```
:-----------------------------------------------------------:
:                                                           :
:            M A K I N G    T H E    M O S T    O F        :
:                 R S T S / E    S Y S T E M S              :
:            ====================================           :
:                                                           :
:                 Written by   :   Captain Hack             :
:                              :   of Melbourne,            :
:                              :   Australia.               :
:                 Written on   :   01-Feb-86.               :
:                 File No.     :   V01A-01                  :
:                                                           :
:-----------------------------------------------------------:
```

Originally Displayed in the U.S. on:
P-80 Int'l Information Systems

INTRODUCTION.

This file is a tutorial on making the the most of a RSTS/E system, making the most could mean anything from making the system do so neat tricks, to using it to you advantage, to taking it over completely; depending on your needs!

For most of the examples you will need an account, obviously non-privilaged, else you would not be reading this tutorial. Bear in mind that most, if not all, of the techniques described can be changed by the sysop. I found this out while trying them, but most sysop's don't realize everythings full potential and how it will be used; needless to say that I most likely have missed out on things. Anyway I hope you like the tutorial and you have an educational experience! I will rely on also using your imagination and ingenuity, as this is often needed.

OBTAINING OLD FILES.

If ever you have a valuble file that you don't want people to see the contents of for one reason or another, always write other information (random of fixed) over the entire file before deleting it. When the system creates a file it likes to have it continuous if possible, which means many blocks will be consecutive. When a file is created the

system alters information in a system file indicting that a particular block or set of blocks have been allocated so as they will not be over-written. The directory knows which which blocks are associated with which file, but when you delete a file, the system flags the used blocks as available and delete the directory entry. The system doesn't wipe the information.

To dig up these old blocks, write yourself a program to open a large file, I will leave the size up to you. You use the filesize option in the OPEN statement to do this, then just read in the blocks. When possible use block I/O for file manipulation because of its speed and convienience. Look through the blocks any if one if worth keeping save if to another file.

Seeing as BASIC programs have line numbers, as long as you find all the blocks, file reconstruction is easy.

There of course is no guarantee that you will find them all. Some may have been reallocated, but it's amazing what you can find! The bigger the block cluster size used in files the easier it is to reconstruct them. Just experiment!

ANNOYING THE USERS.

A way to annoy the users of the system is a techinque discovered just after we found out about the block recovery. After finding remnants of some of my data strewn across the system I decided the best thing to do was to zero all the available disk space. One Sunday night we wrote a program to open a very large file (all free disk space in fact) and proceeded to zero this. This was sucessful, except when the system was supposed to kill the file something went wrong and the file was not deleted. The next morning, before the sysops arrived, the users tried to login. Because there was no free space to write login records or do anything! No-one could login! Apparently this had the operators mystified why they couldn't get into their system. They had to reboot, or so I heard, they later worked out what had happended. They then asked why!

THE PSEUDO KEYBOARD.

The pseudo keyboard (device PKnn:) must be about the most useful and versatile device. This is the device to be used for the perfect hack! Originally, like many people, I saw it in a manual and really didn't read about it but as usual when I was going over the manuals with a fine tooth comb I read into it. Although most of the weak-points discussed in this tutorial can be removed by patching implemented by the system operator, they are likely to ignore them. The pseudo keyboard is a keyboard which doesn't physically exist! When you open the channel and do I/O's with it it appears like another keyboard. It even has a keyboard number. This is useful for extracting inormation for your programs that can only be accessed in command mode. An example of this is SYSTAT. I recommend that you get hold of a PROGRAMMING MANUAL and read it thoroughly, including the section on pseudo keyboards.

PASSWORD CATCHING.

Password catching is always desirable if you want access to a privilaged account! When I first started out we ran a crude program which did I/O's to the desired terminal and gave the responses that the system would give. A number of problems that we ran into was that you cannot fake the whole system, that is impossible using those methods, and also this did not allow for timing delays which most users were used to and expected.

With these old programs, the passwords obtained never lasted long because the user almost always knew they had been caught because users get suspicious when they get an INVALD ENTRY - TRY AGAIN message when they put in their correct password. This problem left me pondering how could you write the "perfect" password catcher. When I discovered pseudo keyboards I was thinking of applications, then it hit me. Why not simulate the whole job of another user? It was possible too!

To do this you write a program to open a channel to the keyboard where the person will enter the desired account number and password. You also open a pseudo keyboard. Basically from then on you pass the data from one to the other, and you keep checking what is being typed and when the account number and password are detected save them to a disk file encoded or what ever. You should continue to simulate the job until the person logs off. There are a few things to

be careful about. When the person runs SYSTAT make sure the output is sent to the terminal replaces the pseudo keyboard number with his keyboard number. Make sure he doesn't see a channel is open to his terminal and other things like that. Another things to be careful of is that he doesn't run programs to tell him his keyboard number or a few other things like that or a program where the terminal where it is run from affects its operation, if he does it could be quite hard to deal with. All of the techniques I describe will need practice and perfection. Perfect things before using them and don't tell people what you are doing. Another thing to be careful of are operators who look at the files in your account. A simple way to deal with those people is to encrypt/code those files and keep decoded copies online for as little time as possible. If you can't stay at a terminal to have the password catcher program running don't despair because I will show you how to detach jobs later. My biggest piece of advice is always stalk and watch the target system first for a while. Get to know what most or all of the programs in the project 1 accounts do, and only after you are sure should you try and pull a stunt like I decribe. Once you have a privilaged password you must use your own imagination to how you use it. Remember always be security conscious. Don't take unneccessary risks and by the time you get to a privilaged account you should know all the SYS functions possible and how to use them to your advantage. If you system keeps login and logout records remember to edit them, because depending on what you do, your activities should be able to go on undetected!

DETACHING JOBS.

There is one way a non-privilaged user can detach a job. This is done using pseudo keyboards. You will need to read the PROGRAMMING MANUAL (the RSTS/E bible for hackers!) and write yourself a program to almost simulate a job except you do it to your terminal and you don't log the passwords etc..

When you are logged into your account and you run this program it will look on your terminal like you are logged out again and you will have to log into the system again. Log into the account which you want to detach the job from and you must have access to the program from this account. You should execute the program you want detached then have a

special key sequence that will close the channel to the pseudo keyboard. You detacher program will finish and that is how you do it. ut you say that doesn't work, it just kills the job you started when you closed the channel to the pseudo keyboard, but you didn't wait for the most important piece! When opening the pseudo keyboard you must have included MODE 1 in the OPEN statement which tells the system to detach the job when you close the channel instead of killing the job. When you do a SYSTAT you will see your job running detached. Don't forget opertors mightn't like you detaching too many jobs so do it when they aren't around. For a job to terminate itself you may try getting it to run LOGOUT, but when it tries to output something like a message saying your disk space or have a nice afternoon it will sit there helpless in a HB wait state until someone attaches to it (like you or an operator) or an operator kills it. To get around this take notice of the message you get when you log in and you have a job detached. The system tells you that and asks if you want to attach. So you what you do is make you detached program open a pseudo keyboard in a mode that won't detach it and get that job to log into your account. Don't worry about entrusting your password to the program as others can't obtain it. Anyway when you get the new job to log in make it attach to the job you wanted killed, then when your original program closes the channel to the pseudo keyboard the job running on the pseudo keyoard is killed and it effectively kills itself as it attached to itself in a manner of speaking, and thus the job disappears! (Well it worked on the system I tried it on)

RECOMMENDATIONS.

When you attempt to do all of this I advise you to get hold of (buy ($20) or borrow) the RSTS/E PROGRAMMING MANUAL. There would be at least one with the system and are also available from DEC (Digital Equipment Corporation).

Finally, non of the above methods and techniques are guaranteed as they can be removed or altered by the system operators. All of the techniques are valid and are not bugs in the operating system. Whether your operator knows about them or what they can do is a different matter! Anyhow have fun, RSTS/E is a good operating system, and don't do anything that I wouldn't do!

:-------------------------THE-END-------------------------: