Chapter 1

Recovering lost passwords

It is now commonplace to use encryption algorithms to prevent unauthorized access to computer data. Even many novice computer users are able to benefit from the use of encryption. If we have something we wish to keep private from others but that we may also need access to in the future, we often use the password protection built into editing or compression programs. More advanced users frequently employ sophisticated programs dedicated solely to encryption, which can cost tens or even hundreds of dollars. Such applications are becoming increasingly popular as users have come to recognize the need for privacy. It's no surprise, because just as the number of opportunities provided by the internet has grown, so has the number of potential threats. Ultimately nearly all users will wish to acquire the tools necessary to secure their accumulated data so that nobody except the owner will have access to them.

Are password-encrypted data secure enough? This is one of the questions that occur to nearly every computer user. As time passes and new technologies appear, the answer to this question becomes more difficult. That's because despite the appearance of newer and more secure encryption methods, no-one can be certain that the current security methods can't be broken. Just as some people are constantly working on newer and more effective forms of security, others are constantly working to break them. As a result, very few of the encryption methods that human ingenuity has devised will prove to be reliable long-term protection for a broad variety of data.

So how can we check the security of our data? What do we do if we have lost or forgotten the password? The answer should be obvious – we have to try to crack our own security! By using applications that allow us to discover our

forgotten password and recover useful data, we will discover how easy it would be for an intruder to do this. This process will be helpful in creating a more secure password.

Unfortunately, at the moment of writing there is no free software available for Unix systems that could help us perform these tasks. So the only option available to us is to use an application designed for closed systems. Such applications usually are not free, meaning that their availability to the average user is severely limited. We will, however, try to describe the most popular tools and indicate approximately how to crack the password.

To do this, we will use a software package developed by Elcomsoft that includes a broad range of applications and available functions. Among these commercial programs is the Advanced Password Recovery series, which, however, is also available in a shareware version (with slightly limited functionality), making it particularly useful to us.
The interface of the shareware version is no different from the commercial version, so once we have described examples of two applications for cracking passwords to RAR archives and DOC files, readers should be able to use other applications in this series without difficulty. The shareware versions are fully functional for 30 days after installation. The applications are available from the following address:

```
http://www.elcomsoft.com
```

Using Advanced RAR Password Recovery

First we must equip ourselves with the program mentioned in the previous section. We can download it from the publisher's site:

```
http://www.elcomsoft.com/download/archpr.zip
```

We can unpack the archive using commonly available programs such as WinZip or WinRAR, and then install the application. After starting the program we will have a look at its interface.

Along the top of the window is the toolbar, where we will choose our options to begin. First, however, to test things out we should create an example RAR archive and secure it with a simple password in order to try cracking it. We can create it using WinRAR, which is available at:

```
http://www.rarlabs.com
```

Next we open our archive, and clicking on the small key icon at the bottom of the window, we give the file a password. Choose, for example, "abc". We can also set the password when creating an archive in WinRAR by choosing Advanced and then Set Password. We enter the password and thus protect the archive we have created with the character sequence "abc."

Now we can go to the password cracker. We select an encrypted RAR file to open, under **Encrypted RAR-file** at the top left of the window.

First we will try a brute-force attack. How this kind of attack works is the computer creates all possible combinations of characters of different length within a given range (in ARCHPR the default range of length is 1-5 characters). In doing this it uses a character set defined in the **Range** field. It is a very time-consuming attack, especially with RAR files created with newer versions of WinRAR (2.9 and above), due to the very robust coding mechanism these use. We can check how long decoding will take on our computer by clicking the button labeled **Benchmark** – for a while the program will analyze test passwords, after which it will display a summary. The statistics display the number of passwords to be generated and checked with the current brute-force settings, as well as statistics for test duration and average time to crack the password. Normally, even with a fast computer, the number of password attempts varies between only 20 and 30 per second. Thus we see that encoding, even within the default settings of five characters and a range of upper and lowercase letters of the Latin alphabet, would take a long time – on the tested system around two hours.

But let's return to our attempt. In the real world, whenever we have to recover a password for secure files, we usually remember approximately how long the password we used could have been and which characters it probably

contained. Our "abc" is the maximum length of three characters, which we set in the **Length** field. We also set the minimum string length to three, because we are sure about the length of our password. We also know that we used a password composed only of lowercase letters, so we choose all small letters (a-z). Let's try Benchmark again.

If we set the length and the range correctly, it should take no more than a second to get a working password, because the program found it by testing passwords generated at random on the archive.

Keeping this duration in mind, we should check how long it would take to generate a password with more challenging conditions. Let's assume that we encoded an important file and the only thing we know about the password is the fact that we used:

- Upper- and lowercase letters
- Numbers
- And around eight to twelve characters

We set the appropriate options for length and range, and once again use Benchmark. On the author's computer it would require about four hours to crack this password. Even so, this really is not a lot of time, considering the importance of the data.

So if we need to crack a password to some encoded archive, try to remember, as much as possible, which characters it could be made up of. This will save us a lot of time needed to guess the sequence using the application. We set the appropriate options on fields that divide the ARCHPR window into very readable sections. We talked about the Range field and now know that we use it to choose the sets of characters the password we are looking for is composed of. Next, in the Length field we have options available enabling us to define the password size – we can set both a lower and an upper limit. We will discuss the Dictionary section in a while. First let's click on the **Auto-Save** tab. In this field we choose the file in which we define the time interval in which the data are automatically saved. This allows us to continue searching from the point where we left off. In **Options** we can specify the

priority for the application. Priority means the attribute of the process that allows us to define its importance. The process with the highest priority is assigned the greatest available share of resources and is given the largest share of processor time. We also have two buttons that allow us to update the program to the latest version as well as register it via the internet. After registering, more options become available in the application. In the **Advanced** field particular attention is drawn to the **Use code optimized for** option, in which we choose the type of processor (the closest to the one we have). Because of this, the password-cracking procedure can take advantage of hardware-specific options and CPU-level routines, resulting in better performance and increased password-cracking speed.

We have now described the basics of brute-force attacks. Now it's time to introduce the concept of dictionary password cracking. The dictionary method is much faster and more effective than brute force, but it is not always successful in finding the password. In this case, the password cracking occurs in a completely different way. By starting the cracking procedure the application asks us to choose a dictionary or loads one automatically. This is a text file with a very simple format and rules, the content of which is a list of terms in alphabetical order taken from a standard dictionary. Each of these begins on a new line, so the dictionary structure is ordered along the lines of:

```
(...)
comp
computational
computer
computing
(...)
```

The dictionaries used for cracking passwords generally also include many non-standard but relevant terms, for instance "admin," "hax0r," etc. Because of this we can have greater certainty that we will find the correct character sequence.

While using the dictionary, the program does not generate any terms but loads them according to their sequence in the file. To use this option, in ARCHPR we go to the **Type of Attack** pull-down menu at the top right of the window, and choose Dictionary. Now let's have a look at all the tabs. We

should notice that the first two – **Range** and **Length** – are no longer available. The dictionary does not contain a limit on length – all the terms contained in it are attempted.

Click on the **Dictionary** tab to activate that field. Let's use the setting available in it and define the details. Under **Dictionary file path** we can indicate which list of terms the program should use. At the beginning we should use the included standard list, the default value for this option. **Start line #** defines the line in the dictionary file from which we should begin checking the terms. We don't need to enter anything here but can enter a line number if required. The **Smart mutation** field activates the function by which the program combines terms, increasing the probability of correctly guessing a more complicated password. We may also choose to use **Try all possible upper/lower case combinations** depending on the length of the password. The application will then try all possible permutations of a given word.

We have now set all the options that we need to begin dictionary cracking. We start the whole process now by activating the **Start!** button, in exactly the same way we did with brute force. The testing procedure now begins. We see in real time which term is currently being checked, how much time has elapsed and an estimate of the time remaining, as well as how many terms have been processed and what the average application speed is. After not too long we should see a window with the process results – our password has been guessed correctly. In the case of frequent combinations, such as the example "abc" or similar, the dictionary method has proved to be ideal. We must be aware, however, that in the case of a non-English-based password, the default word list will not produce satisfactory results. We must instead use a more appropriate dictionary file. Such files can be found without difficulty on the internet.

We have now learned what brute-force and dictionary attacks are and how they are used. We have also become familiar with the basic functionality of the ARCHPR application and the principles of its use. We have also become familiar with its interface. We invite the reader to try out the other functions of this program. Not all the possible examples of its use have been described

here because we believe that people learn best by trying out all the options at their own pace. From experience we know that forcing one's own ideas and ways of doing things on others does not produce the best results because it limits the opportunities and the desire to use one's own intellect to test the range of software functions.

As for the ARCHPR application, it is a very useful program, and for many reasons it is among the highest-rated software of its kind. Firstly, although the full version is not free of charge, the shareware version does not place many restrictions on use, which is a very important feature if we do not need the program in a commercial setting, just for occasional personal use. Secondly, the program offers high performance. The third plus is, the current version can deal with a wide variety of RAR archives – from those created with the earliest versions of WinRAR to those from the newest, WinRAR 3.00.

Recovery of DOC files with Advanced Office Password Recovery

As we mentioned already, we will be using an application from the Advanced Password Recovery series for Microsoft Office for this purpose. We can download it from:

```
http://www.elcomsoft.com/download/aopr.zip
```

The interface of this program is almost identical to that of ARCHPR, so we will not bother describing how to use it again. We will provide a few examples of password cracking.

Let's start with brute force, as in the case of RAR archives. At the beginning we will create a Microsoft Word DOC file containing some sort of text and protect it with a password. We do this in Word by choosing (in versions XP or 2003) Options from the Tools menu and in the Security tab we enter the password to protect the document from being opened. We enter our standard "abc" then click OK, confirm the password and save the file. Now we run Advanced Office Password Recovery. We chose the **Open** file option. And what happens? The password that we used for the file, "abc" appears immediately on the screen! Now we will try a longer password. For example, "syncmaster755ds", which is the maximum length for this encryption

method. At first the program will try to crack the password with the "quick" method that produced "abc" on the first attempt. Unfortunately, the attempt will not produce any results. We now click OK and set the cracking options.

With a brute-force attack we choose the **Brute-force** option on the Recovery tab. Under Brute-force there are many options. Under **Password Length** we set the minimum and the maximum likely password lengths. Under **Character Set** we can choose the characters that the program is to use in attempting the passwords. We can use the **Define custom charset...** option, which can be useful for non English character sets. We select a-z and 0-9. In the **Starting Password** field we specify the word from which the cracking of the password will begin. In **Mask/Mask Character** we can enter all the parts of the password that we know - this option is, however, of little use because it is limited to cases in which we know the length of the password. For instance, in the case of an 8-character password that we know begins with "al", we enter the mask "al******". There is a more detailed description in the program's help section.

We do not have to enter anything under **Options**. We have now set the options for a brute-force attack. Next we have to click the **Start** button to begin the cracking procedure. Unfortunately, we encounter the limitations of the trial version here. It can only crack passwords with a maximum length of four characters.

In the case of the dictionary option, we choose this mode in the **Recovery** field. All other settings are found in the **Dictionary** field. As mentioned earlier, in the Dictionary tab we can indicate the dictionary file to be used (under **Dictionary file**) and if needed change the default dictionary (in the field below) or enter the line at which the cracking should begin (**Start from line number**). We now choose **Start**. After a short time, the whole dictionary will have been attempted, but our password will not have been found because it is a relatively unusual combination. For the purposes of our test, we will write it to the dictionary file, placing it somewhere in the middle of that document. Once again we will start the cracking procedure. Unfortunately, in the trial version, after finding the matching password the application informs

the user about its excessive length. Fortunately the correct password is displayed in the area above the progress bar until the window with this message is closed.

Before we start the dictionary method, we suggest writing a list of your own most frequently used passwords terms at the very beginning of the dictionary file, entering each on a separate line. This operation will save you a lot of time in testing your own encrypted archives.

With this we end the short presentation of the possibilities of the A*PR software. Of course, these applications can be used successfully in many configurations, which we will leave it to the reader to discover. Experimenting increases one's own knowledge and leads to better understanding of these programs. So we would suggest: Test, test, and test again. And if these programs prove to be useful to you, you should consider purchasing the full versions.

In summary, thanks to the applications in the A*PR family, we can carry out tasks such as recovering lost passwords relatively easily – we suggest, however, to run them whenever possible on fast computers so that the cracking process itself is performed in a bearable time, as quickly as possible.