Ghiribizzo’s Cracking Tutorial

OCU 1 : Nag Screen Trainer

Open Cracking University

This is the first in a series of tutorials made especially for the OCU. Rather than find programs which demonstrate nag screens, I decided to write my own in assembly. This has several advantages: download time, suitability for tutorial and legality.

PGP and Signed Tutorials

My tutorials and programs should be signed electronically using PGP. PGP 5 supports DSS/Diffie-Hellman keys. These keys are not supported by previous versions of PGP.

You should check the signature to make sure that the tutorial and especially its program files have not been tampered with. All cracks, tutorials and zip files I release will be signed. This will prevent tampering and will hopefully reduce the chances of viral infection.

My signature will also be the only way you can identify me as my email address will often change.

My Web Site:	http://Ghiribizzo.home.ml.org
My Email:	Ghiribizzo@geocities.com
My Backup Email:	Ghiribizzo@hotmail.com

Nag Screen Trainer

Here’s the file to be cracked (newbies you need to copy the section below and UUdecode it!):

```
begin 644 trainer1.com
[illegible]
end
```

This lesson is intended to allow you to familiarise yourself with using a debugger. I suggest using Turbo Debug for this. Those of you who are familiar with assembly can try using only Hiew. Those who have had experience with cracking should use only Hiew.

Rules

1. Don’t produce a dead listing (for those learning to use a debugger). To be honest a dead listing isn’t necessary to crack this - Hiew is more than adequate.
2. When patching the file with a hex editor, you may not change bytes 1F1-1FB inclusive. This is to prevent simply nopping of the last nagscreen.
3. When I say no nops, I don’t just mean no 090h bytes. No overwriting unwanted code with filler byte sequences (e.g. inc ax, dec ax etc.). Circumvent the code using jumps, calls, rets, etc.

That’s all, just go ahead and crack it now.

Feeling the Code

I want to teach you to move on from just mechanistic cracking and understand more of the program and the programmer - it is surprising what you can find out about how the program was written and the personality of the programmer! Take a closer look at my program. There are some things which I altered. Questions:

1. What have I altered? (clue: look in the data area)
2. Why have I left it like that?
3. What does that tell you about me? (a four letter word beginning with L)

Parting Shots

One of the bytes changed is now 0Dh; the password for the following block of PGP is what that byte was originally (written in decimal). I learnt something about how the compiler works. I’m sure you can figure out what that was from the PGP message. Clue, I changed one of the procedures to save 3 bytes of code - I could do this because of where the procedure was located (a clue for question 2, above!).

```
-----BEGIN PGP MESSAGE-----
Version: 2.6.2i

pgAAAj0u3079CNLKvv59E1hXSASLzYIIINnVfjY+aFqwXIQbi+Kz1+OIjkGs9T1W
jmf96k+782/UF6YiwQLnx15YheaymNHj8Iof08RFvSN0ozJaWQ+QyntNeYQ2n5lZ
W691eHJd3VonPBaMpV4hbLSCByEI/oeWfXSL44pIanruNZTvvewo04i7VwwDONXe
Yj4Woxcu1E9HbVwBlHWT76Hm/gVPttFJbS1j/U0aNo+EFQRX1z2sg4slha0/eYPV
SzNp9MWySRsZU1qDnHK6fnt8W3eywOGogMOLwAzORsqAx5CqMQf+1aNc+SyFbmAm
p/vUWOcTEmMXBCE9kbky46GzXgkjvE6ODR6WvrkPrhbsFVuqrz1uMHAo6N/bCk3h
V2KXMHaIJUovK+v/wvZLKMXeZ6VIV3QGCFh1Ewp5h8Q1L3hWXYXyZXKslOfDWEsy
hERl8cp0uvjQ54+f9ha64s+9+ke02RfFN2aBTxT/zfzI/jhyhdDBP1yIJskIu027
OeRUVwdThpF46jTILj5cugbob+ZFTVaklox951kdTp+RlZn2mur50S8GQ4TjORTn
4HO+vXTUGLgFnqZ2LU1kyams2MNilXn5mMWZWQ/WOWjQ5bRDyHpQPZRQsi5XyD6a
mXJX05koBOPwyd3pQYgNTamDv942XXdeLCbVUpWBgy4LAXPXBwjJRBzNx9RhzJ6K
/eAyv+ao6pOtRzwCAciirFChsHs14nCBaJmKV1fo5Z1/bOOp0nzyBkOaqp3Qh+aZ
XmM=
=iXzV
-----END PGP MESSAGE-----
```