FreeBSD and Windows Environments

FreeBSD is Uniquely Positioned to Help Deploy, Virtualize, and Serve Microsoft Windows Production Environments

Michael Dexter

AsiaBSDCon 2024, Taipei, Taiwan

Abstract

The FreeBSD open source operating system[1] provides a powerful set of features to facilitate the deployment, virtualization, and serving of Microsoft Windows[2] environments ranging from home research labs to enterprise deployments. FreeBSD's exemplary integration with the OpenZFS file system and volume manager, its bhyve hypervisor, its third party ports and packages, and its overall consistent administrative experience empower operators when performing tasks relating to:

Deployment

- Windows Product Registration Key Retrieval
- Automated Installation
- Remote Console and Desktop Access

Virtualization

- Server Virtualization
- Desktop Virtualization

Serving

- Bi-directional, SMB, NFS, and iSCSI Sharing
- Active Directory Domain Participation
- Active Directory Domain Serving
- NTFS Mounting and Management
- Data Synchronization and Replication
- Native Windows OpenZFS Deployment

This paper describes each of these facilities with examples of their usage.

Introduction

FreeBSD and Microsoft Windows have had significant impacts on computing: BSD Unix, from which FreeBSD descends helped produce the interoperable Internet and the first hardware-agnostic computing, while Microsoft Windows has dominated business and home desktop computing. The communities surrounding these two ecosystems are often unrecognizable to one another, but were united with the seminal year 2000 publication of "The FreeBSD Corporate Networker's Guide"[3] which provides inspiration for this paper. With FreeBSD-derived systems such as TrueNAS CORE and pfSense/OPNsense providing *critical* infrastructure to Windows environments around the world, we must appreciate the decades-long symbiotic relationship between these ecosystems.

Deployment

Hardware Data Collection

The *observability* of FreeBSD and Microsoft Windows systems to an operator could not be more antithetical. A FreeBSD user is greeted by a detailed boot log of kernel and system initialization, while a Windows user is greeted by a metaphor for a rotary telephone and a scenic landscape. This stark contrast has long provided an

opportunity for Unix-like tools such as the Cygwin [4] GNU environment, and single-purpose tools as `ifconfig(8)`, `acpidump(8)`, `smartctl(8)`, OpenSSH, `rsync(8)`, `dmidecode(8)`, and storage-related utilities such as FreeBSD `gpart(8)` and `nvmecontrol(8)`. These tools can be used to retrieve MAC addresses from network interfaces, Windows Product Registration keys from ACPI tables, provide shell access, synchronize data, provide hardware information, format disks, and manage NVMe devices respectively. While some of these tools have been ported to Windows, some of them must be run in hardware or virtualized FreeBSD environments for maximum effectiveness. For example, FreeBSD's `gpart(8)` GPT partition management tool is vastly more capable and intuitive than any Windows-native tool, and `nvmecontrol(8)` can provide basic NVMe namespace management and block size configuration in ways that are not possible with standard Windows utilities. As these are often one-time administrative data collection tasks, they are best performed on a host that is temporarily booted to FreeBSD. For example, to retrieve a Windows Product Registration Key with the FreeBSD `sysutils/acpica-tools` package:

```
/usr/local/bin/acpidump | \
grep -A5 MSDM | tail -n3 | \
cut -c60-75 | xargs echo | \
sed -e 's/^\.*//' \
-e 's/ //g'
```

This ability should be trivial to add to the in-base `acpidump(8)`.

NVMe Block Size Configuration

To identify an NVMe device and format for 4Kn block size: (WARNING: This will erase all data on the device)

```
nvmecontrol devlist
 nvme0: CT500P3PSSD8
    nvme0ns1 (476940MB)
nvmecontrol identify nvme0ns1 | \
grep "LBA Format"
Number of LBA Formats: 2
Current LBA Format: LBA Format #00
LBA Format #00: Data Size: 512
Metadata Size: 0 Performance: Better
LBA Format #01: Data Size: 4096
Metadata Size: 0 Performance: Best

nvmecontrol format -f 01 nvme0ns1
```

Reboot the system to enable the new block size.

Partition Deletion

To destroy partitions on storage device `/dev/ada0` as is often required to allow for further management by Windows or macOS graphical disk management utilities.

```
gpart destroy -F /dev/ada0
```

S.M.A.R.T. Storage Device Health

The `smartctl(8)` storage device S.M.A.R.T. health monitoring utility from the `sysutils/smartmontools` package is used in a similar manner both within and outside of Windows, using Linux-style device naming on Windows:

```
smartctl.exe -a /dev/hda
```

Storage Device Imaging

Prior to system deployment or redeployment, some systems may need to be preserved in their previous state for archival or forensic purposes. The in-base FreeBSD `camdd(8)` command provides disk imaging facilities that consistently match the block count of a storage device as reported by `diskinfo(8)`:

```
diskinfo da1
da1   512   4000787029504
7814037167  4096  0     486401255
63

camdd -i file=/dev/da1,bs=1M \
-o file=4tb.raw

ls -l
-rw-------  1 root wheel
4000787029504 Sep 23 16:20 4tb.raw
```

If a storage device proves unreliable when imaging, the `ddrescue(1)` utility from the `sysutils/ddrescue` port provides resumable, best effort recovery with options such as `-d` direct access without caching and `-r3` retry three times:

```
ddrescue -d -r3 /dev/da1 \
4tb-ddrescue.raw ddrescue.log
```

The native windows VHDX disk image format can be converted to the universal raw format with the `qemu-img(1)` utility from the `qemu-tools` package.

```
qemu-img convert -f vhdx \
      -O raw 4tb.vhdx 4tb.raw
```

The resulting raw disk image can be attached with `mdconfig(8)`, booted in the QEMU[5] emulator, or the `bhyve(8)` and FreeBSD/Xen hypervisors. It can also be transferred to a ZFS volume or hardware storage device using `camdd(8)` or `dd(1)`.

Storage Device Mounting

The FUSE-based `ntfs-3g(8)` utility from the `sysutils/fusefs-ntfs` package compliments this imaging ability with a reliable, albeit limited-performance facility to mount NTFS-formatted partitions in FreeBSD.

```
mdconfig 4tb.raw
md0
gpart show md0
```

```
=>        34  7814037100  md0  GPT  (3.6T)
          34      262144    1  ms-reserved  (128M)
      262178        2014       - free -  (1.0M)
      264192  7813771264    2  ms-basic-data  (3.6T)
  7814035456        1678       - free -  (839K)
```

```
kldload fusefs

ntfs-3g -o ro /dev/md0p2 /mnt


ls /mnt/

$RECYCLE.BIN WindowsImageBackup
```

Bitlocker Considerations

Bitlocker is the native Windows data encryption at rest solution that, like all data encryption solutions, is prone to unintentionally isolating authorized users from critical data. It is important to back up and protect Bitlocker decryption keys and always be aware if a system has Bitlocker enabled. Trusted Platform Module (TPM)-equipped systems preserve Bitlocker keys in hardware for convenient decryption using passphrases or PINs, but will lose those keys if hardware is replaced and in some cases, if system firmware is updated. Automatic firmware updates to systems that are not known to be Bitlocker-enabled can result in catastrophic data loss. Bitlocker-encrypted file systems can be mounted on FreeBSD with the `devel/libbde` package:

```
bdemount -p <password> /dev/ada0p2 /mnt
```

Automated Windows Installation

Microsoft desktop and Server Windows installations targeting the QEMU emulator or the `bhyve(8)` and FreeBSD/Xen hypervisors can be automated with the inclusion of a `autounattend.xml` file located in the top directory of an installation ISO image. `xmllint(1)` from the `textproc/libxml2` package can validate the `autounattend.xml` file and `7z(8)` from the `archivers/7-zip` package, and `mkisofs(8)` from the `sysutils/cdrtools` package can be used to perform installation ISO extraction and remastering.

```
mkdir iso
cd iso
7z x /path/to/windows.iso
xmllint /path/to/autounattend.xml
cp /path/to/autounattend.xml .
cd ..
```

```
mkisofs \
-b boot/etfsboot.com \
-no-emul-boot -c BOOT.CAT \
-iso-level 4 -J -l -D \
-N -joliet-long \
-relaxed-filenames -v \
-V "Custom" -udf \
-boot-info-table -eltorito-alt-boot \
-eltorito-platform 0xEF \
-eltorito-boot \
efi/microsoft/boot/efisys_noprompt.bin \
-no-emul-boot \
-o install.iso ./iso
```

Note that while FreeBSD's `libarchive(3)` will support 7z and ISO9660 formats, it does not yet support the UDF format of windows installation DVDs. Nor does `makefs(8)` support all attributes of the UDF DVD.

Furthermore, the `sysutils/wimlib` package includes utilities to manipulate Windows Imaging (WIM) archives for further deployment customization.

Remote Access

Recent versions of Microsoft Windows include the OpenSSH[6] suite of remote shell access utilities such as `ssh(8)`, `scp(8)`, and `sftp(8)`, and the `sshd(8)` daemon as "optional" features. If not included, these tools can be obtained from various sources and remain invaluable regardless of their origin. RDP remote desktop access is supported by the open source `xfreerdp(1)`[7] (`net/freerdp`), and `remmina(1)`[8] (`net/remmina`) clients, which are available on FreeBSD and most Unix-like operating systems. When configured with familiar facilities such as pre-shared SSH keys, FreeBSD and Windows systems can perform as indistinguishable, bi-directional peers for more-consistent administration.

If OpenSSH is neither pre-installed nor an available "option", it can be searched for, installed, and enabled with the following syntax:

```
Get-WindowsCapability -Online | ? Name -like
'OpenSSH*'
Add-WindowsCapability -Online -Name
   OpenSSH.Client~~~~0.0.1.0
Add-WindowsCapability -Online -Name
   OpenSSH.Server~~~~0.0.1.0
Start-Service sshd
Set-Service -Name sshd -StartupType 'Automatic'
Get-NetFirewallRule -Name *ssh*
```

Virtualization

The FreeBSD hypervisor `bhyve(8)`[9] has supported Microsoft Windows guests in production environments[10] since 2015 and performs admirably with VirtIO-backed storage and networking, or SR-IOV network hardware virtualization, and NVMe storage emulation. Windows desktop and Server virtual machines are both well-supported, and the built-in bhyve UEFI-GOP VNC server can be replaced with native Windows RDP for remote desktop access over secure protocols. Beyond traditional server-based host/guest virtualization, FreeBSD is institutionally used for desktop virtualization where the underlying system is FreeBSD for real-time computing purposes but is visually and operationally Windows to the end-user[11]. With nearly a decade of production-use experience, FreeBSD's bhyve is a proven alternative to proprietary virtualization solutions such as Microsoft Hyper-V and Broadcom VMware.

Windows on bhyve Considerations

Early Windows on bhyve deployments were very particular with regard to configuration but now only require that a few conditions be met:

- The Low Pin Count (lpc) device must be on PCI slot 31
- Windows does not ship with VirtIO drivers and emulated devices must be used until they are the drivers installed
- TPM pass-through is supported on FreeBSD 14.0 and later, and TPM emulation is under active development

While not a requirement, Windows virtual machines have been observed to perform particularly well[12] with bhyve's NVMe emulation thanks to its authentic multiple queue emulation.

OpenZFS Storage

While the OpenZFS[13] file system and volume manager provides unrivaled data safety guarantees when used for the backing storage of Windows virtual machines, its snapshot and rollback facilities are invaluable with Windows file servers to mitigate:

- Ransomware attacks
- Accidental data deletion
- Staged OS and application installation
- Application data restoration
- Failed OS updates

Snapshotting a virtual machine at its storage level at every stage of installation allows for instantaneous "undo" of misconfigurations or failed installations. Applications that update their data during software update can take advantage of OpenZFS rollbacks to achieve "fresh" installations:

- Install and snapshot the operating system
- Install old application version
- Import application data
- Upgrade the application
- Export application data
- Roll back the operating system
- Install new application version
- Re-import application data

OpenZFS also provides efficient data replication via its snapshot `zfs-send(8)` and `zfs-receive(8)` facilities, allowing for efficient on-premises and off-site backups. A virtualized Windows desktop backed by OpenZFS storage would also have the advantage of being efficiently backed up in its entirety, and being bootable on a remote virtual machine that has access to a participating backup.

Serving

As outlined in the FreeBSD Corporate Networker's Guide and confirmed with the popular TrueNAS CORE[14] storage operating system, FreeBSD has long provided SMB sharing services to Windows guests with the option of Active Directory participation.

FreeBSD itself can mount SMB shares with the in-base `mount_smbfs(8)` utility, with limited permissions preservation:

```
mount_smbfs -W MYDOMAIN \
//user@myserver/mysmb_share /mnt
```

If the SMB share proves to be of a level not handled by `mount_smbfs(8)`, the `smbnetfs` utility from the `sysutils/fusefs-smbnetfs` package is available and can be used as follows:

```
mkdir ~/.smb
```

```
cp /usr/local/share/doc/smbnetfs-0.6.3/smbnetfs.conf \
~/.smb/
```

```
vi ~/.smb/smbnetfs.auth

  auth 10.0.0.20 <user> <password>
```

```
chmod 600 ~/.smb/smbnetfs.auth
```

```
vi ~/.smb/smbnetfs.host

  host 10.0.0.20 visible=true
```

```
mkdir ~/mnt
```

```
kldload fusefs
```

```
smbnetfs ~/mnt
```

```
ls ~/mnt/10.0.0.20
HelloWorld.txt
```

```
umount ~/mnt/10.0.0.20
```

SMB Sharing

A Samba[15] Server Message Block (SMB) share can configured using the traditional POSIX/Unix permissions model but Windows Active Directory participation requires specific Access Control List configuration and specifically use of the `aclmode=restricted` ZFS property:

```
zfs create -o \
mountpoint=/data/share \
-o xattr=sa -o dnodesize=auto -o \
relatime=on -o \
aclmode=restricted -o \
aclinherit=passthrough \
zroot/data/share

setfacl -R -m
owner@:full_set:fdI:allow,group@:m
odify_set:fdI:allow,everyone@::fdI
:allow /data/share

setfacl -m
owner@:full_set:fd:allow,group@:mo
dify_set:fd:allow,everyone@::fd:al
low /data/share
```

The accompanying Samba configuration file reads:

```
cat /usr/local/etc/smb4.conf
[global]
       vfs objects = zfsacl streams_xattr
       nfs4:chown = true
       use sendfile = yes
       block size = 4096
       server smb encrypt = desired
       server string = Samba Version %v
       netbios name = nas
       realm = dc.mydomain.com
       workgroup = MYDOMAIN
       security = ADS
       winbind enum groups = Yes
       winbind enum users = Yes
       winbind nss info = rfc2307
       idmap config *:range = 2000-9999
       idmap config * : backend = tdb
[NAS]
       path = /data/share
       writable = yes
       browsable = yes
       read only = no
       public = no
       create mode = 0666
       directory mode = 0755
```

Active Directory Participation

Samba provides the ability for a host to join a Windows Active Directory domain for user authentication, and to serve as a secondary or primary domain controller. As with any Active Directory participation, it is exceedingly important that time be synchronized on all participating systems and that forward and reverse DNS be correctly configured. Samba provides a built-in DNS server and all participating hosts must be configured to first query the DNS server of the authoritative domain controller.

Active Directory Domain Membership

For this Samba host to join a Windows Active Directory domain controller for user authentication, `/etc/nsswitch.conf` must be modified to read:

```
group: files winbind
passwd: files winbind
```

The join is performed with:

```
net ads join -U \
Administrator%<password>
```

Recent version of Samba allow this operation to be performed with `samba-tool(8)`:

```
samba-tool domain join \
samdom.example.com MEMBER \
-U administrator
```

Active Directory Domain Controller

Similar to an Active Directory join, Samba can configure a domain controller service with the interactive or scripted use of `samba-tool(8)`:

```
samba-tool domain provision
```

The author's `freebsd-ad` project[16] automates this process with an interactive provisioning procedure and suggested diagnostics syntax.

NFS Mounting and Sharing

Windows 7 Enterprise and later include Network File System client support[17] and Windows Server 2016 and later can provide NFS version 4.1-compatible shares[18].

iSCSI Sharing

Windows has a proven history of robust iSCSI and Fiber Channel block storage support for want of robust file-based sharing protocols. FreeBSD provides robust iSCSI and Fibre Channel services and is a proven platform for situations that require native NTFS file systems that are backed by OpenZFS file systems and volumes (ZVOLs).

With FreeBSD able to provide an iSCSI target to Windows, the `ntfs-3g(8)` package allows for the backing image or volume to be mounted locally, providing "round trip" data portability:

```
truncate -s 10G /tmp/iscsi10G.raw
```

Generate a simple `ctl.conf` configuration file:

```
portal-group default {
        discovery-auth-group no-authentication
        listen 10.0.0.20
}

target iqn.2014-09.org.freebsd:target0 {
        auth-group no-authentication
       portal-group default
        lun 0 {
                path /tmp/iscsi10G.raw
               size 10G
        }
}
```

Validate the file:

```
ctld -f ctl.conf -t
```

Launch `ctld(8)` in debug mode:

```
ctld -f ctl.conf -d
```

Connect to the target with iSCSI Initiator under Windows, format it with Disk Management, and disconnect. Attach the image in FreeBSD with `mdconfig(8)` and mount it with `ntfs-3g(8)`:

```
mdconfig  -a /tmp/iscsi10G.raw
md0

gpart show md0
=>      34  20971453  md0  GPT  (10G)
        34     32734    1  ms-reserved  (16M)
     32768  20934656    2  ms-basic-data  (10G)
  20967424      4063       - free -  (2.0M)

kldload fusefs
ntfs-3g /dev/md0p2 /mnt
ls /mnt
$RECYCLE.BIN
```

The same procedures can be performed with a ZFS volume if the `vfs.zfs.vol.mode` `sysctl` is set to "`0`" while shared via iSCSI, and "`1`" if accessed on the FreeBSD host:

```
zfs create -V 10g t14/tgt

file /dev/zvol/t14/tgt
/dev/zvol/t14/tgt: character special (0/128)

sysctl vfs.zfs.vol.mode: 1
sysctl vfs.zfs.vol.mode=0
vfs.zfs.vol.mode: 1 -> 0
```

Modify `ctl.conf` to include:

```
path /dev/zvol/t14/tgt
```

Upon return from the "round trip", set `vfs.zfs.vol.mode` back to "`1`" and it will be visible as a GEOM provider and can be mounted on the FreeBSD host:

```
gpart show
=>      34 20971453 zvol/t14/tgt GPT  (10G)
        34    32734              1 ms-reserved  (16M)
     32768 20934656              2 ms-basic-data  (10G)
  20967424     4063                - free -  (2.0M)

ntfs-3g /dev/zvol/t14/targetp2 /mnt
ls /mnt
$RECYCLE.BIN
```

Data Synchronization and Replication

Microsoft Windows has an unfortunate history of failing to preserve date stamps when copying files, motivating the use of external tools such as the included `robocopy`, and `rsync`[19] via cwRsync[20] or DeltaCopy[21]. While these tools provide schedulable data replication, they are ultimately obsoleted by OpenZFS replication which provides far greater data integrity guarantees and efficiency.

Example `rsync`-like `robocopy` syntax:

```
robocopy D:\
E:\backups\previous-d-drive /MIR
/FFT /R:1 /W:1 /Z /XJD
```

OpenZFS on Windows

Experimental OpenZFS support[22] exists for Windows desktop and Server, and the author was the first known user of OpenZFS on Windows on physical hardware, providing feedback to its developers for years. OpenZFS on Windows or "ZFSin" promises to be a significant step forward in Windows file systems given the limited adoption of its native ZFS-alternative, ReFS. To view available disks by device name for use with OpenZFS and to create a zpool:

```
wmic diskdrive list brief

... DeviceID ...

\\.\PHYSICALDRIVE1

zpool.exe create -O \
casesensitivity=insensitive -O \
normalization=formD -O \
compression=lz4 -O atime=off -o \
ashift=12 tank PHYSICALDRIVE1
```

Conclusions

FreeBSD is an excellent compliment to Windows environments and can provide production-ready deployment, virtualization, and serving resources and facilities to them. The majority of the tools mentioned are permissively-licensed and all of them are freely-usable at no cost.

Acknowledgements

The author would like to thank the many FreeBSD and related developers who advance open source operating systems and add-on software to help create robust, production ready environments.

References

1. FreeBSD.org
2. Microsoft.com/windows
3. FreeBSD-Corp-Net-Guide.com
4. Cygwin.com
5. QEMU.org
6. OpenSSH.org
7. FreeRDP.com
8. Remmina.org
9. bhyve.org
10. bhyvecon.org/bhyvecon2019-Tubnor.pdf
11. Beckhoff.com/en-us/products/automation/twincat-bsd-hypervisor
12. KlaraSystems.com/articles/virtualization-showdown-freebsd-bhyve-linux-kvm
13. OpenZFS.org
14. TrueNAS.org
15. Samba.org
16. GitHub.com/michaeldexter/freebsd-ad
17. learn.microsoft.com/en-us/windows-server/storage/nfs/nfs-overview
18. learn.microsoft.com/en-us/windows-server/storage/nfs/deploy-nfs
19. Rsync.samba.org
20. itefix.net/cwrsync
21. AboutMyIP.com/AboutMyXApp/DeltaCopy.jsp
22. GitHub.com/openzfsonwindows/openzfs/releases