Proton: a deep technical history of Windows gaming on Linux

Valve's Proton compatibility layer, first released in August 2018, transformed Linux from a gaming afterthought into a commercially viable platform — one now powering millions of Steam Deck handhelds. Proton is not a single piece of software but a carefully orchestrated stack: a patched Wine build, Vulkan-based graphics translators (DXVK, VKD3D-Proton), sync primitive optimizations, Wikipedia a container runtime, and dozens of subsystem reimplementations. Its technical lineage traces back to 1993 and the Wine project's founding, and its evolution has been shaped by the economics of hardware (Steam Machines' failure, Steam Deck's success), the politics of kernel development (the ntsync saga), and the adversarial dynamics of anti-cheat. What follows is a detailed account of how all these pieces fit together.

Wine's three decades from Usenet to PE conversion

The Wine project originated in June 1993 from discussions on the comp.os.linux Usenet group. inertz corp. **Bob Amstadt** and Eric Youngdale announced the project — initially targeting Windows 3.1 (Win16) applications Wikipedia — with Amstadt posting the first status update to comp.windows.x.i386unix on September 29, 1993. The project drew inspiration from Sun Microsystems' Wabi for Solaris. Wikipedia **Alexandre Julliard** assumed maintainership in 1994 and has led the project ever since, CodeWeavers WineHQ now as CTO of CodeWeavers.

Wine's fundamental design principle — recursive acronym "Wine Is Not an Emulator" — defines its architecture. Wine does not virtualize CPU instructions. Wikipedia When a Windows .exe runs under Wine, its x86/x86-64 machine code executes natively on the host CPU. Wine intercepts Windows API calls at the DLL boundary and translates them to POSIX/Linux equivalents. This makes Wine a reimplementation of the Windows API, not an emulator, and means its performance ceiling is theoretically identical to native execution for CPU-bound workloads.

The project spent 15 years reaching version 1.0 on June 17, 2008. Wikipedia Subsequent milestones arrived on an accelerating schedule: Wine 2.0 (January 24, 2017) brought an annual release cadence, 3.0 (January 18, 2018) improved Direct3D 11, 4.0 (January 22, 2019) added Vulkan support via vkd3d, 5.0 (January 21, 2020) began the PE conversion, Steam Community and 6.0 (January 14, 2021) accelerated it. Wine 7.0 (January 18, 2022) bundled roughly 9,100 changes including a Vulkan renderer for WineD3D. WineHQ Wine 8.0 (January 24, 2023) declared the PE conversion "finally complete" after four years of work. WineHQ GitHub Wine 9.0 (January 16, 2024) Wikipedia introduced an experimental Wayland driver and WoW64 improvements. The Register Wine 10.0 arrived in January 2025, and Wine 11.0 (January 13, 2026) shipped ntsync support and declared WoW64 at full feature parity. WineHQ

Two commercial entities profoundly shaped Wine's trajectory. **CodeWeavers**, founded by Jeremy White in May 1996 in St. Paul, Minnesota, pivoted to full-time Wine development in 1999 after White discovered the project. CodeWeavers employs Julliard and claims authorship of roughly **two-thirds of all Wine commits**. Their CrossOver product, Wikipedia launched in 2002, CodeWeavers funds this work and returns all changes upstream under the LGPL. On the other side, **TransGaming Technologies** (founded 2001, Toronto) forked Wine under the MIT/X11 license as WineX — later renamed Cedega (June 22, 2004) — focused on DirectX gaming. TransGaming's refusal to contribute back En Academic was a direct catalyst for Wine's license change from MIT/X11 to LGPL Slashdot in March 2002. Cedega gradually fell behind upstream Wine in compatibility; its last release was version 6.1 (September 23, 2008), the subscription service was retired February 28, 2011, WikiMili and TransGaming itself was shuttered in 2016. HandWiki Wine-staging, maintained since September 2015 as an official experimental branch PCGamingWiki (transferred to WineHQ maintainer Alistair Leslie-Hughes in January 2017), served as the testing ground for patches too aggressive for upstream Wikipedia — including early versions of the sync primitives that would later become central to Proton.

From Steam Machines' failure to Proton's launch

Valve's Linux journey began in earnest in 2012. Gabe Newell publicly called Windows 8 **"a catastrophe for everyone in the PC space"** in July 2012, GamingOnLinux framing Linux as a strategic hedge against Microsoft's platform lock-in. That August, Valve's famous "Faster Zombies!" blog post demonstrated Left 4 Dead 2 running faster on Linux with OpenGL than on Windows 7 with Direct3D. GamingOnLinux The Steam for Linux client entered limited beta in November 2012 GamingOnLinux and launched publicly on February 14, 2013. GamingOnLinux

Valve announced SteamOS and Steam Machines in September 2013. Wikipedia SteamOS 1.0 beta — Debian-based — shipped December 13, 2013. Wikipedia Multiple OEM partners (Alienware, Origin PC, and others) built Steam Machine hardware. The devices launched alongside the Steam Controller and Steam Link on **November 10, 2015**, Wikipedia and promptly failed. The core problem was circular: SteamOS could only run native Linux games, and there were too few of them because the platform had no users. The confusing "Good, Better, Best" branding across OEMs, the Steam Controller's steep learning curve, and the $50 Steam Link cannibalizing the living-room use case compounded the issue. As Newell later acknowledged in March 2020, "the hardware we were pushing for was super-incomplete at the time." Wikipedia

The quiet period following Steam Machines' failure was not idle. Valve hired Pierre-Loup Griffais ("Plagman") as a core engineer, contracted CodeWeavers for Wine development, sponsored Mesa/RADV contributors, and — crucially — began funding **Philip Rebohle**, creator of DXVK. For over two years before Proton's announcement, Valve and CodeWeavers developed the compatibility layer in private. ResetEra

On **August 21, 2018**, Griffais published the Steam Play announcement. Proton 3.7 shipped as a beta with 27 whitelisted titles (Fandom) (DOOM 2016, Quake, Final Fantasy VI, S.T.A.L.K.E.R., among others). Its initial composition was straightforward: a patched Wine build, DXVK for D3D9/10/11-to-Vulkan translation, and esync patches for threading performance. Major releases followed a cadence roughly aligned with upstream Wine versions: Proton 4.2 (March 2019, D9VK integration), 5.0 (March 2020, VKD3D-Proton for DX12), 6.3 (April 2021, major compatibility push), 7.0 (February 2022, aligned with the Steam Deck launch), 8.0 (March 2023, Vulkan 1.3 required, HDR support), 9.0 (March 2024, updated DXVK and VKD3D-Proton), and 10.0 (January 2026, Wine 10.0 rebase, DLSS 3 Frame Generation). Proton Experimental, launched December 2020 as a perpetual rolling-release branch, receives bleeding-edge fixes targeting newly released AAA titles. (Star Oceans)

The **Steam Deck** transformed Proton from a power-user curiosity into a commercial necessity. Announced July 15, 2021, (Wikipedia) and launched February 25, 2022 (Wikipedia) (starting at $399), the Deck runs SteamOS 3.x — now based on **Arch Linux** rather than Debian, (Wikipedia) with an immutable root filesystem and Collabora-built update system. The shift to Arch was pragmatic: Debian's large, infrequent releases suited servers, not a consumer device needing rapid driver and compatibility fixes. The Deck Verified program categorized games as Verified, Playable, Unsupported, (ShopSavvy) or Unknown, (Steam Deck) directly driving Proton development priorities toward anti-cheat support, controller UX, and media playback. The Steam Deck OLED (Wikipedia) (November 16, 2023) (Gematsu) added an HDR display, pushing Proton's HDR pipeline work. Estimated sales reached **3.7–4 million units** by February 2025, (Wikipedia) and Linux's Steam survey share surpassed 3% by October 2025. (Grokipedia)

Wine's internal architecture and the PE conversion

Understanding Proton requires understanding Wine's layered design. At the lowest level, Wine's **PE loader** (rooted in `dlls/ntdll/loader.c`) parses the Portable Executable format — DOS header, PE signature, COFF headers, section tables, import tables — and maps PE sections into the process's virtual address space (Readmex) via `NtCreateSection`/`NtMapViewOfSection` (internally backed by `mmap`). `LdrLoadDll()` resolves imports, (GitHub) walks the initialization order list, and calls `DllMain(DLL_PROCESS_ATTACH)` for each loaded module. Wine supports both "native" (actual Windows) and "builtin" (Wine-reimplemented) DLLs, with load order configurable via `WINEDLLOVERRIDES`. (WineHQ)

Wine's **ntdll.dll** is the keystone — the lowest-level NT system call interface. It provides `Nt*` syscall stubs (`NtCreateFile`, `NtWaitForMultipleObjects`, `NtCreateSection`), the PE loader (`Ldr*` functions), virtual memory management, exception handling (translating Unix signals to Windows SEH exceptions), and the Thread/Process Environment Blocks (TEB/PEB). As of 2026, Wine's ntdll implements approximately **93% of documented NTDLL functions** (~1,400 of ~1,500). (Winehq) Above ntdll sit KERNEL32/KernelBase (process/thread management, file I/O), USER32 (windowing, input), and GDI32 (graphics), mirroring the real Windows DLL hierarchy. (Readmex)

The **wineserver** is a separate, single-threaded daemon managing Win32 kernel-object state shared across Wine processes — analogous to the Windows NT kernel's object manager. It handles kernel objects (handles, Events, Mutexes, Semaphores), the Wine registry, IPC, and window management coordination. (Readmex) Wine processes communicate with wineserver over **Unix domain sockets** in a synchronous RPC model: each operation that touches kernel-object state requires a round-trip (client → server → client), involving two context switches. (Collabora) This IPC overhead is the fundamental bottleneck that esync, fsync, and ntsync were created to address.

The most significant architectural change in Wine's history was the **PE conversion**, spanning Wine 5.0 through 8.0 (2020–2023). Previously, Wine DLLs were built as ELF shared objects (`.so` files) using the host GCC compiler. These "fake PE" files appeared to be PE binaries but were actually ELF, allowing them to call Unix APIs (glibc, X11) directly. This broke copy protection and anti-cheat systems that inspect PE headers in memory, prevented a clean separation between Windows and Unix address spaces, and made WoW64 impossible without 32-bit ELF libraries.

After the PE conversion, Wine DLLs are built as **real PE binaries** using MinGW cross-compilers. (WineHQ) Each module splits into a PE side (with proper PE headers and Windows calling conventions) and a Unix side (thin `.so` shims interfacing with the host OS). The **syscall dispatcher** (`__wine_syscall_dispatcher`) mediates transitions between these two sides using a `syscall_frame` structure to preserve register state (DeepWiki) and a `SYSTEM_SERVICE_TABLE` mapping syscall IDs to Unix implementations. (DeepWiki) Architecture-specific assembly (x86, x86-64, ARM, ARM64) (DeepWiki) minimizes overhead. (DeepWiki) This matters enormously: real PE binaries pass integrity checks (Linux Uprising) from Denuvo, EasyAntiCheat, and BattlEye, and enable the WoW64 architecture for running 32-bit Windows apps on 64-bit-only hosts. (Linux Uprising)

Graphics translation: DXVK, VKD3D-Proton, and the shader pipeline

DXVK: Direct3D 9/10/11 on Vulkan

DXVK, created by **Philip Rebohle** ("doitsujin") in late 2017/early 2018, is the workhorse of Proton's graphics stack. Frustrated by WineD3D's poor OpenGL-based D3D11 performance, Rebohle built a from-scratch D3D-to-Vulkan translator. Valve began sponsoring him full-time in 2018, and DXVK shipped with Proton from its first release.

DXVK replaces the system DLLs (`d3d9.dll`, `d3d11.dll`, `dxgi.dll`) with its own implementations. Its architecture consists of three layers: an API layer implementing COM interfaces (`ID3D11Device`, `ID3D11DeviceContext`), a core abstraction layer (`DxvkDevice`, `DxvkContext`, `DxvkBuffer`) managing state and submitting commands via an asynchronous **command stream (CS) thread**, and direct Vulkan calls beneath. D3D11 state (blend, rasterizer, depth-stencil, input layout) is tracked and combined into Vulkan pipeline state objects at draw time.

DXVK contains its own **DXBC-to-SPIR-V compiler** (`src/dxbc/`) that parses the DXBC container format (Shader Model 4/5 bytecode) and directly emits SPIR-V. For D3D9, a separate DXSO compiler (`src/dxso/`) handles Shader Model 1–3 bytecode, emulating D3D9's fixed-function state (fog, texture coordinate generation, lighting) in shader code. D9VK, originally a separate project by Joshua Ashton implementing D3D9-to-Vulkan, was **merged into DXVK** Wikipedia **1.5** (December 16, 2019). Wikipedia Direct3D 8 support followed in **DXVK 2.4** (July 10, 2024).

Shader compilation stutter has been DXVK's most notorious user-facing issue. Vulkan requires a complete graphics pipeline (all shader stages plus fixed-function state) compiled into a monolithic `VkPipeline` before any draw call using that combination. Phoronix When a new combination is first encountered at draw time, synchronous driver compilation causes visible hitches. The community-maintained dxvk-async patch (by Sporif) addressed this by simply skipping draw calls whose pipelines weren't ready GitHub — objects would be invisible for one to two frames. This was controversial: some considered it rendering modification that could provide competitive advantages, and Valve never shipped it officially.

The proper solution arrived with **VK_EXT_graphics_pipeline_library (GPL)**, supported in **DXVK 2.0** (November 2022). GPL splits pipeline compilation into four independent stages (vertex input interface, pre-rasterization shaders, fragment shader, fragment output interface). DXVK compiles shader stages when games call `CreateVertexShader()`/`CreatePixelShader()` — typically during loading screens — then rapidly fast-links them at draw time. GitHub A fully optimized pipeline compiles asynchronously in the background and hot-swaps in when ready. NVIDIA supported GPL from driver 520.56.06; AMD's RADV enabled it by default in Mesa 23.1. Phoronix This largely eliminated stutter for games that precompile shaders during loading screens, though **Unreal Engine 4 titles** that create shaders at draw time remain problematic, Steam Community requiring Valve's **Fossilize**-based shader pre-caching system (which collects pipeline state from users and redistributes it via Steam).

DXVK 2.0 also raised the minimum requirement to **Vulkan 1.3** Boiling Steam and added D3D11 Feature Level 12_1 support. DXVK 2.5 (November 2024) rewrote resource and memory management, reducing peak VRAM usage by up to **1 GiB**. DXVK 2.7 (March 2025) rewrote descriptor management to use `VK_EXT_descriptor_buffer` on newer AMD/NVIDIA hardware for significantly reduced CPU overhead, and **removed the legacy state cache** entirely in favor of relying on GPL. The latest release is DXVK 2.7.1 (August 2025).

VKD3D-Proton: Direct3D 12 on Vulkan

Direct3D 12 translation presents a harder problem than D3D11. The upstream vkd3d library at WineHQ, with primary development by Henri Verbeet (CodeWeavers) and the late Józef Kucia, prioritizes clean Wine integration and broad driver compatibility. **VKD3D-Proton** is a fork maintained by **Hans-Kristian Arntzen** (HKA, contracted by Valve) and Philip Rebohle, aggressively targeting performance via modern Vulkan extensions. GitHub

The fork's core technical challenges include mapping D3D12's monolithic **descriptor heaps** to Vulkan's bindless descriptor model. VKD3D-Proton implements this with up to seven persistent Vulkan descriptor sets — six for CBV/SRV/UAV views (split by resource type: sampled images, storage images, uniform buffers, storage buffers, uniform texel buffers, storage texel buffers) and one for samplers — requiring a minimum of **1,000,000 UpdateAfterBind descriptors**. GitHub When `VK_EXT_descriptor_buffer` is available, descriptor sets are replaced with plain VkBuffers for even lower overhead. D3D12 **root signatures** map root constants to Vulkan push constants, root descriptors to buffer device addresses (`VK_KHR_buffer_device_address`), and descriptor table pointers to offsets into bindless sets.

DXR (DirectX Raytracing) support progressed from experimental (VKD3D-Proton 2.3, requiring `VKD3D_CONFIG=dxr`) to enabled-by-default in v2.11 (November 2023). DXR 1.0 is feature-complete via `VK_KHR_ray_tracing_pipeline` and `VK_KHR_acceleration_structure`. DXR 1.1 inline raytracing is fully implemented. Cyberpunk 2077, Metro Exodus Enhanced, and Control all work with ray tracing. Mesh shader support via `VK_EXT_mesh_shader` arrived in v2.7. Linux Gaming Central VKD3D-Proton 2.11 exposed the full **DX Ultimate feature set** (Feature Level 12_2) on RDNA2+ and Turing+ GPUs. GamingOnLinux

VKD3D-Proton 3.0 (November 17, 2025) was a landmark release. Phoronix Rebohle completely rewrote the **DXBC shader backend**, replacing the legacy vkd3d-shader path Yahoo! with a new `dxbc-spirv` library shared between DXVK and VKD3D-Proton. How-To Geek Phoronix This fixed many previously broken D3D12-mode games (notably Red Dead Redemption 2). How-To Geek +2 The release also added experimental **D3D12 Work Graphs** emulation via compute shaders GamingOnLinux — no native Vulkan work graph extension exists yet GitHub — along with FSR4 support via AGS WMMA intrinsics, GitHub AMD Anti-Lag support, Phoronix and Shader Model 6.7/6.8 coverage.

For DXIL-to-SPIR-V translation (D3D12 Shader Model 6+), HKA's **dxil-spirv** library parses DXIL — which is based on LLVM 3.7 IR bitcode — using a custom minimal LLVM C++ API subset (~2 MB binary rather than full LLVM's ~40 MB). A critical component is the CFGStructurizer, which transforms LLVM's unstructured control flow into SPIR-V's required structured form via dominance analysis, loop detection, and merge/ladder block creation. HKA documented the complexity extensively in his blog series "My personal hell of translating DXIL to SPIR-V" (themaister.net, September–November 2021).

DXVK-NVAPI and WineD3D

DXVK-NVAPI (repo: jp7677/dxvk-nvapi) emulates NVIDIA's NVAPI to enable NVIDIA-specific features. It does not implement DLSS or Reflex directly but forwards calls to the appropriate subsystems. DLSS 2.x support works for D3D11 and D3D12 on NVIDIA Turing+ GPUs, requiring `nvngx.dll` in the Wine prefix (Proton copies these automatically). Reflex is supported via `VK_NV_low_latency2` for both D3D11 and D3D12 paths. Included in Proton since 6.3-6, latest version 0.9 (March 2025).

WineD3D, Wine's built-in OpenGL-based Direct3D implementation (Wikipedia) (maintained by Henri Verbeet), remains as a fallback. DXVK typically delivers **2–5× the frame rate** of WineD3D for D3D11 titles, and WineD3D's OpenGL path has substantially higher CPU overhead. Fedora made DXVK the default over WineD3D in version 33+. (The Fedora Project)

The ntsync saga: from wineserver bottleneck to kernel driver

Windows NT provides synchronization primitives — Events (auto-reset and manual-reset), Mutexes (recursive, ownership-tracking), Semaphores (counting), and critically `WaitForMultipleObjects()`, which atomically waits on up to 64 objects with wait-any and wait-all semantics. (XDA Developers) Traditional Wine routed every sync operation through wineserver IPC: two context switches per operation. (LWN.net) (Collabora) Modern games invoke thousands of sync operations per second across many threads, (XDA Developers) making the single-threaded wineserver a severe bottleneck.

Esync (2018, by Zebediah Figura at CodeWeavers) replaced wineserver round-trips with Linux `eventfd` (XDA Developers) file descriptors and `poll()`/`epoll()` for `WaitForMultipleObjects`. (GitHub) This reduced each operation to approximately one syscall. (Collabora) The drawback was **file descriptor exhaustion**: each sync object consumed an fd, and games creating many objects could blow past (XDA Developers) `ulimit -n` (often 1024 by default; users needed to raise it to 524,288+). Certain NT semantics — `NtPulseEvent()`, atomic wait-for-all — could not be correctly emulated. (XDA Developers) Esync was never upstreamed to Wine; it remained an out-of-tree patch set enabled via `WINEESYNC=1`.

Fsync (Proton 4.11, late 2019) substituted Linux `futex()` for eventfd, eliminating fd exhaustion. (Collabora) The initial version used `futex_wait_multiple`, an out-of-tree kernel operation requiring custom kernel patches (XDA Developers) (TkG, Xanmod). After Collabora engineers André Almeida and Gabriel Krisman Bertazi upstreamed the `futex_waitv()` syscall into **Linux 5.16** (Collabora) (January 2022), fsync was updated to use this mainline interface. (GamingOnLinux) (Collabora) Fsync shared esync's semantic limitations — it approximated NT behavior using primitives not designed for the job. As Figura noted, "sometimes there is a positive performance difference, sometimes a negative one, and often no measurable difference at all." (Collabora) Both esync and fsync remained out-of-tree because upstream Wine rejected Linux-specific hacks with known correctness issues.

Ntsync is the definitive solution. Developed by **Elizabeth Figura** (the same person, formerly Zebediah Figura) at CodeWeavers, it is a proper Linux kernel driver providing NT-compatible sync primitives. Russell Clare XDA Developers Each `open()` of `/dev/ntsync` creates an isolated NT instance. LWN.net GitHub The driver provides ioctls for creating semaphores, mutexes, and events LWN.net (returned as file descriptors), plus `NTSYNC_IOC_WAIT_ANY` and `NTSYNC_IOC_WAIT_ALL` implementing proper `WaitForMultipleObjects` semantics. LWN.net lwn Crucially, the kernel has the authority to implement correct wait queues: `NtPulseEvent()` (instantaneous set-and-reset), atomic wait-for-all (acquire all or none without partial holding), and proper mutex ownership/abandonment all work correctly lwn — unlike esync and fsync. Russell Clare

The road to mainline was long. Figura first proposed "winesync" on wine-devel in January 2021. The patch series was submitted to LKML as "ntsync" in January 2024 (RFC v1), with an LWN article by Jonathan Corbet published February 16, 2024. LWN.net lwn Patches were queued into char/misc-next by Greg Kroah-Hartman for Linux 6.10 Phoronix (June 2024) but marked `BROKEN`. After multiple revisions (v5 through v7) and approximately six months of review stall, v7 was accepted into the Linux trunk for **Linux 6.14**, LinuxMusicians which released in **March 2025**. The Wine-side merge request (#7226) was submitted January 29, 2025, GitLab and **Wine 11.0** (January 13, 2026) shipped ntsync as a headline feature WineHQ — auto-detected, no environment variable needed.

Benchmark comparisons against vanilla Wine (without any sync optimization) show dramatic improvements: Dirt 3 went from 110.6 to **860.7 FPS** (678% improvement), Call of Juarez from 99.8 to 224.1 FPS, Tiny Tina's Wonderlands from 130 to 360 FPS. XDA Developers For Proton users already on fsync, the gains are more modest but the semantic correctness eliminates subtle bugs in multithreaded games. XDA Developers lwn

---

Anti-cheat: the structural wall

Anti-cheat has been the single largest compatibility blocker for Proton. The core issue is architectural: kernel-mode anti-cheat (EAC, BattlEye, Vanguard) installs Windows kernel drivers (`.sys` files) that hook into NT kernel internals — `PspLoadImageNotifyRoutine` callbacks, `ObRegisterCallbacks`, memory manager structures, and syscall tables. Wine/Proton provides a user-space Windows API translation but **does not implement the Windows kernel or its driver model**. When anti-cheat tries to load a `.sys` driver or call kernel-level interfaces, there is nothing to interact with.

The breakthrough came in September 2021. **Epic Games announced EAC Proton/Linux support** It's FOSS GamingOnLinux on September 23, 2021, and **BattlEye followed** It's FOSS Regata OS Magazine on September 24. Both work via user-mode implementations: Linux.org EAC provides a Linux shared library (`easyanticheat_x64.so`) running within Wine's process space; BattlEye operates through a "Proton BattlEye Runtime" distributed via Steam. Neither loads a kernel driver on Linux. The critical limitation is **developer opt-in**: each game publisher must explicitly enable Proton support. For EAC, games must migrate to the EOS (Epic Online Services) version of the SDK. For BattlEye, developers email BattlEye to request enablement.

This opt-in requirement means many high-profile titles remain unsupported. Notable successes include Elden Ring (EAC), DayZ (BattlEye), Arma 3 (BattlEye), and Fall Guys (EAC). Notable holdouts include **Destiny 2** (BattlEye, Bungie has refused with no public explanation) Windows Forum Steam Community and Fortnite. Tim Sweeney explained on February 7, 2022 that Fortnite relies on kernel-mode EAC on Windows for maximum protection, PC Gamer and the user-mode Proton version provides "comparatively high-trust configuration, where only part of the anti-tampering protections of EAC are active." GamingOnLinux With Linux's open kernel — users can compile custom kernels and bypass any kernel-level check — he argued the threat model is inadequate for a game with 60M+ active players. GameSpot

Riot's Vanguard represents the hardest case. Grokipedia Unlike EAC/BattlEye, which load on demand when a game launches, Vanguard's kernel driver (`vgk.sys`) starts at **Windows boot** (`SERVICE_BOOT_START`), observing every driver that loads after it. S4dbrd Medium It enforces TPM/Secure Boot, blocks known vulnerable drivers, and is researching HVCI (Hypervisor-Protected Code Integrity) enforcement. This boot-time trust chain verification is fundamentally impossible under Wine/Proton, where there is no Windows boot process. Medium Since Patch 14.4 (early 2024), League of Legends requires Vanguard, ending all Linux play — which previously supported an estimated ~700 Wine users playing ~6,000 games per day. Other blockers include nProtect GameGuard (kernel-mode, ~15+ Asian MMOs), XIGNCODE3, and FACEIT Anti-Cheat (boot-time driver similar to Vanguard). ACM Digital Library

The trend toward **hypervisor-based anti-cheat** — running below the OS to counter DMA-based cheating devices — will widen the gap. Vanguard is already exploring HVCI enforcement. Remote attestation via TPM 2.0 and Secure Boot is a more promising long-term direction Medium potentially compatible with Linux (SteamOS supports Secure Boot), ResetEra but requires significant infrastructure work.

Container and runtime infrastructure

Proton runs inside a **pressure-vessel** container — Valve's bubblewrap (bwrap)-based launcher developed by Collabora (primarily Simon McVittie). Its primary purpose is library compatibility, not security: it creates a predictable Debian-based filesystem overlay regardless of the host distro. fosdem

The Steam Linux Runtime ships in named versions (after Team Fortress 2 classes): **scout** (SLR 1.0, Ubuntu 12.04-era, used via `LD_LIBRARY_PATH` for legacy native games), **soldier** (SLR 2.0, Debian 10, first to use pressure-vessel, used by Proton 5.13–7.0), **sniper** (SLR 3.0, Debian 11, used by Proton 8.0+ and Counter-Strike 2/Dota 2), GitHub and **medic** (SLR 4.0, Debian 12, still in early stages as of 2026).

Pressure-vessel uses Linux namespaces to create an isolated mount namespace where `/usr` comes from the runtime, then selectively bind-mounts the game directory, Steam client data, Proton files, and — crucially — host GPU drivers. The tool `capsule-capture-libs` enumerates the host's OpenGL/Vulkan ICDs and makes them available inside the container. For Mesa (AMD/Intel), the open-source driver `.so` files are captured; for NVIDIA proprietary, the corresponding proprietary libraries are passed through. If the host driver requires a newer glibc than the runtime provides, the runtime can override its own glibc with the host's.

Wine's security model and its limitations

Wine's security posture is explicitly minimal. Its FAQ states: **"Wine is NOT intentionally secure. It can and will do EVERYTHING that the calling user can do in the base operating system."** WineHQ Wine processes run with the full privileges of the invoking Unix user. WineHQ By default, Wine maps `Z:\` to `/` — the entire Linux filesystem — giving any Windows application read/write access to all user-accessible files. Removing `Z:\` is described as only a "weak defense" because Wine provides internal APIs to access Linux paths regardless. Linux Mint Forums

Research by Duncan et al. (2019, Journal of Computer Virology and Hacking Techniques) tested 10 Windows malware samples in Wine and found **5 of 10 successfully compromised the system**, performing file operations, network communications, and registry changes comparable to Windows execution. Known CVEs include CVE-2018-12932 and CVE-2018-12933 (heap-based buffer overflow and out-of-bounds write in `PlayEnhMetaFileRecord()`, both in Wine 3.7), CVE-2006-3507 (arbitrary code execution via WMF file handling), and CVE-2005-2260 (world-readable temp files). The recurring vulnerability classes are metafile/image parsing (EMF, WMF), font parsing, and buffer overflows in reimplemented DLLs.

Pressure-vessel provides some mitigation through mount and IPC namespace isolation, but McVittie's own FOSDEM 2020 presentation states explicitly on slide 10: **"Not a security boundary — Would be nice, but not a priority right now."** The container shares the X11 socket (enabling keylogging and screenshot capture of other applications), the D-Bus session bus (potentially allowing command execution via systemd), network access, and audio. No seccomp filtering is applied by default. The known sandbox escape vectors are well-understood: X11 eavesdropping, D-Bus exploitation, and shared `/tmp` (required for X11 socket and XIM). Wayland improves the X11 problem — SteamOS uses Gamescope, a Wayland compositor — but X11 fallback remains common on desktop Linux.

The **anti-cheat-as-attack-surface** argument deserves mention here. On Windows, kernel-mode anti-cheat drivers with Ring 0 access represent an enormous attack surface. CVE-2020-36603 (mhyprot2.sys, Genshin Impact's anti-cheat driver) allowed local privilege escalation to SYSTEM; in August 2022, ransomware actors weaponized this signed driver to disable antivirus across enterprise networks. The ESEA competitive gaming platform infamously distributed a Bitcoin miner within its anti-cheat client in 2013. An ARES 2024 academic paper analyzing BattlEye, EAC, FACEIT AC, and Vanguard found all four share significant technical characteristics with rootkits. This attack surface is a core reason Linux kernel developers resist adding kernel anti-cheat support — combined with the practical impossibility of maintaining proprietary modules across Linux's unstable internal ABI.

Media, audio, input, HDR, and gamescope

Media Foundation has been one of the most persistent compatibility gaps. Many Windows games use MF APIs for FMV cutscenes (typically H.264/AAC in MP4 containers). Wine's **winegstreamer** bridges Media Foundation and DirectShow APIs to GStreamer, with pioneering work by Zebediah Figura at CodeWeavers. Proton-GE historically carried additional MF patches (including proprietary codec workarounds) before they were upstreamed. Proton 10 introduced **winedmo** as a new default media backend, with `PROTON_MEDIA_USE_GST=1` available as a fallback to the winegstreamer path. H.264, H.265, WMV, VP9, WMA, and AAC generally work when proper GStreamer plugins are available.

Wine's **audio** path routes through mmdevapi (the Multimedia Device API endpoint enumerator) to backend drivers: winepulse (PulseAudio, also works with PipeWire's PulseAudio compatibility), winealsa (ALSA), or the newer winepipewire (native PipeWire, still maturing). Default latency is ~80ms for safety; with rtkit and PipeWire, ~3.3ms has been demonstrated in WASAPI mode, though underruns can occur.

Input handling combines Wine's XInput/DirectInput implementations with Steam Input, Valve's controller abstraction layer that remaps any controller (DualShock, DualSense, Switch Pro, Steam Controller) to appear as an XInput device before it reaches the game. Wine's winebus.sys enumerates host HID devices and presents them as Windows HID devices.

HDR works end-to-end on SteamOS via the pipeline: game → DXVK/VKD3D-Proton (HDR swapchain, enabled with `DXVK_HDR=1`) → Vulkan → gamescope (HDR tone mapping) → DRM/KMS → display. Proton 8.0-5 (January 2024) added HDR for numerous titles including Resident Evil 2, Hogwarts Legacy, and Alan Wake 2. On desktop Linux, HDR works via Wine's Wayland driver with compositor support for the xx-color-management-v4 protocol (KDE Plasma 6+ has early support), but GNOME lacks scRGB support, NVIDIA has known issues with gamescope HDR, and no standardized Linux color management userspace API exists yet.

Gamescope, the Wayland micro-compositor used by SteamOS, was forked from steamcompmgr and rewritten for Wayland/Vulkan on wlroots. Pierre-Loup Griffais is the primary developer, with Joshua Ashton contributing color/HDR work. It runs in two modes: embedded (direct DRM/KMS, SteamOS Game Mode) and nested (inside another compositor, desktop use). Gamescope handles FSR 1.0 upscaling, integer scaling, frame rate limiting, resolution spoofing, and HDR tone mapping. On AMD hardware, color transforms use AMD Display Core hardware planes at scanout time for zero overhead.

Ecosystem: CodeWeavers, Proton-GE, and the launcher landscape

CodeWeavers' role cannot be overstated. Most Wine development for Proton is performed by CodeWeavers engineers under Valve contract, with changes upstreamed to Wine and then integrated into both Proton and CrossOver. Key individuals include Alexandre Julliard (CTO), Arek Hiler (Proton lead developer), Elizabeth Figura (ntsync), and Andrew Eikum. In May 2023, CodeWeavers transitioned to an employee ownership trust.

Proton-GE, maintained by Thomas Crider (GloriousEggroll, a Red Hat engineer), is the most prominent community fork. Available on GitHub with ~13,900 stars, it carries additional media foundation patches, protonfixes (per-game automated workarounds), Wine-Wayland patches, ntsync enablement, and bleeding-edge rebases of Wine/DXVK/VKD3D-Proton. It typically releases every two weeks based on Proton Experimental bleeding-edge builds. Crider also created the Nobara Project (Fedora-based gaming distro) and umu-launcher (Unified Linux Wine Game Launcher for non-Steam use).

The broader ecosystem includes **Lutris** (universal launcher managing Wine/Proton runners with community installer scripts), **Bottles** (GTK4-based Wine prefix manager with snapshot/restore), and **Heroic Games Launcher** (GOG, Epic, and Amazon games launcher popular on Steam Deck). All leverage the same underlying stack: Wine/Proton, DXVK, VKD3D-Proton, and container runtimes.

What remains technically hard

Shader compilation stutter persists despite GPL. Games that create shaders at draw time rather than during loading screens — notably many Unreal Engine 4 titles — still cause hitches. Valve's Fossilize-based pre-caching helps but requires per-game, per-GPU, per-driver-version data collection. Geometry and tessellation shaders are blacklisted from GPL pre-compilation because Vulkan forces them to compile together.

32-bit support deprecation is an ongoing pressure point. Arch Linux already transitioned Wine packages to pure WoW64 builds in 2025. Fedora and Debian have discussed dropping i686 support. Wine's WoW64 mode (build with `--enable-archs=i386,x86_64`) runs 32-bit Windows apps inside 64-bit host processes by thunking 32-bit NT syscalls to 64-bit Unix implementations. Wine 11.0 declared WoW64 at full feature parity and deprecated pure 32-bit prefixes. The remaining penalty: OpenGL has a thunking overhead for 32-bit apps, though Vulkan and DXVK paths are unaffected.

ARM64 support is strategically important. **FEX-Emu**, a fast usermode x86/x86-64 emulator for ARM64 Linux, received **Valve sponsorship** (confirmed December 2025) to power the Steam Frame VR headset (Qualcomm Snapdragon 8 Gen 3). FEX uses an advanced binary recompiler with custom IR, supports SSE4.1 and AVX/AVX2, and forwards GPU calls to native ARM64 Vulkan/OpenGL drivers to avoid emulation overhead for graphics. Combined with Wine's WoW64 mode, FEX enables running 32-bit x86 Windows games on ARM64 Linux. Early benchmarks on ARM64 desktops show Portal 2 exceeding 100 FPS and DOOM 2016 at ~60 FPS, limited by CPU translation rather than GPU. Box64 offers an alternative x86-64 emulator supporting ARM64, RISC-V, and LoongArch.

DirectStorage is partially implemented. VKD3D-Proton 2.10 (September 2023) implemented `VK_NV_memory_decompression` for GPU-accelerated GDeflate decompression on NVIDIA GPUs. AMD uses a CPU fallback when the extension is unavailable. Full NVMe-direct-to-GPU DMA (bypassing CPU RAM) is not available via Vulkan.

D3D12 Work Graphs, announced by Microsoft at GDC 2024, are experimentally emulated in VKD3D-Proton 3.0 via normal compute shaders. No native Vulkan equivalent exists. Ironically, HKA reports the emulation "can massively outperform native driver implementations" in tested scenarios, and he has expressed skepticism about work graphs as a paradigm.

The **performance gap** versus native Windows is typically within **0–5%** for well-supported titles. DXVK and VKD3D-Proton are translators rather than emulators, so overhead is minimal for well-mapped APIs. In CPU-limited scenarios, Linux's lower OS overhead and DXVK's efficient Vulkan usage can actually outperform native D3D11. The gap widens for shader compilation stutter, WoW64 OpenGL thunking, and games with complex D3D11 state management. Subsystems that remain weak include printing, accessibility (Windows UI Automation and MSAA are barely implemented), obscure COM interfaces, DRM-laden installers (SafeDisc, StarForce), and .NET/Mono edge cases.

Conclusion

Proton's technical achievement is not any single component but the integration of dozens of independently complex systems into a stack that makes most Windows games work on Linux with minimal user intervention. The project's success rests on three pillars: the three-decade foundation of Wine's API reimplementation, now architecturally modernized through the PE conversion; the Vulkan-based graphics translators (DXVK and VKD3D-Proton) that eliminated OpenGL as a bottleneck; and the sync primitive evolution from wineserver IPC through esync and fsync to ntsync, which finally delivered both correct semantics and kernel-level performance.

The remaining barriers are increasingly structural rather than engineering gaps. Kernel-mode anti-cheat requires Windows kernel interfaces that Wine fundamentally does not and cannot provide, and hypervisor-based anti-cheat trends will widen this divide. HDR on desktop Linux awaits Wayland protocol standardization. Shader compilation stutter is an inherent mismatch between D3D's incremental and Vulkan's monolithic pipeline models, addressable but not eliminable. ARM64 support via FEX-Emu and Wine WoW64 points toward a future where the x86 dependency itself may be removable.

What distinguishes Proton from earlier efforts like Cedega is not ambition but institutional structure. Valve funds CodeWeavers to do the Wine work, employs DXVK and VKD3D-Proton developers, contracts Collabora for runtime infrastructure, and upstreams nearly everything — ntsync into the Linux kernel, PE conversion into Wine, shader compiler libraries shared across projects. The Steam Deck created the commercial incentive to sustain this investment. The result is a compatibility layer that is technically closer to a managed Windows subsystem than to the ad-hoc translation layers of prior decades, and one whose trajectory is defined less by what it cannot yet do than by the rate at which remaining gaps close.