Megatouch Force 2006.5 Instructions

- Assumes the following software has been installed: `V21_51T070522_1743`

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.

Gaining Root Privileges

Unlike the software from 2007 up, this version has two separate accounts - "root" and "maxx". Furthermore, "maxx" has user/group id 100, which is not root. The password of root is not the same as "maxx", so we're unable to log into root. However, we can gain root privileges by doing the following:

1. On machine startup, hold "alt" until you get to a red bootloader screen, then enter the following command:
   ```
   linux init=/bin/sh
   ```

2. Once you have access to the terminal (“sh-3.00#” shows), make sure you’re following the right instructions by checking the binary.
```
md5sum /usr/local/bin/start
# Should return 5807e7940eea780757231788e790f790
```

3. change the user and group id of “maxx” to root by enter the following commands:
```
mount -o rw,remount /
sed -i.bak 's/100:100/0:0/g' /etc/passwd
sync
/sbin/shutdown -h -n now
```

4. Reboot the machine by pressing ctrl + alt + delete. If this does not work, manually shutdown the computer.

Obtaining access to the shell

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

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

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. Make the hard drive writable:
```
mount -o rw,remount /
```

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 f9d0cfd0edfa4d2bcce2cdc22a49e089
```

3. Run the game. It should hang shortly after with the error message “FATAL ERROR: CANNOT ACCESS I/O BOARD”.
```
/usr/local/bin/start
```

4. Hold Ctrl + Alt and press F2 to gain access to another terminal. Log in with user “maxx” and password “maxx”.

5. Replace the wrapper binary with the main 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 b993682e384289f16c3cb6c306602291
```

Altering the graphics startup process

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

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

Extract the encrypted data from your actual key

Check that you’re able to type in the terminal. If not, touch the terminal, and try again.

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

2. Setup a breakpoint in USBIO::USBReadKeyData after key data is read in.
```
break *0x8372a32
```

3. Run the program in debugging mode.
```
run
```

4. Once at breakpoint, **continue execution of the program again**. *(In my experience, this code was run twice (the first run appears to be a failed attempt, from examining the KeyManager::Read() method that calls USBIO::USBReadKeyData). If you find the program boots into the game selection screen without hitting this code a second time, then rerun gdb and skip this step).*

```
continue
```

5. Once at the breakpoint again, dump the data from the key to file.
```
dump memory /.key $ebp-0xfc $ebp-0x10
```

6. Exit the program.
```
quit
```

7. Confirm the data was written to file (should be 236 bytes).
```
ls -lash /.key
```

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

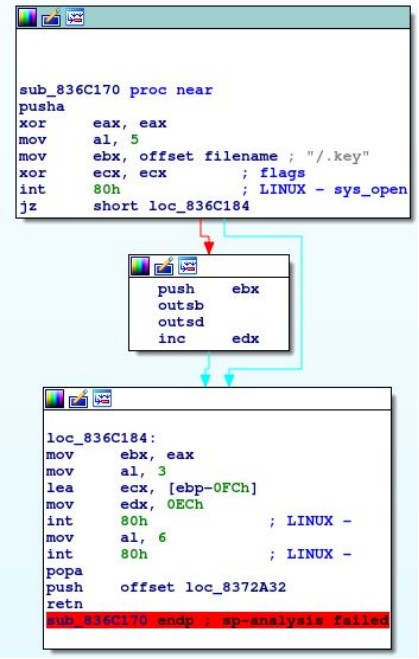
1. Patch KeyManager::Check() to always return true.
```
perl -e 'print "\x31\xC0\x40\xC3"' | dd bs=1 count=4 seek=3293538 of=/usr/local/bin/start conv=notrunc
```

2. Patch USBIO::USBConfirmKeyID() to always return true.
```
perl -e 'print "\x31\xC0\x40\xC3"' | dd bs=1 count=4 seek=3318992 of=/usr/local/bin/start conv=notrunc
```

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

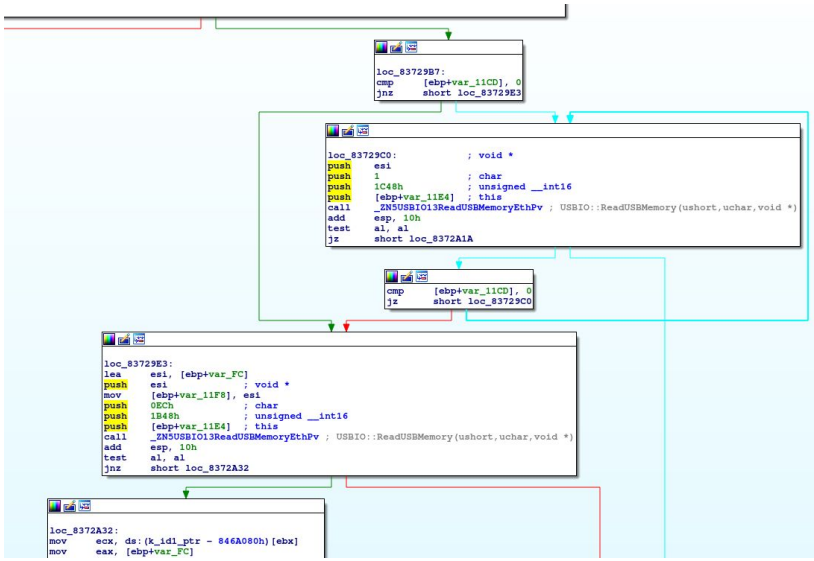
The code below adds a function (shown on the right) to read the data stored in the `/.key` file when retrieving data from the security key. We’ll be adding this function to the now-unused code leftover from KeyManager::Check().

```
perl -e 'print "\x2F\x2E\x6B\x65\x79\x00\x00\x00\x00\x00\x60\x31\xC0\xB0\x05\xBB"' | \
    dd bs=1 count=16 seek=3293542 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x66\xC1\x36\x08\x31\xC9\xCD\x80\x74\x04\x53\x6E\x6F\x42\x89\xC3"' | \
    dd bs=1 count=16 seek=3293558 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xB0\x03\x8D\x8D\x04\xFF\xFF\xFF\xBA\xEC\x00\x00\x00\xCD\x80\xB0"' | \
    dd bs=1 count=16 seek=3293574 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x06\xCD\x80\x61\x68\x32\x2A\x37\x08\xC3\x68\x61\x63\x6B\x65\x64"' | \
    dd bs=1 count=16 seek=3293590 of=/usr/local/bin/start conv=notrunc
```



Patching USBIO::USBReadKeyData() to read the key data from disk

Now, we'll replace the data-reading code from USBReadKeyData to instead call the newly created function above. The part of the method we'll be patching is shown on the right.



The "loc_83729C0" block reads one byte of data in a loop. We did not capture this in the previous steps, nor does it seem needed.

The "loc_83729E3" block is what reads the configuration data from the key.

We'll call the function we created at "loc_83729B7". The function is setup to return to "loc_8372A32" when completed. Once patched, "loc_83729B7" will look like the following:

```
loc_83729B7:
lea     esi, [ebp+var_FC]
mov     [ebp+var_11F8], esi
push    offset sub_836C170
retn
```

The extra lines before the call (taken from the "loc_83729E3") block are essential, otherwise the program will seg fault when run.

Run the following to patch the function:

```
perl -e 'print "\x8D\xB5\x04\xFF\xFF\xFF\x89\xB5\x08"' | dd bs=1 count=9 seek=3320247 of=/usr/local/bin/start conv=notrunc
perl -e 'print "\xEE\xFF\xFF\x68\x70\xC1\x36\x08\xC3"' | dd bs=1 count=9 seek=3320256 of=/usr/local/bin/start conv=notrunc
```

Now check that the changes so far have been properly added.

```
md5sum /usr/local/bin/start
# Should return 2528d11528edfc9f67a8281211fb8e8b
```

Patching USBIO::USBReadKeyId()

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=3319368 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=3319399 of=/usr/local/bin/start conv=notrunc
   perl -e 'print "\x44\x33\x22\x11"' | dd bs=1 count=4 seek=3319406 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
   cp /etc/passwd.bak /etc/passwd
   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.