Megatouch Force 2003 Instructions

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). These instructions are also more complex than those for versions 2004 and up.

The instructions 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 / unplayable 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).

Prerequisites

Unlike instructions for years 2006 and up, not all the steps here can be done with a USB keyboard. ***A PS/2 keyboard is needed, or a USB to PS/2 adapter!*** Otherwise, you will not be able to enter commands in the terminal when a graphical environment is brought up. In my experience, only a low-powered wired USB keyboard worked when using a USB to PS/2 adapter.

Even with a low-powered USB keyboard connector on a PS/2 adapter, I experienced issues with typing in the graphical terminal. If this happens, I found the following to work:

1. Plug the usb keyboard in a usb port, and press several buttons (nothing happens).
2. Plug the usb keyboard back into the PS/2 adapter, it should work again.

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 ("init-2.04#" shows), make sure you're following the right instructions by checking the binary.
```
md5sum /usr/local/bin/start
# Should return 750c1cf606a392ac5fdde8f08d4e04d6
```

3. Change the user and group id of "maxx" to root by entering the following commands:
```
mount -n -o rw,remount /
cp /etc/passwd /etc/passwd.bak  # Keep a backup.
sed 's/100:100/0:0/g' /etc/passwd.bak > /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 -n -o rw,remount /
```

2. Set up the machine so that on sequential boots a graphical environment is loaded with a terminal (this is needed for upcoming steps):
```
cp ~/.xinitrc ~/.xinitrc.bak  # Keep a backup.
echo -e "mount -n -o rw,remount /\nxterm" | cat - ~/.xinitrc.bak > ~/.xinitrc
```

3. Patch the wrapper binary to not delete the main binary once executed:
```
cp /usr/local/bin/start /usr/local/bin/start.bak  # Keep a backup.
perl -e 'print "\x90" x 5' | dd bs=1 count=5 seek=3675 of=/usr/local/bin/start conv=notrunc
md5sum /usr/local/bin/start
# Should return 005b2d53a3ee434843eda0a9a7af49b5
```

4. Run the game. After a few seconds, the machine should hang at a black screen.
   ```
   /usr/local/bin/start
   ```

5. Wait about 10-20 seconds on the black screen, restart the machine. It will automatically boot into a terminal in a graphical environment. If you're using a USB keyboard on a PS/2 adapter and cannot type, try the instructions in the "Prerequisites" section.

6. ***Make a backup of the 'dstart' binary, this is needed for a later step. Do not skip this step.***
   ```
   cp /tmp/dstart /usr/local/bin/dstart.bak
   ```

7. Replace the wrapper binary with the main binary.
   ```
   cp /tmp/dstart /usr/local/bin/start
   chmod 4755 /usr/local/bin/start
   md5sum /usr/local/bin/start
   # Should return b09673ace38b17f351181b981c84dabb
   ```

Extract the encrypted data from your actual key

Unlike the instructions for versions 2004 and up, we cannot rely on "gdb" command to dump the required memory contents (the key's data and the decryption value) to files. Instead, we'll patch in a function that will populate these files for us.

Patch the program to dump the key's contents to a "/.key" file and the decryption value to a "/.decrypt" file (note: each command from "perl" to "conv=notrunc" can be run on a single line, the `\` character is used to split up a command across multiple lines):

```
perl -e 'print "\x60\x31\xC0\xB0\x05\xBB\x65\x7F\x16\x08"' | \
    dd bs=1 count=10 seek=1179392 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x31\xC9\xB1\x01\xB5\x06\x31\xD2\xB6\x07"' | \
    dd bs=1 count=10 seek=1179402 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xCD\x80\x89\xC3\xB0\x04\x8D\x8D\xB8\xFE"' | \
    dd bs=1 count=10 seek=1179412 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xFF\xFF\xBA\xEC\x00\x00\x00\xCD\x80\xB0"' | \
    dd bs=1 count=10 seek=1179422 of=/usr/local/bin/start conv=notrunc
```

```
perl -e 'print "\x06\xCD\x80\x31\xC0\xB0\x05\xBB\x6B\x7F"' | \
    dd bs=1 count=10 seek=1179432 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x16\x08\x31\xC9\xB1\x01\xB5\x06\x31\xD2"' | \
    dd bs=1 count=10 seek=1179442 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xB6\x07\xCD\x80\x89\xC3\xB0\x04\x8B\x75"' | \
    dd bs=1 count=10 seek=1179452 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x08\x8D\x8E\x80\x03\x00\x00\xBA\x04\x00"' | \
    dd bs=1 count=10 seek=1179462 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x00\x00\xCD\x80\xB0\x06\xCD\x80\x61\x8D"' | \
    dd bs=1 count=10 seek=1179472 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x85\xB8\xFE\xFF\xFF\x68\xD6\xE0\x28\x08"' | \
    dd bs=1 count=10 seek=1179482 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xC3\x2F\x2E\x6B\x65\x79\x00\x2F\x2E\x64"' | \
    dd bs=1 count=10 seek=1179492 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x65\x63\x72\x79\x70\x74\x00"' | dd bs=1 count=7 seek=1179502 of=/usr/local/bin/start conv=notrunc
```

Now, check that the program was patched properly.

```
md5sum /usr/local/bin/start
# Should return 731d96d721c91679feb24f80597ce348
```

If the md5sum does not match, restore the program from the backup, and re-enter the commands again.

```
cp /usr/local/bin/dstart.bak /usr/local/bin/start
chmod 4755 /usr/local/bin/start
```

Once the md5sum matches, create empty files for the patched program to write to, and run the program.

```
touch /.key
touch /.decrypt
/usr/local/bin/start
```

The program should crash after a few seconds and bring you back to the terminal.

Confirming key data and decryption key

Let's make sure the data was properly saved to file. Check that the /.key file was made.

```
wc -c /.key
```

The output of the above command should be `236 /.key`.

Check the contents of the /.decrypt file.

```
hexdump /.decrypt
```

The output should look something like the following (instead of `1122 3344`, you should see random-ish characters, like `7b8f 7a62`):

```
00000000 1122 3344
00000004
```

Once the contents of /.key and /.decrypt are confirmed, ***restore the 'dstart' binary so that it no longer crashes:***

```
cp /usr/local/bin/dstart.bak /usr/local/bin/start
chmod 4755 /usr/local/bin/start
md5sum /usr/local/bin/start
# Should return b09673ace38b17f351181b981c84dabb
```

Patching USBIO::USBConfirmKeyId() to always return "OK" or "Success"

This method checks that the serial was read from the security key properly. Let's hardcode this method to always return a success.

```
perl -e 'print "\x31\xC0\x40\xC3"' | dd bs=1 count=4 seek=1177264 of=/usr/local/bin/start conv=notrunc
```

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

We need a function that can read the "/.key" file and load it into memory. Let's add this method right after the patched ConfirmKeyId method:

```
perl -e 'print "\x2F\x2E\x6B\x65\x79\x00"' | dd bs=1 count=6 seek=1177268 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x60\x31\xC0\xB0\x05\xBB\xB4\x76\x16\x08"' | \
    dd bs=1 count=10 seek=1177274 of=/usr/local/bin/start conv=notrunc
```

```
perl -e 'print "\x31\xC9\xCD\x80\x74\x04\x53\x6E\x6F\x42"' | \
    dd bs=1 count=10 seek=1177284 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x89\xC3\xB0\x03\x8D\x8D\xB8\xFE\xFF\xFF"' | \
    dd bs=1 count=10 seek=1177294 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xBA\xEC\x00\x00\x00\xCD\x80\xB0\x06\xCD"' | \
    dd bs=1 count=10 seek=1177304 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x80\x61\x68\x00\x7F\x16\x08\xC3"' | \
    dd bs=1 count=8 seek=1177314 of=/usr/local/bin/start conv=notrunc
```

Now patch the USBReadKeyData() method to read data from /.key (calling the above function) instead of from the physical security key:

```
perl -e 'print "\x68\xBA\x76\x16\x08\xC3"' | dd bs=1 count=6 seek=1178913 of=/usr/local/bin/start conv=notrunc
```

If you entered the commands properly so far, you should see the following with the md5sum command:

```
md5sum /usr/local/bin/start
# Should return 3091b328a157fb8436cd299575364950
```

Patching USBIO::GetDecryptVal()

In order for the program to read the /.key contents properly, we must also hardcode the decryption value.

The following patches the method to return that hard-coded value (assuming the output of `hexdump /.decrypt` was `1122 3344`):

```
perl -e 'print "\x55\x89\xE5\x8B\x45\x08"' | dd bs=1 count=6 seek=1181216 of=/usr/local/bin/start conv=notrunc
perl -e 'print "\xC7\x80\x80\x03\x00\x00"' | dd bs=1 count=6 seek=1181222 of=/usr/local/bin/start conv=notrunc
perl -e 'print "\x22\x11\x44\x33\x5D\xC3"' | dd bs=1 count=6 seek=1181228 of=/usr/local/bin/start conv=notrunc
```

Remember to replace **`\x22\x11\x44\x33`** with your specific value (eg: if your decrypt value was `7b8f 7a62`, then **`\x22\x11\x44\x33`** should be **`\x8F\x7B\x62\x7A`**.

Unfortunately, since the decryption value changes each time, we're unable to verify these changes via `md5sum`.

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
```

First, make sure the hardcoded serial key is always used.

```
perl -e 'print "\xEB"' | dd bs=1 count=1 seek=1177779 of=/usr/local/bin/start conv=notrunc
```

Second, add your hardcoded serial to the method.

```
perl -e 'print "\x56\x8B\x75\x0C\xC7\x06\x88\x77\x66\x55"' | \
    dd bs=1 count=10 seek=1177840 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\xC7\x46\x04\x44\x33\x22\x11\x5E\x90\x90\x90"' | \
    dd bs=1 count=11 seek=1177850 of=/usr/local/bin/start conv=notrunc
```

Remember to replace **\x88\x77\x66\x55** and **\x44\x33\x22\x11** with your specific key's serial.

Confirm the game boots without the key

1. Make sure filesystem changes are made.
   ```
   sync
   ```

2. Poweroff the machine.
3. Remove your key.
4. Turn the machine back on.
   At the terminal, start the game.
   ```
   /usr/local/bin/start
   ```

5. If the game menu selection screen appears, it's done! Follow the "Cleanup" instructions. Otherwise, follow the handling errors section.

Cleanup

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

2. Restart the machine.

Handling “Invalid key for version …”, crashes, hangs, or other errors

It’s very likely there was a typo when issuing one of the many commands throughout this guide. It’s likely in the decryption values or the key id misread (can be hard to read if there’s significant wear on the key).

Restore the original “dstart” binary:

```
cp /usr/local/bin/dstart.bak /usr/local/bin/start
chmod 4755 /usr/local/bin/start
```

Remove the added memory files:

```
rm /.key
rm /.decrypt
```

To try the patching again, start over from the section called “Extract the encrypted data from your actual key”.
To restore the machine back to normal (requiring the key), go to the “Cleanup” section.