UNIX Password Security

Walter Belgers

`walter@giga.win.tue.nl`
December 6, 1993

Introduction

This document was written to make system administrators aware of the importance of well-chosen passwords. Easy-to-guess passwords offer hackers[1] the possibility to enter a system. More and more computers are being connected to the world-wide Internet (the latest estimations speak of about 1.5 billion systems). This means there will be more and more users, and therefore more and more hackers. By means of good password-security, one can protect a system from newbie hackers.

There are many types of systems, and every one has a lot of security aspects to it. I will restrict myself to the password-security of UNIX systems. The reason being that UNIX systems are very popular, especially in an educational environment, where one can expect an increased concentration of hackers due to the openness that is appreciated in such environments. This is in contrast to a commercial environment, where data has to be protected. E.g. against competitors. There are many ways to hack a UNIX system, and there are many programs for finding a user's password. These programs can be used by people who have little knowledge of UNIX. Choosing good passwords can therefore help in keeping newbie hackers out. ('Advanced hackers' are often capable of entering a system without using passwords. This implies that the security of a system depends not solely on well-chosen passwords.)

Besides the importance of good passwords (i.e. not easily guessable) we will take a look at how they work. Next we will see an actual example which shows how bad passwords are usually chosen. To conclude I will give several methods for chosing good passwords.

The importance of good passwords

The goal of a hacker is usually obtaining the superuser-status ('root'). The normal strategy to do this is using badly installed software, bugs in (system)software and human errors. There are several ways to hack a computer, but most ways require extensive knowledge. A (relatively) easy way is logging in as a normal user and searching the system for bugs to become superuser. To do this, the hacker will have to have a valid usercode/password combination to start with.

It is of utmost importance that all (!) users on a system choose a password that is not easy to guess. The security of each individual user is closely related to the security of the whole system. Users often have no idea how a multi-user system works and don't realise that they, by chosing an easy to remember password, indirectly make it possible for an outsider to manipulate the entire system. It is essential to educate the users well to avoid attitudes like the one described in [Muf]: "It doesn't matter what password I use on **my** account, after all, I only use it for laserprinting…".

[1] 'Crackers' is a better term, because 'hackers' is historically reserved for people who take the most out of their computer because of their great knowledge of soft- or hardware. I will continue to use the term 'hacker' because this is common in literature.

The users have to get involved with the security of the system they are working on. In [Pet] we read "Users have a responsibility to employ available security mechanisms and procedures for protecting their own data. They also have a responsibility for assisting in the protection of the systems they use". It also says that it is important to notify the users of the security guidelines. A solution might be giving new users a limited course. Or at least make them understand why good passwords are essential. This can be done e.g. when a user gets his or her initial password from the system administrator.

How a hacker finds a password

Most UNIX systems don't use the so-called shadow passwordfiles which we will look at in a moment. In most cases, the passwords are stored encrypted in the file `/etc/passwd`, or, if the system is a client, on the server. In the latter case, one can get the passwordfile by giving the command `ypcat passwd`.

A line from the passwordfile looks like this:

```
account:coded password data:uid:gid:GCOS-field:homedir:shell
```

A user with account gigawalt, crypted password `fURfuu4.4hY0U`, userid 129 (a user with userid 0 (of which there can be more than one) is superuser), groupid 129, information (GCOS) Walter Belgers, homedirectory `/home/gigawalt` and shell `/bin/csh` will have an entry in `/etc/passwd` like this:

```
gigawalt:fURfuu4.4hY0U:129:129:Walter Belgers:/home/gigawalt:/bin/csh
```

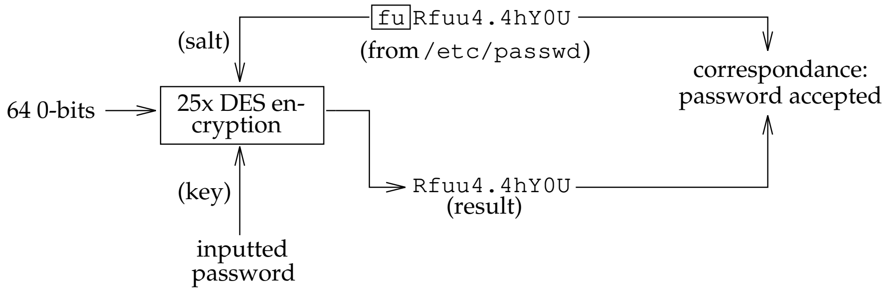
Things are a bit different when a shadow passwordfile is being used. All coded password data is replaced with a `*` in /etc/passwd, and a second file (the shadowfile) is used to store the original data. This file can only be read with sufficient privileges so that normal users cannot get the crypted passwords.

Passwords are being crypted using the DES algorithm. DES ('Data Encryption Standard') is an American encryption standard since 1979. With DES it is possible to encrypt and decrypt data using a key. UNIX password encryption uses the DES algorithm 25 times in a row. The first DES round uses 64 0-bits as input and encrypts them with the password the user inputs, with a permutation taking place during the encryption process. There are 4096 possible permutations. The permutation used is randomly chosen for each user. The chosen permutation is coded into two bytes called 'salt'. The salt is stored in the passwordfile. The output is used as input for the next DES round, which uses the same key and permutation. This process repeats until there is a final output from the 25th DES round. This output is coded into eleven bytes which are put in the passwordfile. So the coded password data in the passwordfile consists of thirteen bytes, first the salt and then the encrypted password.

This method of encryption is almost irreversible, which means that it is easy to encrypt a string, but it is impossible to find the original of a string encrypted in the way described above, except by systematically trying out all possible keys and salts. (By impossible we mean here that there is no known method that does this quicker.) It is however possible to find the original of a string encrypted using *single* DES. For more information concerning single DES encryption see [Til]. If it's impossible to decrypt the encrypted password, how can a user log in? This works as follows: the user inputs his or her password which is used as key to crypt 64 0-bits using the aforementioned method, using the salt as it can be found in the passwordfile for that particular user. If the output corresponds to the eleven bytes that represent the crypted password in the passwordfile the password is considered valid and the user will be permitted to access the system.

For more information on the precise working of DES on UNIX see [Fel2].

Schematically it looks like this:



As I just pointed out it is not possible (in real life) to decrypt passwords. However, it is possible to encrypt 64 0-bits with some words and see if the result 'incidentally' is the password. Then the account is hacked. One could remark that it must be possible to check all possible passwords this way. However, this would take the fastest computer longer than the time the universe exists. By trying out only passwords consisting of six lowercase characters, it becomes possible to try out all combinations in reasonable time, provided one has an extremely powerful computer. The latest record for passwords consisting of six lowercase characters stands at one hour per user. Passwords of accounts that are attractive to hackers (the ones with a lot of privileges) should therefore never consist solely of lowercase characters!

We see that the usual way to find passwords is by guessing them. So you have to make sure users do not use easy-to-guess passwords, i.e. passwords that can be found in lists (a dictionary, an encyclopedia, files with astronomical terms, flora and fauna, etc.). It is very easy to obtain such lists via the Internet.

Suppose a line from the passwordfile looks like this:

```
gigawalt:fURfuu4.4hY0U:129:129:Walter Belgers:/home/gigawalt:/bin/csh
```

Then passwords that should **not** be used are e.g.:

- all English words and derivatives (so 'laugh', 'laughs', 'laughing', etc.)
- all words from a foreign language (it is easy to get foreign dictionaries)
- words that can be found in the passwordfile itself like Walter, Belgers, gigawalt, etc.
- patterns like 123456, qwerty, etc.
- geographical names
- words from an encyclopedia ('Socrates')
- the license plate of a car, the roomnumber, the phonenumber or other things that have something to do with the owner of the account
- given names
- variations of these (walter, WALTER, retlaw, Walter, wAlter, walter0, walt3r, Retlaw4,...) Also take into account doubling words or adding a random character.

An actual example

To show how users choose their passwords badly, I used a password guessing program on a passwordfile of a system in operation.

The program I used was Crack v4.1 with ufcrypt (ultra-fast crypt, a fast implementation of the DES algorithm) on a network of SUN ELC computers. The performance of these computers (20 MIPS) is comparable to that of a modern PC. The program was stopped before it was finished after almost 60 hours. The passwords that were found were found within the first 25 hours.

Results:

Type of machines:	11x SUN ELC
Total number of accounts:	521
Number of hacked accounts:	58 (11.1%) (with interactive shell 56 (10.7%))
Total time:	59:13 (real time, not CPU time)

1	lists	42	(7.2%)
2	common names	1	(0.2%)
3	user/account name	5	(0.9%)
4	phrases and patterns	3	(0.5%)
5	women's names	2	(0.3%)
6	men's names	4	(0.7%)
7	cities	1	(0.2%)

Passwords found:

1. cyclades, paardens, fiesta, regen, gnosis, police, fuselier, ballon, smaragd, marques, farao, kasteel, valent, adagio, clematis, gehannes, koeien, gnomen, onderkin, zeilboot, druppel, fietsen, testen, marathon, tamtam, global, vrijheid, wolf, kwiek, basket, stones, klomp9, fiets9, Zoutje, Biefstuk, neenee, tnbrg (this is 'tonbrug' without vowels).

2. fischer.

3. guest had password guest. This is not a user's fault of course, but the fault of the system administrator. It has to be seen whether or not logging in as 'guest' with password 'guest' is criminal in the Netherlands.

4. qwerty, unesco.

5. heather, joanne.

6. piet, atilla, Frans2, vatsug (this is 'gustav' spelled backwards).

7. adelaide.

Some people have studied the amount of passwords that is easily guessable in the past. In [Kle] Daniel Klein finds 21% of 15,000 passwords using one week of CPU time. The first 2.7% was found within 15 minutes (people who used their account as password, e.g. account gigawalt with password gigawalt). The categories in which over 1% of the 15,000 passwords were found are:

lists	7.4%
common names	4.0%
user/account name	2.7%
phrases and patterns	1.8%
women's names	1.2%
men's names	1.0%
machinenames	1.0%

Comparing these results to the ones above is of little use because of the limited extent of this investigation.

A somewhat more extensive research (see [Far]) was concerned with passwordfiles of several `.COM` systems (computers owned by US companies). One would expect companies to have good security measurements, but the passwords kept coming in, with the first root-password (!) after little more than an hour. (There was a total of 1594 passwords of which 50 had been guessed within 15 minutes, and 90 after 35 minutes.)

Picking good passwords

The above illustrates the importance of a good password for every user. We will look at some methods for choosing good passwords. A good password consists of 8 characters (a UNIX password can be up to eight characters, any extra characters will be discarded, making the passwords 'Still won't talk, eh, Spiff?' and 'Still wo' mutually interchangable). It has to be hard to guess but easy to remember, because otherwise, users will be tempted to write down their password which totally takes away the function of a it.

Pick a password that not only consists just of upper- or lowercase characters, or only one capital ('seCret' is thus a bad password). It is preferable to use a non-alfanumeric character in the password (%,=,*, etc.). The use of control characters is possible, but not all control characters can be used, and it can give rise to problems with some networking protocols.

A few methods:

- Concatenate two words that together consist of seven characters and that have no connection to eachother. Concatenate them with a punctuation mark in the middle and convert some characters to uppercase. Examples: 'Pit+idEa', 'plOVer#me'.
- Use the first characters of the words of a certain (not too common) sentence. When we use the sentence 'My goldfish are called Justerini and Brooks!' as example, we would get the password 'MgacJaB!'. (Also in this case make sure you use an eight-character password with uppercase characters and/or punctuation marks.)
- Alternately pick a consonant and one or two vowels resulting in a pronouncable (and therefore easy to remember) word. Examples: 'koDupaNy', 'eityPOop'.

Reducing break-in possibilities

It is important for users to have hard to guess but in the meantime easy to remember passwords. There are methods for generating such passwords. System operators should inform the users about the importance of good passwords.

To reduce the risk of a break-in there are several possibilities:

- Make sure the users know why a good password is important and how they can choose one.
- Install a new `/bin/passwd` (or `yppasswd`) that checks whether the password is not too obvious (by checking if it contains punctuation marks, or by investigating if the password can be found in standard wordlists).
- Install a shadow passwordfile (this involves changing some software).
- Let passwords expire, for example after three months for regular users, after a month for users with extra privileges. The timespan a password lives should not be chosen too small. What will still exist is the danger of users using series of passwords, like 'Secret1', 'Secret2',... making it easy for a hacker to, once he has obtained a password, guess the successor.
- Use a program that hacks passwords to check if some users have guessable passwords. Let those users visit you personally to inform them about the fact that a good password is everyone's concern.
- Switch to single-use passwords (this is radical and requires some investments, see [Ven]).
- Use passwords of accounts with privileges like that of root on the console only to avoid eavesdropping the network. When impossible, try to avoid logging in on such accounts from computers or terminals that are connected to a LAN segment on which people can easily and/or anonymously wiretap the network, like classrooms.
- Keep in mind that total system security is as weak as the weakest chain. Also keep in mind that a system with good passwords alone is not yet a secure system.

Bibliography

[Bel] WALTER BELGERS, *Password Security – A Case Study* (in Dutch), TimeWasters Online Magazine #5, march 9, 1993, can be obtained by sending email with Subject '`TOM5`' to `timewasters-request@win.tue.nl`.

[Cur] DAVID A. CURRY, *UNIX System Security*, Addison-Wesley 1992.

[Far] DAN FARMER, WIETSE VENEMA, *Improving the Security of Your Site by Breaking Into it*, USENET newsgroup `comp.security.unix`, can be obtained by anonymous ftp from `ftp.win.tue.nl` as `/pub/security/admin-guide-to-cracking.Z`, 1993.

[Fel1] DAVID C. FELDMEIER, *A High-Speed Software DES Implementation*, can be obtained by anonymous ftp from `thumper.bellcore.com` as `/pub/crypt/des.ps.Z`, 1989.

[Fel2] DAVID C. FELDMEIER, PHILIP R. KARN, *UNIX Password Security – Ten Years Later*, Proceedings of Advances in Cryptology – CRYPTO '89, 1989.

[Kle] DANIEL V. KLEIN, *"Foiling the Cracker": A survey of, and Improvements to, Password Security (revised paper)*, Proceedings of the USENIX Security Workshop, summer 1990.

[Muf] ALEC E. MUFFET, *Almost Everything You Ever Wanted To Know About Security (but were afraid to ask!)*, USENET newsgroup `alt.security`.

[Pet] R. PETHIA, S. CROCKER, B. FRASER, *RFC1281: Guidelines for the Secure Operation of the Internet*, november 1991.

[Til] HENK C.A. VAN TILBORG, *An Introduction to Cryptology*, Kluwer Academic Publishers, 1988.

[Ven] WIETSE VENEMA, *Using SecurID tokens in an open multi-host UNIX environment*, can be obtained by anonymous ftp from `ftp.nic.surfnet.nl` as `/surfnet/net-security/docs/securid.ps`, 1993.