Engineering NetBSD 9.0

AsiaBSDCon 2020
Tokyo, Japan

Kamil Rytarowski
The NetBSD Foundation
kamil@NetBSD.org

Abstract

Multics first appeared in 1965, UNIX First Edition in 1969, 1BSD in 1977 and NetBSD 0.8 in 1993, followed by NetBSD 1.0 in 1994. The more modern versions: NetBSD 5.0 was released in 2009, 6.0 in 2012, 7.0 in 2015 and 8.0 in 2018. 8.0 was officially announced on July 16th, 2018. The NetBSD-8 branch was being developed as NetBSD-current between June 4th, 2017 and July 21st, 2019. NetBSD-9 was branched on 30th July and 708 pull requests (884 commits) were processed in the branch until the final release on February 14th 2020. The development period lasting 1.5 years brought a rich set of features to the distribution. These include: support for new hardware devices, massive improvements for the ARM 32-bit and AArch64 architecture, stub of RISC-V support, new hardware-assisted virtualization options, improved Wine support, kernel and userland sanitizers and fuzzers, security hardening and CPU bug mitigations, graphics stack upgrade, ZFS upgrade, debugging interfaces refinement, packet filter policy, 3rd party software upgrades, many bug fixes, generic enhancements and removals of obsolete code.

1. Timeline

NetBSD 9.0 was announced on February 14th, 2020 and informally called the “Valentine Release”. This is the seventeenth major release of NetBSD.

NetBSD 8.0 was released on Tuesday July 17th, 2018. That is, 1 year and 7 months earlier. The span between 7.0 and 8.0 releases was 2 years and 10 months. The release cycle is becoming quicker.

One of the reasons making this possible was the availability of a dedicated developer Martin Husemann working on the release engineering process part-time.

There are initial plans to release NetBSD 10.0 later this year, possibly reducing the length of the cycle below one year.

2. Raw src repository changes

The src repository is the main catalogue containing the NetBSD distribution. Two other important repositories are xsrc and pkgsrc. The xsrc repository contains traditionally X11 (Xorg) files and pkgsrc contains a set of rules to build 3rd party software on top of NetBSD, although it was extended to work on other UNIX-like Operating Systems (Linux, Darwin, other BSDs. Solaris, etc).

The following statistics are restricted to the src repository only. There are developers that are active only in pkgsrc or on the wiki. Their contributions are still valuable and this analysis does not aim to reduce credit for their involvement.

18819 commits were made between Sunday June 4th 2017 (branching point of NetBSD-8 “8.99.1” by snj@), and the final 9.0 release commit on 14th February 2020 (“Welcome to NetBSD 9.0 - the "Valentine Release"” by martin@).

Throughout this period, the following changes were introduced:

- 102,952 files changed,
- 12,728,116 insertions(+),
- 12,082,323 deletions(-).

114 developers made at least a single commit in this branch.

Top 10 active committers are:

- 2426 christos (12.89% total)
- 1749 maxv (9.29% total)
- 1531 jmcneill (8.16% total)
- 1248 martin (6.61% total)
- 1087 msaitoh (5.78% total)
- 996 riastradh (5.29% total)
- 975 mrg (5.18% total)
- 712 kamil (3.78% total)
- 626 skrll (3.33% total)
- 568 kre (3.02% total)

These people introduced 63.33% of the total commits.

Top 20 active developers introduced 80.77% of the total commits.

Top 20% of active committers (approximately 22-23 people) introduced 83.08%-84.20% of the total commits.

If we ignore 9 committers that made no more than a single commit, we have got 105 active developers. This makes 20% of active committers (21 people) introducing 81.94% changes.

If we ignore 18 committers that made no more than commits, we have got 96 active developers. This makes 20% of active committers (19-20 people) introducing 79.71%-80.90% changes.

This presents that the NetBSD-9 development period has long tail distribution of the active committers with infrequent activity and core of very active people making (in the raw number of commits) significant change to the codebase.

This analysis illustrates the Pareto principle also known as the 80/20 rule that roughly 80% of the effects come from 20% of the causes.

It also shows that there is a long tail of developers with overall 20% of activity in the project and this leads to harmonic situation. For every very active developer we get around 4 people contributing from time to time.

3. Hardware support

The hardware requirements are contentiously raising and computer vendors release new generations of computers and their peripherals. NetBSD as a general purpose Operating System follows the market and improves its compatibility with the modern products.

3.1. ARM architecture

NetBSD 9.0 is the first release with formal support for AArch64 (64-bit ARMv8) machines.

- Support for "Arm ServerReady" compliant machines (SBBR+SBSA)
- Symmetric and asymmetrical multiprocessing support (aka big.LITTLE)
- Support for running 32-bit binaries via COMPAT_NETBSD32 on CPUs that support it
- Single GENERIC64 kernel supports ACPI and devicetree based booting
- Supported SoCs:
  - Allwinner A64, H5, H6
  - Amlogic S905, S805X, S905D, S905W, S905X
  - Broadcom BCM2837
  - NVIDIA Tegra X1 (T210)
  - QEMU "virt" emulated machines
  - Rockchip RK3328, RK3399
  - SBSA/SBBR (server-class) hardware using ACPI. Successfully test on, for example: Amazon Graviton and Graviton2 (including bare metal instances), AMD Opteron A1100, Ampere eMAG 8180, Cavium ThunderX, Marvell ARMADA 8040, QEMU w/ Tianocore EDK2
- Support for up to 256 CPUs

The ARMv7 and earlier generation of the ARM line was improved.

- Symmetric and asymmetrical multiprocessing support (aka big.LITTLE)
- UEFI bootloader
- Single GENERIC kernel supports devicetree based booting
- Supported SoCs:
  - Allwinner A10, A13, A20, A31, A80, A83T, GR8, H3, R8
  - Amlogic S805
  - Arm Versatile Express V2P-CA15
  - Broadcom BCM2836, BCM2837
  - Intel Cyclone V SoC FPGA
  - NVIDIA Tegra K1 (T124)
  - Samsung Exynos 5422
  - TI AM335x, OMAP3
  - Xilinx Zynq 7000
- Support for up to 8 CPUs

A large number of ARM SOCs were switched to GENERIC and GENERIC64 kernel configurations with associated DTS configurations. These kernel configurations produce single kernel images ready to run on various ARM boards.

3.2. Direct Rendering Management Kernel Mode Setting

The DRMKMS stack for graphics devices on x86 was updated to match Linux 4.4. This brings support for many recent Intel, nVidia and AMD cards.

New GPU device drivers for ARM include:

- DRM/KMS modesetting drivers for Allwinner DE2, Rockchip VOP, TI AM335x LCDC
- Basic framebuffer driver for Arm PrimeCell PL111, TI OMAP3 DSS
- Simple framebuffer support for reusing linear FBs configured by the bootloader

3.3. RISCV

The RISCV support has reached a functional boot in an emulator, however there is still no userland nor real hardware support. This code was not mature enough to be merged into NetBSD 9.0 and is awaiting future releases, possibly NetBSD 10.0.

3.4. Other changes

Other noteworthy changes include:

- cpuctl(8), the program to control CPUs, was ported to the sparc and sparc64 ports.
- The atari port gained Milan device support.
- The support for Mac G5 was enhanced in the macppc port.
- VAXstation 4000 gained support for the TURBOchannel device, which allows connecting USB devices to VAX computers.
- A number of processors (ARMv7, ARMv8, x86 AMD, x86 Intel) gained support for hardware assisted Performance Monitor Counter.
- The SATA subsystem was reworked to support multiple commands in transit (NCQ).
- New device drivers for networking cards were added: Broadcom Full-MAC wireless devices; Amazon Elastic Network Adapter, Mellanox ConnectX-4 Lx EN, ConnectX-4 EN, ConnectX-5 EN, and ConnectX-6 EN ethernet adapters.

4. Virtualization

There is a rich set of virtualization enhancements in NetBSD 9.0.

4.1. Xen

The traditional NetBSD/xen port is still actively developed and receives maintenance.

The primary changes in this area adapted the existing Xen support code to use more native x86 code paths. Another chunk of patches corrected build and boot of Xen kernels.

Large parts of the xen port were refactored and optimized. Xen non-PAE-i386 is now removed, leaving amd64 and i386+PAE as the only two supported modes.

Many code paths were adapted for newer __XEN_INTERFACE_VERSION__ versions larger or equal to 0x00030201.

4.2. NetBSD Virtual Machine Monitor

NVMM is a hypervisor platform that provides hardware-accelerated virtualization support for NetBSD. It consists of a Machine Independent frontend that connects with Machine Dependent backends. A virtualization API is provided by libnvmm, that makes it possible to easily create and manage virtual machines.

The Qemu package in pkgsrc (emulators/qemu) has been modified to leverage this virtualization API and provide fast emulation using NVMM. The upstream Qemu developers reviewed the NetBSD hypervisor patchset and it has a good chance of being part of the future releases of Qemu.

Two additional components are shipped as demonstration programs, toyvirt and smallkern. The former is a toy virtualizer that executes the 64bit ELF binary given as argument in a VM, while the latter is an example of such a binary.

NVMM is supported on modern Intel and AMD64 64-bit CPUs.

NVMM has been reported to support a number of guest Operating Systems:

- NetBSD 32-bit
- NetBSD 64-bit
- FreeBSD 32-bit
- FreeBSD 64-bit
- Windows XP 32-bit
- Windows 7 32-bit
- Windows 8 32-bit
- Windows 8 64-bit
- Windows 10 32-bit
- Windows 10 64-bit
- Linux 64-bit

The Linux guest requires disabling the sanity checks for hwadware timer, from the bootloader level. The same is already done for other hypervisors such as KVM.

Not all Operating Systems are supported by the combination of NVMM + Qemu, two notable examples being Windows 95 and Windows 98.

4.3. Intel Hardware Accelerated Execution Manager

HAXM is a cross-platform hardware-assisted virtualization engine (hypervisor), widely used as an accelerator for Android Emulator and QEMU.

It has always supported running on Windows and macOS, and has been ported to Linux and then NetBSD.

There is a native support for NetBSD 9.0 and untested support for older releases on NetBSD, although it is probably unstable.

The Qemu package in pkgsrc (emulators/qemu) already ships with the HAXM/NetBSD support out of the box.

The HAXM package is distributed as an external piece of software in pkgsrc/emulators/haxm.

The NetBSD developers decided to promote NVMM as the primary choice and keep HAXM as an optional fallback.

Traditionally HAXM supported only a single guest Operating System – Android, a variation of the Linux kernel. Today the number of other Operating Systems is supported, among others:

- NetBSD 64-bit
- Windows 7 32-bit
- Linux 64-bit
- DragonFlyBSD 64-bit
- FREEDOS

The guest support for 64-bit Windows 10 is in active development by the Intel developers and the Open Source community.

4.4. Wine 64-bit support

The missing kernel support of USER_LDT was implemented. This means that the appropriate wine64 package is now a matter of packaging in pkgsrc and bug fixing.

During the Google Summer of Code 2019, a student packaged and improved wine for modern NetBSD. The deliverable has been verified by a successful execution of a number of Windows applications.

Unfortunately the kernel option is still disabled by default in the kernel configuration as the development of the Wine support is still ongoing.

4.5. NetBSD guest support

Qemu firmware configuration device (fw_cfg) can be mounted via mount_qemufwcfg(8). It provides the Qemu fw_cfg configuration files as a file system tree. This utility is associated with the qemufwcfg(4) device driver that allows Qemu to provide data items and files to guest operating systems.

The virtio support for MMIO and PCI was improved for modern ARM ports.

Hyper-V support for x86 was introduced. This opens the possibility of using NetBSD on the Microsoft Azure cloud.

5. Security, Quality Assurance

The process of engineering the NetBSD Operating System has been improved with this release and the developers gained new tools to catch software bugs more effectively and efficiently.

The NetBSD Operating System is also more secure thanks to new features and developer activities.

5.1. Kernel Address Space Layout Randomization

A full Kernel ASLR has been implemented for the amd64 port along with the GENERIC_KASLR build configuration file. This implementation is one of the most advanced available to date among all Operating Systems.

The GENERIC (default) kernel configuration is also shipped with a partial Kernel ASLR enabled by default. An intermediate bootloader called prekern has been developed for full KASLR.

5.2. Kernel Leak Sanitizer

KLEAK has been designed around the LLVM/GCC Sanitizer Coverage feature to track passing copies of uninitialized memory to userland.

5.3. Kernel Address Sanitizer

Address Sanitizer is a feature of GCC/LLVM that catches accesses to unmanaged memory (use-after-free, out-of-bound access, etc).

The first functional BSD-licensed kernel implementation, landed the NetBSD source tree. It includes support for two ports: amd64 and evbarm aarch64.

5.4. Kernel Undefined Behavior Sanitizer

UBSan is a GCC/LLVM sanitizer that catches unspecified semantics during runtime of the code.

The NetBSD kernel implementation uses the µUBSan runtime that has been created by the NetBSD developers. The µUBSan version is implemented in plain C as a single file. It avoid depending on the environment as much as possible and is reasonable. This runtime is reused in ATF regression tests for standalone userland programs and continuously verified against the host compiler (GCC or Clang).

kUBSan happens to be considered controversial as many issues caught by it are considered uninteresting and de facto well defined behavior by a number of developers. For example architectures such as x86 are generally not sensitive to misalignment.

5.5. Other kernel quality assurance activities

Noteworthy goals achieved in this release are:

- Kernel Sanitizer Coverage (KCOV) device to collect tracing data for kernel sanitizers.
- Kernel Heap Hardening.
- Audited network stack – bringing more confidence in the networking components of the kernel.
- Many bug fixes and new features in the ptrace(2) debugger-oriented API for controlling processes.

5.6. Userland quality assurance activities

The support for userland sanitizers has been improved significantly. Address Sanitizer, Memory Sanitizer, Thread Sanitizer and Undefined Behavior Sanitizer have been implemented and enabled in the LLVM-enabled distribution. Address Sanitizer, Leak Sanitizer, Thread Sanitizer and Undefined Behavior Sanitizer have also been enhanced for the GCC-enabled distribution.

Two new build options have been introduced to make use of the sanitizers:

- MKLIBCSANITIZER – that enables using the sanitizer selected via USE_LIBCSANITIZER inside libc to compile userland programs and libraries. It defaults to “undefined”. Currently UBSan is the only supported sanitizer in this mode and it uses the µUBSan runtime.
- MKSANITIZER – that uses the sanitizer selected via USE_SANITIZER to compile userland programs. It defaults to “address” (ASan). The following sanitizers are supported: address, thread, memory, undefined, leak, dataflow, cfi, safestack, scudo.

The exact list of supported features, their completeness and their valid combinations depend on the compiler version and target CPU architecture.

6. Removal of numerous old components

There were various hardware peripherals and kernel subsystems no longer in use. The support for them was removed in this release. Among others:

- NETISDN and related drivers (daic, iavc, ifpci, ifritz, iwic, isic)
- NETNATM and the related midway driver
- several compatibility layers (NDIS, SVR3, SVR4)
- the n8 driver
- vm86
- ipkdb

The result is that the kernel code is now smaller and easier to develop.

7. Other and general improvements

The NetBSD core team decided that the future of packet filtering on NetBSD is npf(4) – a homegrown NetBSD Packet Filter. This obsoletes ipf(4) and pf(4) that are no longer actively maintained.

The USB device drivers have been refactored and a common layer between the drivers has been defined as the usbnet framework.

ZFS is updated and considered now as ready for daily use.

Third party software upgrades involve: GCC 7.4, GDB 8.3, LLVM 7.0.0, OpenSSL 1.1.1d, OpenSSH 8.0, sqlite3 3.26.0.

8. Conclusions

A large team gathered around the Open Source project, with more than 100 people actively involved in the maintainer role engineered a solid product with wide press coverage and good reception from the users.

9. Dedication

This release was dedicated to the memory of Matthias Drochner, who passed away in August 2018, and Eric Schnoebelen, who passed away in March 2019.

References

[1] The NetBSD Project. http://netbsd.org/releases/formal-9/NetBSD-9.0.html (accessed 2020-02-14)

[2] The NetBSD Project. GitHub mirror. https://github.com/NetBSD/src (accessed 2020-02-14)

[3] Maxime Villard. NetBSD Virtual Machine Monitor. https://m00nbsd.net/4e0798b7f2620c965d0dd9d6a7a2f296.html (accessed 2020-02-14)

[4] Intel® Hardware Accelerated Execution Manager (Intel® HAXM). https://github.com/intel/haxm (accessed 2020-02-14)

[5] Kamil Rytarowski. MKSANITIZER - bug detector software integration with the NetBSD userland https://blog.netbsd.org/tnf/entry/mksanitizer_bug_detector_software_integration (accessed 2020-02-14)

[6] Kamil Rytarowski. Introduction to µUBSan - a clean-room reimplementation of the Undefined Behavior Sanitizer runtime https://blog.netbsd.org/tnf/entry/introduction_to_%C2%B5ubsan_a_clean (accessed 2020-02-14)

[7] Maxime Villard. The strongest KASLR, ever? https://blog.netbsd.org/tnf/entry/the_strongest_kaslr_ever (accessed 2020-02-14)