Security Breaches:
Five Recent Incidents at Columbia University

Fuat Baran <fuat@columbia.edu>
Howard Kaye <howie@columbia.edu>
Margarita Suarez <marg@columbia.edu>

Columbia University
Center for Computing Activities
New York, NY 10025

Abstract

During a two-month period (February through March, 1990) Columbia University was involved in five break-in incidents. This paper provides a detailed account of each incident as well as what steps we took, both short-term and long-term, to reduce the likelihood of future incidents.

1. Site description

Columbia's computing environment is networked and distributed. Many departments have their own computing facilities — some have only a few workstations or small minicomputers, and others (like the Computer Science department) have dozens of workstations and several mainframes/super-minis. Departmental facilities are maintained in varying degrees by the departments that own them; the Computer Center provides the central network and tries to notify departments of possible problems. This is complicated by the fact that many departmental facilities are virtually unadministered.

The campus is interconnected via an Ethernet backbone, and through a central ROLM CBX. The phone switch allows people to connect to the campus terminal servers via phone lines, providing virtually unlimited on-campus connections. There are also 64 inbound modems, which can call any host connected to the switch. Unfortunately, the CBX ‘‘hides’’ the origin of inbound modem connections, making them difficult to trace.

Columbia has a variety of external network connections: Internet (via NYSERNet), BITNET, and CCNET (a DECNET network connecting several universities). While all of this connectivity is convenient, it also makes the campus an easy target for break-ins from outside the university.

The computer center (CUCCA) runs the network as well as the central academic and administrative computing environments. We have a cluster of three central Sun-4/280 server machines which are primarily used by the student body. We also run two Encore Multimaxes as staff machines. Most of the users of these staff machines are novice computer users, who use the systems primarily for electronic mail (EMAIL). These five machines make up our central UNIX systems. They are reached via two sets of terminal servers: the Suns are logged into via two cisco terminal servers, and the Encores via six Annex terminal servers. All of these systems mount each other's file systems via NFS. Sun users' home machines are randomly assigned within the cluster. The systems have a total user population of about 3000 users: 1500 students, 250 instructors, 1000 staff, and 200 guests. These academic systems are not used for anything sensitive, that is, there is no proprietary or confidential administrative information on them.

CUCCA also runs several IBM VM systems and a few PS2/Macintosh microcomputer labs. For internal staff use, we have several workstations of various types (Vaxstation II's, NeXT's, IBM RT's, Sun-4/110's) in our offices. The UNIX systems group has five members who support these CUCCA systems. There is also a User Services group of about twenty full-time consultants, as well as around 40 part-time student consultants who support the user community.

2. Incident Reports

During a two-month period (February through March, 1990) Columbia University was involved in five break-in incidents on various Computer Center systems. According to to the Computer Emergency Response Team (CERT) and *The New York Times*,[1] many other Internet sites were apparently experiencing similar problems during this time period.

2.1. Case #1 16-Feb-90

On Friday, February 16, 1990, at around 5 P.M. a member of our UNIX systems group noticed that one of our Multimaxes felt uncharacteristically sluggish for a Friday evening. A quick look at all running processes to try and identify what was using so much of the system revealed a program called *program* running as user *user1*.[2] Since *user1* was a guest account on our system and guests usually use their accounts to read mail, netnews, etc., and not for running CPU intensive programs, we decided to investigate.

A look at *user1*'s *ksh* history file to see what *program* was and where it was run from revealed unusual activity.[3] *user1* had connected to a directory named ‘‘.. ’’ (dot dot space space) and had run *program* from there. This directory contained a copy of our */etc/passwd* file called *funlist*, and a list of 324 words called *list* — containing a lot of first names, names of famous people and teams, and

[1]‘‘Computer System Intruder Plucks Passwords and Avoids Detection,’’ March 19, 1990; ‘‘Caller Says He Broke Computers' Barriers to Taunt the Experts,’’ March 21, 1990.

[2]Specifics such as hostnames, usernames, passwords, etc. will not be mentioned here, though *program* was the actual program name.

[3]The computer center's policy document states: ‘‘When necessary, the computing center full-time staff may access users' files. This will be done only for the purposes of maintenance and security of the system.’’

miscellaneous other words.[4] In this directory we found another copy of the executable file *program*, though the source code was not there. After examining the executable using tools such as *strings* and *nm*, we concluded that *program* was a ‘‘password cracker’’ (or perhaps a ‘‘password checker’’ — depending on your point of view).

```
$Header: pwchkr.c,v 1.2 85/11/30 22:42:07 richl Exp $
shell: %s
%s -- Problem: null passwd:
%s -- Trying "%s" on %s
Problem: GUESSED:
shell: %s passwd: %s
/etc/passwd
```

Figure 2-1: Some of the strings in *program*

```
_chkpw                          _checkgecos
_uandltry                       _checkcase
_try                            _chknulls
_setpwent                       _users
_endpwent                       _chkwords
_pwskip                         _PASSWD
_getpwent                       _EMPTY
_reverse                        _line
_verbose                        _passwd
_singles                        _Curpw
_backwards                      _Wordlist
```

Figure 2-2: Some of the symbols in *program*

Further inspection of *user1*’s *ksh* history file revealed that they had been reading our *crontab* file and the scripts invoked by *cron*. Also, we saw attempts by *user1* to connect to other users’ temporary directories in */tmp*, and an attempt to propagate the *program* executable to other machines via NFS.

At this point we decided to turn off *user1*’s account by setting the login shell to */etc/turnedoff*.[5] We contacted our User Services group to inform them of our actions and found out that *user1*’s account belonged to a Barnard College staff member who was currently on maternity leave. Looking back through our accounting records we determined that *user1*’s account had been in use regularly starting on February 13, and that during this time it had accumulated a CPU charge for over 400 CPU minutes of usage, all running *program*. Login records showed that all logins to our machine were via one of our cisco Systems terminal servers.

Strangely, *user1*’s password was not in the list of passwords that we found. We later discovered that a few months prior to this she had forgotten her password, and contrary to our policy, a staff member

---

[4] a complete list of these words, plus other words found elsewhere, has been compiled by CERT

[5] */etc/turnedoff* informs a user at login that their account has been turned off for usage restriction violations, and gives them a phone number to contact to discuss reinstatement.

had reset her password to be her username, with the understanding that she would immediately change it to something else.[6] She neglected to change it, and according to our logs her password had remained the same until the break-in, when on February 14 and 15 someone, presumably the ‘‘cracker,’’ changed it a couple of times.

After turning off *user1*, we examined the current day’s logs, and we noticed a brief *su* process being charged to *user1* with a timestamp *after* the ID was turned off. From the tty name associated with the charge record we saw that this attempt was made from *user2*’s account, which had just been logged into, also from one of the cisco terminal servers. This was strange for two reasons: 1) *user2* was a guest account belonging to a former staff member whom we knew, and 2) the login was from a cisco terminal server rather than an Annex terminal server — legitimate users of our systems predominantly use the Annex terminal servers to get to this staff machine, while the cisco’s are used by students to get to the student machines. A quick call to *user2*’s Wall Street office confirmed that he wasn’t using his ID at the time. It turns out that *user2*’s password was a short, personal name which was in *list*.

We decided to watch this session for a while by tailing the shell history file and saw *user2* trying to access the files in *user1*’s ‘‘.. ’’ directory, as well as wandering around trying to get into other guests’ home directories. He then tried to send mail to an off-campus site. We intercepted this message when an incorrect address caused it to bounce, and we removed the bounce notice. In this message, which was signed with a pseudonym, *user2* informed a friend that this ID was being used without the owner’s permission.

We traced the connection from the cisco terminal server to a 2400 baud inbound modem on our university CBX attached to a trunk line to New York Telephone. Tracing it further would have required the cooperation of our Telecommunications department as well as New York telephone. We have since found out that getting NYTel to do a phone trace would be extremely difficult (if at all possible) and time consuming due to New York State laws restricting caller identification.

To determine which other accounts besides *user1* and *user2* may have fallen into the hands of the ‘‘cracker’’ we decided to run the word list against our rather large password file. We split the task up and ran a total of 12 password crackers simultaneously on several Sun-4/280’s and Encore Multimaxes. We completed this by 3 A.M. and identified 16 accounts which had passwords in the list. We modified the shells on all these ID’s to */etc/badpasswd*, which informed the users at login to go to our Business Office to get their ID’s turned back on after verification of their identity. We then turned off *user2* as well.

According to our weekend logs, there were over 25 failed attempts to login as *user1* and *user2*, which leads us to suspect that information on break-ins was being widely disseminated — it did not seem

[6] Our */bin/passwd* won’t let users pick ‘‘dumb’’ passwords, but another utility allows staff to set arbitrary passwords.

like one person would stubbornly try the ID so many times after getting a message that said the ID was turned off. At one point on Saturday someone claiming to be *user2* even called up a consultant in one of our terminal rooms asking for information on access to ‘‘anonymous email.’’

Details of this break-in were provided to CERT as they became known to us.

2.2. Case #2 18-Feb-90

During the weekend following case #1 the UNIX Systems group performed periodic spotchecks to see if there was any other unusual activity on the systems. On Sunday, February 18, 1990, we noticed a student, *user3*, on one of the instructional Suns running some CPU intensive processes. Also, these programs were being run with usernames as arguments — in particular, familiar staff names (which are at the top of the passwd file). A quick check of this user’s shell history and directory revealed a password cracker which used a 215 word list.

By looking at this user’s other processes, we suspected that it was the account owner himself that was running the password cracker (he was doing his homework while running the cracker in the background). We contacted him by phone, at which time he admitted to running the password cracker. His account was invalidated, and after a meeting with the Manager of User Services his case was referred to his dean. Though the student did not think he had done anything seriously wrong, his dean held a disciplinary hearing, and, on our recommendation, placed the student on probation. The student’s account has since been revalidated.

At the meeting with the Manager of User Services, *user3* said that he had gotten the idea to break passwords partly through a microcomputer bboard message, which was written by someone whom he didn’t know personally who was using a pseudonym. He had also corresponded about password cracking with a friend at a university in the United Kingdom.

2.3. Case #3 20-Feb-90

On Tuesday, February 20, 1990 (two days after case #2), another guest account, *user4*, belonging to a former university employee was using an unusual amount of CPU cycles running a program called *square*. The home host for *user4* was the same machine that *user1* and *user2* were on, though *user4* was logged in on one of the student Suns and accessing her files remotely over NFS. Due to the events of the past week, we had been on the lookout for such processes and immediately became suspicious. Looking at *user4*’s shell history file revealed unusual activity. Her home directory contained many encrypted files, and she was decrypting them one at a time and using them as input into the program *square*. The only plaintext file was the one currently in use. We examined it and it turned out to be a passwd file for some foreign system — the *gecos* field of some users had 812 area code phone numbers (Indiana). We later confirmed with Indiana University that the file was indeed a copy of the passwd file to a system there which had recently been broken into from a site in Texas.

```
_pw_name
_crypt
_fgetpwent
_pw_passwd
_input_file
_strcmp
Enter number:
%d squared equals %d
Enter your guess:
Correct is %s
```

Figure 2-3: Some of the strings in *square*

We also found a plaintext C source file called *getch.c* which contained the sources to *square*. This program prompted for a magic number (32768) and when it got anything else it just printed out the square of the number and exited. When given the magic number, it opened a file (with the name hardcoded, which is probably why the source file was in plaintext, since it had to be recompiled for each passwd file), did an *fgetpwent* on the descriptor, prompted for a password guess and encrypted it for each user in the passwd file trying to see if it was the password for any of the users.

We decided to turn off *user4* (by setting the login shell to */etc/turnedoff*) and went through our logs and through the various files in *user4*'s directory. We saw recent activity on this account, which had previously been pretty much unused. The *.newsrc* file here revealed that the various *comp.bugs.**, *comp.dcom.**, *comp.protocols.**, and *comp.unix.** newsgroups, as well as *soc.culture.china*, *soc.women*, *alt.hackers*, *alt.cyberpunk*, *comp.virus*, and *misc.security*, were read. A few days after the ID was turned off, we found queued mail (deferred because the destination was temporarily unreachable) to someone in response to a *comp.dcom.telecom* article referring to a phone hackers' newsletter called *Phrack*.

While we were looking at *user4*'s previous activities, we found a copy of one of the data files and the program *square* in */usr/tmp*. This file was owned by *user5*, a student account from the past semester. Looking around some more we found a file called *.newshell* in *user4*'s home directory, which was a world executable copy of */bin/sh* that was setuid *user4*. *user5* had apparently used this to access the files in *user4*'s directory. Immediately after a failed attempt to login to *user4*, we saw a login to *user5*, so we suspected that the same person had broken into both accounts. The shell history file shows *user5* connecting to *user4*'s directory and running *.newshell*. He also *grep*'ed for *user4* in */etc/passwd* and saw that the shell had been changed to */etc/turnedoff*. We subsequently turned off *user5* as well. To date, *user4* has not gotten in touch with us about her turned-off account, and *user5*'s has since expired.

As always we informed CERT about all that occurred. The encrypted files in *user4*'s directory had short names that sounded like possible hostnames, so with the help of *nslookup* we tried to determine the fully qualified names and made a list of potential sites. We sent mail to people in charge of those systems warning them that their passwd files might have been stolen to be processed by a password cracker.

One of the system administrators we contacted suggested we use *Crypt Breakers' Workbench* (CBW) to decrypt the files we found. We compiled CBW, but did not have the time to learn how to use it properly.

2.4. Case #4 22-Mar-90

On Thursday, March 22, 1990 (after a quiet month) we received electronic mail from a system administrator at the University of Virginia telling us that they were experiencing break-ins on their system originating from two of our machines. One was a Sun-4 student system, and the other was a cisco gateway. We contacted the system administrator by phone and obtained more information. It turns out they had been informed by another university of break-ins originating at Virginia which Virginia traced to a guest ID on their system. This ID had been logged into from our machines on several occasions, at various times, mostly early in the morning between 3 A.M. and 7 A.M. Virginia provided us with exact login times for the most recent occurrences, one of which turned out to be at 7 A.M., when exactly one student, *user6*, was logged in on the machine in question.

user6's directory did not contain anything interesting other than the fact that it lacked a shell history file. The user's shell profile did not rename the history file (via the *HISTFILE* environment variable), nor did it do anything fancy to automatically remove it at login or logout, which led us to suspect that it had been manually deleted. One other interesting item was that the logins were on a machine other than *user6*'s home machine and had occurred regularly over a two-week period of time, almost always from our cisco terminal servers. We noticed regular logins on the home machine that ended approximately one week before we were notified of the break-in. Login records also showed a few logins (on the non-home system) from Virginia at times which we later confirmed coincided with break-in times at Virginia.

It is interesting to note that the particular machine at Virginia that got broken into had a name that was also the name of one of the encrypted files we had found in *user4*'s directory (see case #4 above). We mentioned this to the administrator at Virginia who said he had heard that his passwd file had been stolen a while ago.

We turned off *user6*'s account, and that afternoon there was a single login attempt made on it. A week later the student who owned the account got in touch with us in response to the message he got when he attempted to login. We set up a meeting with him and the Manager of User Services. He told us that he had been on vacation in Texas during the past week (spring break) and that he didn't even know he could login to any machine other than his home system. He was apparently a novice computer user taking his first CS class. We asked him what his password was, and it turned out to be the name of a musical group. We believe it may have been guessed by a password cracker (though this particular name was not in any lists we had seen). We subsequently reported this incident along with the guessed password to CERT (and his ID was turned back on with a new password).

2.5. Case #5 26-Mar-90

Four days later, in the evening of Monday, March 26, 1990, a member of our User Services group noticed what appeared to be a runaway *mm* process.[7] This sometimes happens when a user gets logged out from a terminal server and the host end does not clean up properly. To determine if the user was still logged in he *finger*'ed the Annex terminal server that this person had logged in from. On the Annex he noticed a suspicious outbound *telnet* connection (to a foreign IP address), and by *finger*'ing the IP address found that it belonged to an Annex terminal server at an east coast university. The only session on that Annex (with our Annex as the source IP address) had an outbound *telnet* connection to a cisco gateway at another east coast organization. Checking on *that* gateway, again with *finger*, showed an outbound *telnet* connection to a host at the University of California, San Francisco (UCSF).

Annex ts at Columbia	→	Annex ts at *site1*	→	cisco gw at *site2*	→	host at UCSF

Figure 2-4: multi-hop connection to UCSF

Using the *tap* feature of our Annex terminal server, we were able to silently monitor all traffic on the line that this person was using, and logged it to an *xterm* log file. We watched as he logged into an account at UCSF, checked up on the logged in users (*w*), running processes (*ps -eafg*), mail for the user he logged in as (*cat*'ed */usr/spool/mail/user7*), another *w*, *cd*'ed to */etc* and got a directory listing. He then *cat*'ed the *hosts.equiv* file and *grep*'ed for ‘‘gw’’ (gateway) in the */etc/hosts* file. After another *w*, he *grep*'ed for various users in */etc/passwd*. After one final *w*, he disconnected.

We spent the next hour or so trying to get in touch with someone at UCSF in order to report the suspected break-in. We tried getting a phone number using the Domain Name System (DNS) *Start of Authority (SOA*) records as well as the *whois* database at *nic.ddn.mil*. Mostly we reached ‘‘phonemail’’ systems, since it was after business hours. We left a few messages and continued trying to reach someone in person. Eventually, we *finger*'ed all the sites in the *hosts.equiv* file we saw and found a phone number for an operations account. We called, and the night supervisor got in touch with the system manager for the broken-in machine, who called us back.

We wrote up a report of the incident and sent copies to CERT, UCSF, and the two intermediate sites that were involved. One site responded the next day, saying that they had just put a password on inbound connections to the gateway, while the other never responded. When we checked a month later, nothing had changed on their Annex.

---

[7] *Columbia-MM* is our locally written mail user agent.

We turned terminal server access restrictions back on (see section 4.8). They had been temporarily disabled due to hardware problems on the security host which the Annex would query for access restrictions.

3. Summary of the Five Cases

3.1. How They Got In

The illicit access to our systems occurred predominantly over dialup lines. Connections were made from our inbound modems to both Annex and cisco terminal servers, from which the crackers logged into our large UNIX systems (Encore Multimaxes, Sun-4/280's). In one case, both dialup lines and remote *telnet* sessions from another site were used (see case #4).

In all cases the accounts were broken into by password guessing. Despite our password selection policies, some people were able to get around the restrictions (either intentionally in the case of staff, or unintentionally — they forgot their password and a staff member changed it to something guessable, like a last name). In other cases, the dictionary of disallowed passwords we were using didn't have all the words in the cracker's list. No software security holes (e.g. *sendmail*, *ftpd*, etc.) were used. This may be because we try to fix known security holes as soon as we find out about them, or it may simply reflect on the type of break-ins that we saw. We suspect that once a user/password pair was cracked, this information was often being shared by a large number of people (see case #1).

Of the accounts that were broken into, two belonged to former staff members (now guests), one to a guest from an affiliated organization and two to students (one an ID from a previous semester and about to expire, the other that of a current student — though the illegal logins were on a non-home system). In all cases the accounts broken into were not being actively used.

The student in case #2 was using his own account in an attempt to crack passwords on our system (while simultaneously doing his homework).

3.2. What They Did

In three of the five cases we caught people running various forms of password crackers. None of the programs were especially sophisticated, and all the programs were written by the cracker or easily obtainable elsewhere. The program in case #1 seemed to be the best of the lot. It seemed to check the *gecos* field (to get first and last names, uppercased and lowercased), and used a list of passwords to try against our whole password file. The password cracker in case #2 tried a list of passwords on a single user (specified on the command line). The program in case #3 used a hardwired filename, accepted a guess from the cracker, and tried that on all password entries in the file.

In case #4 the account on our system was used to *telnet* out to remote sites. We found no evidence of any other activity (though history files were not available for the past two weeks' activity) and accounting data we had showed nothing other than *telnet*. In case #5 only our Annex terminal server was used. We were the initial network access point for a multi-hop break-in.

In case #2 we found mail files in which the concept of password cracking was being discussed with a friend. In cases #1 and #3 the cracker made use of electronic mail to either get in touch with a friend or try to obtain information heard on usenet.

Case #	**Ran Password Cracker**	**Guessed Local Password**	**Used Us As Intermediary**
case #1	our /etc/passwd	2 guests	no
case #2	our /etc/passwd	no	no
case #3	foreign /etc/passwd's	1 student, 1 guest	used our CPU
case #4	no	1 student	to/from Virginia
case #5	no	no	used Annex ts

Figure 3-5: Summary of cases

3.3. How They Were Detected

A major component in detection can be attributed to pure luck — being on the right system at the right time, noticing uncharacteristic sluggishness in system responses and investigating the cause, seeing suspicious processes being run by guests, etc. Accounting records, which aren't normally reviewed on a daily basis, also contained corroborating traces of break-ins once we consulted them. In the later cases, past problems had alerted us, so we were more suspicious and checked the systems more often.

Once we started investigating a particular user, we were able to find plenty of traces in the form of accounting records and files left around (‘‘hidden’’) in various places. None of the people in the above five cases had access to anything other than a couple of student or guest accounts (i.e. no *root* break-ins), so they were unable to cover their trails by modifying system logs and accounting records. The best that they did was to encrypt all incriminating files and delete source code, but even then, one or two files were left around in plaintext.

4. Lessons Learned

We have always been concerned about the security of our systems, especially since the university has such a large and transient user community. However, this rash of attacks on our systems has taught us some important lessons, and as a result, we have discovered ways to make our systems even more secure. We would like to share some of the methods we have implemented to try to prevent these attacks, as well the stop-gap measures we took when we noticed crackers on our systems.

4.1. Turn Off Abusers

CUCCA has a policy that disallows sharing of accounts. This means that if we detect that someone other than the owner of an account is using that account, we immediately turn it off. We have this restriction precisely so that we will be able to track down abusers and intruders. Our usual method of dealing with possible abusers is to set their shell to */etc/turnedoff*, change the mode of their home directory to 0, and kill their processes. The program */etc/turnedoff* displays a message telling the user who to contact to discuss getting the account turned on. That way we can verify the identity of the user in person.

4.2. Turn Off Potential Targets

In addition to turning off the obvious abusers, we found it necessary to turn off the potential targets — such as guest accounts with guessable passwords and all root IDs — and have the users come to us in person to have their passwords changed. We took advantage of this opportunity to reinforce security-consciousness in people who had privileged accounts.

4.3. Monitor System Activity

When we noticed the first breakins, there was a period when we found ourselves staying up late to monitor system activity. Some of this ‘‘paranoia’’ paid off: we were able to catch *user3* and *user4* because of their excessive CPU usage late at night, a time when few staff are usually logged in.[8] Unfortunately, there is a limit to how long a system administrator can stay awake, or type ‘‘ps -alxww,’’ for that matter. Luckily, our nights of vigil ended when we had apparently plugged enough holes, and the cracker activity subsided.

4.4. Eliminate Stupid Passwords at the Source

One precautionary measure we had taken was to modify the */bin/passwd* program to check new passwords against a dictionary[9] and strings in the *gecos* field, as well as for length and robustness. The BSD *passwd* program has a rather arbitrary set of criteria for determining ‘‘richness’’ of character composition, in addition to a well-known workaround that lets you set any password you desire, if you insist.

We created a new module called *esp*, for ‘‘eliminate stupid passwords.’’ This library is linked in with all password-changing user programs. By disallowing usernames, personal names in the *gecos* field, and dictionary words as passwords, we ‘‘force’’ people to choose better passwords. In addition we require passwords to be sufficiently long (at least six characters) or ‘‘rich’’ — containing digits, mixed-

---

[8]See case #2 and case #3.

[9]In fact, the comments in the BSD *passwd* program mention that this should be done.

case, and special characters.

This measure was in some cases defeated when a staff member used a utility which did not have *esp* linked in, but we believe that since the few cracked passwords were guessable, such password checking has protected the majority of our accounts from casual attacks. CERT was able to crack about 300 more accounts running their password crackers on our passwd file, and we have since incorporated their word list into our password checker. Currently we are using a dictionary of about 52,000 words. The list consists of standard English words from the *ispell* dictionary, words known to be used by password crackers (from the list given to us by CERT), and site-specific words such as hostnames. The code was borrowed from the *ispell* sources, and it uses a hash table, so it is relatively efficient. While some may argue that such restrictions leave no ‘‘good’’ passwords, changing one character to uppercase is usually sufficient to make an otherwise guessable password be acceptable.

4.5. Force Users to Change Stupid Passwords

Since *user1* and *user2* had both been broken into because of easily guessable passwords, we wanted to be sure that all guessable passwords were changed before the crackers got to them. In the beginning, we had been turning off accounts as we found them to have guessable passwords. This was necessary because we had no way of knowing whether the accounts had already been cracked or would soon be cracked. We set the shells of such accounts to */etc/badpasswd,* which was essentially the same as */etc/turnedoff.* The */etc/badpasswd* program simply displayed a message saying that the user’s password needed to be changed, and that they should visit our Business Office in person to get it changed to a better password. This worked all right for the first few password changes, but it was getting a bit tedious for the Business Office to have to deal with so many users — these password changes had to be done by hand, and the users had to come in person and present their Columbia University Identification card. After we received a list of more than 300 accounts whose passwords were broken by CERT, we implemented a simple password changing shell to automate this process, thereby sparing the Business Office a deluge of password changers.

We have a convention of setting the password of new accounts to be the user’s Columbia University Identification Number (CUID), so that they will know their initial password without us having to write it down. Obviously, in a university environment it is not unlikely that one can learn someone else’s CUID number, so we require the user to change their password immediately when they first log in. The program */etc/ksh.first,* which is the shell given to newly created accounts, prints a message about the importance of a good password, asks new users to change their passwords before resuming normal login, then finally resets the shell to be */bin/ksh.*

In the event that the user has been found to have a bad password, we set their shell to our new program */etc/ksh.badpasswd*, which is similar to */etc/ksh.first*. This program prints a message notifying the user that their password is easily guessable and that they should change it to a better password. Since we cannot be certain that the person trying to log in is not a cracker, we have them pass a simple identification verification test before allowing them to change their password and log in: we ask for their CUID number. All attempts to log in using this program, whether failed or successful, are logged, and the log is reviewed periodically to check for repeated failed attempts (to check for CUID-crackers!). We have found this method to be quite adequate because it does not require that the user wait until business hours to get their password changed.

The shells */etc/ksh.first* and */etc/ksh.badpasswd*, as well as the */bin/passwd* program, were modified to use the new 52,000 word dictionary (which incorporated CERT's list). Obviously, no dictionary can be big enough to include all the words a cracker could try, but eliminating the obvious ones makes it a little bit harder for a cracker to break in.

4.6. Hide Passwords

We finally bit the bullet and implemented a shadow password scheme to hide the encrypted password strings from would-be attackers. In three cases, the cracker used our machines as a CPU to crack password files from ours and other hosts.[10] By making encrypted password strings available only to the superuser, we make it more difficult for someone to break into our machines by simply stealing a copy of the password file, and then crunching it at their leisure on some other machine. As an added feature, we placed randomly generated strings into the public password file to let potential crackers waste time trying to decrypt fake passwords. This allows us to watch for password crackers because a noticeable amount of CPU time would be used trying to break a fake password file.

Switching over to shadow passwords was fairly straightforward. We modified the *getpwent()* family of routines to check the effective *uid* of the process — the real password string can only be retrieved by root. Programs like */bin/passwd* update both the flat */etc/passwd* file and the shadow password file, */etc/shad/passwd*, placing the fake password string in the flat file, and hiding the real password string in */etc/shad*, a directory readable only by the superuser.

We obtained a copy of the sources to Berkeley's implementation of a shadow password scheme. However, due to the size of our password file, Berkeley's method was extremely slow, so we reimplemented it using a hashed database. The improved code has been sent back to Berkeley and will hopefully be available in their next release.

---

[10] See case #3.

4.7. Check for Bad *.rhosts* Files

To be sure that crackers could not use accounts gained on our machines to hop over to other sites and vice versa, we ran a special spot-check of users' *.rhosts* files. We wrote a simple *perl* script which checked for lines in users' *.rhosts* files which contained machines outside of the Columbia domain or which included a username different from this user, and recorded bad lines along with the owner in a log. After scrutinizing the output, we then moved ''bad'' *.rhosts* files to be *.rhosts.bad*, and sent mail to each user informing them of the change and giving an explanation of safe *.rhosts* files.

This is especially important for staff accounts, since staff are likely to have accounts on many different hosts (e.g., at other departments, or even at other universities). We would not like to see staff accounts compromised because of poor security elsewhere.

4.8. Turn On Gateway and Terminal Server Security

Our terminal servers and gateways had been used by the crackers in a couple of cases as ''invisible'' routes to other machines. We implemented passwords on each of our terminal servers and gateways to prevent this happening in the future.

A password is now required in order to telnet into our gateways and terminal servers. Also, we allow access only to local hosts, in order to prevent intruders from using these machines to hide their origins as they try to break into hosts outside our domain.

4.9. Report the Problem

As always, we kept close contact with CERT during this whole ordeal. We warned system administrators at other sites as soon as we realized they might be affected. In order to determine who the system administrators were at other sites, we used tools such as *whois*, *nslookup*, and *finger*, as well as the SMTP *VRFY* and *EXPN* commands.[11]

4.10. Educate Users

During this time of emergency, we tried to maintain close communication with our user community. In system bulletin boards, electronic mail, and the message of the day, we kept users informed of what was happening, explaining the importance of keeping their passwords secret and unguessable, warning them of unsafe *.rhosts* files, and apologizing for the inconvenience of having their userids turned off. We also tried to point out ways the users themselves could help detect if there had been breakins on our system: keeping an eye out for strange files in their directories, using *last* to verify the time they last logged in, etc. In addition to user education, we did a lot of staff education, as it is most

[11] See the appendix for a sample session where we look up some system administrators in the *columbia.edu* domain.

important that users with privileges (such as being in the wheel group or having a root id) have secure passwords and *.rhosts* files. Also, staff are likely to maintain several accounts on different machines, and therefore they are likely to have little-used accounts which they should take care to keep secure. All in all, staff userids are likely to be enticing targets for malicious intruders.

5. Loose Ends

5.1. Who Were They?

Most of the attacks which we suffered were rather anonymous. People didn't break in from other hosts, but through dial-in lines and terminal servers. Those that we did trace didn't use their own names, but stuck with pseudonyms. Data files which were left around were, for the most part, encrypted. When we were able to watch a break-in attempt, we did not seem to be confronted with expert UNIX ‘‘crackers.’’ Often, MS-DOS style commands were used, and then followed up by their UNIX equivalents. Mail sent from penetrated IDs bounced: simple precautions were not taken to take care of little details like these. These initial attacks seemed to be followed by a flurry of similar attempts.

On the other hand, other attacks left dozens of encrypted password files from other sites sitting on our systems. These attacks seemed to be more careful, though history files were still found. We made a quick attempt at decrypting these files using the *Crypt Breakers' Workbench*, but as we had had no experience with the program, and since we already had a fair idea of the files' contents, it did not seem worth the time.

From the methods used and the traces left behind, as well as the repeated attempts to use the same entry points, we got the impression that we were dealing with several gangs of fairly unsophisticated crackers. In several cases crackers called attention to themselves through excessive CPU usage during hours when staff members were logged in. Alternatively, one cracker, trying to obscure his path, used our machine to reach other sites during a time when he was the only logged in user, making him especially easy to trace from login records.

In the end, all we could do was pass on what little data we had about these crackers to CERT, and hope that it would correlate with other information they might have. The only ‘‘cracker’’ who was identified was a local student, and he did not successfully crack any passwords.

5.2. Caller Identification

We ran into problems trying to trace the crackers via the telephone lines. First, we had to get past our CBX to find which New York Telephone lines were receiving the calls. Then, even when we had gotten that information using our CBX console software, we were unable to get telephone traces made. The university was in touch with NYTel about phone traces, but were eventually told that a court order was needed for any trace, and that the line the trace was being requested on had to be in an individual's name, rather than an organization's. Clearly, getting our dial-in modem numbers traced would be next to impossible, though our telecommunications department is still trying to work something out.

6. Futures

6.1. Preventative Measures

System administrators charged with the task of maintaining a secure working environment for their users need to keep up-to-date on potential security holes. This requires them to have access to channels that disseminate information on system security, such as security mailing lists, newsgroups, and other publications. Likewise, vendors need to be responsible to their customers by keeping current on these issues and providing timely fixes to any security holes that may be uncovered.

System administrators should take an active role in insuring the integrity of their systems. Periodic audits of the system as well as routine examination of log files can go a long way in detecting problems when they occur, and helping to minimize damage. Once a problem and its solution have been identified, fixes should be installed in a timely fashion. Perhaps to encourage this, once a fix is widely available it should be publicized. Methods of applying pressure on vendors should be developed to encourage them to provide fixes as soon as possible. How to do this is a sensitive issue that needs to be examined in depth. There is a fundamental dilemma in that it seems to require publicizing security holes to get vendors to take action, though this means that the holes can be abused in the interim.

We need better organization. Perhaps a single organization (like CERT) should coordinate contacts with law enforcement and telephone organizations. Perhaps there should be a distinction between tracing data and voice telephone calls. The laws for getting calls traced differ across state borders. Similarly, the procedures for getting traces are different in different places. The contacts and experience necessary for getting traces started can be more easily developed by a central organization than by individual institutions.

We need better cooperation throughout the Internet. When break-ins here pointed at other sites, it was often difficult (and merely through chance) that we were able to find administrators (and their phone numbers) at the other sites. The *whois* database should be expanded to contain accurate and up-to-date information. Several contacts and their emergency numbers should be included for each site. An automated periodic review process could get in touch with the listed contacts to verify entries.

In the event of network connectivity problems, the DDN Network Information Center may not be reachable. The information contained there should be replicated at several places; perhaps the database should be set up as a distributed database along the lines of DNS where each domain would maintain its own entries.

6.2. Ethics

Society needs to develop an awareness of the moral and ethical issues of computer usage. Breaking into a system should be judged with the same set of values as breaking into someone's home. Both are violations of privacy and cause severe distress as well as possible monetary damage to the victim. Just as burglary when committed by teenagers is not considered a ''childish prank,'' breaking into a computer system, regardless of how inadequately protected it may be, should not be considered the harmless activities of a precocious youngster. Movies such as *War Games* which glorify crackers ultimately do the community a disservice.

The Internet has grown from a relatively small community of trustworthy citizens to a cosmopolitan network with its share of vandals and other unsavoury individuals. We should try our best to educate the community as well as to deploy preventative measures to minimize the damage in the event of the inevitable attack.

7. Acknowledgements

We would like to thank Melissa Metz for helping during the incidents; our colleagues for proofreading this document and making extensive comments and suggestions; Benjamin Fried for *tap*'ing in Case #5; the members of CERT for moral support and processing our password file, and the members of our Telecommunications Department for contacting New York Telephone.

Appendix A. Locating a System Administrator

Below are excerpts from the transcript of a session in which we try to determine a night-time or emergency phone number for a system administrator within the *columbia.edu* domain.

First we query the NIC *whois* database for information on the *columbia.edu* domain.

```
% telnet nic.ddn.mil      # You can also use the whois cmd
Trying...
Connected to nic.ddn.mil.
Escape character is '^]'.
*  -- DDN Network Information Center --
*
*  >>> Our new address is 192.67.67.20.
*  >>> Service on our MILNET address (26.0.0.73) will end June 1.
*
*  For TAC news, type:                    TACNEWS <return>
*  For user and host information, type:   WHOIS <return>
*  For NIC information, type:             NIC <return>
*
*  For user assistance call (800) 235-3155 or (415) 859-3695
*  Report system problems to ACTION@NIC.DDN.MIL or call (415) 859-5921

 SRI-NIC, TOPS-20 Monitor 7(21242)-4
 The system will go down Thu 17-May-90  6:00pm until Thu 17-May-90  9:00pm
 for For Preventive Maintenance
@
@whois columbia-dom              # columbia domain
Columbia University (COLUMBIA-DOM)
   450 Computer Science
   New York, NY 10027

   Domain Name: COLUMBIA.EDU

   Administrative Contact, Technical Contact, Zone Contact:
      Cattani, Robert  (BC14)  cattani@CS.COLUMBIA.EDU
      (212) 854-2736

   Record last updated on 10-Aug-88.

   Domain servers in listed order:

   COLUMBIA.EDU                 128.59.16.1, 128.59.32.1
   HARVARD.HARVARD.EDU          128.103.1.1
   CUNIXE.CC.COLUMBIA.EDU       128.59.40.143

Would you like to see the known hosts under this secondary domain? n
Whois:
@logout
 The system will go down Thu 17-May-90  6:00pm until Thu 17-May-90  9:00pm
 for For Preventive Maintenance
Killed Job 19, TTY 152, at 14-May-90 17:34:40
 Used 0:00:04 in 0:01:37
Connection closed by foreign host.
%
```

Since it's after hours, calling the phone number given may reach an answering machine. Also, since the record hasn't been updated in two years, the number may be wrong. Using the information from the *whois* database we *finger* the contact address to see if his finger plan has an emergency phone number listed.

```
% finger cattani@cs.columbia.edu
[cs.columbia.edu]
Login name: cattani                     In real life: Bob Cattani
Directory: /u/cs/cattani                Shell: /bin/ksh
Last login Mon May 14 08:34 on ttyp7 from williamsburg.cs.
New mail received Mon May 14 19:22:00 1990;
  unread since Mon May 14 18:15:08 1990
Plan:
Robert Cattani                          EMail:  cattani@cs.columbia.edu
450 Computer Science
Columbia University                     Phone:  212-854-8107 (office)
New York, NY  10027

Computer Science Department
        Director of Research Facilities

%
```

We may not have found what we are looking for yet (this is another office number), so we try the Domain Name System (DNS) next, looking up a *Start of Authority* record.

```
% nslookup
Default Server:  cunixf.cc.columbia.edu
Address:  128.59.40.130

> set querytype=SOA
> columbia.edu.
Server:  cunixf.cc.columbia.edu
Address:  128.59.40.130

columbia.edu    origin = columbia.edu
     mail addr = hostmaster.columbia.edu
        serial=182, refresh=3600, retry=1800, expire=3600000, min=3600
> ^D
%
```

The mail address (*hostmaster@columbia.edu*) specified in the SOA record looks like an alias, so we will try to expand it using the *SMTP EXPN* (or *VRFY*) command.

```
% telnet columbia.edu 25      # port 25 is for SMTP
Trying...
Connected to columbia.edu.
Escape character is '^]'.
220 columbia.edu Sendmail 5.59++/0.3 ready at Mon, 14 May 90 20:35:42 EDT
expn hostmaster
250-<hostmaster@cunixf>
250-<alan@cunixc.cc.columbia.edu>
250-<chris@cs>
250 <cattani@cs>
quit
221 columbia.edu closing connection
Connection closed by foreign host.
%
```

The *hostmaster* alias on *columbia.edu* expanded to a few users and another *hostmaster* alias on *cunixf*. We try expanding the alias there next.

```
% telnet cunixf.columbia.edu 25
Trying...
Connected to cunixf.cc.columbia.edu.
Escape character is '^]'.
220 cunixf.cc.columbia.edu Sendmail 5.59/FCB ready \
                      at Mon, 14 May 90 20:37:40 EDT
expn hostmaster
250-Howie Kaye <howie@ivory.cc.columbia.edu>
250-Howie Kaye <\howie>
250-<jbaltz@cunixe>
250-Fuat C. Baran </f/sy/fuat/.backup/backup.mail>
250 Fuat C. Baran <\fuat>
quit
221 cunixf.cc.columbia.edu closing connection
Connection closed by foreign host.
%
```

We then try *finger*'ing again.

```
% finger fuat@cunixf.cc.columbia.edu
[cunixf.cc.columbia.edu]
fuat   Fuat C. Baran    Last login May 13 12:58 from ttyqd (sparky.cc.columb)
Project: CUCCA UNIX Systems Group and postmaster
Mail forwarded to \fuat, /f/sy/fuat/.backup/backup.mail

I am a member of the CUCCA UNIX Systems Group and CUCCA postmaster.  I
am one of the authors of Columbia-MM.

Electronic mail is the best way to get in touch with me.

For fastest results, send electronic mail related queries to 'postmaster',
general questions to 'consultant', mm problems to 'bug-mm' please.
In an emergency call the Operations Shift Supervisor at 854-2652 and have
him page someone from the UNIX Systems Group.

Passwords -- ''Use them like a toothbrush.  Change them often and don't
               share them with friends.''
                                        --Clifford Stoll
----
Internet: fuat@columbia.edu        U.S. MAIL: Columbia University
  BITNET: fuat@cunixf                         Center for Computing Activities
    UUCP: ...!rutgers!columbia!cunixf!fuat    712 Watson Labs, 612 W115th St.
   Phone: (212) 854-5128  Fax: (212) 662-6442 New York, NY 10025
%
```

Here we finally find an emergency phone number. If none of the above had yielded a phone number, we could also try *finger*'ing common usernames such as *root*, *operator*, *operate*, *system*, etc. and hopefully one of them would have a finger plan. If all else fails, we can try calling directory assistance for the city.

A.1. Is Your Information Available?

When dealing with the incidents mentioned above, we had the need to contact system administrators at other sites. In some cases we followed all of the preceding steps and still had trouble reaching somebody. It is a good idea to periodically check places such as the *whois* database, the SOA record in the DNS, and finger plans of system IDs and administrators to make sure that someone outside your site can reach you in an emergency with minimal effort. It is also a good idea to make sure that your local phone company has a general information number for your institution listed, as well as verifying that your internal phone operator knows the number of the computer center.

When all else fails someone may have to resort to sending electronic mail, so make sure IDs such as *root* are forwarded to people who are in charge of the system. However, in some cases sending email is inadvisable, in the event that the message may be intercepted.

Table of Contents

List of Figures