Attack Surfaces in Operating Systems

Vajradeva

***Abstract*—This paper provides a comprehensive technical analysis of the attack surfaces inherent to modern operating systems (OS). It deconstructs this surface from foundational definitions and formal models to a deep examination of core components, including the kernel, user interfaces, network stack, and file system. The analysis is extended to the expanded boundaries introduced by third-party software, virtualization, containerization, and cloud integration, which introduce new layers of abstraction and leveraged risk. A detailed comparative analysis of the security architectures of Windows, Linux, and macOS reveals distinct philosophical approaches to mitigation, examining specific technologies like Virtualization-Based Security (VBS), SELinux, and System Integrity Protection (SIP). The paper concludes by synthesizing these findings into a framework for strategic mitigation, grounded in the principles of least privilege and attack surface minimization, and supported by a suite of technical hardening controls. Future trends, such as the shift to memory-safe languages, hardware-enforced security, and the impact of artificial intelligence, are also discussed.**

***Index Terms*—Operating Systems, Cybersecurity, Attack Surface Management (ASM), System Hardening, Kernel Security, Virtualization Security, Container Security, Cloud Security, Mandatory Access Control (MAC), Principle of Least Privilege (PoLP).**

I. A Formal Model of the Operating System Attack Surface

The operating system (OS) serves as the foundational layer upon which all applications and user interactions depend. It manages hardware resources, provides essential services, and enforces security boundaries. Consequently, the security of the entire computing stack is predicated on the integrity of the OS. Understanding and mitigating the attack surfaces inherent to an operating system is, therefore, a paramount concern in modern cybersecurity. This document provides a comprehensive technical analysis of these attack surfaces, from the core components of the kernel to the expanded boundaries introduced by virtualization, containerization, and cloud computing. It further details the platform-specific mitigation strategies and overarching principles of system hardening required to build resilient, defensible systems.

A. Defining the Attack Surface: A Synthesis of NIST and Academic Frameworks

To conduct a rigorous analysis, a precise definition of the "attack surface" is essential. The National Institute of Standards and Technology (NIST) provides an authoritative baseline, defining the attack surface as, "The set of points on the boundary of a system, a system element, or an environment where an attacker can try to enter, cause an effect on, or extract data from, that system, system element, or environment" [1]. This definition establishes the attack surface as the complete interface through which a system interacts with its environment. The Open Worldwide Application Security Project (OWASP) offers a complementary perspective, describing it as "all of the different points where an attacker could get into a system, and where they could get data out," [2] highlighting the bidirectional nature of threats.

While conceptually sound, a more granular model is required for technical analysis. Pioneering work by Manadhata et al. provides such a model, deconstructing the attack surface into a function of three primary elements [3]:

1) **Entry and Exit Points (Methods):** These are the interfaces through which data and control flow into and out of the system. This includes system calls, application programming interfaces (APIs), and any function that processes external input. For example, the `read()` system call is an entry point that takes a file descriptor, a buffer, and a size as input. A vulnerability in the kernel's handling of this call could lead to an information leak.
2) **Channels:** These are the communication pathways used by the system, such as network sockets, pipes, and inter-process communication (IPC) mechanisms. A TCP socket listening on port 443 is a channel.
3) **Untrusted Data Items:** This encompasses all data that originates from outside the system's trust boundary. This includes files, registry keys (on Windows), configuration settings, and data stored in databases. An XML file parsed by a system service is an untrusted data item.

This formal model, represented as a function of the sets of methods, channels, and untrusted data items (M, C, and I), allows for a more structured assessment [4]. A grounded theory (GT) study by Moshtari, analyzing 1,444 CVEs, categorized attack surface components based on how real-world vulnerabilities manifest [5]–[7]. This resulted in a practical framework comprising three core categories: Entry Points (e.g., APIs, UI), Targets (e.g., memory, configuration files), and Mechanisms (e.g., lack of input validation, improper error handling). This shift from a conceptual boundary to a measurable set of components precipitated the rise of Attack Surface Management (ASM) as a formal discipline [8], [9].

B. The Anatomy of an Attack

A clear distinction between attack surface, vector, and vulnerability is critical.

- **Attack Surface:** The sum total of all possible points of entry and exposure [10]. For a web server, this includes the OS it runs on, the web server software (e.g., Apache), open ports (80, 443), and all application code.
- **Attack Vector:** The specific path or method an adversary uses. For the Heartbleed bug, the attack vector was a

malicious heartbeat request sent over a TLS connection [15].
- **Vulnerability:** The underlying weakness. In Heartbleed (CVE-2014-0160), the vulnerability was a missing bounds check in the OpenSSL heartbeat extension implementation, allowing an attacker to read up to 64KB of memory from the server [16].

An attacker uses a vector to exploit a vulnerability on the attack surface.

C. Conceptual Categorization

The total attack surface can be divided into three domains:
- **Digital Attack Surface:** All software and network components. This is the realm of code, protocols, and configurations [13].
- **Physical Attack Surface:** All hardware an attacker can physically access. Vectors include theft of devices, "baiting" with malicious USB drives, and cold boot attacks where an attacker reboots a machine into a controlled environment to extract cryptographic keys from RAM before it degrades [10]–[12].
- **Social Engineering Attack Surface:** This targets the human element. Vectors include phishing, pretexting (creating a fabricated scenario to obtain information), and exploiting insider threats [11], [12].

A holistic security strategy must address all three domains.

II. The Core Digital Attack Surface: Kernel and User Space

The digital attack surface of a modern OS is vast. At its heart lie the kernel and the primary user-mode interfaces. A compromise at this level can lead to a complete loss of system integrity.

A. The Kernel as the Ultimate Target

The OS kernel is the central, most privileged component, operating in the hardware's most protected context, "Ring 0," with unrestricted access to all hardware and memory [20]. In contrast, user applications run in the less-privileged "Ring 3." A transition from Ring 3 to Ring 0 via a system call is a critical security boundary crossing.

The monolithic architecture of Windows and Linux, while performant, creates a single, large, implicitly trusted execution domain. A vulnerability in one minor component, such as a third-party device driver, can be leveraged to compromise the entire system. This is a key difference from microkernel architectures, which aim to minimize the code running in Ring 0, though often at a performance cost [21].

Key Kernel Attack Surface Components:
- **System Call Interface:** The boundary between user space and the kernel. Failure to validate pointers or sizes from hundreds of system calls can lead to critical vulnerabilities, such as allowing a user-mode process to read or write to arbitrary kernel memory [24].
- **Device Drivers:** Often developed by third parties, these execute with full kernel privileges and are a notorious source of vulnerabilities due to complex, asynchronous hardware interactions and potentially lower security standards [22].
- **Memory Management Subsystems:** Complex kernel memory allocators (e.g., Linux's SLUB, Windows' Pool) are a fertile ground for sophisticated heap-based exploits like Use-After-Free (UAF) or Out-of-Bounds (OOB) writes [26]. A UAF occurs when the system deallocates a memory structure but a pointer to it (a "dangling pointer") remains and is later used, potentially after the memory has been reallocated for a different, attacker-controlled object.

Common Kernel Exploitation Vectors include memory corruption, race conditions, and logical flaws, all aimed at achieving privilege escalation [16].

B. User-Mode Interfaces: CLI and GUI

- **Command-Line Interface (CLI):** Adversaries use built-in CLIs (PowerShell, `wmic`, `netsh`, bash) to "live off the land," avoiding detection [29]. The primary vector is OS Command Injection [31].
- **Graphical User Interface (GUI):** The GUI stack (e.g., X11, Wayland, GDI) is an attack surface [32]. Vulnerabilities can range from code execution via a malicious font file to attacks that abuse accessibility APIs to log keystrokes or inject input [34].

C. The Network Stack: Protocols, Ports, and Services

For a remote attacker, the OS network stack is the primary attack surface. Vulnerabilities in the implementation of protocols like SMB (e.g., MS17-010, exploited by WannaCry) or RPC (e.g., MS08-067, exploited by Conficker) have led to widespread compromises. These are often buffer overflows in code written in C/C++ [9], [12], [37].

D. The File System: Permissions and Access Controls

The file system's attack surface is defined by its directory hierarchy and permissions model. Key attack vectors include:
- **Insecure Permissions (CWE-732):** World-writable directories like `/tmp` can allow an unprivileged user to place malicious code where a privileged process might execute it [39], [40].
- **Link-Based Attacks (TOCTOU):** Abusing symbolic links in a Time-of-Check-to-Time-of-Use race condition can trick a privileged program into operating on a sensitive file. An attacker can swap a benign file with a symlink to a sensitive file between the time the system checks permissions and the time it uses the file.
- **Alternate Data Streams (ADS):** A feature of NTFS on Windows, ADS can be used to hide malicious code behind legitimate files, evading some detection tools.

III. The Expanded Attack Surface: Modern Architectures

Modern IT environments have expanded the OS attack surface. Virtualization, containerization, and cloud integration introduce new layers of abstraction and "leveraged risk," where a single vulnerability in an underlying layer can compromise dozens or hundreds of workloads.

A. Third-Party Applications and Supply Chain Risk

Software supply chain attacks are an insidious threat. Examples include the 2020 SolarWinds attack and the 2018 compromise of the `event-stream` npm package, where malicious code was added to a popular library to steal cryptocurrency wallet credentials [46]. A key mitigation is the adoption of a Software Bill of Materials (SBOM), an inventory of all components in a piece of software.

B. Virtualization Security: Hypervisor and VM Escape

Virtualization introduces the hypervisor (Type-1/bare-metal like VMware ESXi, or Type-2/hosted like VirtualBox). A compromise of the hypervisor is catastrophic [48], [50]. The most critical threat is **VM escape**, where an attacker in a guest OS exploits a hypervisor vulnerability to execute code on the host [51]. A famous example is VENOM (CVE-2015-3456), a buffer overflow in the virtual floppy disk controller code used by QEMU, which affected numerous virtualization platforms.

C. Containerization Security and Breakout Vectors

Containers share the host OS kernel, a critical security trade-off [59]. Isolation is provided by software constructs like kernel namespaces (pid, net, mnt, user, etc.) and cgroups. A host kernel vulnerability can be exploited from one container to compromise all others. Key attack vectors, as defined in NIST SP 800-190 [60], include:

- **Insecure Container Images:** Using images from public repositories with known vulnerabilities [64].
- **Runtime Misconfigurations:** Running containers with the `--privileged` flag, which disables most isolation, or mounting the Docker socket (`/var/run/docker.sock`), which allows the container to control the Docker daemon and escape its confinement.
- **Container Breakout:** Exploiting a vulnerability in the runtime or host kernel. CVE-2019-5736 ("runcescape") was a critical vulnerability that allowed a malicious container to overwrite the host's runc binary, leading to root access on the host.

D. Cloud Integration and the Dissolving Perimeter

Integrating with public cloud services creates a hybrid attack surface where the traditional network perimeter dissolves [69]. Key challenges include navigating the shared responsibility model and securing cloud-specific vectors like IAM [71], [72]. A major threat is the abuse of cloud metadata services. In the 2019 Capital One breach, an attacker used a Server-Side Request Forgery (SSRF) vulnerability to query the EC2 metadata service from a compromised web server, obtaining temporary credentials that were then used to access and exfiltrate data from S3 buckets.

IV. Comparative Analysis of OS Security Architectures

Windows, Linux, and macOS have evolved distinct security architectures. Windows prioritizes centralized, enterprise-grade control; Linux prioritizes flexibility; and macOS prioritizes usability and out-of-the-box security [22].

A. Microsoft Windows: VBS and WDAC

The modern Windows security architecture leverages hardware virtualization to create a zero-trust execution environment.

- **Windows Defender Application Control (WDAC):** A strict application allowlisting technology operating on a "default deny" principle [75], [76].
- **Virtualization-Based Security (VBS):** Uses the Hyper-V hypervisor to create an isolated environment, Virtual Secure Mode (VSM). VBS places critical security processes inside this protected environment. Even if the main kernel is compromised, an attacker cannot access the VSM. This protection is dependent on hardware features like an IOMMU (Input-Output Memory Management Unit) to prevent DMA attacks. Key VBS features include:
  - **Hypervisor-Enforced Code Integrity (HVCI):** Runs the kernel-mode code integrity engine inside VSM, preventing a compromised kernel from disabling WDAC or loading a malicious driver.
  - **Credential Guard:** Isolates and protects secrets managed by the Local Security Authority Subsystem Service (LSASS), such as NTLM hashes and Kerberos tickets, inside VSM, preventing pass-the-hash attacks.

B. Linux: SELinux and AppArmor

The Linux Security Modules (LSM) framework allows for pluggable Mandatory Access Control (MAC) systems.

- **SELinux (Security-Enhanced Linux):** A powerful but complex label-based MAC system. Every process and object is assigned a security context (e.g., `user:role:type:level`). The policy consists of explicit rules defining allowed interactions between types [23], [80].
- **AppArmor:** A more user-friendly, path-based MAC system. Profiles are associated with executables and define allowed file access and capabilities. A simple profile for `/usr/bin/foo` might contain: `/usr/bin/foo` { `r, /home/user/data r, /tmp/** w,` }. It is less granular than SELinux but more accessible [80], [82].
- **seccomp-bpf:** Another LSM that can restrict the system calls a process is allowed to make, reducing the kernel surface exposed to a potentially compromised application.

C. Apple macOS: SIP, Gatekeeper, and XProtect

Apple's "walled garden" model leverages tight hardware/software integration [83].

- **System Integrity Protection (SIP):** A kernel-level technology that protects critical system locations (`/System`,

`/bin`, etc.) from modification, even by root. It uses extended filesystem attributes to mark protected files [87], [89].
- **Gatekeeper:** Enforces code signing and notarization. Notarization is an automated Apple service that scans software for malicious components, providing a higher level of assurance than simple developer ID signing [90].
- **XProtect:** A built-in, signature-based anti-malware tool [84].

V. Strategic Mitigation and System Hardening

A comprehensive defense requires a proactive, multi-layered strategy focused on systematically reducing exposure and enforcing a ”default deny” posture.

A. Foundational Strategies: PoLP and Attack Surface Minimization

- **The Principle of Least Privilege (PoLP):** Grant any user, process, or component only the absolute minimum permissions necessary for its function [91]. Implementation involves dedicated service accounts (e.g., a `www-data` user for a web server that cannot access user home directories), role-based access control (RBAC), and avoiding persistent high-privilege accounts [93].
- **Attack Surface Minimization:** The practical application of PoLP. Reduce entry points by removing all non-essential software, services, protocols, and accounts [94]. Start with a minimal OS installation (e.g., Debian netinstall, Windows Server Core) and add only necessary components [96].

B. Proactive Defense: Attack Surface Management (ASM)

ASM is the continuous process of identifying, analyzing, and mitigating risks. The lifecycle consists of three phases:

1) **Discovery:** Continuously discover all assets using tools for External Attack Surface Management (EASM), vulnerability scanners, and asset inventory systems to find ”shadow IT” and ”orphaned IT” [12].
2) **Analysis and Prioritization:** Analyze assets for vulnerabilities, prioritizing remediation based on risk, exploitability (e.g., using the Exploit Prediction Scoring System - EPSS), and business impact [11].
3) **Remediation and Monitoring:** Apply patches and hardening controls, with continuous monitoring for new risks.

C. Technical Hardening Controls (Cross-Platform)

1) Kernel and Memory Protections:

- **Address Space Layout Randomization (ASLR):** Randomizes memory locations of key data areas to foil exploits that rely on predictable addresses [25].
- **Data Execution Prevention (DEP) / W^X:** Marks memory regions as non-executable, preventing code injection attacks [25].
- **Supervisor Mode Execution/Access Prevention (SMEP/SMAP):** Hardware features that prevent the kernel (supervisor mode) from executing code in or accessing data from user-space pages, directly mitigating ”ret2usr” attacks where a kernel exploit attempts to redirect execution to shellcode in user memory [26].

2) Network and File System Hardening: Configure host-based firewalls with a ”default deny” policy [94]. Regularly audit file system permissions and use tools like `auditd` (Linux) to log access to critical files [99].

3) Application Whitelisting and Control Flow Integrity: Deploy application whitelisting technologies like WDAC, AppArmor, or SELinux [75]. Additionally, Control Flow Integrity (CFI) is an emerging mitigation that prevents attackers from hijacking a program's execution flow by ensuring that indirect branches and calls only go to legitimate locations. Hardware implementations like Intel's Control-flow Enforcement Technology (CET) provide more robust protection for both forward-edge (indirect calls/jumps) and backward-edge (returns) control transfers [37].

D. Advanced Mitigation in Virtualized Environments

The hypervisor must be treated as the most critical security component. Adhere to hardening guides like the CIS Benchmarks. Minimize its attack surface by disabling non-essential features and patch it aggressively [50].

For containers, align with NIST SP 800-190 guidance:

- **Image Security:** Use minimal base images (e.g., distroless) from trusted registries and scan images for vulnerabilities in the CI/CD pipeline [64].
- **Runtime Security:** Run containers as a non-root user (using the `USER` directive in the Dockerfile), drop unnecessary Linux capabilities, and run with a read-only root filesystem (`--read-only`) where possible [59].
- **Network Security:** Create custom networks to segment containers and use network policies to enforce a ”default deny” posture for inter-container communication [67].

VI. Conclusion and Future Outlook

Securing an OS is a continuous endeavor requiring deep architectural understanding and a proactive, multi-layered mitigation strategy. The defensibility of an OS is a function of its architecture, configuration, and the expertise of its administrators. Looking forward, the landscape will be shaped by several key trends:

- **Shift to Memory-Safe Languages:** A significant industry effort to adopt languages like Rust for kernel development to eliminate memory corruption vulnerabilities by design. Both Google (in Android) and Microsoft (in Windows) are actively integrating Rust into their OS components [101].
- **Hardware-Enforced Security:** New hardware features like Intel CET and ARM's Pointer Authentication Codes (PAC) will provide more robust defenses against control-flow hijacking.
- **Dual Impact of AI:** AI will be used by both attackers (to create evasive malware and find bugs via fuzzing) and defenders (for AI-driven behavioral analysis and anomaly

TABLE I
COMPARATIVE ANALYSIS OF OS MANDATORY ACCESS CONTROL SYSTEMS

Control System	Mechanism	Policy Granularity	Management Complexity	Primary Use Case	Bypass/Evasion Difficulty
WDAC (Windows)	Application Allowlisting (Whitelisting)	High (File Hash, Publisher, Path)	Medium to High (Requires careful policy creation and maintenance)	Enterprise Endpoint Lockdown, Server Hardening	High (Protected by VBS, requires signed policy to prevent admin bypass)
SELinux (Linux)	Label-based MAC	Very High (Controls processes, files, ports, syscalls)	Very High (Requires deep expertise and complex policy authoring)	Hardened Servers, High-Security Environments (e.g., Military, Finance)	High (Requires kernel exploit or severe policy misconfiguration)
AppArmor (Linux)	Path-based MAC	Medium (Controls file access and capabilities per application)	Low to Medium (Profiles are human-readable and can be auto-generated)	Application-Specific Confinement on Servers and Desktops	Medium (Relies on correct path definitions; can be bypassed by some complex attacks)
SIP (macOS)	Filesystem/Kernel Restriction	Low (System-wide, protects predefined critical paths)	Low (Enabled by default, not user-configurable)	Consumer/Prosumer Device Protection, Core OS Integrity	Very High (Requires physical access to boot into recovery mode to disable)

TABLE II
OS COMPONENT ATTACK SURFACE MATRIX

Core OS Component	Common CWEs	Relevant MITRE ATT&CK Techniques	Primary Mitigation Strategy	Key Hardening Tool/Technology
Kernel	CWE-121 (Stack Overflow), CWE-122 (Heap Overflow), CWE-416 (Use-After-Free)	T1068: Exploitation for Privilege Escalation	Memory Safety, Hardware-Enforced Isolation	KASLR, DEP, SMEP/SMAP, VBS (Windows)
Network Stack	CWE-120 (Buffer Copy without Checking Size), CWE-287 (Improper Authentication)	T1190: Exploit Public-Facing Application	Default-Deny Firewall, Service Minimization	iptables/nftables (Linux), Windows Defender Firewall
File System	CWE-732 (Incorrect Permission Assignment), CWE-276 (Incorrect Default Permissions)	T1222: File and Directory Permissions Modification	Principle of Least Privilege (PoLP), Auditing	chmod/icacls, auditd (Linux), Access Control Lists (ACLs)
User Interfaces (CLI/GUI)	CWE-78 (OS Command Injection)	T1059: Command and Scripting Interpreter	Input Validation, Application Allowlisting	WDAC, AppArmor, SELinux
Third-Party Software	CWE-1104 (Use of Unmaintained Third Party Components)	T1195: Supply Chain Compromise	Supply Chain Security, Vulnerability Management	Software Composition Analysis (SCA), Container Scanning
Hypervisor/ Container Runtime	CWE-284 (Improper Access Control)	T1610: Deploy Container, T1611: Escape to Host	Runtime Security, Configuration Hardening	SELinux/AppArmor Profiles, Seccomp, Hypervisor Hardening Guides

detection to identify threats that bypass traditional signatures).

- **Proliferation of Embedded OSs:** The growth of IoT and ICS devices introduces a massive new attack surface of specialized OSs (RTOS) that often lack mature security features, robust update mechanisms, and community scrutiny [16].

The fortress of the operating system can be defended, but only through constant vigilance, rigorous engineering, and a security-first mindset at every layer.

REFERENCES

[1] "Attack surface," *Glossary — CSRC*, National Institute of Standards and Technology. Available: https://csrc.nist.gov/glossary/term/attack_surface.

[2] "Attack Surface," *Glossary*, Ivanti. Available: https://www.ivanti.com/glossary/attack-surface.

[3] A. A. Al-Dhaqm, "Attack Surface Score for Software Systems," *MDPI*, vol. 17, no. 7, p. 305, 2025.

[4] J. Li, "A Network Attack Surface Evaluation Method Based on Optimal Attack Strategy," *MDPI*, vol. 14, no. 2, p. 274, 2025.

[5] S. Moshtari, "Characterizing and Detecting Software Attack Surface Components," M.S. thesis, Dept. Comput. Sci., Rochester Institute of Technology, Rochester, NY, USA, 2025.

[6] S. Moshtari, "Characterizing and Detecting Software Attack Surface

Components," *RIT Digital Institutional Repository*. Available: https://repository.rit.edu/cgi/viewcontent.cgi?article=12804&context=theses.
[7] S. Moshtari, "A Grounded Theory Based Approach to Characterize Software Attack Surfaces," *arXiv*, 2112.01635, 2025.
[8] "Guide to Attack Surface Management," Redjack. Available: https://redjack.com/resources/quide-to-attack-surface-management.
[9] "What is attack surface management and why is it important?," Outpost24. Available: https://outpost24.com/blog/what-is-attack-surface-management-asm/.
[10] "What Is an Attack Surface?," Picus Security. Available: https://www.picussecurity.com/resource/glossary/what-is-attack-surface.
[11] "What Is Attack Surface Management?," Palo Alto Networks. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-attack-surface-management.
[12] "What is an Attack Surface?," IBM. Available: https://www.ibm.com/think/topics/attack-surface.
[13] "Attack Surface," *Glossary*, Beyond Identity. Available: https://www.beyondidentity.com/glossary/attack-surface.
[14] "What Is an Attack Surface? Types, Components & Best Practices," IONIX. Available: https://www.ionix.io/guides/what-is-attack-surface-management/attack-surface/.
[15] "What is an attack surface?," Cloudflare. Available: https://www.cloudflare.com/learning/security/what-is-an-attack-surface/.
[16] "Operating System Vulnerabilities: Understanding and Mitigating the Risk," Sternum IoT. Available: https://sternumiot.com/iot-blog/operating-system-vulnerabilities-understanding-and-mitigating-the-risk/.
[17] "Defining Your Organization's Attack Surface: The 4 Types of Attack Surfaces," Nightfall AI. Available: https://www.nightfall.ai/blog/4-types-of-attack-surfaces.
[18] "ICS Techniques," MITRE ATT&CK®. Available: https://attack.mitre.org/techniques/ics/.
[19] "What is an Attack Surface in Cybersecurity?," Rapid7. Available: https://www.rapid7.com/fundamentals/attack-surface/.
[20] "Kernel level attack?," *Information Security Stack Exchange*. Available: https://security.stackexchange.com/questions/273216/kernel-level-attack.
[21] A. Kurmus, "Quantifiable Run-time Kernel Attack Surface Reduction," in *Proc. DIMVA*, 2014.
[22] "Is Linux actually less secure than Windows/MacOS?," *Reddit*. Available: https://www.reddit.com/r/linuxquestions/comments/1bh582b/is_linux_actually_less_secure_than_windowsmacos/.
[23] "What is SELinux?," Red Hat. Available: https://www.redhat.com/en/topics/linux/what-is-selinux.
[24] "Research of The Linux Operating System — Part 3: Attack Surfaces," VerSprite. Available: https://versprite.com/vs-labs/part-3-comprehensive-research-of-linux-vulnerabilities-attack-surfaces/.
[25] "Protecting the Core Kernel Exploitation Mitigations," Census Labs. Available: https://www.census-labs.com/media/bheu-2011-wp.pdf.
[26] "Exploiting the Linux Kernel," OffensiveCon. Available: https://www.offensivecon.org/trainings/2025/exploiting-the-linux-kernel.html.
[27] "Exploiting the Linux Kernel // Andrey Konovalov," Ringzero. Available: https://ringzer0.training/bootstrap25-exploiting-the-linux-kernel/.
[28] "The Cybersecurity Professional's Guide to Kernel Exploits," AppSecEngineer. Available: https://www.appsecengineer.com/blog/the-cybersecurity-professionals-guide-to-kernel-exploits.
[29] "Command-Line Interface, Technique T0807 - ICS," MITRE ATT&CK®. Available: https://attack.mitre.org/techniques/T0807/.
[30] "MITRE ATT&CK T1059 Command Line Interface," Picus Security. Available: https://www.picussecurity.com/resource/blog/picus-10-critical-mitre-attck-techniques-t1059-command-line-interface.
[31] "OWASP Desktop App Security Top 10," OWASP. Available: https://owasp.org/www-project-desktop-app-security-top-10/.
[32] "Command Line Interface (CLI) vs. Graphical User Interface (GUI)," Cybrary. Available: https://www.cybrary.it/blog/command-line-interface-cli-vs-graphical-user-interface-gui.
[33] "API, CLI, GUI, oh my! Understanding User Behavior Across Surfaces," UX Booth. Available: https://uxbooth.com/articles/api-cli-gui-oh-my-understanding-user-behavior-across-surfaces/.
[34] M. Liss, "Security of User Interfaces: Attacks and Countermeasures," *Research Collection ETH Zurich*, 2025.
[35] "What is a Buffer Overflow — Attack Types and Prevention Methods," Imperva. Available: https://www.imperva.com/learn/application-security/buffer-overflow/.
[36] "Buffer Overflow," OWASP Foundation. Available: https://owasp.org/www-community/vulnerabilities/Buffer_Overflow.
[37] "What is a Buffer Overflow?," Portnox. Available: https://www.portnox.com/cybersecurity-101/what-is-a-buffer-overflow/.
[38] "Buffer Overflow Attack: Prevention and Detection," Indusface. Available: https://www.indusface.com/learning/what-is-buffer-overflow-attack/.
[39] "Insecure File Permissions," *Amazon Q, Detector Library*. Available: https://docs.aws.amazon.com/codeguru/detector-library/go/insecure-file-permissions/.
[40] "CWE-732: Incorrect Permission Assignment for Critical...," CWE. Available: https://cwe.mitre.org/data/definitions/732.html.
[41] "Insecure Windows Service Permissions," Tenable®. Available: https://www.tenable.com/plugins/nessus/65057.
[42] "What Are Excessive Permissions?," Zluri. Available: https://www.zluri.com/blog/excessive-permissions.
[43] "What Is an Attack Surface? Key Components and How to Manage," Wiz. Available: https://www.wiz.io/academy/attack-surface.
[44] "What is Third-Party Software Security and Breach Examples," Veracode. Available: https://www.veracode.com/security/what-is-third-party-software-security/.
[45] "What is a supply chain attack?," Cloudflare. Available: https://www.cloudflare.com/learning/security/what-is-a-supply-chain-attack/.
[46] "Supply Chain Attacks: 7 Examples and 4 Defensive Strategies," BlueVoyant. Available: https://www.bluevoyant.com/knowledge-center/supply-chain-attacks-7-examples-and-4-defensive-strategies.
[47] "The Hidden Danger in Your Software: Understanding Supply Chain Attacks," RSA Conference. Available: https://www.rsaconference.com/library/blog/the-hidden-danger-in-your-software-understanding-supply-chain-attacks.
[48] "Hypervisor Security Vulnerabilities and Precautions," Hostragons. Available: https://www.hostragons.com/en/blog/hypervisor-security-vulnerabilities-and-precautions/.
[49] "What Is Hypervisor Network Security?," ITU Online IT Training. Available: https://www.ituonline.com/tech-definitions/what-is-hypervisor-network-security/.
[50] "Compute and hypervisor security," Mirantis Documentation. Available: https://docs.mirantis.com/mcp/q4-18/mcp-security-best-practices/openstack/compute-and-hypervisor-security.html.
[51] "Virtualization Under Siege: A Deep Dive into VMware's Hypervisor Security Nightmare," CyberSRCC. Available: https://cybersrcc.com/2025/03/11/virtualization-under-siege-a-deep-dive-into-vmwares-hypervisor-security-nightmare/.
[52] "Understanding VM Escape: A Threat to Virtualized Environments," Blue Goat Cyber. Available: https://bluegoatcyber.com/blog/understanding-vm-escape-a-threat-to-virtualized-environments/.
[53] "Understanding VM Escape: Risks and Precautions," Spyboy blog. Available: https://spyboy.blog/2024/09/17/understanding-vm-escape-risks-and-precautions.
[54] "Hardening the virtualization layers," Security Guide documentation. Available: https://docs.openstack.org/security-guide/compute/hardening-the-virtualization-layers.html.
[55] "Multiple zero-days in VMware products actively exploited," Field Effect. Available: https://fieldeffect.com/blog/zero-days-vmware-actively-exploited.
[56] "Breaking the Virtual Barrier: From Web-Shell to Ransomware," Sygnia. Available: https://www.sygnia.co/threat-reports-and-advisories/breaking-the-virtual-barrier-web-shell-to-ransomware/.
[57] "Update your VMware ESXi products now," Kaspersky official blog. Available: https://www.kaspersky.com/blog/vmware-critical-vulnerabilities-cve-2025-22224/53167/.
[58] "CVE-2025-22224 VMware TOCTOU VM Escape Vulnerability," Fidelis Security. Available: https://fidelissecurity.com/vulnerabilities/cve-2025-22224/.
[59] "Docker Security," OWASP Cheat Sheet Series. Available: https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html.
[60] "Use NIST SP 800-190 policy constraints," Google Cloud. Available: https://cloud.google.com/kubernetes-engine/enterprise/policy-controller/docs/how-to/using-nist-sp-800-190.
[61] "Application Container Security Guide," NIST. Available: https://www.nist.gov/publications/application-container-security-guide.
[62] "Using Minimus to Achieve NIST SP 800-190 Container Security Compliance," Minimus. Available: https://www.minimus.io/post/using-minimus-to-align-with-nist-sp-800-190.
[63] "NIST SP 800-190 Application Container Security," Sysdig. Available: https://sysdig.com/blog/nist-sp-800-190-compliance-assurance-with-sysdig-secure/.

[64] ”8 Container Security Best Practices,” Wiz. Available: https://www.wiz.io/academy/container-security-best-practices.

[65] ”A Complete Guide: Docker Security Best Practices,” Sonatype. Available: https://www.sonatype.com/resources/guides/docker-security-best-practices.

[66] ”Docker Security: 5 Risks and 5 Best Practices for Securing Your Containers,” Tigera. Available: https://www.tigera.io/learn/guides/container-security-best-practices/docker-security/.

[67] ”Container Security-Part II- Kubernetes -OWASP Top 10,” System Weakness. Available: https://systemweakness.com/container-security-part-ii-kubernetes-owasp-top-10-99a340694437.

[68] ”OWASP Cloud-Native Application Security Top 10,” OWASP. Available: https://owasp.org/www-project-cloud-native-application-security-top-10/.

[69] ”Top 7 Cloud Security Challenges and How to Overcome Them,” Spot.io. Available: https://spot.io/resources/cloud-security/top-7-cloud-security-challenges-and-how-to-overcome-them/.

[70] ”12 Cloud Security Issues: Risks, Threats & Challenges,” CrowdStrike. Available: https://www.crowdstrike.com/en-us/cybersecurity-101/cloud-security/cloud-security-risks/.

[71] ”Securing Core Cloud Identity Infrastructure,” CISA. Available: https://www.cisa.gov/news-events/news/securing-core-cloud-identity-infrastructure-addressing-advanced-threats-through-public-private.

[72] ”Cloud Security Threats: Detection and Challenges,” Palo Alto Networks. Available: https://www.paloaltonetworks.co.uk/cyberpedia/cloud-security-threats-detection-and-challenges.

[73] ”Windows Vs macOS Vs Linux: Best OS For Cybersecurity,” Analytics India Magazine. Available: https://analyticsindiamag.com/ai-trends/windows-vs-macos-vs-linux-for-cybersecurity/.

[74] ”Which is the Most Secure Operating System?,” SentinelOne. Available: https://www.sentinelone.com/blog/which-is-more-secure-windows-linux-or-macos/.

[75] ”What is Windows Defender Application Control (WDAC)?,” NinjaOne. Available: https://www.ninjaone.com/blog/understanding-windows-defender-application-control-wdac/.

[76] ”What is Windows Defender Application Control (WDAC)?,” Scalefusion Blog. Available: https://blog.scalefusion.com/windows-defender-application-control/.

[77] ”9 Benefits of Windows Defender Application Control,” Trio MDM. Available: https://www.trio.so/blog/windows-defender-application-control/.

[78] ”Manage Windows Defender Application Control,” Microsoft Learn. Available: https://learn.microsoft.com/en-us/intune/configmgr/protect/deploy-use/use-device-guard-with-configuration-manager.

[79] ”Manage Windows Defender Application Control (WDAC) Enforced Infrastructure,” Microsoft Learn. Available: https://learn.microsoft.com/en-us/windows-server/manage/windows-admin-center/use/manage-application-control-infrastructure.

[80] ”AppArmor vs SELinux: Compare the Differences in Linux Security,” TuxCare. Available: https://tuxcare.com/blog/selinux-vs-apparmor/.

[81] ”Comparison Between AppArmor and Selinux,” *Information Security Stack Exchange*. Available: https://security.stackexchange.com/questions/29378/comparison-between-apparmor-and-selinux.

[82] ”AppArmor vs SELinux - Quick Comparision with Top 10 Advantages,” *Reddit*. Available: https://www.reddit.com/r/CentOS/comments/102ma38/apparmor_vs_selinux_quick_comparision_with_top_10/.

[83] ”Windows vs macOS Security in 2025,” WebAsha. Available: https://www.webasha.com/blog/windows-vs-macos-security-which-operating-system-should-you-trust.

[84] ”Linux vs. Windows vs. Mac: An In-Depth Comparison of OS,” Wbcom Designs. Available: https://wbcomdesigns.com/linux-vs-windows-vs-mac/.

[85] ”Windows vs macOS vs Linux,” Crucial.com. Available: https://www.crucial.com/articles/pc-users/windows-vs-macos-vs-linux.

[86] ”Windows vs. macOS vs. Linux: The Ultimate Operating System Showdown,” Nahil. Available: https://nahil.com.sa/windows-vs-macos-vs-linux-the-ultimate-operating-system-showdown/.

[87] ”About System Integrity Protection on your Mac,” Apple Support. Available: https://support.apple.com/en-us/102149.

[88] ”MacOS System Integrity Protection - What is it and how to control it?,” 6’ Networks, LLC. Available: https://www.youtube.com/watch?v=_yW_VHX3THO.

[89] ”System Integrity Protection - Adding another layer to Apple’s security model,” Der Flounder. Available: https://derflounder.wordpress.com/2015/10/01/system-integrity-protection-adding-another-layer-to-apples-security-model/.

[90] ”Gatekeeper and runtime protection in macOS,” Apple Support. Available: https://support.apple.com/guide/security/gatekeeper-and-runtime-protection-sec5599b66df/web.

[91] ”The Principle of Least Privilege (PoLP),” Aqua Security. Available: https://www.aquasec.com/cloud-native-academy/application-security/the-principle-of-least-privilege-polp/.

[92] ”Understanding the Principle of Least Privilege (POLP),” Legit Security. Available: https://www.legitsecurity.com/aspm-knowledge-base/what-is-the-principle-of-least-privilege-polp.

[93] ”Least Privilege Principle: Benefits, Risks & Implementation,” Beta Systems Software AG. Available: https://www.betasystems.com/resources/blog/principle-of-least-privilege.

[94] ”OS Hardening: 15 Best Practices,” Perception Point. Available: https://perception-point.io/guides/os-isolation/os-hardening-10-best-practices/.

[95] ”What Are System Hardening Standards?,” RSI Security. Available: https://blog.rsisecurity.com/what-are-system-hardening-standards/.

[96] ”How Hardening is Reflected in the Different NIST Standards,” CalCom Software. Available: https://calcomsoftware.com/how-hardening-is-reflected-in-the-different-nist-standards/.

[97] ”What Is an Attack Surface? Definition & Management Tips,” Proofpoint. Available: https://www.proofpoint.com/us/threat-reference/attack-surface.

[98] ”Enhanced Operating System Security Through Efficient and Fine-grained Address Space Randomization,” USENIX. Available: https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final181.pdf.

[99] ”Restrict File and Directory Permissions, Mitigation M1022,” MITRE ATT&CK®. Available: https://attack.mitre.org/mitigations/M1022/.

[100] ”docker security best practices for production environments,” Medium. Available: https://medium.com/@subhasmitadas696/docker-security-best-practices-for-production-environments-a5994e29a8cb.

[101] ”Guidelines for system hardening,” Cyber.gov.au. Available: https://www.cyber.gov.au/resources-business-and-government/essential-cybersecurity/ism/cybersecurity-guidelines/guidelines-system-hardening.

[102] ”Linux Kernel Vulnerabilities Expose 78 Subsystems to Attack,” Electropages. Available: https://www.electropages.com/blog/2025/02/126-linux-kernel-vulnerabilities-lets-attackers-exploit-78-linux-sub-systems.