwords and after each use of an "untrusted" machine. Unfortunately many users are neither told this, nor how to change the password. Be sure both to tell you users how to change their passwords and include these instructions in the on-line documentation in an obvious place. Users should not be expected to realize the password changing is (1) an option for directory maintenance under TOPS-20 and many versions of CMS, (2) is spelled `passwd` under unix or (3) is an option to `set` under VMS.

2.2 Old Password Files

It is often the case at sites running shadow password systems, someone forgets to prevent the shadow password file from being publically readable. While this is easy to prevent by having a batch job that routinely revokes read permissions that were accidently granted, there is an interesting variant of this problem that is harder to prevent.

When password files are edited, some editors leave backup files that are publically readable. In fact when a new system is installed a password file is often created by extracting information from the password files of many existing systems. The collection of password files is all too often left publically readable in some forgotten disk area where it is found by an attacker weeks or months later. The attacker then uses this data to break into a large number of machines.

2.3 Dormant Accounts

While requiring annual password changes does eventually remove dormant accounts, it is worthwhile to try a more active approach for their removal. The exact nature of this approach will vary from center to center.

2.3.1 VMS

In VMS, the account expiration field is a good method of retiring dormant accounts, but care should be taken as no advance notice is given that an account

is near expiration.

Also VMS security auditing makes the removal of expired users a bad idea. Because one of the most common errors is typing the password on the username line, DEC suppresses any invalid username from the logs until a breaking attempt is detected. But if the username is valid and the password wrong, the username is logged.

2.4 Default Accounts and Objects

One of the joys of many operating systems is that they come complete with prebuilt accounts and other objects. Many operating systems have enabled either accounts or prelogin facilities that present security risks.

The standard "accounts" for an attacker to try on any system include the following:

Open. A facility to automatically create new accounts. It is often set by default to not require either a password or system manager approval to create the new accounts.

Help. Sometimes the pre-login help is too helpful. It may provide phone numbers or other information that you wouldn't want to advertise to non-users.

Telnet. Or Terminal. An account designed to let someone just use this machine as a stepping stone to get to another machine. It is useful for hiding origins of an attack.

Guest. Many operating systems are shipped with guest accounts enabled.

Demo. Not only are several operating systems shipped with a demo account, but when installing some packages, a demo account is automatically created. All too often the demo account has write access to some of the system binaries (executable files).

Games. Or Play. Often the password is Games when the account name is Play. In some cases this account has the ability to write to the Games directory allowing an attacker to not only play games, and snoop around, but to also insert Trojan horses at will.

Mail. Quite often a system is shipped with or is given an unpassworded mail account so that people can report problems (like their inability to login) without logging in. In two-thirds of the systems that I have observed with such an account, it was possible to break into the main system through this account.

Often these default accounts are normal accounts with an initialization file (`.login`, `.profile`, `login.cmd`, `login.bat`, etc.) or alternate command line interpreter to make it do something non-standard or restrict its action. These are generally called, "Captive Accounts" or "Turnkey Logins." Setting up a restricted login so that it stays restricted is very hard. It should of course be very easy, but in most cases a mistake is made.

Subjobs. It is often the case that a restricted account is set up to only run a single application. This single application program is invoked by a startup script or instead of the standard command interpreter. Very often this program has an option to spawn a subprocess.

In some cases this might be an arbitrary job (e. g. the `/spawn` option to Mail in VMS or ":!" to vi in unix) or might be limited to a small number of programs. In the former case the problem is immediate, in the latter case, it is often the case that one of these programs in turn allows arbitrary spawning.

A carefully written subsystem will prevent this (and all other such problems). Generally these subsystems are created quickly rather than carefully.

Editors. Most editors are sufficiently powerfully that if the restricted system can use an editor, a way can be found to cause problems.

Full Filenames. Many restricted subsystems presume that by resetting the set of places the command interpreter looks for executable programs (called its "search path") functionality can be restricted. In unix this might be done by altering the `Path` variable or the logical names table in VMS.

All too often the clever attacker is able to defeat this plan by using the complete filename of the file of interest. Sometimes non-standard names for the file are necessary to circumvent a clever restriction program.

Removable Restriction Files. When a system relies on an initialization file to provide protection, it is important that this file cannot be altered or removed. If an restricted application is able to write to its "home directory" where these initialization files are kept it can often free itself.

Non-standard Login. Some network access methods do not read or respect the startup files. Among these are many file transfer systems. I have often been able to gain privileged access to a machine by using the the login and password from a captive account with the file transfer facility that didn't know that these accounts weren't "normal." Many file transfer facilities have methods for disabling the use of selected accounts.

Interrupts. It is sad that a number of the captive accounts won't withstand a single interrupt or suspend character. Try it just to be sure.

Making sure that you have not made any of the above listed mistakes is of course not sufficient for having a perfectly safe system. Avoiding these mistakes, or avoiding the use of captive accounts at all, is enough to discourage the vast majority of attackers.

Each operating system for each vendor has some particular default accounts that need to be disabled or otherwise protected.

2.4.1 Unix

Under unix there are a lot of possible default accounts since there are so many different vendors. Below is a partial list of the default accounts that I have successfully used in the past that are not mentioned above.

Sysdiag. Or `diag`. This is used for doing hardware maintenance and should have a password.

Root. Or `Rootsh` or `rootcsh` or `toor`. All to often shipped without a password.

Sync. Used to protect the disks when doing an emergency shutdown. This account should be restricted from file transfer and other net uses.

Finger. Or `Who` or `W` or `Date` or `Echo`. All of these have legitimate uses but need to be set up to be properly captive.

Among the things that one should do with a new unix system is

```
grep ::  /etc/passwd
```

to see what unpassworded accounts exist on the system. All of these are worth special attention.

2.4.2 VMS

Since VMS is available from only one vendor, the default account here are better known. On large systems, these appear with standard well known passwords. On smaller systems, these accounts appear with no passwords at all. With the exception of `Decnet`, all have been eliminated on systems newer than version 4.6.

Decnet

System

Systest

Field

UETP

Many of the networking and mail delivery packages routinely added to VMS systems also have well know password. In the past six months these accounts have been commonly used to break into VMS systems.

MMPONY

PLUTO

The password on all of these accounts should be reset when a new system is obtained. There are many problems with the `DECNET` account and the with the `Task 0` object. System managers should obtain one of the standard repair scripts to remove these vulnerabilities.

2.4.3 CMS

It has been many years since I have seriously used CMS. At last glance the default configuration seemed to include well know passwords for two accounts.

rcsc

operator

2.5 File Protections

With file protections simple measures can avoid most problems. Batch jobs should be run on a regular basis to check that the protections are correct.

Writable Binaries and System Directories. The most common problem with file protections is that some system binary or directory is not protected. This allows the attacker to modify the system. In this manner, an attacker will alter a common program, often the directory listing program to create a privileged account for them the next time that a privileged user uses this command.

When possible the system binaries should be mounted read-only. In any event a program should systematically find and correct errors in the protection of system files. "Public" areas for unsupported executable should be moderated and these executable should *never* be used by privileged users and programs. System data files suffer from similar vulnerabilities.

Readable Restricted System Files. Just as the encrypted passwords need to be protected, the system has other data that is worth protecting. Many computers have passwords and phone numbers of other computers stored for future use. The most common use of this type of information is for network mail being transported via UUCP or protected DECNET. It is difficult to rework these systems so that this information would not be necessary and hence it must be protected. You have an obligation to protect this data about your neighbors just as they have a responsibility to protect similar data that they have about you.

Home Dir's and Init Files Shouldn't Be Writable. Checking that these directories and files can be written only by the owner will prevent many careless errors. It is also worthwhile to check that peoples mail archives are not publically readable. Though this is not directly a security threat, it is only one more line of code while writing the rest of this.

In many versions of the common operating systems special checks are placed in the command interpreters to prevent them from using initialization files that were written by a third party. In this case there are still at least two types of interesting attacks. The first is to install a Trojan horse in the person's home directory tree rather than in the initialization file itself and the second is to simple remove the initialization files themselves. Often security weaknesses are remedied through the proper initialization file and without these files the vulnerabilities are re-introduced.

No Unexpected Publically Writable Files or Directories. There are of course places and individual files that should be publically writable but these are stable quantities and the script can ignore them. In practice user seems to react well to being told about files that they own that are publically overwritable.

When Parents aren't Owners. While it is not unusual for someone to have a link to a file outside of his directory structure, it is unusual for there

to be a file to be in his home directory that is owned by someone else. Flagging this when the link-count is "1" is worthwhile.

Automated scripts can find these errors before they are exploited. In general a serious error of one of the types described above is entered into a given cluster university system every other week.

2.6 Well Known Security Holes

While hundreds of security holes exist in commonly used programs, a very small number of these account for most of the problems. Under modern version of VMS, most of them relate to either DECNET or creating Mailboxes.

Under unix, a handful of programs account for most of the problems. It is not that these bugs are any worse or easier to exploit than the others, just that they are well known and popular. The interested reader is referred to the Hackman Project for a more complete listing.

Set-Uid Shell Scripts. You should not have any set-uid shell scripts. If you have system source, you should consider modifying chmod to prevent users from creating set-uid programs.

FTP. The file transfer utilities has had a number of problems both in terms of configuration management (remembering to disallow accounts like "sync" from being used to transfer files) and legitimate bugs. Patched version are available for most systems.

Login on the Sun 386i and under Dec Ultrix 3.0, until a better fix is available,

```
chmod 0100 /bin/login
```

to protect yourself from a serious security bug.

Sendmail. Probably the only program with as many security problems as the yellowpages system itself. Again a patched version should be obtained for your system.

TFTP. This program should be set to run as an unprivileged user and/or chrooted.

Rwalld. This program needs to be set to run as an unprivileged user.

Mkdir. Some versions of unix do not have an atomic kernel call to make a directory and hence can leave the inodes in a "bad" state if it is interrupted at just the right moment. If your system is one of these it is worthwhile to write a short program that increases the job priority of a job while it is making a directory so as to make it more difficult to exploit this hole.

YP & NFS. Both present giant security holes. It is important to arrange to get patches as soon as they become available for these subsystems because we can expect more security problems with them in the future. Sun has recently started a computer security group that will help solve this set of problems.

While the ambitious and dedicated system manager is encouraged to fix all of the security problems that exist, fixing these few will discourage most of the attackers.

2.7 New Security Holes

New security holes are always being found. There are a number of computer mailing lists and advisory groups the follow this. Three groups of particular interest are CERT, ZARDOZ and CIAC.

2.7.1 CERT

Cert is a DARPA sponsored group to help internet sites deal with security problems. They may be contacted as `cert@cert.sei.cmu.edu`. They also maintain a 24 hour phone number for security problems at (412) 268-7090.

2.7.2 ZARDOZ

Neil Gorsuch moderates a computer security discussion group. He may be contacted as `zardoz!security-request@uunet.UU.NET` or `security-request@cpd.com`.

2.7.3 CIAC

CIAC is the Department of Energy's Computer Incident Advisory Capability team led by Gene Schultz. This team is interested in discovering and eliminating security holes, exchanging security tools, as well as other issues. Contact CIAC as `ciac@tiger.llnl.gov`.

2.8 Excess Services

Every extra network service that a computer offers potentially poses an additional security vulnerability. I am emphatically *not* suggesting that we remove those services that the users are using, I am encouraging the removal of services that are unused. If you are not getting a benefit from a service, you should not pay the price in terms of system overhead or security risk. Sometimes, as with `rexecd` under unix, the risks are not immediately apparent and are caused by unexpected interactions that do not include any bugs per se.

2.9 Search Paths

If a user has set his search path to include the current directory (".” on Unix), he will almost always eventually have a serious problem. There are a number of security vulnerabilities that this poses as well as logistical ones. Searching through the all of the users initialization files and/or through the process table (with `ps -e` on unix) can detect this problem.

2.10 Routing

Routing can provide a cheap partial protection for a computer center. There are some machines that don't need to talk to the outside world at all. On others, one would might like to be able to initiate contact outward but not have any real need to allow others to contact this machine directly.

In an academic computer when administrative computers are placed on same network as the student machines, limiting routing is often a very good idea. One can set up the system such that the users on administrative machines can use the resources of the academic machines without placing them at significant risk of attack by the student machines.

Ideally one would wish to place the machines that need to be protected on their own local area net with active routers to prevent an attacker from "listening in" on the broadcast net. This type of an attack is becoming increasingly popular.

2.11 Humans

In almost all technological systems, the weakest link is the human beings involved. Since the users, the installers and the maintainers of the system are (in the average case) all humans, this is a serious problem.

2.11.1 Managers

Managers, bosses, center directors and other respected people are often given privileged accounts on a variety of machines. Unfortunately, they often are not as familiar with the systems as the programmers and system maintainers themselves. As a result, they often are the targets of attack. Often they are so busy that do not take the security precautions that others would take and do not have the same level of technical knowledge. They are given these privileges as a sign of respect. They often ignore instructions to change passwords or file protections

The attackers rarely show this level of respect. They break into the unprotected managerial account and use it as a vector to the rest of the system or center. This leads to an embarrassing situations beyond the break-in itself as the manager is made to look personally incompetent and is sometimes accused of being unfit for his position.

Prevent this type of situation form occurring by giving privileges only to people that need and know how to use them.

2.11.2 Secretaries

Secretaries are often give their bosses passwords by their bosses. When a secretary uses his bosses account, he has all the privileges that his boss would have and generally does not have the training or expertise to use them safely.

It is probably not possible to prevent bosses from giving their passwords to their secretaries. Still one can reduce the need for this by setting up groups correctly. One might consider giving "bosses" two separate accounts one for

routine use and one for privileged access with a hope that they will only share the former with their secretary.

2.11.3 Trojan Horses

Having an "unsupported" or "public" area on disk where users place binaries for common use simplifies the placement of Trojan horse programs. Having several areas for user maintained binaries and a single user responsible for each reduces but does not eliminate this problem.

2.11.4 Wizards

Wizards and system programmers often add their own security problems. They are often the ones to create privileged programs that are needed and then forgotten about without being disabled. Thinking that an account doesn't need to be checked/audited because it is owned by someone that should know better than to make a silly mistake is a risky policy.

2.11.5 Funders

Funders are often giving accounts on the machines that they "paid for." All to often these accounts are never used but not disabled even though they are found to be dormant by the procedures discussed above. Again, this is a mistake to be avoided.

2.12 Group Accounts

A group account is one that is shared among several people in such a way that one can't tell which of the people in the group is responsible for a given action.

Those of you familiar with Hardin's "The Tragedy of The Common" will understand that this is a problem in any system computer or otherwise. Part of the problem here is with passwords.

1. You can't change the password easily. You have to find everyone in the group to let them know.
2. If something Dumb happens you don't know who to talk to about it.
3. If someone shares the group password with another person, you can never find out who did or who all the people who knew the password were.

Group accounts should *always* be avoided. The administrative work to set up several independent accounts is very small in comparison to the extra effort in disaster recovery for not doing so.

One must not only avoid the explicit group accounts, but also the implicit ones. This is where an individual shares his password with dozens of people or allows dozens, perhaps hundreds of them to use his through proxy logins or `.rhosts`.

2.13 .rhosts and proxy logins

Just as some people trust each other, some accounts trust each other and some machines trust each other. There are several mechanism for setting up a trust relationship. Among these are `hosts.equiv`, `.rhosts`, and proxy logins.

These mechanisms essentially allow a user to login from one machine to another without a password. There are three basic implications to this.

1. If you can impersonate a machine, you can gain access to other machines without having to provide passwords or find bugs.

2. Once you get access to one account on one machine, you are likely to be able to reach many other accounts on other machines.

3. If you gain control of a machine, you have gained access to all the machines that trusts it.

Various experiments have shown that by starting *almost anywhere interesting*, once one has control of one medium size machine, one can gain access to tens of thousands of computers. In my most recent experiment, starting from a medium size timesharing system, I gained immediate access to 150 machines and surpassed 5000 distinct machines before completing the second recursion step.

2.14 Debugging

About one third of the security holes that I have come across depend on a debugging option being enabled. When installing system software, always check that all the "debugging" options that you are not using are disabled.

2.15 Getting People Mad at You

It is sad but true that a small number of sites have gotten groups of hackers angry at them. In at least two cases, this was because the hackers had found an interesting security hole, had tried to contact the administrators of the center and were given a hard time when they were seriously trying to help.

When one is given a "tip" from someone that won't identify themselves about a security problem, it is generally worth investigating. It is not worth trying to trick the informant into giving his phone number to you. It almost never works, and it is the "type of dirty trick" that will probably get people mad at you and at the very least prevent you from getting early warnings in the future.

3 Pre-Planning your Incident Handling

3.1 Goals

Despite your best plans to avoid incidents they may very well occur. Proper planning can reduce their serverity, cost and inconvenience levels. There are about half dozen different goals that one can have while handling an incident.

1. Maintain and restore data.
2. Maintain and restore service.
3. Figure out how it happenned.
4. Avoid the future incidents and escalation.
5. Avoid looking foolish.
6. Find out who did it.
7. Punish the attackers.

The order shown above is what I believe the order of priorities generally should be. Of course in a real situation there are many reasons why this ordering might not be appropriate and we will discuss the whens and why of changing our priorities in the next section.

For any given site, one can expect that a standard goal prioritization can be developed. This should be done in advance. There is nothing so terrible as being alone in a *cold* machine room at 4 on a Sunday morning trying to decide whether to shut down the last hole to protect the system or try to get a phone trace done to catch the attacker. It is similarly difficult to decide in the middle of a disaster whether you should shut down a system to protect the existing data or do everything you can to continue to provide service.

Noone who is handling the technical side of an incident wants to make these policy decisions without guidance in the middle of a disaster. One can be sure that these decisions will be replayed an re-analyzed by a dozen "Monday Morning Quarterbacks" who will explain what should have been done could not be bothered to make up a set of guidelines before.

Let us look at each of these goals in a little more detail.

3.1.1 Maintaining and restoring data

To me, the user data is of paramount importance. Anything else is generally replacable. You can buy more disk drives, more computers, more electrical power. If you lose the data, though a security incident or otherwise, it is gone.

Of course, if the computer is controlling a physical device, there may be more than just data at stake. For example, the most important goal for the computer in Pacemaker is to get the next pulse out on time.

In terms of the protection of user data, there is *nothing* that can take the place of a good back-up strategy. During the week that this chapter was written, three centers that I work with suffered catastrophic data loss. Two of the three

from air conditioning problems, one from programmer error. At all three centers, there were machines with irreplacable scientific data that had never been backed up in their lives.

Many backup failures are caused by more subbtle problems than these. Still it is instructive to note that many sites *never* make a second copy of their data. This means than any problem from a defective disk drive, to a water main break, to a typing mistake when updating system software can spell disaster.

If the primary goal is that of maintaining and restoring data, the first thing to do during an incident needs to be to check when the most recent backup was completed. If it was not done very recently, an immediate full system dump *must* be made and the system must be shutdown until it is done. Of course, one can't trust this dump as the attacker may have already modified the system.

3.1.2 Maintaining and restoring service

Second to maintaining the data, maintaining service is important. Users have probably come to rely on the computing center and will not be pleased if they can't continue to use it as planned.

3.1.3 Figuring how it happenned

This is by far the most interesting part of the problem and in practice seems to take precident over all of the others. It of course strongly conflicts with the two preceeding goals.

By immediately making a complete copy of the system after the attack, one can analyze it at one's leisure. This means that we don't need to worry about normal use destroying evidence of about the attacker re-entering to destroy evidence of what happenned.

Ultimately, one may never be able to determine how it happenned. One may find several ways that "could have happenned" presenting a number of things to fix.

3.1.4 Avoiding the Future Incidents and Escalation

This needs to be an explicit goal and often is not realized until much too late. To avoid future incidents one of course should fix the problem that first occurred and remove any new security vulnerabilities that were added either by the attackers or by the system staff while trying to figure out what was going on.

Beyond this, one needs to prevent turning a casual attacker who may not be caught into dedicate opponent, to prevent enticing other attackers and to prevent others in one's organization and related organizations from being forced to introduce restrictions that would be neither popular nor helpful.

3.1.5 Avoiding looking foolish

Another real world consideration that I had not expected to become an issue is one of image management. In practice, it is important not to look foolish in the press, an issue that we will discuss more fully in an appendix. Also it

is important for the appropriate people within the organization to be briefed on the situation. It is embarrising to find out about an incident in one's own organization from a reporter's phone call.

3.1.6 Finding out who did it

This goal is often over emphasized. There is definitely a value in knowing who the attacker was so that one can debrief him and discourage him from doing such things in the future.

In the average case, it effort to determine the attackers identity than it is worth unless one plans to prosecute him.

3.1.7 Punishing the attackers

This merits of this goal have been seriously debated in the past few years. As a practical matter it is very difficult to get enough evidence to prosecuter someone and very few succesful prosecutions. If this is a one of the goals, very careful record keeping needs to be done at all times during the investigation, and solving the problem will be slowed down as one waits for phone traces and various court orders.

3.2 Backups

It should be clear that accomplishing most of the goals requires having extra copies of the data that is stored on the system. These extra copies are called "Backups" and generally stored on magnetic tape.

Let us consider two aspects of keeping backup copies of your data. First, we will look at why this important and what the backups are used for and then we will examine the charateristics of a good backup strategy.

3.2.1 Why We Need Back Ups

Good back ups are needed for four types of reasons. The first three of these are not security related per se, though an insufficeint back up strategy will lead to problems with these first three as well.

If a site does not have a reliable back up system, when an incident occurs, one must seriously consider immediate shutdown of the system so as not to endanger the user data.

User Errors. Every once in a while, a user delete a file or overwrites data and then realizes that he needs it back. In some operating systems, "undelete" facilities or version numbering is enough to protect him, if he notices his mistake quickly enough. Sometimes he doesn't notice the error for a long time, or deletes all of the versions, or expunges them and then wants the data back.

If there is no backup system at all, the users data is just plain lost. If there is a perfect backup system, he quickly is able to recover from his mistake. If there is a poor back up system, his data may be recovered in a corrupted form or with incorrect permission set on it.

There have been cases where back up systems returned data files to be publically writeable and obvious problems have ensued from it. Perhaps as seriously, there are sites that have stored all of the back up data in a publically readable form, including the data that was protected by the individual user.

System Staff Errors. Just as users make mistakes, staff members do as well. In doing so, they may damage user files, system files or both. Unless there is a copy of the current system files, the staff must restore the system files from the original distribution and then rebuild all of the site specific changes. This is an error prone process and often the site specific changes including removing unwanted debugging features that pose security vulnerabilities.

Hardware/Software Failures. Hardware occassionally fails. If the only copy of the data is on a disk that has become unreadable it is lost. Software occasionally fails. Given a serious enough error, it can make a disk unreadable.

Security Incidents. In this document, our main concern is with security incidents. In determining what happen and correcting it, backups are essential.

Basically, one would like to return every file to the state before the incident except for those that are being modified to prevent future incidents. Of course, to do this, one needs a copy to restore from. Naively, one would think that using that modification date would allow us to tell which files need to be updated. This is of course not the case. The clever attack will modify the system clock and/or the timestamps on files to prevent this.

In many attacks, at one the following types of files are modified.

- The system binary that controls logging in.
- The system authorization file lists the users and their privileges.
- The system binary that controls one or more daemons.
- The accounting and auditing files.
- User's startup files and permission files.
- The system directory walking binary.

Now that we understand why we need back ups in order to recover

3.2.2 How to form a Back Up Strategy that Works

There are a few basic rules that provide for a good backup strategy.

- Every file that one cares about must be included.
- The copies must be in non-volitile form. While having two copies of each file, one on each of two separate disk drives is good for protection from simple hardware failures, it is not defense from an intelligent attacker that will modify both copies, of from a clever system staffer who saves time by modifying them both at once.

- Long cycles. It may take weeks or months to notice a mistake. A system that reuses the same tape every week will have destroyed the data before the error is noticed.
- Separate tapes. Overwriting the existing backup before having the new one completed is an accident waiting to happen.
- Verified backups. It is necessary to make sure that one can read the tapes back in. One site with a programming bug in its back up utility had a store room filled with unreadable tapes!

3.3 Forming a Plan

While the first major section (avoidance) contained a lot of standard solutions to standard problems, planning requires a great deal more thought and consideration. A great deal of this is list making.

Calls Lists. If there a system staffer suspects security incident is happening right now, who he should call?

And if he gets no answer on that line?

What if the people are the call list are no longer employees or have long since died?

What if it Christmas Day or Sunday morning?

Time–Distance. How long will it take for the people who are called to arrive?

What should be done until they get there?

This a user notices. If a user notices something odd, who should he tell?

How does he know this?

Threats and Tips. What should your staffers do if they receive a threat or a tip-off about a breakin?

Press. What should a system staffer do when he receives a call from the press asking about an incident that he, himself doesn't know about?

What about when there is a real incident underway?

Shutting Down. Under what circumstances should the center be shutdown or removed from the net?

Who can make this decision?

When should service be restored?

Prosecution. Under what circumstances do you plan to prosecute?

Timestamps. How can you tell that the timestamps have been altered?

What should you do about it?

Would running NTP (the network time protocal) help?

Informing the Users. What do you tell the users about all this?

List Logistics. How often to you update the incident plan?

How does you system staff learn about it?

3.4 Tools to have on hand

File Differencing Tools
Netwatcher
Spying tools
Backup Tapes
Blanks Tapes
Notebooks

3.5 Sample Scenarios to Work on in Groups

In order to understand what goal priorities you have for you center and as a general exercise in planning, let us consider a number of sample problems. Each of these is a simplified version of a real incident. What would be appropriate to do if a similar thing happenned at your center? Each new paragraph indicates new information that is received later.

- A system programmer notices that at midnight each night, someone makes 25 attempts to guess a username–password combination

  Two weeks later, he reports that each night it is the same username–password combination.

- A system programmer gets a call reporting that a major underground cracker newsletter is being distributed from the administrative machine at his center to five thousand sites in the US and Western Europe.

  Eight weeks later, the authorities call to inform you the information in one of these newsletters was used to disable "911" in a major city for five hours.

- A user calls in to report that he can't login to his account at 3 in the morning on a Saturday. The system staffer can't login either. After rebooting to single user mode, he finds that password file is empty.

  By Monday morning, your staff determines that a number of privileged file transfer took place between this machine and a local university.

  Tuesday morning a copy of the deleted password file is found on the university machine along with password files for a dozen other machines.

  A week later you find that your system initialization files had been altered in a hostile fashion.

- You receive a call saying that breakin to a government lab occurred from one of your center's machines. You are requested to provide accounting files to help trackdown the attacker.

  A week later you are given a list of machines at your site that have been broken into.

- A user reports that the last login time/place on his account aren't his.

  Two weeks later you find that your username space isn't unique and that unauthenticated logins are allowed between machines based entirely on username.

- A guest account is suddenly using four CPU hours per day when before it had just been used for mail reading.

  You find that the extra CPU time has been going into password cracking.

  You find that the password file isn't one from your center.

  You determine which center it is from.

- You hear reports of computer virus that paints trains on CRT's.

  You login to a machine at your center and find such a train on your screen.

  You look in the log and find not notation of such a feature being added.

  You notice that five attempts were made to install it within an hour of each before the current one.

  Three days later you learn that it was put up by a system administrator locally who had heard nothing about the virus scare or about your asking about it.

- You notice that your machine has been broken into.

  You find that nothing is damaged.

  A high school student calls up and apologizes for doing it.

- An entire disk partition of data is deleted. Mail is bouncing bouncing because the mail utilities was on that partition.

  When you restore the partition, you find that a number of system binaries have been changed. You also notice that the system date is wrong. Off by 1900 years.

- A reporter calls up asking about the breakin at your center. You haven't heard of any such breakin.

  Three days later you learn that there was a breakin. The center director had his wife's name as a password.

- A change in system binaries is detected.

  The day that it is corrected they again are changed.

  This repeats itself for some weeks.

4 Incident Handling

The difficulty of handling an incident is determined by several factors. These include the level of preparation, the sensitivity of the data, and the relative expertise levels of the attacker(s) and the defender(s). Hopefully, preliminary work in terms of gathering tools, having notification lists, policies and most importantly backup tapes, will make the actual handling much easier.

This section is divided into three parts. The first of these deal with general principles. The second presents some particular (simple) techniques that have proven useful in the past. Finally, the third section presents a description of a simulation exercise based a set of real attacks.

4.1 Basic Hints

There are a number of basic issues to understand when handling a computer incident. Most of these issues are present in handling most of these issues and techniques are relevant in a wide variety of unusual and emergency situations.

4.1.1 Panic Level

It is critical to determine how much panic is appropriate. In many cases, a problem is not noticed until well after it has occurred and another hour or day will not make a difference.

4.1.2 Call Logs and Time Lines

All (or almost all) bad situations eventually come to an end. At that point, and perhaps at earlier points, a list of actions and especially communications is needed to figure out what happened.

4.1.3 Accountability and Authority

During an incident it is important to remind people what decisions they are empowered to make and what types of decisions that they are not. Even when this is explicitly discussed and formulated in a contingency plan, people have a tendency to exceed their authorities when they are convinced that they know what *should* be done.

4.1.4 Audit Logs

Audit logs need to be copied to a safe place as quickly as possible. It is often the case that an attacker returns to a computer to destroy evidence that he had previously forgotten about.

4.1.5 Timestamps

The second most powerful tool (second only to backup tapes) in an incident handlers arsenal is timestamps. When in doubt as to what to do, try to understand the sequencing of the events. This is especially true when some of the actions will change the value on the system clock.

4.2 Basic Techniques

There are five basic sets of techniques for understanding what has happened.

4.2.1 Differencing

Differencing is that act of comparing the state of a part of the computer system to the state that it was in previously. In some cases we have compared every executable system file with the corresponding file on the original distribution tape to find what files the attacker may have modified. Checksums are often used to decrease the cost of differencing. Sometimes people look only for differences in the protection modes of the files.

4.2.2 Finding

Finding is generally cheaper than differencing. Finding is the act of looking at a part of a computer system for files that have been modified during a particular time or have some other interesting property.

4.2.3 Snooping

Snooping is the act of placing monitors on a system to report the future actions of an attacker. Often a scripting version of the command line interpreter is used or a line printer or PC is spliced in to the incoming serial line.

4.2.4 Tracking

Tracking is the use of system logs and other audit trails to try to determine what an attacker has done. It is particularly useful in determining what other machines might be involved in an incident.

4.2.5 Psychology

A wide range of non-technical approaches have been employed over the years with an even wider range of results. Among these approaches have been leaving messages for the attacker to find, starting talk links, calling local high school teachers, etc.

4.3 Prosecution

Prosecution has historically been very difficult. Less than a year ago, the FBI advised me that it was essentially impossible to succeed in a prosecution. More recently, FBI agent Dave Icove, (`icove@dockmaster.cnsc.mil`, 703–640–1176) has assured me that the FBI will be taking a more active role in the prosecution of computer break-ins and has expressed interest in lending assistance to investigation where prosecution is appropriate.

4.4 Exercise

The bulk of this class hour is reserved for an incident handling simulation. A facility will be described. A consensus policy for incident handling will be agreed upon and then the simulation will begin.

During the simulation, the effects of the attackers actions and those of third parties will be described. The participants can choose actions and take measurements and will be informed of the results of those actions and measurements. In a sufficiently small working group that had several days, we would run a software simulation; but as many of the actions take hours (egà full system comparison to the original distribution), we will proceed verbal in the short version of this workshop.

5 Recovering From Disasters

Incident recovery is the final portion of the of the incident handling process. Like the other portions of incident handling, it is not particularly difficult but is sufficiently intricate to allow for many errors.

Telling everyone that is over. For a large incident, it is not unusual to have contacted people at a dozen or more sites. It is important to let everyone know that you are done and to be sure to give your colleagues the information that they need. It is also important that your staff knows that things are over so that they can return to normal work. Generally a lot of people need to thanked for the extra hours and effort that they have contributed.

Removing all Tools. Many of the tools that were installed and using during an incident need to removed from the system. Some will interfere with performance. Others are worth stealing by a clever attacker. Similarly a future attacker that gets a chance to look at the tools will know a lot about how you are going to track him. Often extra accounts are added for handling the incident. These need to be removed.

File and Service Restoration. Returning the file system to a "known good state" is often the most difficult part of recovery. This is especially true with long incidents.

Reporting Requirements. Often, especially if law enforcement agencies have become involved, a formal report will be required.

History. After everything is over, a final reconstruction of the events is appropriate. In this way, everyone on your staff is telling the same story.

Future Prevention. It is important to make sure that *all* of the vulnerabilities that were used in or created the incident are secured.

Just after an incident, it is likely to be a good time to create sensible policies where they have not existed in the past and to request extra equipment or staffing to increase security. Similarly, it is a logical time for someone else to demand stricter (nonsensical) policies to promote security.

A Micro Computers

While the bulk of this book and class has concerned multi-user computers on networks, micro computers are also worth some attentions.

Basically there are four issues that cause concern.

Shared Disks. In many settings, micro computers are shared among many users. Even if each user brings his own data, often the system programs are shared on communal hard-disk, network or library or floppies. This means that a single error can damage the work of many people. Such errors

might include destruction of a system program, intentional or accidental modification of a system program or entry of a virus.

To combat this, systematic checking or reinstallation of software from a known protected source is recommended. In most shared facilities, refreshing the network, hard-disk or floppy-library weekly should be considered. Shared floppies should be write protected and the original copies of programs should be kept under lock and key and used only to make new copies.

Trusted server the provide read only access to the system files have been successfully used in some universities. It is absolute critical that these machines be used only as servers.

Viruses. A number of computer viruses have been found for micro-computers. Many experts consider this problem to be practically solved for Macintoshes an soon to be solved for IBM-style PC's.

Two basic types of anti-viral software are generally available. The first type is installed into the operating and watches for virus's trying to infect a machine. Examples of this on the Mac include Semantic's SAM (Part 1), Don Brown's vaccine and Chris Johnson's Gate Keeper.

The second type of anti-viral software scans the disk to detect and correct infected programs. On the Mac, SAM (Part 2), H. G. C. Software's Virex, and John Norstab's Disinfinct are commonly used disk scanners.

On the PC type of machines we find three types of virus. The first of these is a boot sector virus that alters the machine language start up code found on the diskette. The second infects the `command.com` startup file and the third alters the `exe` (machine language executable files).

Flu Shot Plus by Ross Greenberg is an example of a program to deal with `command.com` & some `exe` virus. Novirus and cooperatively built by Yale, Alemeda and Merit is one of the boot track repair systems.

There are a number of electronic discussion groups that deal with computer virus. On BITNET (and forwarded to other networks), `virus-l` supports discussion about PC and Mac virus, while `valert` is used to announce the discovery of new ones. Compuserve's `macpro` serves as a forum to discuss Macintosh viruses.

Network. The third is issue is the placement of single user computers on networks. Since there is little or no authentication on (or of) these machines, care must be taken to not place sensitive files upon them in such a configuration.

Reliability. Finally there is a reliability issue. Most single user computers were never designed for life and time critical applications. Before using such a computer in such an application, expert advise should be sought.

In the use of single user computers, there are some basic issues that need be considered and some simple advice that should be given.

In the advice column, there are a few basic points.

1. Where practical, each user should have his own system disks and hence be partially insulated from potential mistakes.

2. When people are sharing disks have an explicit check out policy logging the users of each disk. Be sure to set the write-protect them and teach the users how to write protect there own system disks. (Most PC programs are sold on write-protected disks, this is not true of most Macintosh programs.

3. Keep a back up copy of all system programs and system programs to allow for easy restoration of the system.

4. Write lock originals and keep them under lock and key for emergency use only.

5. Have an explicit policy and teach users about software theft and software ethics.

6. Teach users to back up their data. Just as with large computers, the only real defense from disaster is redundancy.

Even when the computer center is not providing the machines themselves, it should generally help to teach users about backups, write protection, software ethics and related issues. Most PC users do not realize that they are their own system managers and must take the responsibility of care for their systems or risk the consequences.

B VMS Script

This script is courtesy of Kevin Oberman of Lawrence Livermore National Labs. It is used on DEC VMS systems to close a number of the standard created by the normal installation of DECNET. Rather than typing this in by hand, please request one by electronic mail. This DCL script is provided for reference purposes only and is not guaranteed or warranted in any way.

```
$ Type SYS$INPUT

This procedure changes the password for the default DECnet account and
sets up a new account for FAL activity. It prevents unauthorized users
from making use of the default DECnet account for any purpose except
file transfer.

This procedure assumes a default DECnet account named DECNET using a
directory on SYS$SYSROOT. If this is not the case on this system, do
not proceed!  It will use UIC [375,375]. If this UIC is already in
use, do not continue.

$ Read/End=Cleanup/Prompt="Continue [N]: " SYS$COMMAND OK
$ If .NOT. OK Then Exit
$ Say := "Write SYS$OUTPUT"
$ Current_Default = F$Environment("DEFAULT")
$ Has_Privs = F$Priv("CMKRNL,OPER,SYSPRV")
$ If Has_Privs Then GoTo Privs_OK
$ Say "This procedure requires CMKRNL, OPER, and SYSPRV."
$ Exit
$Privs_OK:
$ On Control_Y Then GoTo Cleanup
$ On Error Then GoTo Cleanup
$ Set Terminal/NoEcho
$ Read/End=Cleanup/Prompt="Please enter new default DECnet password: " -
 SYS$Command DN_Password
$ Say " "
$ If F$Length(DN_Password) .GT. 7 Then GoTo DN_Password_OK
$ Say "Minimum password length is 8 characters"
$ GoTo Privs_OK
$DN_Password_OK:
$ Read/End=Cleanup/Prompt="Enter new FAL password: " SYS$COMMAND FAL_Password
$ Say " "
$ If F$Length(FAL_Password) .GT. 7 Then GoTo FAL_Password_OK
$ Say "Minimum password length is 8 characters"
$ GoTo DN_Password_OK
$FAL_Password_OK:
$ Set Terminal/Echo
$ Type SYS$INPUT

The FAL account requires a disk quota. This quota should be large
enough to accomodate the the files typically loaded into this account.
Should the qouta be exhausted, the system will fail to perform default
DECnet file transfers.
```

It is also advisable to clear old files from the directory on a daily basis.

```
$ If .NOT. F$GetSYI("CLUSTER_MEMBER") Then GoTo Not_Cluster
$ Say "This system is a cluster member.
$ Read/Prom="Has this procedure already been run on another cluster member: "-
  SYS$INPUT Cluster
$ If Cluster Then GoTo No_Create
$Not_Cluster:
$ Read/End=Cleanup -
 /Prompt="Disk quota for FAL account (0 if quotas not enabled): " -
  SYS$COMMAND Quota
$ If F$Type(Quota) .EQS. "INTEGER" Then GoTo Set_Quota
$ Say "Diskquota must be an integer"
$ GoTo FAL_Password_OK
$Set_Quota:
$ Say "Setting up new FAL account."
$ Set Default SYS$SYSTEM
$ Set NoOn
$ UAF := "$Authorize"
$ UAF Copy DECNET FAL/Password='FAL_Password'/UIC=[375,375]/Directory=[FAL]
$ Create/Directory SYS$SYSROOT:[FAL]/Owner=[FAL]
$No_Create:
$ NCP := "$NCP"
$ NCP Define Object FAL USER FAL Password 'FAL_Password'
$ NCP Set Object FAL USER FAL Password 'FAL_Password'
$ If (Quota .eq. 0) .OR. Cluster Then GoTo NO_QUOTA
$ Say "Entering disk quota for FAL account.
$ Set Default SYS$SYSTEM
$ PID = F$GetJPI("","PID")
$ Open/Write Quota SET_QUOTA'PID'.COM
$ Write Quota "$ Run SYS$SYSTEM:DISKQUOTA"
$ Write Quota "Add FAL/Perm=''Quota'"
$ Close Quota
$ @SET_QUOTA'PID'
$ Delete SET_QUOTA'PID'.COM;
$No_Quota:
$ Say "Resetting default DECNET account password"
$ NCP Define Executor Nonpriv Password 'DN_Password'
$ NCP Set Executor Nonpriv Password 'DN_Password'
$ UAF Modify DECNET/Password='DN_Password'
$Cleanup:
$ Set Default 'Current_Default'
$ Set Terminal/Echo
$ Exit
```

C Highly Sensitive Environments

An computing environment should be considered highly sensitive when it is potentially profitable to covert the data or when great inconvenience and losses could result from errors produced there. In particular, you should consider you site sensitive if any of the following conditions apply:

1. You process data that the government considers sensitive.
2. You process financial transactions such that a single transaction can exceed $25,000.00 or the total transactions exceed 2.5 Million dollars.
3. You process data whose time of release is tightly controlled and whose early release could give significant financial advantage.
4. Your function is life critical.
5. Your organization has enemies that have a history of "terrorism" or violent protests.
6. Your data contains trade secrete information that would be of direct value to a competitor.

Essentially money is more directly valuable than secrets and a "vilian" can potentially steal more from one successful attack on one financial institution than he will ever be able to get selling state secrets for decades. There is significant concern that the electrical utility companies and and bank conducting electronic funds transfer will be targets of terrorists in thee next decade.

For centers the must support sensitive processing it is *strongly* advised to *completely* separate the facilities for processing this data from those facilities used to process ordinary data and to allow absolutely no connection from the sensitive processing systems to the outside world. There is *No* substitute for physical security and proper separation will require an attacker to compromise physical security in order to penetrate the system. Techniques for coping with the remaining "insider threat" are beyond the scope of this tutorial.

In analysis of computing in sensitive environments, there are two different security goals. The first is that of protecting the system. All of the advice in this booklet should be considered as a first step towards that goal. The second goal is the protection of job or "Technical Compliance." This is is the goal of showing that all of the regulations have been followed and that protecting the system has been done with "due diligence."

It is important to realize that these two security goals are separate and potentially conflicting. It may be necessary to work towards the latter the goal and that is often more a legal and bookkeeping question than a technical one. It is also beyond the scope of this work.

D Handling the Press

Often media inquiries can absorb more time than all of the others issues in incident handling combined. It is important to understand this and to use your public affairs office if it exists. In the excitement, people, especially those who are not experience speakers will often forget that they are not empowered to speak for the center and that nothing is ever really said, "Off the record."

D.1 Spin Control

The phrase "Spin Control" was first used in political circles. It refers to altering the perceptions about an incident rather than the delaying with the facts of the incident themselves. Consider the two statements.

1. *To keep our machines safe, we decided to disconnect them from the network.*
2. *We were forced to shut down our network connections to prevent damage to our machines.*

I have found that the giving the press a state like the former tends to produce a laudatory piece about one's staff while a statement like the latter, produces an embarrassing piece. The two statements are of course essentially identical.

Your public affairs group is probably familiar with these issues and can help you form press statements

D.2 Time Control

With a sufficiently large incident, the media attention can absorb almost unbounded amounts of time. The press will often call employees at home. It is important the staff that are solving a problem understand that the solving the incident is more important that dealing with the press. At the very least insist that all press representatives go through the public affairs often so that the standard questions can be easily and time-efficiently be answered.

D.3 Hero Making

The press likes to find outstanding heroes and villains. As a result, the media will tend to make one of your staff members into a hero if at all possible from them to do so. It is more likely than not that the Hero will not be the person who has worked the hardest or the longest.

D.4 Discouraging or Encouraging a Next Incident

The attention that an incident receives greatly affect the likelihood of future incidents at that particular site. It probably also influences the decision process or potential future crackers in the community at large. Claiming that your site is invulnerable is an invitation to a future incident. Giving the media step by step instructions on how to break in to a computer is also not a wonderful idea.

I (personally) suggest stressing the hard work of your staff and the inconvenience to the legitimate users and staff members. To the extent practical portray the cracker as inconsiderate and immature and try to avoid making him seem brilliant at one extreme or the attack seem very simple at the other.

D.5 Prosecution

If you considering prosecution, you need to consult with your legal counsel and law enforcement official for advise on press handling.

D.6 No Comment

One common strategy for avoiding (or at least bounding) time loss with the press is to simply decline to comment on the situation at all. IF you are going to adopt this approach, your public affairs office can advise you on techniques to use. It is important to tell everyone who is involved in the incident that they should not discuss the situation; otherwise people will leak things accidently. Also, without correct information from your center, the press may print many inaccurate things that represent their best guesses.

D.7 Honesty

I recommend against trying to mislead the press. It is hard to keep a secret forever and when and if the press finds that you have lied to them, the negative coverage that you may receive will probably far exceed the scope of the actual incident.

E Object Code Protection

To keep object code safe from human attackers and virus, a variety of techniques may be employed.

Checksums. Saving the checksums of each of the system files in a protected area an periodically comparing the stored checksum with those computed from the file's current contents is a common and moderately effective way to detect the alteration of system files.

Source Comparisons. Rather than just using a checksum the complete files may be compared against a known set of sources. This requires a greater storage commitment.

File Properties. Rather the computing a checksum, some facility store certain attributes of files. Among these are the length and location on the physical disk. While these characteristics are easy to preserve, the naive attacker may not know that they are important.

Read-Only Devices. Where practical, the system sources should be stored on a device that does not permit writing. On many system disk partitions may be mounted as "Read-Only."

Dates. On many systems the last modification date of each file is stored and recent modifications of system files are reported to the system administrator.

Refresh. Some system automatically re-install system software onto there machines on a regular basis. Users of TRACK often do this daily to assure that systems have not be corrupted.

F The Joy of Broadcast

The majority of the local area nets (LAN's) use a system called broadcast. It is somewhat like screaming in a crowded room. Each person tends to try to ignore messages that weren't meant for them.

In this type of environment, eaves-dropping is undetectable. Often passwords are sent unencrypted between machines. Such passwords are fair game to an attacker.

Various cryptographic solutions including digital signature and one time keys have been used to combat this problem. Kerberos, developed at the MIT Athena project is available without cost and presents one of the few promising potential solutions to the broadcast problem.

G Guest Accounts

The computer center guest policy is among the most hotly debated topics at many computer centers. From a security standpoint, it should be obvious that an attacker who has access to a guest account can break into a computer facility more easily.

G.1 Attack Difficulty Ratios

Basically it is a factor of ten easier to break into a machine where you can easily get as far as a login prompt that one where you can't. Being able to reach the machine through a standard networking discipline and open connections to the daemons is worth another order of magnitude. Access to a machine that is run by the same group is worth another factor of three and access to a machine on the same LAN would grant a factor of three beyond that. Having a guest account on the target machine makes the attack still another order of magnitude easier.

Essentially, having a guest account on the target simplifies an attack at least a thousand fold from having to start cold.

G.2 Individual Sponsors

I strongly suggest requiring each guest to have an individual staff sponsor who takes responsibility for the actions of his guest.

G.3 The No Guest Policy

In centers that prohibit guests, staff members often share their passwords with their guests. Since these are generally privileged accounts, this is a significant danger.

H Orange Book

You have doubtlessly by now heard of the "Orange Book" and perhaps of the whole rainbow series.

Much of the "Orange Book" discusses discretionary and mandatory protection mechanism and security labeling. Another section deals with "covert channels" for data to leak out. While most of these issues are not important in a university, the ideas of protecting password files (even when encrypted), individual accountability of users and password aging are worth implementing in an unclassified environment.

I Acknowledgements

– Help of a lot of people. – copies were sent out to 48 people for peer review

Jerry Carlin. For examples from his training course.

Joe Carlson. For help with spelling and grammar.

James Ellis. For help with organization.

Alan Fedeli.

Paul Holbrook. For help getting this document distributed.

David Muir. For help with spelling, grammar and comments about computer games.

Kevin Oberman. For help with VMS issues, spelling and grammar.

Mike Odawa. For help with the microcomputers section.