

Voodoo XNU Kernel

User Documentation
Revision 1



6 Dec 2008

Table of contents

Introduction.....	I
System requirements.....	I
Using the kernel.....	I
Who should use this kernel?.....	I
Installing the kernel automatically.....	2
Installing manually.....	2
Kernel features.....	2
SSE3 emulation.....	3
On-the-fly opcode patching.....	3
64-bit support on AMD.....	3
Improved real-time clock.....	3
Kext black-listing.....	4
Miscellaneous changes.....	4
Troubleshooting.....	4
Kernel boot-time arguments.....	4
fsb=.....	5
busratio=.....	5
busratiopath=.....	5
tcsync=.....	5
c1ramp=.....	5
kprintf=I.....	6
std_dyld=.....	6
patcher_opts=.....	6
-notscverify.....	6
-tscpanic.....	6

-amd.....	6
-emulateintel.....	6
Known Issues.....	7
64-bit applications will not run on AMD.....	7
I am getting occasional or consistent application crashes.....	7
Running Marvin's AMD utility crashes the system.....	7
I get a kernel panic during heavy disk activity.....	7
USB devices don't respond or hot-plugging them does not work.....	7
VMware (OS X) does not run when running on Voodoo kernel.....	7
Some of my system preferences seem to have been reset.....	7
My Pentium 4 is reporting incorrect cache information.....	7
My AMD / Pentium D / Celeron processor is being reported as an Intel Core / Core 2 processor!.....	8
System Profiler crashes.....	8
I get a kernel panic on booting, with the message "Cannot find driver for platform ACPI".....	8
My computer cannot sleep / cannot resume after sleeping / cannot sleep or resume in 64-bit mode.....	8
My AMD Phenom processor is being detected as having 100 MHz FSB frequency and twice the bus ratio.....	8
My system boots up to the blue screen and doesn't proceed further.....	8
Frequently asked questions.....	8
Can I update straight from Apple Software Update if I use this kernel?.....	8
Do I need to remove AppleIntelCPUPowerManagement.kext or install NullCPUPowerManagment.kext or a similar disabler?.....	9
Does the kernel have in-built decryption?.....	9
Can you tell me a solution for decrypting encrypted binaries?.....	9
Where can I find a "stock" System.kext / AppleSMBIOS.kext / other kext?.....	9
Is it beneficial to do offline CPUID patching to avoid on-the-fly patching and gain some performance?.....	9
Can I use the kernel on an already-patched AMD system?.....	9
Will you help me with installing retail on my AMD system?.....	9
Will the source-code for the SSE3 emulator be available?.....	9
Is this kernel legal?.....	9
I am a distro-maker. Can I include your kernel on my DVD?.....	9
In conclusion.....	10
Core contributors.....	10
Other contributors.....	10
How can I contribute?.....	10
Suggested kexts.....	11
Getting in touch.....	11

Introduction

The Voodoo XNU kernel is a fork of the XNU kernel shipped with Apple® Darwin operating system – the open-source foundation of Mac OS X.

Voodoo kernel aims to expand the range of processors supported by XNU, and provide technologies to enable Darwin and associated applications to run on a variety of x86 platform combinations. Technologies included in the Voodoo kernel range from emulation of SSE3 instructions, to on-the-fly patching of loadable binaries. The kernel also fixes issues with multi-processor timing synchronization, SpeedStep™ support and sleep/resume.

System requirements

The Voodoo kernel supports almost all recent, and many not-so-recent processors. The list of processors the kernel has been tested on includes :

Intel® processors:	AMD® processors:
• Pentium 4 / D / M	• Athlon 64 and X2
• Celeron	• Phenom
• Core Solo / Duo	• Opteron
• Core 2 range	• Sempron
• Atom	

Most SSE2 capable or better processors should work with the kernel. Intel Core i7 is known to not work yet. You will also need to use an EFI boot-loader. Chameleon, netkas PC_EFI and dfe boot-132 should work well. This kernel does not have any in-built decryption, so you will need a solution for that.

Using the kernel

The Voodoo kernel is based on xnu-1228.7.58 (Darwin 9.5.0), and is built to work with Mac OS X Leopard 10.5.5. The kernel should work adequately on any version of Leopard from 10.5.1 onwards, however AMD users in particular are recommended to update fully to 10.5.5. You must also use a 10.5.5 System.kext (included), *regardless* of your version of Leopard.

Who should use this kernel?

Anyone who cannot run Apple's stock kernel (aka 'vanilla' kernel) should use the Voodoo kernel. This includes all processors except Apple-supported Intel Core/ Core2 range and Xeon. Some people who *can* use vanilla kernel might want to try the Voodoo kernel as well, which could help fix some problems, specially related to multiple-core synchronization, restart issues etc.

All other kernels have been superseded by the Voodoo kernel so everyone is advised to upgrade.

Installing the kernel automatically

The best way to install the Voodoo kernel is to run the included installer package. This backs up your existing kernel as **/mach_kernel.original**, installs the Voodoo kernel in your root folder as **/mach_kernel.voodoo**, sets it as default (by editing `com.apple.Boot.plist`) and creates a symlink `/mach_kernel` which points to the Voodoo kernel (to enable VMware support). A matching `System.kext` is also installed in your extensions folder.

Once the installation has finished, reboot your machine. The system will boot-up using the Voodoo kernel and you do not need to do anything extra. This is the recommended method for most users.

Installing manually

If you wish to install the kernel manually, or have a non-standard setup, simply copy the kernel binary to your system (boot) partition. The default kernel is **/mach_kernel**. It is recommended that you copy the Voodoo kernel to your system partition as **/mach_kernel.voodoo**. Software updates from Apple frequently update the default kernel, which might result in an un-bootable system if your only working kernel is overwritten. It is a good idea to keep different fall-back kernels in your root folder.

Note: If you want to use VMware Fusion, your kernel *must* be named **/mach_kernel**. This is a bug in VMware, which requires the running kernel to use the default name. The recommended workaround is to rename the existing kernel, and create a symlink as follows (in Terminal) :

```
sudo mv /mach_kernel /mach_kernel.original
sudo ln -s /mach_kernel.voodoo /mach_kernel
```

Once you have copied the Voodoo kernel to your root folder, you can boot into the kernel in one of two ways:

1. At the bootloader prompt, type the name of the kernel (ie. **`mach_kernel.voodoo`**). You need to do this every time you boot up.
2. Modify your **`/Library/Preferences/SystemConfiguration/com.apple.Boot.plist`** file and change the Kernel key (the `<string>` tag following the `<Kernel>` tag) to point to the Voodoo kernel.

You will also need to use a `System.kext` from Leopard version 10.5.5 if you would like to have USB hotplug working. If you are running 10.5.5 you probably already have this. If not, make sure you install the appropriate version. We have bundled a copy of the `kext`.

For more specific information on manual installation, please make use of various forums or the IRC channels.

Kernel features

Several new technologies are available in the Voodoo kernel. Many of them are described below, along with some notes.

SSE3 emulation

Many OS X applications make use of SSE3 instructions (Finder, Safari, Logic Pro, Quicktime, iTunes etc.). The kernel includes a redesigned emulator which is 2.5 to 3 times faster than previous emulators, and can handle an unlimited number of concurrent threads. The distinction between SSE2 and SSE3 processors has been closed.

The emulator is activated *only* on SSE2 processors. If you have an SSE3 capable processor, you will *not* lose any performance.

On-the-fly opcode patching

Several applications from Apple do a CPU identification and refuse to work on non-Intel processors. AMD users traditionally had to use manually patched binaries which replaced these CPUID instructions with an emulation call. The Voodoo kernel has the ability to patch Apple binaries automatically while loading them. Sophisticated design has made possible patching of regular Mach-O executables as well as dynamically loaded libraries.

This enables AMD users to run OS X just like Intel users – no binaries need to be manually patched anymore.

64-bit support on AMD

Due to an incompatibility with the SYSENTER instruction on AMD (vs. Intel), AMD users were unable to run Leopard in 64-bit mode. Using a combination of emulation and opcode-patching, the Voodoo kernel enables all capable processors to run in 64-bit mode.

By default, AMD processors will run in 32-bit mode. To enable 64-bit mode on AMD, boot with boot flag **-force64**. *Note that there are known incompatibilities with several kexts and applications. If you encounter problems in 64-bit mode, please switch to 32-bit mode.*

In 64-bit mode on AMD processors, **translate** (required for Rosetta) will not run. This will be fixed in a later release. If you need to run PowerPC applications, please boot in 32-bit mode (default).

Improved real-time clock

The Voodoo kernel now reads your processor clock speed, bus ratio and bus frequencies directly from the processor and from the EFI tree exported by the bootloader. On several systems this means more reliable timing. SpeedStep™ is fully supported by the kernel, which is specially important for Pentium M and similar processors which need their clock to be re-calibrated between frequency throttles. The kernel also fully supports clock speeds lower than 1 GHz, which makes it possible for netbook users to boot Leopard without any timing issues.

Because the kernel depends on receiving the FSB frequency from EFI, it is required to use an EFI-capable bootloader.

Kext black-listing

The Voodoo kernel now includes the ability to block specific kernel extensions which are known to cause problems on non-Apple hardware. Following kexts are currently blocked by the kernel :

1. AppleIntelCPUPowerManagement
2. AppleHWSensor
3. Dont_Steal_Mac_OS_X

No special configuration is needed to take advantage of this feature. This means you should be able to use Apple® Software Update to update your system directly without worrying about using a disabler kext.

Miscellaneous changes

There are countless miscellaneous changes in the kernel which enable it to be run on processors not directly supported by Apple. If you are interested in finding out about these, you are encouraged to check out the source code. Refer to the *How can I contribute?* section for more information.

Troubleshooting

You just installed the Voodoo kernel, booted into it and now something is not working! The first thing you should do is refer to the *Known Issues* section and see if your problem is known and a solution is available. If there is no mention of your problem, figure out whether this is a regression compared to a non-Voodoo kernel. If you can run vanilla kernel, please boot into it and check whether the problem persists. Alternatively, you could try a ToH / modbin or other similar kernel to compare against. If you are sure the problem is with the Voodoo kernel, proceed as follows:

The recommended plan is to first post on the forums, and check the Issues list on the Google Code page to see if this problem has already been reported and a solution is known. If not, please do your best to find the cause of the problem, try to fix it with help from the community.

If you believe you have found a bug which cannot be fixed without modifying the Voodoo kernel, please file a bug report on:

<http://code.google.com/p/xnu-dev/issues/entry>

Note that Voodoo Labs *cannot* help you with general trouble-shooting.

Kernel boot-time arguments

This section describes new or modified boot flags that the Voodoo kernel supports. This is meant for informational or trouble-shooting purposes only. Most users should never need to use any of these boot flags.

Note that this is not an exhaustive list of all kernel options – it only contains options that are new or have changed in Voodoo kernel.

fsb=

Specifies the FSB frequency in Hertz. Use this if the boot-loader is passing an incorrect frequency to the kernel. Typical range: 100000000 to 800000000. Example: **fsb=200000000** (for 200 MHz)

busratio=

This is used to specify your processor's clock multiplier. This is also known as the bus ratio, and is the ratio of your FSB frequency to your processor's highest frequency. If you need to specify a non-integer bus ratio, simply omit the decimal point. Typical range: 3 to 30. Examples: **busratio=13** (multiplier 13) or **busratio=105** (multiplier 10.5)

busratiopath=

Specifies the method used to find the processor's bus ratio (clock multiplier). This is crucial to the kernel's timing code. Audio, graphics, USB and other time-critical performance depends on this value and the FSB frequency. The "bus ratio path" is one of 7 methods that the kernel can use, depending on your particular setup. Ordinarily the kernel will automatically choose the most appropriate method for your computer, however you might want to force the kernel to use a particular path. The path **0** (zero) means the bus ratio is specified via the busratio bootflag (described above). Other values are as follows:

1. Used for AMD Athlon/Opteron and other K8 class processors.
2. Used to read the CPU frequency exported by some newer versions of Chameleon (and perhaps other) bootloaders.
3. Used for AMD Phenom/Shanghai and other K10/K11 class processors.
4. Used for Pentium M or newer Intel CPUs. Reads bus ratio directly from the CPU. This is the method used by stock Apple kernel.
5. This is a dummy value used if you want the kernel to choose the best bus ratio path for you. This is the default for Voodoo kernel.
6. Used on some models of Pentium 4 processors. Typically not recommended in favor of path 7.
7. Used as the fallback option. This will work on all processors. Calculates the bus ratio by timing your processor for 50 ms and measuring the number of clock ticks.

Please use this boot option only if absolutely necessary. Using an incorrect path might cause the kernel to fail to boot.

tscsync=

Certain multi-core CPUs suffer from a timestamp counter drift between the cores, causing unpredictable timing behaviour or audio stutters. The kernel automatically accounts for this. Use this option to enable or disable this feature. Pass **0** (zero) to disable or **1** (one) to enable tsc synchronization. Example: **tscsync=0**

clramp=

Certain AMD dual-core systems (notably Athlon X2) have issues with timestamp counter synchronization between the cores (in addition to tsc drift as described

above). The kernel has a workaround which fixes this problem by *disabling* `cr` clock ramping. On affected systems, `cr` ramping is automatically disabled. If you wish to force enable or disable `cr` ramping, specify a value of `1` to enable or `0` to disable. Example: **`cramp=0`** (force-disable clock ramping).

`kprintf=1`

This is only used for special-purpose debugging. This enables print-out of kernel `kprintf` messages on the screen instead of being sent via serial port. By default this is off.

`std_dyld=`

On-the-fly opcode patching requires an elaborate technique to patch dynamic libraries. Because of this, the kernel includes its own specialized copy of `dyld`, and chooses the best one depending on your CPU. Use this option to force the kernel to use standard `dyld` (pass value `1`) or its specialized copy. (pass value `0`), regardless of your CPU type. *Note: AMD installs might fail to function if you specify this boot-flag!*

`patcher_opts=`

This option is used to control the opcode patcher options. By default the kernel automatically chooses the options based on your CPU type. If you wish to force certain features of the patcher, use a sum of these values :

<i>Value</i>	<i>Description</i>	<i>Default</i>
1	Enable verbose debug output	off
2	Force-enable opcode patcher	off
16	Enable CPUID patching	\$
32	Enable SYSENTER patching (<i>required for 64bit on AMD</i>)	\$
64	Enable LDDQU patching (<i>disabled in Voodoo Release 1</i>)	N/A
128	Enable FISTTP patching (<i>disabled in Voodoo Release 1</i>)	N/A

\$: *Enabled for AMD processors, disabled for Intel processors*

Example: **`patcher_opts=17`** (`16+1` = enable CPUID patching and debug output)

`-notscverify`

The kernel verifies CPU clock speed after initializing the timestamp counter (TSC), in an attempt to automatically detect and correct for incorrect data passed by the bootloader. If you do not wish to perform this verification, use this bootflag.

`-tspanic`

This is used during troubleshooting. If you specify this boot flag, the kernel will induce a panic at the very beginning of the boot process and print out information related to the real-time clock.

`-amd`

This option should *not* be used by anyone and has been left from early alpha testing. This forces the kernel to think it's running on an AMD system (even on Intel). This does not affect user applications – it only affect the kernel's own internal algorithms.

`-emulateintel`

This option should *not* be used by anyone and has been left from early alpha testing. This returns GenuineIntel as the processor vendor string for things which query the kernel (e.g. `sysctl`). This does not affect most applications.

Known Issues

Before you head for the forums or post a bug report, please go through this section.

64-bit applications will not run on AMD

Because of an incompatibility with the kernel's custom dyld, and "translate," Rosetta is unable to run in 64-bit mode at present, on AMD processors. If you are booting the kernel in 64-bit mode, PowerPC applications will not run. Please boot the kernel without any boot-flags (or using the `-legacy` bootflag) to run PowerPC applications on Voodoo.

I am getting occasional or consistent application crashes

If you are running on an AMD processor, please make sure you have updated your system to Leopard 10.5.5. If you are running your computer in 64-bit mode, please boot without any boot-flags (or boot using `-legacy`) to force the kernel to start in 32-bit mode.

Running Marvin's AMD utility crashes the system

This is a known incompatibility with this utility (and possibly with similar applications compressed with `upx`). There is no solution at the moment, please do not run Marvin's utility on AMD systems if the opcode patcher is active. As a reassurance, offline AMD patching utilities are not required since the kernel does on-the-fly patching automatically.

I get a kernel panic during heavy disk activity

This is a known bug with certain ATA kexts in 64-bit mode (specially with more than 4GB RAM). Refer to the forums to get advice on upgrading to a fixed kext. A starting point is <http://forum.insanelymac.com/index.php?showtopic=127611>. Another alternative is to boot using the `-legacy` boot-flag.

USB devices don't respond or hot-plugging them does not work

To fix USB problems, you must use a matching System.kext with this kernel. Since this kernel is based on XNU version 9.5.0 you must install System.kext version 9.5.0 (available with Leopard 10.5.5). A copy is included with this package.

VMware (OS X) does not run when running on Voodoo kernel.

VMware requires the running kernel to be named `/mach_kernel`. This is a known bug in VMware. Either install your Voodoo kernel as `/mach_kernel` (not recommended), or remove `/mach_kernel` and create a symbolink link in its place (alias) which points to Voodoo kernel (`/mach_kernel.voodoo`). If you used the automatic installer, this has been done for you. If you installed the kernel manually, refer to the *Installing the Kernel* section for more details.

Some of my system preferences seem to have been reset

This is a known issue which happens occasionally when switching kernels. Please set your preferences again.

My Pentium 4 is reporting incorrect cache information

This is a known issue with certain Pentium 4 processors, specially those with HyperThreading enabled. These processors report the cache size incorrectly. Since this is just cosmetic, the kernel currently makes no effort to correct for it.

My AMD / Pentium D / Celeron processor is being reported as an Intel Core / Core 2 processor!

This is done on purpose. Several applications and installers (e.g. Silverlight, Parallels) refuse to work if the CPU type is “Unknown.” The kernel can only report from among the Apple-supported CPUs. Hence, other CPUs are reported as either Intel Core Solo, Core Duo or Core 2 Duo.

System Profiler crashes

This might happen on certain processors which report no or incorrect cache information. This causes System Profiler to crash with a “Divide by zero” error. A workaround will be available in a later version of Voodoo.

I get a kernel panic on booting, with the message “Cannot find driver for platform ACPI”

This happens usually because you did not clear the caches after installing the kernel. Please delete:

```
/System/Library/Extensions/Extensions.mkext  
/System/Library/Extensions/Extensions.kextcache  
/System/Library/Caches/com.apple.kernelcaches/*
```

And then reboot with `-f` boot-flag. Please also ensure you are using an up-to-date version of stock AppleACPIPlatform.kext.

My computer cannot sleep / cannot resume after sleeping / cannot sleep or resume in 64-bit mode

This can happen because of a multitude of reasons. Please refer to the forums to help you trouble-shoot these issues. If you are running the kernel on AMD in 64-bit mode, please run the kernel in 32-bit mode which might fix the problem.

My AMD Phenom processor is being detected as having 100 MHz FSB frequency and twice the bus ratio

This is a known issue with some bootloaders which report 100 MHz instead of 200 MHz as the FSB frequency on Phenoms. This does not affect the working of your system in any way and can safely be ignored.

My system boots up to the blue screen and doesn't proceed further

This might be because the kernel does not handle decryption of encrypted binaries. Refer to the forums for decryption solutions.

Frequently asked questions

Can I update straight from Apple Software Update if I use this kernel?

For the most part, yes. If you have installed the kernel as recommended, you should be able to install any updates directly from Software Update, on Intel as well as AMD platforms. However, *please backup your data and any extra kernel extensions you might be using* in case things don't work out.

Do I need to remove AppleIntelCPUPowerManagement.kext or install NullCPUPowerManagement.kext or a similar disabler?

No.

Does the kernel have in-built decryption?

No.

Can you tell me a solution for decrypting encrypted binaries?

No, please refer to the community forums for this information.

Where can I find a "stock" System.kext / AppleSMBIOS.kext / other kext?

A matching System.kext is included with this package. For other kexts, refer to forums.

Is it beneficial to do offline CPUID patching to avoid on-the-fly patching and gain some performance?

The kernel always switches on the patcher if it is running on AMD processor, but it only patches Apple binaries. The opcode patcher is very fast and the performance hit is unnoticeable in our testing (e.g. it takes 200 ms to patch a 40 MB executable). The time taken to read your entire hard-drive, disassemble, patch and rewrite the binaries will probably be longer than leaving the opcode patcher on all the time, hence we do not recommend patching your binaries offline.

Can I use the kernel on an already-patched AMD system?

Yes. Patched and un-patched binaries can be freely mixed.

Will you help me with installing retail on my AMD system?

Unfortunately, no. We have made the technology available, but leave the rest to the community.

Will the source-code for the SSE3 emulator be available?

The SSE3 emulator is available as a binary blob. If you would like to see the source-code, please disassemble it using ndisasm or another tool of your choice. The source is very concise and should be easy to comprehend for anyone with the required knowledge.

Is this kernel legal?

Absolutely. This document is not a proper place to debate on the legality, however. If you have further questions, please contact us directly.

I am a distro-maker. Can I include your kernel on my DVD?

In short: Yes. Longer reply: We have taken great care to ensure that everything about this project is legal. We do not support any activity which might jeopardize our efforts. Therefore, we make it *clear* that if you include or distribute this kernel in any medium, *the Voodoo Labs team is not in any way responsible nor does it endorse such activity.*

In conclusion

Core contributors

The core team of Voodoo Labs is :

mercurysquad

Project maintainer, TSC timing code, cache detection, NX bit, SSE3 emulation

kaitek

Lead programmer, on-the-fly opcode patcher, 64-bit support on AMD, code fixes

Dense

Maintainer, code fixes, source audit, scripting support, build process

Turbo

SSE3 emulation, RTC timing code, technical know-how

Other contributors

Eureka

Get the model name directly from property instead of calling `getModelName()`.

ovof / paulicat

Disable all cores on shutdown to prevent the system hanging

mifki / netkas – modified by kaitek

bcopy SSE3 64-bit version based on LModo

David Elliott aka **dfc**

Modifies the in-kernel HPET initialization to check if a device exists at `00:1f.0` before reading the RCBA from it

Peter Bartoli

Allow raw network packet support

semthex – modified by mercurysquad

CPUID interrupt handler

Kabyl

Fixes for stability issues with Intel based nForce chipsets

Tireless testers: **bhast2**, **skevyn**, **lord_muad_dib**, **XyZ** *et al*

Also thanks to **zef**, **iNDi** *et al* from the Chameleon team and **Galaxy** for his crazy ideas.

How can I contribute?

The source code for the Voodoo kernel is released as a separate package. Please check the Google Code webpage (“Source” tab) for more information.

Together with the source package, Dense’s **voodoo-build** script is bundled which makes building XNU as easy as typing one command. We hope a lot of you will join in!

Suggested kexts

- We recommend using ChunNan/Eureka's AppleSMBIOSEFI.kext along with stock SMBIOS.kext.
- We recommend using stock System.kext from Leopard 10.5.5 (included)
- We recommend using stock AppleACPIPlatform.kext as far as possible

Getting in touch

Google Code homepage

<http://code.google.com/p/xnu-dev>

IRC: #xnu-chat on irc.osx86.hu and #xnu-chat on Freenode.

InsanelyMac and InfiniteMac forums.

E-mail : voodoo@mercurysquad.com

Hope you guys like the kernel!

We appreciate your feedback and support.