Impact of Debloating on Operating System Performance and Privacy

Parth Mital[1], Anshul Naphade[2]
[1]*Bachelor of Technology – Computer Science, Vellore Institute of Technology, Vellore, India*
[2]*Bachelor of Technology – Computer Science, Vellore Institute of Technology, Vellore, India*
parth.mital2023@vitstudent.ac.in
anshul.sandeep2023@vitstudent.ac.in
October 2025

Abstract

Bloatware, defined as preinstalled or unnecessary applications and services, has long been a concern in modern operating systems. It consumes processing power, memory, storage, and network bandwidth, while also introducing potential privacy risks through telemetry and data collection. This study investigates the effects of debloating on system performance and privacy through comprehensive testing of Windows 10, Windows 11, and various Linux distributions under controlled virtual machine environments. Using standardized benchmarks across identical hardware configurations, the research compares baseline system metrics with debloated environments. Findings indicate that debloating significantly improves boot time, RAM usage, battery efficiency, and overall responsiveness while reducing telemetry endpoints and data leakage risks. Stock Windows 10 Pro exhibited reductions of up to 36% in idle RAM usage and 31% faster boot times after debloating, while Windows 11 showed moderate improvements with 39% RAM reduction and 32% boot time improvement. Custom Windows builds including Ghost Spectre, AtlasOS, and ReviOS demonstrated even more aggressive optimization at the cost of certain features and update compatibility. Linux distributions, inherently more modular, started with lower resource footprints but still benefited from selective debloating. However, debloating can also introduce instability and reduce support for updates if performed aggressively without careful consideration of system dependencies.

1. Introduction

This case study originates from an extensive engagement with system performance enhancement and resource optimization within operating systems. The primary focus is the debloating and optimization of modern systems to achieve improved efficiency and longevity. Over the years, consistent experimentation with software, tools, and operating systems has revealed significant potential for performance gains through systematic reduction of unnecessary components.

Optimization represents the balance between functionality, performance, and sustainability. It encompasses both hardware and software processes to ensure maximum utilization of available resources. Given that the operating system governs all device operations, its optimization yields the most substantial improvements in overall system performance and user experience.

Contemporary operating systems frequently include bloatware—unnecessary preinstalled applications, redundant background services, and intrusive telemetry processes—that consume processing power, memory, and network bandwidth. These elements degrade performance, reduce battery life, and raise privacy concerns through continuous data collection. Despite these effects, general users often remain unaware of the extent of the issue or its implications for system efficiency and security.

A representative example of long-term optimization outcomes can be observed in an Acer Nitro 5 laptop (Intel Core i7-9750H, NVIDIA GTX 1650, 16 GB RAM, dual storage: 256 GB SSD and 1 TB HDD) purchased in 2020 and maintained in daily operation for over five years. Through systematic debloating, software streamlining, and performance tuning, the device continues to perform at or beyond its original operational efficiency.

This study seeks to analyse, in a structured and empirical manner, the impact of debloating on both performance metrics and privacy implications across multiple operating systems, establishing a methodological framework for sustainable optimization practices.

1.1 Research Problem

In recent years, operating systems have evolved from lightweight, task-oriented platforms into complex ecosystems containing extensive built-in applications, background services, vendor-specific integrations, and telemetry frameworks. While these additions are often marketed as improvements in usability, security, or cloud connectivity, they have introduced a substantial layer of software bloat—commonly referred to as *bloatware*.

Bloatware consumes valuable system resources such as CPU cycles, RAM, disk space, and network bandwidth, often operating continuously in the background. Beyond the immediate performance degradation, these preinstalled

services frequently transmit telemetry and diagnostic data to remote servers, raising privacy concerns regarding user data collection, behavioural profiling, and the lack of transparent user consent.

This issue is particularly evident in modern consumer versions of Microsoft Windows, which include services such as Cortana, Microsoft Teams integration, OneDrive sync, Windows widgets, and the Xbox platform—many of which remain active even when unused. Windows 10 and 11 maintain persistent outbound telemetry connections even after users disable tracking settings through the interface, with processes including DiagTrack and Connected User Experiences contributing to background CPU and network utilization. Similar, though less severe, trends have been observed in popular Linux distributions like Ubuntu, which include analytics and feedback mechanisms by default. As a result, users experience slower system responsiveness, longer boot times, and increased background network activity, while also being exposed to potential privacy risks.

Although the global technology community has long discussed "debloating" as a countermeasure—through manual configuration, registry edits, package removal, or third-party utilities—most available evidence is anecdotal and lacks methodological rigor. Numerous online tutorials and forums claim that debloating dramatically improves system speed and reduces telemetry, but these claims are rarely supported by consistent, reproducible measurements conducted under controlled conditions.

Furthermore, few studies compare the quantitative effects of debloating across multiple operating systems under identical hardware or virtualized conditions. Without a controlled experimental framework, it remains unclear how much of the perceived improvement arises from genuine optimization versus placebo effects or uncontrolled testing environments.

Therefore, the core research problem addressed in this study is:

To what extent does debloating objectively improve operating system performance and reduce telemetry-based privacy risks across different versions and configurations of Windows and Linux when evaluated in standardized virtual machine environments?

This study seeks to fill that gap by conducting structured, repeatable tests using consistent configurations, focusing on measurable outcomes such as boot time, idle resource utilization, and network telemetry activity before and after systematic debloating.

1.2 Objectives

The primary goal of this research is to conduct a systematic evaluation of how operating system debloating influences both performance and privacy. While optimization and debloating have been widely discussed in online communities, there is a lack of structured data quantifying their measurable effects under controlled experimental setups. To bridge this gap, the study is designed around the following specific objectives:

1. **To evaluate the impact of debloating on key system performance metrics.** This includes a detailed analysis of quantifiable parameters such as boot time, idle and active memory usage, CPU utilization, disk I/O activity, and battery efficiency before and after debloating. By maintaining identical hardware configurations through virtual machines, the study ensures that any performance variation can be attributed solely to the removal of bloatware and background services.
2. **To assess the effect of debloating on user privacy and background telemetry.** The research analyses outbound network traffic, active service counts, and telemetry-related processes using monitoring tools such as Wireshark, netstat, and system logs. The objective is to identify reductions in unsolicited data transmission, background network connections, and automated vendor communication after debloating.
3. **To compare stock, debloated, and custom operating system builds across Windows and Linux platforms.** Different versions and distributions of operating systems including Windows 10 Pro, Windows 11 Pro, Windows LTSC editions, custom Windows builds (Ghost Spectre, AtlasOS, ReviOS), Ubuntu LTS, Linux Mint, and Arch Linux are tested under identical virtual machine conditions. This comparison highlights how system design philosophy—monolithic versus modular—affects bloat levels and optimization potential.
4. **To determine the trade-offs, risks, and limitations of debloating.** While debloating can improve responsiveness and privacy, it may also introduce side effects such as instability, missing dependencies, broken update mechanisms, or reduced system functionality. This objective involves identifying which components can be safely removed and which are critical to maintaining OS integrity and user experience.

1.3 Scope

The scope of this study is deliberately defined to ensure experimental consistency, technical accuracy, and relevance to contemporary operating system environments. It focuses exclusively on desktop-class operating systems and excludes mobile or embedded platforms to maintain methodological precision. The parameters of the study are as follows:

1.3.1 Operating Systems

The research examines selected versions of Microsoft Windows and Linux operating systems that represent varying degrees of bloat, modularity, and system design philosophy. Specifically, the study includes:

- **Windows 10 Pro** (standard consumer edition, debloated manually using Chris Titus Tech Winutil, and LTSC variant)
- **Windows 11 Pro** (stock, debloated using Winutil, and custom ISO builds including Ghost Spectre, AtlasOS, and ReviOS)
- **Linux distributions** including Linux Mint 22.1 Cinnamon, Ubuntu 22.04 LTS with GNOME desktop, and Arch Linux with XFCE, all from standard official ISOs

This variety allows comparative analysis between proprietary and open-source ecosystems, as well as between systems designed for general consumer use and those built for minimalism and user control.

1.3.2 Testing Environment

All experiments are conducted within secure virtual machine environments configured with identical hardware specifications to ensure fair comparison. VMware Workstation is used as the hypervisor platform with custom, consistent resource allocation: 4 virtual CPU cores, 8GB RAM, 50GB virtual SSD storage, and bridged gigabit networking. The virtualization layer ensures isolation, reproducibility, and protection of the host system. Baseline snapshots are created for each OS before debloating to maintain experimental integrity and allow for precise before-and-after comparisons.

While all comparative benchmarks for this research were performed in VMware Workstation virtual machines, my initial years of hands-on debloating experimentation were conducted on my actual personal laptop—the Acer Nitro 5 mentioned earlier—running Windows 11 as my daily driver operating system. This real-world experience informed the methodology and provided practical validation of the VM-based findings.

1.3.3 Debloating Methods

The study employs both manual and tool-assisted debloating procedures to achieve systematic optimization. After extensive experimentation with various debloating tools over the years, I settled on Chris Titus Tech's Windows Utility (Winutil) as my definitive standard for Windows debloating. This tool has become my go-to solution for everything concerning Windows optimization and privacy enhancement.

Manual debloating involves disabling or uninstalling unnecessary background services, startup tasks, and telemetry components using native tools such as Windows PowerShell, services.msc, Task Scheduler, or Linux terminal commands including apt purge, systemctl disable, and pacman remove.

Tool-based debloating for Windows utilizes Chris Titus Tech's Winutil, which consolidates comprehensive debloating, privacy tweaks, application removal, and system optimization into a single, well-documented, community-driven PowerShell script with an intuitive graphical interface. For Linux systems, custom shell scripts and package manager commands achieve similar streamlining of services and background processes.

Each method is documented to maintain consistency and transparency across test cases, ensuring reproducibility of results.

1.3.4 Metrics of Evaluation

Two primary dimensions of analysis are considered to provide a comprehensive assessment of debloating effectiveness:

Performance Metrics include boot time measured from VM power-on to idle desktop, idle and active RAM usage captured at system stabilization, CPU utilization percentage during idle state, disk I/O rates during background operations, and battery efficiency estimated through idle power consumption where applicable.

Privacy Metrics include telemetry activity measured through outbound network connection monitoring, number of active background services at boot, frequency and destination of automated data transmissions, and identification of vendor-specific processes engaged in unsolicited communication.

These metrics collectively provide a comprehensive view of how debloating influences both system efficiency and user privacy across different operating system platforms and configurations.

1.4 Limitations

While this study aims to present a comprehensive and controlled evaluation of debloating's effects on operating system performance and privacy, certain constraints are acknowledged that may influence the generalizability of the findings.

1.4.1 Virtualized Testing Environment

All experiments are conducted within virtual machines to maintain uniform system configurations and ensure repeatability. However, virtualization introduces a degree of abstraction between the guest and host systems. Consequently, certain hardware-dependent behaviours—such as thermal throttling, power management efficiency, and battery drain on portable devices—cannot be accurately replicated or measured. As a result, while the trends observed are representative of real-world

performance, absolute values may differ slightly on physical hardware.

1.4.2 Exclusion of Mobile and Embedded Platforms

This study is limited to desktop-class operating systems including Windows and Linux distributions, and intentionally excludes mobile platforms such as Android and iOS, as well as embedded systems including IoT devices and routers. These environments operate under fundamentally different architectures, kernel optimizations, and power constraints, making direct comparisons inappropriate for the scope of this research.

1.4.3 Hardware and Configuration Variability

Although all tests are performed under standardized virtual configurations, real-world performance and privacy outcomes can vary depending on specific hardware components, firmware optimizations, and user-applied modifications. Factors such as CPU generation, storage type (HDD versus SSD versus NVMe), network interface performance, and even BIOS settings may lead to deviations from the measured results when replicated on physical systems.

1.4.4 Software and Update Dependencies

Operating systems evolve continuously through updates, patches, and version changes. Some results presented in this study may be time-sensitive, as future updates could alter service behaviour, telemetry frequency, or performance characteristics. Similarly, certain debloating tools used in the study may become deprecated or modified over time, affecting reproducibility. The data presented reflects the state of each operating system and tool as of October 2025.

2. Background and Related Work

2.1 Overview

Operating system performance and privacy have long been subjects of research and community interest. As modern OS distributions grow in complexity, the inclusion of bundled applications, telemetry systems, and vendor-specific services has drawn increasing criticism from both researchers and end users. The concept of *debloating*—selectively removing non-essential components—has emerged as a practical response to these concerns. This section reviews prior academic research, industry initiatives, and community-driven projects relevant to operating system debloating and optimization.

2.2 Prior Research on OS Bloatware and Telemetry

Several studies have examined the impact of preinstalled software and background services on system performance and privacy. Research conducted by privacy advocates and independent analysts has shown that Windows 10 and 11 maintain persistent outbound telemetry connections even after users disable tracking settings through the standard interface. These processes, including DiagTrack (Connected User Experiences and Telemetry service) and dmwappushservice, contribute to background CPU and network utilization even during idle states. Microsoft's own documentation acknowledges the collection of diagnostic data at multiple levels, with the "Required" setting still transmitting device health metrics, installed software inventories, and performance data.

Academic papers and benchmarking studies have demonstrated that removing non-critical background services can lead to measurable gains in boot time, memory availability, and process responsiveness. System performance analysis across different Windows editions has shown that stock installations exhibit significantly higher resource consumption compared to debloated or LTSC variants. Research comparing distributions such as Ubuntu, Fedora, and Arch Linux has highlighted that system modularity directly affects resource consumption. Minimal distributions with reduced daemons and startup services consistently demonstrate lower idle resource usage and faster system initialization, often by 30-50% compared to full-featured desktop environments.

These studies establish a foundation for understanding how software bloat influences both quantitative performance metrics and qualitative user experience, validating the need for controlled debloating methodologies.

2.3 Industry Efforts Toward Minimalism

In response to user demand for leaner operating systems, both proprietary and open-source communities have introduced lighter alternatives designed to minimize bloat from the outset.

Windows LTSC (Long-Term Servicing Channel) represents Microsoft's acknowledgment that not all users require constant feature updates and consumer-oriented integrations. LTSC editions are designed for enterprise and industrial environments where stability and predictability outweigh new features. They omit components such as Cortana, the Microsoft Store, Xbox services, Windows widgets, and consumer telemetry frameworks, resulting in improved stability, lower background activity, and reduced attack surface. LTSC receives only security updates for up to 10 years, avoiding the disruptive semi-annual feature updates that often introduce bugs or break existing workflows. Performance comparisons show LTSC boots approximately 15-20% faster than Pro editions with similar hardware, uses 20-30% less RAM at idle, and generates significantly fewer outbound network connections.

Lightweight Linux Distributions embody the modular philosophy that prevents bloat by design. Distributions such as Arch Linux, Alpine Linux, and Debian Minimal provide modular installation options that let users assemble

systems from the ground up, installing only the packages and services they explicitly need. This inherently prevents bloat by giving users control over which packages and services are included. Arch Linux minimal installations can boot in under 15 seconds and consume as little as 400MB of RAM with a basic desktop environment, compared to 1.9GB for stock Ubuntu with GNOME.

Performance-Focused Builds including Fedora Silverblue and Ubuntu Minimal also reflect a growing industry shift toward modular and containerized system architectures, reducing unnecessary system overhead while maintaining security and update mechanisms.

These examples illustrate how the need for optimization and privacy has influenced OS development at an industry level, though they remain niche solutions compared to mainstream consumer editions.

2.4 Community and Open-Source Debloating Projects

A significant portion of debloating research and experimentation originates from community-driven projects. Enthusiasts and developers have created a variety of tools and scripts aimed at streamlining operating systems while preserving core functionality.

Chris Titus Tech's Windows Utility (Winutil) has emerged as one of the most comprehensive and widely-adopted debloating tools for Windows. With over 41,900 stars on GitHub, active development, and extensive community support, Winutil consolidates debloating, privacy enhancement, application installation, system tweaks, and custom ISO creation into a single PowerShell-based tool with both command-line and graphical interfaces. Its advantages over alternatives include comprehensive documentation, modular design allowing selective application of tweaks, regular updates incorporating community feedback, and compatibility across Windows 10 and 11 editions. Unlike fragmented PowerShell scripts or limited registry tweakers, Winutil provides granular control over telemetry services, startup programs, scheduled tasks, and Windows features while maintaining system update capability and driver support.

Other Windows Debloating Tools including O&O ShutUp10, Win11Debloat, NTLite, and MSMG Toolkit each offer distinct approaches. O&O ShutUp10 focuses primarily on privacy settings through registry modifications but lacks comprehensive app removal features. Win11Debloat provides script-driven component removal with granular control but requires higher technical expertise. NTLite excels at deep ISO-level customization for enterprise deployment but is complex and partially paid. Winutil strikes the optimal balance between ease of use, comprehensiveness, and safety, making it the preferred choice for both casual users and advanced system optimizers.

Linux Debloating Approaches leverage the inherent modularity of open-source systems. The community contributes shell scripts and package configurations to minimize startup daemons, remove redundant dependencies, and optimize system boot performance using systemctl, package managers (apt, dnf, pacman), and custom automation scripts. The transparency of open-source systems allows users to precisely understand and control which services run at boot, which packages consume resources, and how telemetry—if any—is configured.

Custom Windows ISO Projects including Ghost Spectre, AtlasOS, and ReviOS represent community-led efforts to create pre-debloated Windows installations. These projects strip telemetry, bundled applications, and non-essential services during installation rather than post-installation cleanup. While offering convenience and aggressive optimization, they also introduce risks related to update compatibility, driver support, and potential instability if critical components are removed.

While these community projects provide valuable practical insights and have helped millions of users optimize their systems, they often lack controlled, data-driven validation—an area this study seeks to address through empirical testing and quantitative measurement.

2.5 Summary

Prior literature and industry trends indicate that bloatware and telemetry have measurable effects on performance and privacy across modern operating systems. Although various solutions exist—ranging from official minimal OS releases to community debloating scripts—most research remains fragmented and anecdotal, lacking the systematic rigor of controlled experimentation. This study builds on that foundation by conducting standardized, reproducible tests across multiple OS versions under identical virtual machine conditions, thereby contributing structured, quantitative data to an area dominated by qualitative observations and user testimonials.

3. Methodology

This section describes the experimental setup, operating systems tested, debloating procedures, performance and privacy metrics, and the standardized procedure used to ensure reproducibility and consistency across all test cases. The methodology has been carefully designed to minimize bias and isolate the effects of debloating from external variables.

3.1 Test Environment

All experiments are conducted in virtualized environments using VMware Workstation to ensure uniformity and reproducibility across different operating systems. The virtualization platform provides system-level isolation,

precise resource control, and the ability to create identical baseline snapshots for fair comparison.

Hardware Specifications: Each virtual machine is configured identically with 4 virtual CPU cores, 8GB of RAM, and a 50GB virtual SSD disk with GPU acceleration enabled where supported. The host system is equipped with an Intel Core i7-9750H processor, 16GB of physical RAM, and SSD storage to minimize I/O bottlenecks and ensure the virtual machines have adequate resources for realistic performance measurement.

Baseline Snapshots: A clean snapshot is taken immediately after each OS installation and initial configuration to serve as the control reference. All experiments are performed using cloned snapshots to maintain consistency and eliminate cumulative changes between runs. This approach allows precise measurement of the impact of debloating without interference from installation variance or configuration drift.

Network Configuration: Virtual machines are configured with bridged networking to accurately monitor outbound connections and telemetry traffic. A clean network environment is maintained to ensure all captured traffic originates from the guest OS rather than host system interference.

This controlled environment allows for a fair and repeatable comparison between stock and debloated system states without interference from hardware variation, background processes on the host, or inconsistent network conditions.

3.2 Operating Systems Tested

A diverse set of Windows and Linux operating systems are selected to represent different philosophies of design, modularity, and resource management. The selection includes mainstream consumer editions, enterprise variants, custom community builds, and minimal Linux distributions.

3.2.1 Windows Variants

Windows 10 Pro (Stock): Standard installation with all default services, bundled applications, and telemetry enabled. This represents the typical consumer experience for Windows 10 users as of October 2025.

Windows 10 Pro (Debloated): Stock edition optimized using Chris Titus Tech's Winutil to remove unnecessary UWP apps, disable telemetry services, eliminate redundant scheduled tasks, and optimize background processes while preserving update functionality and driver support.

Windows 10 LTSC 2021: Long-Term Servicing Channel edition, designed for enterprise stability with reduced feature updates and minimal consumer-oriented bloatware. This version inherently excludes Microsoft Store, Cortana, Xbox services, and most telemetry frameworks.

Windows 11 Pro (Stock): Standard consumer edition with all default telemetry, bundled applications including Teams, widgets, and enhanced Microsoft Account integration. Represents the current mainstream Windows experience.

Windows 11 Pro (Debloated): Manual optimization using Winutil for telemetry removal, service optimization, and non-essential app uninstallation while maintaining core system functionality.

Ghost Spectre Windows 11 SuperLite: Community-modified Windows 11 ISO with aggressive debloating, removal of Microsoft Store, Edge browser integration, telemetry services, and non-essential system components. Designed for gaming and performance with minimal resource footprint.

AtlasOS Windows 11: Ultra-lightweight custom playbook applied to Windows 11, removing extensive background services, telemetry, and features to achieve maximum performance and minimal latency. Targets competitive gaming and resource-constrained systems.

ReviOS Windows 11: Balanced custom playbook focusing on performance optimization while maintaining broader compatibility, update support, and user-friendly features compared to more aggressive alternatives.

3.2.2 Linux Variants

Linux Mint 22.1 Cinnamon (Stock): Standard installation of Linux Mint with the Cinnamon desktop environment, representing a user-friendly, Windows-like Linux experience with moderate resource consumption.

Linux Mint 22.1 Cinnamon (Debloated): Modified installation with selective removal of unnecessary services, background processes, and telemetry using systemctl disable and apt purge for unused packages.

Ubuntu 22.04 LTS (Stock): Standard installation with GNOME desktop environment, default packages, and Ubuntu's telemetry opt-in mechanisms. Represents a mainstream Linux desktop distribution.

Ubuntu 22.04 LTS (Debloated): Optimized installation with removal of snap-related services, telemetry reporting, and non-essential background daemons using systemctl and package removal.

Arch Linux (Minimal Install): Base system installed from scratch using the official Arch installation guide with only essential packages and XFCE desktop environment. Represents a minimal, user-controlled Linux configuration.

This variety enables cross-comparison between proprietary and open-source systems, as well as between general-purpose and minimal OS builds, highlighting the impact of design philosophy on bloat and optimization potential.

3.3 Debloating Procedure

Each operating system undergoes a controlled debloating process using consistent principles to ensure fairness and reproducibility. The goal is to remove non-essential components while preserving core system functionality, security updates, and driver compatibility.

3.3.1 Windows Debloating

Primary Tool: Chris Titus Tech's Windows Utility (Winutil) accessed via PowerShell using the command irm "https://christitus.com/win" | iex. This tool provides a graphical interface for comprehensive Windows optimization.

Components Removed or Disabled:

- **Telemetry Services:** DiagTrack (Connected User Experiences and Telemetry), dmwappushservice, OneDrive sync, and diagnostic data collection services
- **Bundled Applications:** Xbox services, Cortana, Mail and Calendar, Maps, mixed reality components, unnecessary UWP apps (Candy Crush, gaming apps, etc.)
- **Background Tasks:** Redundant scheduled tasks including application telemetry, customer experience improvement, and compatibility assessments
- **Startup Programs:** Non-critical startup entries that consume resources during boot
- **Privacy Settings:** Advertising ID, activity history, location services, voice activation, and feedback frequency

Retention Policy: Core system services including Windows Update, Windows Defender, networking components, display drivers, and audio services are explicitly preserved to maintain usability, security, and stability. Device-specific drivers and manufacturer utilities are retained where necessary for hardware functionality.

Custom ISO Testing: For Ghost Spectre, AtlasOS, and ReviOS, the respective playbooks or pre-modified ISOs are installed according to their official documentation, with no additional debloating performed post-installation to accurately represent each project's optimization approach.

3.3.2 Linux Debloating

Package Removal: Commands such as apt purge, dnf remove, and pacman -Rns are used to uninstall non-essential packages including office suites, games, redundant utilities, and optional desktop applications that are not needed for the testing environment.

Service Optimization: System daemons are audited using systemctl list-unit-files and selectively disabled using systemctl disable to ensure minimal boot-time execution. Services related to printing, Bluetooth, unnecessary networking protocols, and vendor-specific telemetry are disabled where applicable.

Telemetry Removal: For Ubuntu, packages related to Ubuntu's popularity contest and error reporting are removed. For all distributions, automatic update notifications and background connectivity checks are minimized to reduce unsolicited network activity.

Custom Configurations: In some cases, lightweight alternatives to default applications are installed (e.g., replacing LibreOffice with lightweight text editors, or using minimal system monitors instead of resource-heavy alternatives).

Each change is logged in detail, and the resulting configuration is documented to allow reproducibility and transparent reporting of what was modified in each system.

3.4 Metrics Collected

The study focuses on two primary dimensions—performance and privacy—with both quantitative and qualitative data collected for each OS in its stock and debloated states.

3.4.1 Performance Metrics

Boot Time: Measured from VM power-on to idle desktop with all startup processes completed. For Linux systems, systemd-analyze provides precise kernel and userspace boot time breakdowns. For Windows, manual stopwatch timing is used from BIOS handoff to desktop idle state, averaged over three consecutive runs for accuracy.

Idle RAM Usage: Recorded after the system has stabilized for two minutes post-boot with no user applications running. Measured using Task Manager (Windows) and free -m command (Linux). All measurements are taken after system caches have stabilized to ensure representative idle state.

CPU Utilization: Average idle CPU percentage recorded after a two-minute idle period, captured using Task Manager (Windows) and htop (Linux). This metric identifies background process overhead.

Disk I/O Activity: Measured via Windows Resource Monitor and iotop/iostat on Linux under idle conditions to identify unnecessary disk access from background services, indexing, or telemetry logging.

Background Process Count: Total number of running processes at idle, counted using Task Manager (Windows) and ps aux | wc -l (Linux). This provides insight into system complexity and resource overhead from service proliferation.

Battery Efficiency (where applicable): Estimated through idle power consumption metrics in virtual

machines and supplemented by community-reported battery life improvements on physical hardware for validation.

3.4.2 Privacy and Network Metrics

Outbound Network Connections: Monitored at idle using Wireshark packet capture and tcpdump to detect telemetry, analytics, or unsolicited vendor connections. Connections are categorized by destination (Microsoft servers, Canonical servers, update services, etc.) and frequency.

Active Services at Boot: Counted using systemctl list-units --type=service --state=running (Linux) and Get-Service | Where-Object {$_.Status -eq "Running"} PowerShell query (Windows) to assess service proliferation.

Telemetry and Analytics Processes: Identified through process lists (tasklist, ps aux) and network logs to assess privacy impact. Processes explicitly related to diagnostic data, customer experience programs, and usage analytics are documented.

Update and Security Services: Distinguished from telemetry to ensure that legitimate security and update mechanisms are not conflated with privacy-invasive tracking.

3.5 Experimental Procedure

The testing process is standardized to ensure data accuracy and reliability across all operating systems and configurations:

1. **Fresh Installation:** Each OS is freshly installed from official or verified ISO images, updated to the latest available patches as of October 2025, and configured to a baseline state with minimal user customization.
2. **Baseline Measurement:** Before any debloating, the system is rebooted, allowed to stabilize for two minutes, and then all performance and privacy metrics are recorded. This establishes the control baseline.
3. **Snapshot Creation:** A VM snapshot is created to preserve the stock configuration, allowing return to baseline if needed for validation or retesting.
4. **Debloating Application:** The system is then debloated following the procedures defined in Section 3.3, with all changes documented. For tool-based debloating, screenshots and logs are captured to ensure transparency.
5. **Post-Debloat Measurement:** After debloating, the system is rebooted to ensure all changes take effect, allowed to stabilize for two minutes, and then all metrics are collected again under identical conditions.
6. **Multiple Run Averaging:** Each metric is measured across three consecutive runs with system reboots between measurements, and the average value is calculated to minimize random variance from transient system states.
7. **Cross-System Normalization:** All results are recorded in standardized units (seconds for boot time, megabytes for RAM, percentage for CPU, etc.) to allow direct cross-system comparison.

This controlled methodology ensures that performance changes can be directly attributed to the debloating process rather than external factors such as hardware variability, inconsistent network conditions, or transient system load.

4. Results

This section presents the experimental findings derived from the controlled virtual machine tests. Each operating system was analysed in its stock and debloated states, with all performance and privacy metrics recorded over three consecutive runs and averaged to ensure reliability. The focus is on measurable differences in performance, resource utilization, and telemetry activity after debloating.

4.1 Windows 10 Pro

Windows 10 Pro demonstrated the most significant improvement after debloating among all tested configurations. In the stock configuration, the system exhibited high background activity and extensive telemetry connections. After removing redundant services and applications such as DiagTrack, Cortana, OneDrive sync, Xbox services, and bundled UWP apps using Chris Titus Tech's Winutil, noticeable performance gains were observed across all measured metrics.

Idle RAM usage decreased from approximately 2.5GB to 1.6GB, reflecting a 36% reduction in memory consumption. This substantial decrease frees up valuable system resources for user applications and indicates the significant overhead imposed by default Windows services and preloaded applications. Boot time improved dramatically from 42 seconds to 29 seconds, indicating 31% faster system initialization and demonstrating reduced startup process complexity.

Average idle CPU utilization dropped from around 4.2% to 2.1%, effectively halving the background processing load. Idle disk I/O activity reduced from 4.8 MB/s to 1.2 MB/s, a 75% decrease that suggests elimination of unnecessary background indexing, telemetry logging, and service communication. The number of running background processes decreased from approximately 152 to 98, a 35% reduction that correlates directly with lower resource consumption.

Telemetry endpoints, identified through outbound network monitoring during a five-minute idle period, were reduced

from twelve active connections to four. The remaining connections were primarily Windows Update checks and time synchronization services. Eliminated connections included diagnostic data uploads, usage analytics, Cortana backend communication, Xbox Live services, and Microsoft advertising servers. The system remained stable and fully functional throughout testing, with Windows Update, Windows Defender, and all core system services continuing to operate normally. This suggests that most removed components were indeed non-essential for core operation and user productivity.

4.2 Windows 10 LTSC 2021

Windows 10 LTSC 2021, by design, already excludes most consumer-oriented bloatware and telemetry frameworks. As an enterprise-focused edition, it provided an important baseline for understanding how much of Windows 10's bloat is architectural versus optional. In its stock state, LTSC demonstrated boot time of 31 seconds, idle RAM usage of 1.8GB, and only two persistent telemetry endpoints primarily related to security intelligence updates.

Background process count was approximately 85 at idle, significantly lower than Pro edition's 152 processes. CPU idle utilization averaged 2.8% and disk I/O was minimal at 1.5 MB/s. After applying selective debloating using Winutil, further improvements were modest but measurable. Boot time decreased slightly to 28 seconds, RAM usage dropped to 1.5GB, and the single remaining telemetry connection was limited to Windows Update infrastructure.

The LTSC results validate that Microsoft's enterprise edition already embodies many optimizations that manual debloating seeks to achieve, though even LTSC benefits from selective component removal. The diminishing returns on LTSC debloating compared to Pro edition confirm that the majority of Windows bloat exists at the feature and application layer rather than the core OS architecture.

4.3 Windows 11 Pro

Windows 11 Pro showed moderate performance improvements after debloating, primarily because newer editions implement certain architectural optimizations compared to Windows 10. However, the difference between stock Pro, debloated Pro, and LTSC editions was still informative and statistically significant.

The stock Windows 11 Pro configuration exhibited heavy telemetry integration and bundled software, resulting in high background resource usage. Idle RAM consumption averaged 2.8GB with boot time of 44 seconds. The number of active telemetry endpoints during idle reached fifteen, including connections for widgets, Microsoft Teams integration, OneDrive sync, gaming services, advertising, and general diagnostic data. Background process count exceeded 165, higher than Windows 10 Pro despite claimed efficiency improvements.

After debloating using Winutil, Windows 11 Pro showed noticeable improvement in responsiveness and resource consumption. Idle RAM usage fell from 2.8GB to 1.7GB, a 39% reduction. Boot time improved from 44 seconds to 30 seconds, representing a 32% decrease in initialization time. The number of active telemetry endpoints decreased dramatically from fifteen to three, with remaining connections limited to Windows Update and security intelligence. Background process count dropped to approximately 105, a 36% reduction.

CPU idle utilization decreased from 4.0% to 2.2%, and disk I/O activity during idle dropped from 5.2 MB/s to 1.4 MB/s, indicating elimination of background indexing and telemetry logging overhead. The system maintained full functionality with Windows Update, Microsoft Defender, and core services operating normally. User-facing features like the Start menu, File Explorer, and Settings app remained responsive and stable.

4.4 Custom Windows 11 Builds

The custom Windows builds—Ghost Spectre, AtlasOS, and ReviOS—represent community-driven optimization efforts that apply aggressive debloating at the installation or post-installation level. Each targets different user priorities and optimization philosophies.

Ghost Spectre Windows 11 SuperLite demonstrated the most aggressive resource reduction among tested builds. Boot time measured 26 seconds, approximately 41% faster than stock Windows 11 Pro. Idle RAM consumption was only 1.3GB, representing a 54% reduction compared to stock. Background process count was approximately 75, less than half of the stock configuration. Network monitoring revealed virtually zero telemetry connections during idle, with only manual Windows Update checks generating outbound traffic.

However, this optimization came at significant cost. The Microsoft Store was entirely removed, preventing installation of UWP applications. Edge browser integration was stripped, requiring manual installation of alternative browsers. Some Windows features including Windows Hello, certain personalization options, and advanced display settings exhibited reduced functionality or complete absence. Driver compatibility was generally maintained, though some manufacturer-specific utilities failed to install due to missing framework dependencies.

Community feedback indicates Ghost Spectre provides substantial FPS improvements in gaming scenarios, with users reporting 15-30% higher framerates in CPU-bound titles compared to stock Windows. However, long-term update compatibility remains uncertain, as major Windows

feature updates may not apply correctly to modified system files.

AtlasOS Windows 11 exhibited even more extreme optimization targeting competitive gaming and minimal latency. Boot time measured 23 seconds, the fastest among all Windows configurations tested. Idle RAM usage was only 1.1GB with fewer than 65 background processes running. Network monitoring detected zero unsolicited outbound connections, with all telemetry, diagnostic, and analytics services completely disabled.

AtlasOS removes extensive Windows components including Windows Defender, Windows Update automatic functionality, Microsoft Store, system restore, and numerous background services. This achieves maximum performance and minimal resource footprint but introduces significant trade-offs. Users must manually manage security updates, driver installations, and system maintenance. Some games and applications fail to run due to missing .NET frameworks, DirectX components, or Visual C++ redistributables that are normally bundled with Windows.

Performance benchmarks from community testing show AtlasOS delivering 5-10% higher average FPS in games and significantly improved 1% low framerates compared to stock Windows, attributed to reduced CPU scheduler contention and eliminated background interrupts. However, system stability depends heavily on user expertise in maintaining security patches and resolving compatibility issues.

ReviOS Windows 11 strikes a middle ground between aggressive optimization and practical usability. Boot time measured 25 seconds with idle RAM consumption of 1.4GB, representing substantial improvements over stock while maintaining broader compatibility than AtlasOS. Background process count was approximately 90, and telemetry was reduced to minimal Windows Update connectivity.

ReviOS preserves Windows Update functionality, maintains Microsoft Defender as an optional component, and retains framework dependencies necessary for gaming and productivity software. The system demonstrated stable operation across diverse software installations including game launchers, development tools, and office applications. Community reception positions ReviOS as the most balanced custom build, suitable for users wanting performance improvements without sacrificing system maintainability or compatibility.

Comparative gaming benchmarks show ReviOS delivering 10-20% FPS improvements in CPU-bound scenarios compared to stock Windows 11, with excellent stability and broader software compatibility than AtlasOS. Update mechanisms remain functional, allowing users to apply security patches without manual intervention.

4.5 Linux Mint 22.1 Cinnamon

Linux Mint with the Cinnamon desktop environment represents a user-friendly, Windows-like Linux experience with moderate resource consumption relative to other Linux distributions. In its stock configuration, Mint consumed approximately 1.4GB RAM at idle with boot time of 35 seconds. Background process count averaged 65 and idle CPU utilization was approximately 1.7%. Network monitoring revealed minimal telemetry, limited to update availability checks to Linux Mint repositories.

After selective debloating using systemctl to disable unnecessary services and apt purge to remove unused packages including office suites, games, and redundant system utilities, performance improved measurably. Idle RAM usage decreased to 1.1GB, a 21% reduction. Boot time improved to 28 seconds, representing 20% faster initialization. Background process count dropped to approximately 50, and CPU idle utilization decreased to under 1%. Network activity during idle was limited to NTP time synchronization with no analytics or usage reporting detected.

The debloated Mint system maintained full desktop functionality with file management, web browsing, media playback, and system configuration tools operating normally. The Cinnamon desktop environment remained responsive and visually polished. Compared to Windows equivalents, even stock Linux Mint demonstrated significantly lower resource consumption and virtually no privacy-invasive telemetry.

4.6 Ubuntu 22.04 LTS

Ubuntu 22.04 LTS with the GNOME desktop environment exhibited the highest baseline resource consumption among tested Linux systems, reflecting GNOME's feature-rich but resource-intensive design. Stock Ubuntu consumed approximately 1.9GB RAM at idle with boot time of 38 seconds. Background process count exceeded 80, and idle CPU utilization averaged 2.3%. Network monitoring revealed automatic connections to Canonical's update servers, Ubuntu's error reporting service (Apport), and snap store background processes.

After debloating by removing snap-related services, disabling Apport error reporting, purging unused packages, and disabling unnecessary systemd services, Ubuntu's resource footprint improved significantly. Idle RAM usage dropped to 1.3GB, a 32% reduction. Boot time improved to 32 seconds, and background process count decreased to approximately 55. Network activity during idle reduced to occasional update checks with no error reporting or analytics transmission.

The debloated Ubuntu system-maintained compatibility with software installation via apt and flatpak, with desktop functionality fully preserved. However, removal of snap

infrastructure eliminated access to snap-packaged applications, requiring alternative installation methods for some software. Overall, Ubuntu demonstrated substantial bloat in its default configuration but responded well to systematic optimization.

4.7 Arch Linux (Minimal Install)

Arch Linux, installed in its minimal configuration following the official installation guide with base system packages and XFCE desktop environment, represented the leanest configuration among all tested operating systems. Boot time measured only 15 seconds with idle RAM consumption of approximately 400MB. Background process count was fewer than 40, and idle CPU utilization was under 1%. Disk I/O during idle was virtually zero with no background indexing or logging activity.

Network monitoring detected zero unsolicited outbound connections during idle periods. All network activity was user-initiated or limited to manual package update checks. No telemetry, analytics, or diagnostic reporting services were present in the default configuration. The minimal install philosophy of Arch Linux eliminates bloat at the design level, requiring users to explicitly install only desired components.

After selective further optimization including removal of optional services and documentation packages, performance improvements were marginal given the already minimal baseline. RAM usage decreased slightly to 380MB, and boot time improved to 14 seconds. The results confirm that Arch Linux's build-from-scratch approach inherently prevents bloat, making post-installation debloating largely unnecessary for users who carefully select installed packages.

4.8 Comparative Analysis

Across all tested platforms, debloating consistently led to improved system responsiveness, reduced resource consumption, and decreased telemetry activity. However, the degree of improvement varied significantly depending on how bloated the original system was, validating the hypothesis that optimization gains are proportional to initial bloat levels.

Windows 10 Pro benefited the most from debloating, with substantial reductions in RAM usage (36%), boot time (31%), and network telemetry (67% fewer connections). Windows 11 Pro showed similar patterns with 39% RAM reduction and 32% boot time improvement. LTSC editions, already optimized by design, showed diminishing returns with only 17% RAM reduction and 10% boot time improvement.

Custom Windows builds including Ghost Spectre, AtlasOS, and ReviOS demonstrated even more aggressive optimization with 50-60% RAM reductions and 45-48% faster boot times compared to stock Windows 11. However, these gains came with trade-offs in update compatibility, feature availability, and system complexity.

Linux distributions exhibited moderate improvements despite lower baseline bloat. Ubuntu's 32% RAM reduction and Mint's 21% RAM reduction were notable given their already lean default configurations. Arch Linux's minimal 5% improvement confirmed that its design philosophy inherently prevents bloat.

Telemetry reduction was most dramatic in Windows environments, where stock configurations maintained 12-15 persistent connections compared to 2-4 after debloating. Linux systems exhibited minimal telemetry even in stock configurations, with only update checks and optional error reporting generating network activity.

The improvements were not merely anecdotal but quantifiable and reproducible under identical conditions across multiple test runs. Statistical variance between runs was minimal (less than 5% for most metrics), confirming measurement reliability.

4.9 Summary of Findings

The results indicate several key insights supported by quantitative evidence:

1. **Magnitude of Improvement:** The more bloated the initial OS configuration, the greater the observable benefit from debloating. Windows consumer editions showed 30-40% resource reductions while already-lean Linux distributions showed 5-20% improvements.
2. **Performance Gains:** Reductions in RAM usage (20-40%) and boot time (15-35%) were typical across Windows systems. Linux systems showed moderate improvements (5-30%) from already efficient baselines.
3. **Telemetry Reduction:** Windows platforms showed dramatic decreases in background network connections (60-80% reduction) after debloating. Linux systems exhibited minimal telemetry even before optimization.
4. **Platform Differences:** Linux systems started leaner and exhibited inherently better privacy characteristics due to modular design and user control. Windows systems required explicit intervention to achieve comparable efficiency.
5. **Trade-offs:** No critical system instabilities were observed in carefully performed debloating using established tools like Winutil. However, aggressive custom builds sacrificed features, update compatibility, and user convenience for maximum performance.

6. **Tool Effectiveness:** Chris Titus Tech's Winutil proved highly effective for Windows debloating, achieving substantial improvements without compromising system stability or update functionality when applied with recommended settings.

Overall, these findings confirm that debloating has a tangible positive impact on both performance and privacy, particularly for consumer-grade operating systems with extensive preinstalled software and telemetry frameworks.

5. Discussion

The experimental results demonstrate that debloating can significantly improve both system performance and user privacy, particularly in operating systems that include extensive preinstalled components and telemetry frameworks. However, these benefits are accompanied by potential risks relating to stability, usability, and long-term maintainability. This section interprets the quantitative findings presented earlier and discusses their broader implications for users, developers, and the operating system ecosystem.

5.1 Performance Impact

Across all tested platforms, debloating produced measurable improvements in performance, especially in stock Windows editions. The reduction of startup programs, telemetry services, and bundled background applications directly contributed to faster boot times, lower memory consumption, and improved overall responsiveness. These improvements are not superficial but reflect genuine reduction in computational overhead.

The Windows 10 and 11 Pro editions benefited the most, with reductions in boot time by up to 32% and idle RAM usage by 36-39%. These results validate the hypothesis that unnecessary background services and bundled software impose a persistent load on system resources. The CPU and disk I/O reductions observed after debloating confirm that bloatware consumes processing cycles and storage bandwidth even during idle states, creating context switching overhead, cache pollution, and I/O contention that degrades user experience.

Even in Linux environments, which are generally leaner by design, the removal of redundant packages and services led to noticeable gains, particularly in startup speed (20-32% improvement) and idle memory footprint (21-32% reduction). The fact that minimal distributions like Arch Linux showed only marginal improvements (5%) confirms that their build-from-scratch philosophy already achieves what debloating aims to accomplish.

In essence, debloating optimizes the system's execution path by reducing background process scheduling, minimizing memory fragmentation, decreasing disk I/O contention, and eliminating unnecessary network communication. All of these factors contribute directly to a smoother, more responsive user experience with faster application launches and better multitasking performance.

5.2 Privacy Impact

From a privacy standpoint, debloating yielded a clear reduction in unsolicited network activity and telemetry processes. In stock Windows installations, numerous background connections to Microsoft servers were observed immediately after boot, transmitting diagnostic data, usage patterns, advertising identifiers, and device information even when users had ostensibly disabled tracking through the Settings interface. The persistence of these connections despite user configuration changes raises significant questions about informed consent and data sovereignty.

After debloating, the number and frequency of such outbound connections declined by 60-80%, indicating a substantial reduction in automated data collection. Critical services including Windows Update and security intelligence updates continued to function normally, demonstrating that the eliminated connections were primarily analytics and non-essential vendor communication rather than security-critical infrastructure.

Linux distributions, particularly Arch and minimal variants, exhibited almost no telemetry by default. Ubuntu's opt-in error reporting and update checking represented the extent of vendor communication, and even these could be easily disabled through standard system tools. The contrast underscores a fundamental philosophical difference in OS design: proprietary systems prioritize vendor analytics and feature integration, whereas open-source platforms emphasize user control and transparency.

These findings affirm that debloating not only improves performance but also enhances user autonomy by limiting unnecessary background communication and data sharing. For privacy-conscious users, the reduction in telemetry endpoints represents a significant improvement in data sovereignty and protection against potential behavioural profiling or surveillance.

5.3 Risks and Trade-offs

Despite its advantages, debloating carries inherent risks if performed without a clear understanding of system dependencies and critical services. Over-debloating—particularly in Windows environments—can disable or remove critical components, including system update mechanisms, security services, driver frameworks, or APIs required by dependent software. This may lead to system instability, feature loss, application incompatibilities, or security vulnerabilities over time.

Custom or modified ISOs such as Ghost Spectre and AtlasOS that are pre-debloated may exclude essential services or fail to receive official security patches, thereby

compromising long-term reliability and exposing systems to unpatched vulnerabilities. While these builds deliver impressive performance improvements (23-26 second boot times and 1.1-1.3GB RAM usage), they sacrifice system maintainability, update compatibility, and manufacturer support. Users adopting these solutions must possess technical expertise to manually manage security updates, troubleshoot compatibility issues, and restore missing dependencies when applications fail to run.

The ReviOS approach, which balances optimization with compatibility and update preservation, appears to offer the most sustainable middle ground for users seeking performance improvements without sacrificing long-term system health.

In Linux environments, while the modular structure allows for safer optimization, excessive removal of systemd services or library dependencies can still cause boot failures, application crashes, or broken package installations. The process must be guided by understanding of package interdependencies, service relationships, and system architecture. Tools like apt-cache rdepends and pacman -Qi help users understand what depends on a package before removing it, reducing the risk of breaking critical functionality.

Users must therefore balance optimization with sustainability, ensuring that critical update, security, and driver functions remain intact. Debloating should be approached as a careful, informed process rather than aggressive wholesale removal of components.

5.4 Insights from Linux Design

The results from Linux testing highlight the advantages of modular and transparent OS design. Minimal distributions such as Arch Linux, lightweight derivatives, and even mainstream distributions like Linux Mint inherently follow a "build-what-you-need" philosophy, allowing users to tailor the system footprint precisely to their requirements from installation onward.

This approach contrasts sharply with monolithic, vendor-controlled ecosystems like Windows, where the user has limited influence over bundled features and telemetry integration. The findings suggest that modularity and transparency—hallmarks of open-source design—naturally mitigate the need for post-installation debloating by preventing bloat at its root.

Arch Linux's minimal default installation (400MB RAM, 15-second boot time, zero telemetry) demonstrates what is achievable when users control every component installed on their system. Even user-friendly distributions like Linux Mint, despite including full desktop environments and productivity software, consume significantly fewer resources (1.4GB RAM vs. 2.8GB for Windows 11) and generate virtually no privacy-invasive telemetry.

Consequently, the Linux model demonstrates how user agency, configurability, and open development processes can prevent bloat at the design level rather than addressing it as an afterthought. This architectural philosophy offers valuable lessons for future operating system development regardless of platform.

5.5 Tool Selection and Methodology Validation

The effectiveness of Chris Titus Tech's Winutil in achieving substantial performance and privacy improvements without compromising system stability validates the importance of using well-maintained, community-vetted tools for debloating. Unlike fragmented scripts or experimental utilities, Winutil's comprehensive approach, extensive documentation, and active community support make it suitable for both novice and expert users.

The tool's modular design allows selective application of optimizations, reducing the risk of over-debloating. Its preservation of Windows Update functionality and security services ensures that debloated systems remain maintainable and secure over time. The quantitative results—36-39% RAM reduction, 31-32% boot time improvement, and 60-80% telemetry reduction—demonstrate that systematic, tool-assisted debloating can achieve professional-grade optimization without requiring deep technical expertise.

This study's methodology, using identical virtual machine configurations and standardized measurement procedures, successfully isolated the effects of debloating from hardware variance and environmental factors. The consistency of results across multiple test runs (variance under 5%) confirms the reliability of the findings and supports their generalizability to real-world systems.

5.6 Summary

In conclusion, the discussion reinforces that debloating is both a performance optimization strategy and a privacy enhancement technique, with tangible benefits for system efficiency and data control. The quantitative evidence demonstrates that these gains are substantial, reproducible, and statistically significant across diverse operating system platforms.

However, these benefits must be weighed against potential risks to stability and maintainability. The optimal approach involves using established tools like Winutil, understanding system dependencies, preserving critical services, and adopting custom builds only when their trade-offs are acceptable for specific use cases.

The comparative insights between Windows and Linux underline a broader lesson for operating system design: true optimization begins with modularity, transparency, and user choice—principles that minimize the very need for debloating in the first place. Future operating systems would benefit from adopting component-based installation

models that allow users to selectively include only needed features and services, ensuring optimal resource utilization, stronger privacy, and greater long-term maintainability from the outset.

6. Conclusion

The findings of this study clearly indicate that debloating has a substantial and measurable positive impact on both operating system performance and user privacy. By eliminating redundant services, bundled applications, and telemetry processes, systems become leaner, faster, and less intrusive. This effect is particularly evident in Windows environments, where vendor-level bloatware and background analytics are deeply integrated into the system architecture. The data gathered through controlled testing confirms that debloating can significantly reduce idle resource consumption by 20-40%, improve boot times by 15-35%, and minimize unsolicited network communication by 60-80% without compromising core functionality when performed carefully using established tools and methodologies.

Windows 10 Pro showed the most dramatic improvements, with 36% RAM reduction, 31% faster boot times, and 67% fewer telemetry connections after debloating. Windows 11 Pro demonstrated similar benefits with 39% RAM reduction and 32% boot time improvement. Even enterprise-focused LTSC editions, already optimized by design, benefited from selective debloating with 17% RAM reduction. Custom community builds including Ghost Spectre, AtlasOS, and ReviOS achieved even more aggressive optimization with 50-60% resource reductions, though these came with trade-offs in feature availability, update compatibility, and system complexity.

In contrast, Linux distributions—especially minimal or modular builds such as Arch Linux, Linux Mint, and lightweight derivatives—demonstrate that effective design philosophy can inherently prevent bloat. These systems exemplify how user-centric architecture, transparency, and package-level control can yield efficient and privacy-respecting operating systems without requiring extensive post-installation modification. Arch Linux's minimal configuration consumed only 400MB RAM with 15-second boot time and zero telemetry, while even full-featured Linux Mint used only 1.4GB RAM compared to Windows 11's 2.8GB baseline.

The comparative analysis underscores a fundamental difference between proprietary and open-source ecosystems: while one often requires corrective optimization after installation, the other embeds optimization as a core design principle. The modular nature of Linux systems, combined with transparent service management and user control over installed components, naturally minimizes bloat and privacy concerns from the outset.

From a broader perspective, this study highlights the need for a paradigm shift in operating system development. Rather than shipping monolithic systems filled with preinstalled software and telemetry frameworks that users must subsequently disable or remove, future operating systems could adopt a modular, component-based installation model. This would allow users to selectively include only the features and services they require during installation, ensuring optimal resource utilization, stronger privacy protection, and greater long-term maintainability. Microsoft's LTSC editions and Linux's package-based installation models demonstrate the viability of this approach.

The effectiveness of Chris Titus Tech's Winutil as a comprehensive debloating solution for Windows validates the importance of community-driven tools in empowering users to reclaim control over their computing environments. With over 41,900 GitHub stars and active development, Winutil represents the current gold standard for Windows optimization, balancing ease of use with powerful functionality. Its ability to achieve 30-40% resource reductions while preserving system stability and update functionality demonstrates that user-friendly optimization tools can deliver professional-grade results without requiring deep technical expertise.

The privacy implications of this research extend beyond mere performance metrics. The dramatic reduction in telemetry endpoints—from 12-15 connections in stock Windows to 2-4 after debloating—demonstrates that default operating system configurations prioritize vendor analytics over user privacy. The persistence of these connections even after users disable tracking through standard settings interfaces raises important questions about informed consent, data sovereignty, and user agency in modern computing. Debloating thus serves not only as a performance optimization but as an assertion of user control over personal data and system behavior.

However, this study also acknowledges the risks and limitations of debloating. Over-aggressive optimization, particularly when using poorly documented tools or custom ISOs, can compromise system stability, break update mechanisms, introduce software incompatibilities, and create security vulnerabilities. The balanced approach embodied by tools like Winutil—which preserves critical services while removing unnecessary components—represents the most sustainable path forward for most users. Custom builds like AtlasOS and Ghost Spectre offer maximum performance but require technical expertise to maintain security and compatibility over time.

Ultimately, debloating is not merely a performance tweak—it represents a philosophy of digital efficiency and user empowerment. By giving users control over what runs on their systems, we not only enhance performance and privacy but also reaffirm the principle that computing

environments should serve their users, not the other way around. The quantitative evidence presented in this study provides empirical validation for what many users have long suspected: that modern operating systems can and should be leaner, faster, and more respectful of user privacy.

As operating systems continue to evolve, the lessons from this research suggest that transparency, modularity, and user choice must be prioritized alongside new features and cloud integration. The success of minimal Linux distributions and the effectiveness of community debloating tools demonstrate strong user demand for efficient, privacy-respecting computing. Operating system developers would do well to heed this demand, designing systems that optimize resource usage and respect user privacy from the ground up rather than requiring corrective intervention after installation.

The path forward lies not in accepting bloat as inevitable, but in demanding better defaults, more transparent telemetry practices, and modular system designs that empower users to create computing environments tailored to their needs. This research provides the quantitative foundation for that demand, demonstrating conclusively that leaner, more efficient operating systems are not only possible but measurably superior in performance, privacy, and user experience.

7. References

1. Y. Acar, S. Fahl, and M. L. Mazurek, "You are not your developer, privacy and security concerns of end users and developers," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, 2016, pp. 1-17.
2. J. Deng, W. Zhang, and H. Wang, "Performance impact of background services in modern operating systems," *J. Comput. Syst. Appl.*, vol. 45, no. 3, pp. 211-228, 2021.
3. P. Khandelwal and A. Singh, "Evaluating resource efficiency in Linux distributions: A comparative study," *Int. J. Open-Source Syst. Technol.*, vol. 12, no. 2, pp. 34-42, 2019.
4. Microsoft Corporation, "Windows 10 and 11 diagnostic data and privacy FAQ," Microsoft Learn Documentation, 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/privacy/windows-diagnostic-data
5. Microsoft Corporation, "Windows LTSC overview and servicing model," Microsoft Docs, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/whats-new/ltsc
6. Canonical Ltd., "Ubuntu privacy policy and telemetry data collection," Ubuntu Official Documentation, 2024. [Online]. Available: https://ubuntu.com/legal/data-privacy
7. Arch Linux Wiki, "Improving performance and disabling unnecessary services," 2025. [Online]. Available: https://wiki.archlinux.org/title/improving_performance
8. Chris Titus Tech, "Windows utility - debloating and optimization tool," GitHub Repository, 2025. [Online]. Available: https://github.com/ChrisTitusTech/winutil
9. AtlasOS Team, "AtlasOS - optimized Windows for enthusiasts," 2025. [Online]. Available: https://atlasos.net
10. ReviOS Team, "ReviOS - revision of Windows for gaming and performance," 2025. [Online]. Available: https://revi.cc
11. Ghost Spectre Project, "Ghost Spectre Windows custom builds," Community Resources and Documentation, 2025.
12. R. Klein and S. Müller, "User control and privacy in modern software ecosystems," *ACM Trans. Privacy Security*, vol. 25, no. 4, pp. 1-24, 2022.
13. Linux Mint Team, "Linux Mint documentation - performance and resource optimization," 2025. [Online]. Available: https://www.linuxliteos.com/manual/
14. Red Hat, Inc., "Systemd and service management performance guide," 2024. [Online]. Available: https://access.redhat.com/documentation/en-us/
15. Wireshark Foundation, "Wireshark user's guide: network monitoring and analysis," 2025. [Online]. Available: https://www.wireshark.org/docs/
16. XDA Developers, "Chris Titus Windows utility review - best way to debloat Windows 11," 2025. [Online]. Available: https://www.xda-developers.com/chris-titus-windows-utility-best-way-debloat-customize-windows/
17. BetaNews, "Speed up your system: The best custom Windows 10/11 builds for performance," 2024. [Online]. Available: https://betanews.com/2024/08/29/speed-up-your-system-the-best-custom-windows-10-11-builds-for-performance/
18. Aardwolf Security, "Windows 11 privacy settings: Complete setup guide," 2025. [Online]. Available: https://aardwolfsecurity.com/how-to-set-up-windows-11-for-maximum-privacy/
19. Microsoft Learn, "Manage connections from Windows 10 and Windows 11 operating system components to Microsoft services," 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/privacy/manage-connections-from-windows-operating-system-components-to-microsoft-services
20. TweakTown, "You can debloat Windows 11 in just 2 clicks with this new intuitive tool," 2025. [Online]. Available: https://www.tweaktown.com/news/103249/you-can-debloat-windows-11-in-just-2-clicks-with-this-new-intuitive-tool/