The NetBSD Operating system

NetBSD is a free, secure, and highly portable Unix-like Open Source operating system available for many platforms, from large-scale server systems to powerful desktop systems to handheld and embedded devices. Its clean design and advanced features make it excellent in both production and research environments, and the source code is freely available under a business-friendly license.

NetBSD is developed and supported by a large and vivid international community. Many applications are easily available through pkgsrc, the NetBSD Packages Collection.

NetBSD focuses on clean design and well architected solutions. Because of this NetBSD may support certain 'exciting' features later than other systems, but as time progresses the NetBSD codebase is getting even stronger and easier to manage, while other systems that value features over code quality are finding increasing problems with code management and conflicts.

During the development cycle of the NetBSD 5 release, major work was done to improve SMP support; most of the kernel subsystems were modified to be MP safe and use the fine-grained locking approach.

New synchronization primitives were implemented and scheduler activations was replaced with a 1:1 threading model in February 2007. A scalable M2 thread scheduler was implemented, though the old 4.4BSD scheduler is still provided as an option. Threaded software interrupts were implemented to improve synchronization. The virtual memory system, memory allocator and trap handling were made MP safe. The file system framework, including the VFS and major file systems were modified to be MP safe.

Since April, 2008 the only subsystems running with a giant lock are the network protocols and most device drivers.

NetBSD Architectures

CPU	Port
alpha	alpha
arm	acorn26 acorn32 cats evbarm hpcarm iyonix netwinder shark zaurus
hppa	hp700
i386	i386 xen
m68010	sun2
m68k	amiga atari cesfic hp300 luna68k mac68k mvme68k news68k next68k sun3 x68k
mipseb	evbmips ews4800mips mipsco newsmips sbmips sgimips
mipsel	algor arc cobalt evbmips hpcmips playstation2 max sbmips
powerpc	amigappc bebox evbppc ibmnws macppc mvmeppc ofppc prep rs6000 sandpoint
sh3eb	evbsh3 mmeye
sh3el	dreamcast evbsh3 landisk hpcsh
sparc	sparc
sparc64	sparc64
vax	vax
x86_64	amd64 xen

NetBSD Embedded Systems

The NetBSD Operating System is the most portable OS in the world, and many of the supported hardware platforms are suited for embedded applications.

The NetBSD multi-platform operating system supports a wide number of different platforms, many of which can and are already being used in embedded applications. Among the more popular processor families for embedded systems are MIPS, PowerPC, ARM, Xscale and Super-H.

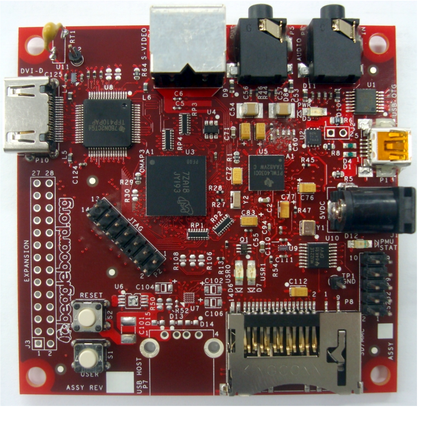
NetBSD Devices

ARM CPU based Toaster running NetBSD

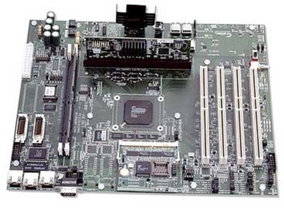
Hitachi Super-h CPU based hp Jornada 680 running NetBSD

ThecusNas running NetBSD

ARM CPU based Sidekick phone running on NetBSD



ARM CPU based Beagleboard



IBM POWERPC CPU based board running on NetBSD

Force10 enterprise class routers running NetBSD

Codian IP VCR 2200 Series enterprise class routers running NetBSD

Property lists

Property lists organize data into named values and lists of values using several object types.

These types give you the means to produce data that is meaningfully structured, transportable, storable, and accessible, but still as efficient as possible.

Property lists are frequently used by applications running on both Mac OS X, iPhone OS and NetBSD.

The property-list programming interfaces for Cocoa and Core Foundation allow you to convert hierarchically structured combinations of these basic types of objects to and from standard XML. You can save the XML data to disk and later use it to reconstruct the original objects.

Proplib library

Proplib library is used for managing property lists defined in Mac OS X documentation. It is clean room reimplementation of property list managing library in the Mac OS X.

The Proplib library was design with multi-thread safeness in mind and it is using mutexes, atomic operations and Read Write lock for it own synchronization. The prolib library have is API which can be used to send receive property list with ioctl routines in kernel/ userspace programs. There are many drivers which encapsulates data into property lists during ioctl communication with kernel e.g. envsys(8), dm(8) and many others.

```
struct _prop_object_type {
        uint32_t pot_type;
        _prop_object_free_rv_t
                (*pot_free)(prop_stack_t, prop_object_t *);
        void    (*pot_emergency_free)(prop_object_t);
        bool    (*pot_extern)(struct _prop_object_externalize_context *,
                  void *);
        _prop_object_equals_rv_t
                (*pot_equals)(prop_object_t, prop_object_t,
                  void **, void **,
                  prop_object_t *, prop_object_t *);
        void    (*pot_equals_finish)(prop_object_t, prop_object_t);
        void    (*pot_lock)(void);
        void    (*pot_unlock)(void);
};
```

prop_object_type definition with lock/unlock routines added

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
 <dict>
    <key>app</key>
    <array>
      <dict>
        <key>app_path</key>
        <string>/usr/sbin/ntpd</string>
        <key>appmod_config</key>
        <array>
          <dict>
            <key>auth_mod</key>
            <string>auth_hash</string>
            <key>auth_mod_data</key>
            <string></string>
          </dict>
          <dict>
            <key>auth_mod</key>
            <string>auth_gid</string>
            <key>auth_mod_data</key>
            <integer>1682</integer>
          </dict>
        </array>
      </dict>
      </array>
 </dict>
</plist>
```


Dictionaries use global red black tree to store keys. It can be referenced from tree. Proplib number objects uses red-black tree fro storing values, too. When number or dictionary entry is released reference counter is dropped. When reference counter is zero dictionary key or number is removed from red-black tree and freed.

Race condition bug

The Proplib library in NetBSD contains race condition vulnerability which can be exploited by calling prop_dictionary_create/prop_number_create and prop_object_release routines.
Race condition was located in a handling of external references of entries in a red-black tree used by dictionary/number.

Internalization bug

There is another bug in proplib library. During converting xml form to binary representation it was possible to crash proplib by using non-existing entry names.

All NetBSD releases with proplib library included was vulnerable to this bug, and it's successful exploitation will lead to Local DoS attack made by arbitrary user. To be able to exploit this bug user must have chance to sent badly formatted plist file to kernel through device-driver which uses proplib as a communication channel. Vulnerable device-drivers are dm(8), envsys(8), drvctl(8).

Vulnerable NetBSD version are **NetBSD**-4.0, **NetBSD**-4.0.1, **NetBSD**-4.1 and not released yet **NetBSD**-5.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//
EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
 <dict>
    <key>app</key>
    <number>1</number>
 </dict>
</plist>
```