Megatouch Force 2004 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 slightly more complex than those for versions 2006 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.

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 e33cdc573fac952e54830b5efa2ef24a
   ```

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  # Nice to keep a backup.
   perl -e 'print "\x90" x 5' | dd bs=1 count=5 seek=4195 of=/usr/local/bin/start conv=notrunc
   md5sum /usr/local/bin/start
   # Should return cc0d9d3c013ac2c32584745b8035fdcf
   ```
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.

6. 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 4891fbd9dbfd1926871c0cebc79a4faa
```

Extract the encrypted data from your actual key

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

2. The debugger will hang due to SIG32 signals coming in. Issue the following command to prevent these hangs.
```
handle SIG32 nostop
```

3. Setup a breakpoint in USBIO::USBReadKeyData after key data is read in. Then dump the contents of the key into a “/.key” file.
```
break *0x816e9a0
command 1
dump memory /.key $ebp-0x11c $ebp-0x30
continue
end
```

4. Unlike the instructions from 2006 and up, the key data alone is not enough to store to file. There is a 4-byte “decryption key” that needs to be stored alongside the key’s data, otherwise the program will fail to properly decode the data. Store this decryption key into a “/.decrypt” file.
```
break *0x816eafc
command 2
dump memory /.decrypt $edx+0x380 $edx+0x384
continue
end
```

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

1. The machine will either boot up to the games menu, or hang on the loading screen. Either way, wait about a minute then restart the machine.

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
```

If either the “wc” or “hexdump” commands give a `No such file or directory` error, you’ll need to rerun all commands in the previous section, starting with `gdb /usr/local/bin/start`. Even if you have one of the files, you’ll need to repopulate both (as their outputs are different with each run of gdb).

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=1205200 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 contents of the /.key file and load it into the proper memory location. Let’s add this method right after the patched ConfirmKeyId method (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 "\x2F\x2E\x6B\x65\x79\x00"' | dd bs=1 count=6 seek=1205204 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x60\x31\xC0\xB0\x05\xBB\xD4\xE3\x16\x08"' | \
    dd bs=1 count=10 seek=1205210 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=1205220 of=/usr/local/bin/start conv=notrunc
```

```
perl -e 'print "\x89\xC3\xB0\x03\x8D\x8D\xE4\xFE\xFF\xFF"' | \
    dd bs=1 count=10 seek=1205230 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=1205240 of=/usr/local/bin/start conv=notrunc

perl -e 'print "\x80\x61\x68\xA0\xE9\x16\x08\xC3"' | \
    dd bs=1 count=8 seek=1205250 of=/usr/local/bin/start conv=notrunc
```

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

```
perl -e 'print "\x68\xDA\xE3\x16\x08\xC3"' | dd bs=1 count=6 seek=1206409 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 1f0528b2a03c481bae1a6b0e25c2e123
```

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=1208000 of=/usr/local/bin/start conv=notrunc
perl -e 'print "\xC7\x80\x80\x03\x00\x00"' | dd bs=1 count=6 seek=1208006 of=/usr/local/bin/start conv=notrunc
perl -e 'print "\x22\x11\x44\x33\x5D\xC3"' | dd bs=1 count=6 seek=1208012 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, 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=1205661 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=1205696 of=/usr/local/bin/start conv=notrunc
perl -e 'print "\xC7\x46\x04\x44\x33\x22\x11\x5E"' | \
    dd bs=1 count=8 seek=1205706 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.
5. At the terminal, start the game.
   ```
   /usr/local/bin/start
   ```
6. 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.

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 with GDB. To restore the machine back to normal (requiring the key), go to the “Cleanup” section.