Megatouch Force 2008.5 Instructions

- Assumes the following software has been installed: `V25_00T080320_1114`

Precautions

Follow these instructions at your own risk. These instructions worked for me, but cannot guarantee the same outcome for others, or that there will be no issues at a later point in time due to the tampering of system files (eg: in-game security checks, maintenance checks, etc).

The instructions also require running with administrator privileges; typos when following instructions can cause serious negative effects, resulting in the machine unable to boot the game or boot at all. ***It is strongly recommended you either***

1. Follow these instructions with a fresh installation on a separate hard drive.
2. Have the DVD installation media ready to use in case the machine goes into an unbootable state.
3. Make a backup of the hard drive before starting (eg: plug the drive into another computer and backup via a tool such as CloneZilla).
4. Make backups of the binaries before modifying them (instructions below do this).

Getting USB Keyboard to work

If the computer is unresponsive to the USB keyboard (ie: the game starts while you're holding "alt" in the following step), you'll need to enable legacy USB support in the BIOS.

1. Start the machine, and press delete to go into BIOS options.
2. Integrated Peripherals → USB Keyboard Support → set to "Enabled".
3. Press F10 to save changes and reboot the machine.

Note that, as part of some software process, this setting may get reverted back to "Disabled" in the future. If this happens, repeat these steps.

Obtaining access to the shell

1. Turn the machine on, and hold "alt" until you get to the bootloader screen, then enter the following command:
   ```
   linux-V25_00 3
   ```

2. When prompted to login, use user "maxx" and password "maxx".

Altering the graphics startup process

1. Ensure you're working with the right version of the binary.
   ```
   md5sum /usr/local/bin/start
   # Should return d572714d13d92c3255070dfa0f520da6
   ```

2. Issue the following commands:
   ```
   cp ~/.xinitrc ~/.xinitrc.bak
   echo -e "mount -o rw,remount /\nxterm" | cat - ~/.xinitrc.bak > ~/.xinitrc
   sync
   ```

3. Restart the computer; it will boot into a graphical terminal.
   ```
   /sbin/reboot
   ```

Obtaining the game code

Similar to the 2011 instructions, the main binary is encrypted in a wrapper binary. We need to decrypt the main binary and replace the wrapper binary with it.

1. Check you're able to type in the terminal. If not, touch the terminal and then you should be able to type.

2. Patch the wrapper binary to not delete the main binary once executed:
   ```
   cp /usr/local/bin/start /usr/local/bin/start.bak  # Nice to keep a backup.
   perl -e 'print "\x90" x 5' | dd bs=1 count=5 seek=2941 of=/usr/local/bin/start conv=notrunc
   md5sum /usr/local/bin/start
   # Should return 9921dcf1abe58af7389c821bd7ffd3f9
   ```

3. Run the game.
   ```
   /usr/local/bin/start
   ```

4. At the game selection screen, restart the computer; it will boot into a graphical terminal.

5. Replace the original "start" binary with the extracted "dstart" binary.
   ```
   cp /tmp/dstart /usr/local/bin/dstart.bak  # Nice to keep a backup.
   cp /tmp/dstart /usr/local/bin/start
   chmod 4755 /usr/local/bin/start
   ```

```
md5sum /usr/local/bin/start
# Should return 810c1806099e4c0f4590573c7276e30e
```

Extract the encrypted data from your actual key

1. Start the binary in debugging mode.
   ```
   gdb /usr/local/bin/start
   ```

2. Setup a breakpoint in USBIO::ReadDS1995KeyData after key data is read in.
   ```
   break *0x8346d1a
   ```

3. Instruct gdb to dump the key data to file once the breakpoint is hit.
   ```
   command
   dump memory /.key $ebp-0x40c $ebp-0xc
   continue
   end
   ```

4. Start the game.
   ```
   run
   ```

5. Once at the game selection screen, restart the machine.

6. You can confirm the data was written to file with the following command (it should show 1.0K bytes written).
   ```
   ls -lash /.key
   ```

Patching key checking functions to always return “OK” or “Success”

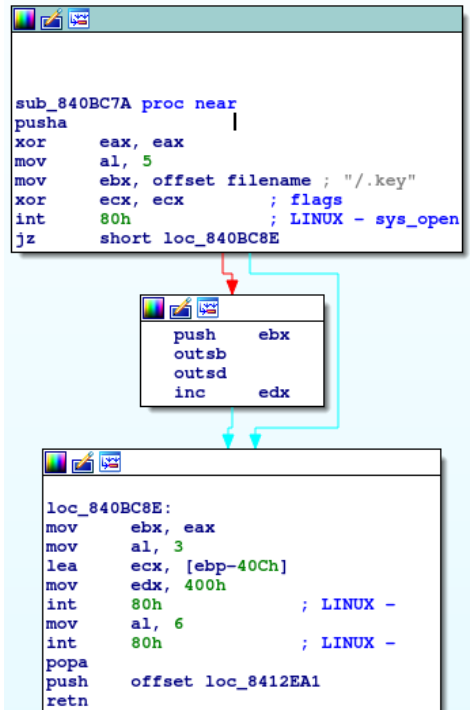
1. Patch KeyManager::Check() to always return true.
   ```
   perl -e 'print "\x31\xC0\x40\xC3"' | dd bs=1 count=4 seek=3102984 of=/usr/local/bin/start conv=notrunc
   ```

2. Patch USBIO::USBConfirmKeyID() to always return true.
   ```
   perl -e 'print "\x31\xC0\x40\xC3"' | dd bs=1 count=4 seek=3139782 of=/usr/local/bin/start conv=notrunc
   ```

Writing machine code to read the the stored key data from disk

The code below adds a function (shown on the right) to read the data stored in the `/.key` file when retrieving data from the security key. We'll be overwriting the actual read from the key with this code.

```
perl -e 'print "\x2F\x2E\x6B\x65\x79\x00"' \
     | dd bs=1 count=6 seek=4554400 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x60\x31\xC0\xB0\x05\x8D\x9B\xDC\x1C\xFC"' \
    | dd bs=1 count=10 seek=3140629 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xFF\x31\xC9\xCD\x80\x74\x04\x53\x6E\x6F"' \
    | dd bs=1 count=10 seek=3140639 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x42\x89\xC3\xB0\x03\x8D\x8D\xF4\xFB\xFF"' \
    | dd bs=1 count=10 seek=3140649 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xFF\xBA\x00\x04\x00\x00\xCD\x80\xB0\x06"' \
    | dd bs=1 count=10 seek=3140659 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xCD\x80\x61\xE9\xA0\x00\x00\x00"' \
    | dd bs=1 count=8 seek=3140669 of=/usr/local/bin/start conv=notrunc
```



Check that the changes so far were applied properly.

```
md5sum /usr/local/bin/start
# Should return 6903f1d2e528f6cc5ee3fd37eb2da416
```

If you get a different result, it means one of the previous instructions was mistyped. It is strongly recommended to copy the original dstart binary back and redo the instructions from the previous two sections (starting with "Patching key checking functions …").

```
# Recover the original dstart binary to start over again, if the above md5sum doesn't match.
cp /usr/local/bin/dstart.bak /usr/local/bin/start
```

Patching USBIO::ReadKeyId()

Now we'll hard-code the key's serial ID into the binary, instead of having the binary read it from the key.
The steps below assume the serial key on the bottom of your physical key looks like the following:

```
11              88
22 33 44 55 66 77
```

1. Path the method to read the hard-coded key:
   ```
   perl -e 'print "\x74"' | dd bs=1 count=1 seek=3140193 of=/usr/local/bin/start conv=notrunc
   ```

2. Hard-code the key:
   ```
   perl -e 'print "\x88\x77\x66\x55"' | dd bs=1 count=4 seek=3140218 of=/usr/local/bin/start conv=notrunc
   perl -e 'print "\x44\x33\x22\x11"' | dd bs=1 count=4 seek=3140225 of=/usr/local/bin/start conv=notrunc
   ```

Unfortunately, because everyone's serial number is different, we cannot validate these changes using md5sum.

Revert initial setup so machine boots up into the game

1. Run the following commands:
   ```
   cp ~/.xinitrc.bak ~/.xinitrc
   sync
   ```

2. Poweroff the machine.
3. Remove your key.
4. Turn back on, and enjoy!

If there's no games on bootup

If you get to the main menu and there are no game categories to select, it's likely that an incorrect serial key was provided. Double check your serial key (taking a photo and zooming on the image is useful) and rerun the steps to patch "ReadKeyId()" if you found that the original serial key was incorrect.

If at any point you want to restore the original game binary, you can do so by following the instructions to get back into a terminal, then coping the backup to its original location:

```
# Restore the original game binary.
cp /usr/local/bin/start.bak /usr/local/bin/start
```