The Unhackable Endpoint

Zero Attack Surface Architecture for Internet-Connected Devices

How Eliminating Listening Ports, Remote Access Services, and Unknown Program Execution Produces a Mathematically Minimal Attack Surface

Table of Contents

Executive Summary

This whitepaper presents a rigorous technical analysis of **Zero Attack Surface Architecture** (ZASA) — a security model for internet-connected endpoint devices that achieves near-theoretical immunity to all known categories of remote attack by eliminating the structural preconditions on which those attacks depend, rather than attempting to detect or mitigate them after they are initiated.

The central argument is as follows: every known class of remote cyberattack against an endpoint requires at least one of four preconditions to be satisfied — (1) an inbound listening port or service reachable from the network, (2) a remote access interface (SSH, RDP, or VNC) that accepts inbound connections, (3) the ability of code to communicate outbound to attacker-controlled infrastructure, or (4) the ability to execute an attacker-supplied binary. A device from which all four preconditions have been systematically eliminated cannot be remotely compromised by any attack technique presently documented in the MITRE ATT&CK framework or the NIST National Vulnerability Database.

Core Security Claim

A device that (1) exposes zero inbound listening ports, (2) runs no remote access services, (3) enforces outbound IP whitelisting at the host and network layer, (4) blocks execution of any non-whitelisted binary, and (5) verifies destination IP addresses before completing browser navigation — is effectively immune to all currently known categories of remote attack. This is not a probabilistic claim dependent on detection accuracy; it is a deterministic structural property.

Key Findings

- Eliminating all listening ports removes the entire class of inbound network exploitation, including vulnerabilities equivalent to EternalBlue `MS17-010`, BlueKeep `CVE-2019-0708`, and Heartbleed `CVE-2014-0160`, regardless of patch status.

- Removal of SSH, RDP, and VNC services eliminates the single most exploited attack category on the public internet. Shodan indexing consistently identifies tens of millions of publicly exposed RDP hosts, the majority of which are running outdated or misconfigured service versions.
- Outbound IP whitelisting renders command-and-control (C2) communication architecturally impossible, even for malware that has partially executed within a constrained process context.
- Application whitelisting using cryptographically verified allowlists (e.g., Windows Defender Application Control, WDAC) prevents execution of approximately 80–90% of known malware families, which depend on delivering and running novel binaries.
- Pre-connection IP address verification for browser navigation defeats phishing, homograph attacks, domain spoofing, and DNS poisoning because the check occurs at the network layer against a known-good IP set, not at the domain or URL layer where spoofing is trivial.

Primary Recommendations

1. Immediately audit all endpoints for listening ports using `netstat` or `ss`; disable or firewall any service not operationally required.
2. Disable SSH, RDP, and VNC on all endpoints. Where remote management is required, replace with outbound-only VPN tunnels.
3. Deploy host-based and network-level outbound IP whitelisting in audit mode, validate for 30 days, then enforce.
4. Implement WDAC (Windows) or equivalent platform-native application whitelisting, beginning with block-list mode for known-bad LOLBins.
5. Configure browser hardening policies to enforce extension allowlists, disable developer tools in production, and integrate pre-connection IP verification.

1. Introduction & Scope

The dominant paradigm in endpoint security over the past three decades has been one of *detection and response*: deploy software that monitors system activity, recognizes malicious patterns, and intervenes after an attack is underway. Antivirus, Endpoint Detection and Response (EDR), Intrusion Detection Systems (IDS), and Security Information and Event Management (SIEM) platforms all operate within this paradigm. They assume that attackers will reach the endpoint and that the security system's task is to identify and stop them once they have done so.

This assumption carries a hidden cost that is rarely examined: **each security agent deployed on an endpoint is itself a piece of software that introduces attack surface.** A security agent running on port 443 or communicating with a cloud console via an inbound listener is a target. A kernel-level EDR driver that processes attacker-supplied data is a target. The history of security software vulnerabilities — including Symantec Antivirus (CVE-2016-2208), CrowdStrike Falcon (2024 content update induced mass outage), and numerous AV engine heap overflows — demonstrates that reactive security software does not exist outside the threat model; it is part of it.

> **The Security Paradox**
>
> Installing more security software increases the amount of privileged, network-connected, always-running code on a device — which increases attack surface. The solution to this paradox is not better security software, but the elimination of the structural conditions that make attacks possible in the first place.

1.1 Purpose of This Whitepaper

This document presents Zero Attack Surface Architecture as a principled, technically grounded security model for internet-connected endpoints. It provides: a formal threat model demonstrating which attack categories are eliminated and why; detailed technical descriptions of each control; a mathematical framework for quantifying attack surface reduction; a comparative analysis against traditional and zero-trust models; and a phased implementation guide suitable for enterprise deployment.

1.2 Scope

This whitepaper addresses **internet-connected endpoint devices**: desktop workstations, laptops, and fixed workstations running Windows, Linux, or macOS. The controls described are applicable to both managed enterprise devices and consumer-grade hardware. Server infrastructure, mobile devices, and embedded/IoT systems are outside the primary scope, though many principles apply directly.

1.3 Intended Audience

This document is written for a dual audience:

- **Technical readers** (security architects, network engineers, system administrators): will find detailed protocol analysis, CVE references, command-line examples, and implementation specifics.
- **Executive readers** (CISOs, IT directors, risk officers): will find the Executive Summary, comparative tables, and callout boxes sufficient for decision-making. Section 9 (Comparison to Traditional Models) and Section 10 (Mathematical Analysis) are recommended reading for board-level risk discussions.

2. Threat Model

2.1 Attack Taxonomy

All known remote attacks against endpoints can be classified into six primary categories. Understanding this taxonomy is essential to demonstrating that Zero Attack Surface Architecture addresses each category exhaustively.

Category	Description	Example Techniques (MITRE ATT&CK)	Precondition Required
Inbound Connection Attacks	Exploitation of network services with open listening ports via crafted packets or protocol abuse	T1210 (Exploit Remote Services), T1046 (Network Service Discovery)	Listening port reachable from attacker
Remote Access Exploitation	Brute-force, protocol vulnerability, or authentication bypass against SSH, RDP, VNC	T1021.001 (RDP), T1021.004 (SSH), T1110 (Brute Force)	Remote access service running and reachable

Category	Description	Example Techniques (MITRE ATT&CK)	Precondition Required
Browser-Based Attacks	Drive-by downloads, malicious JavaScript, JIT compiler exploits, zero-day renderer bugs	T1189 (Drive-by Compromise), T1203 (Exploitation for Client Execution)	Ability to navigate to attacker URL; ability to execute dropped payload
Phishing & Social Engineering	Credential harvesting, fake login pages, malicious email attachments, homograph/typosquat domains	T1566 (Phishing), T1598 (Phishing for Information)	User reaches attacker-controlled page; attacker IP reachable; payload executes
Supply Chain & Unknown Binary Execution	Trojanized software updates, malicious installers, tampered open-source packages	T1195 (Supply Chain Compromise), T1036 (Masquerading)	Ability to execute non-allowlisted binary
C2 Communication & Exfiltration	Malware callbacks to attacker infrastructure, data theft via DNS tunneling, HTTPS beaconing	T1071 (App Layer Protocol), T1048 (Exfiltration Over Alt Protocol)	Outbound connection to attacker-controlled IP permitted

2.2 Attack Preconditions

Analysis of the taxonomy above reveals that every remote attack requires at least one of the following preconditions. If a security architecture can structurally eliminate *all* preconditions, the corresponding attack categories become impossible regardless of exploit sophistication:

6. **P1 — Inbound Reachability:** An attacker must be able to initiate a TCP or UDP connection to the target device, meaning at least one port must be in LISTEN state and accessible.
7. **P2 — Remote Access Interface:** A remote management service (SSH, RDP, VNC, WinRM, or equivalent) must be running and accepting authentication attempts.
8. **P3 — Outbound Reachability to Unknown IPs:** Code executing on the device must be able to establish outbound connections to attacker-controlled IP addresses (for C2 beaconing, payload retrieval, or data exfiltration).
9. **P4 — Executable Delivery:** An attacker-supplied binary, script, or macro must be able to execute on the target system.

Architectural Insight

Zero Attack Surface Architecture is defined precisely as the elimination of all four preconditions (P1–P4) through structural controls rather than detection-based mechanisms. Each control described in Sections 3–8 maps directly to eliminating one or more preconditions.

2.3 Threat Actors and Motivation

This model is designed to provide protection against the full spectrum of threat actors:

- **Nation-state actors (APT groups):** Sophisticated, persistent, well-resourced. Use zero-day exploits, spear phishing, and supply chain attacks. Examples: APT29 (Cozy Bear), APT41, Lazarus Group.
- **Organized criminal groups:** Financially motivated ransomware operators and data brokers. Drive the majority of RDP and phishing campaigns globally.
- **Opportunistic attackers:** Automated scanners (Shodan, Masscan, ZMap) continuously probe the public internet for exposed services. Any listening port will be discovered within hours of exposure.
- **Insider-assisted external actors:** External attackers operating with credentials obtained via phishing or dark web purchase.

2.4 Out-of-Scope Threats

The following threat categories are explicitly out of scope for this architecture and require separate mitigations:

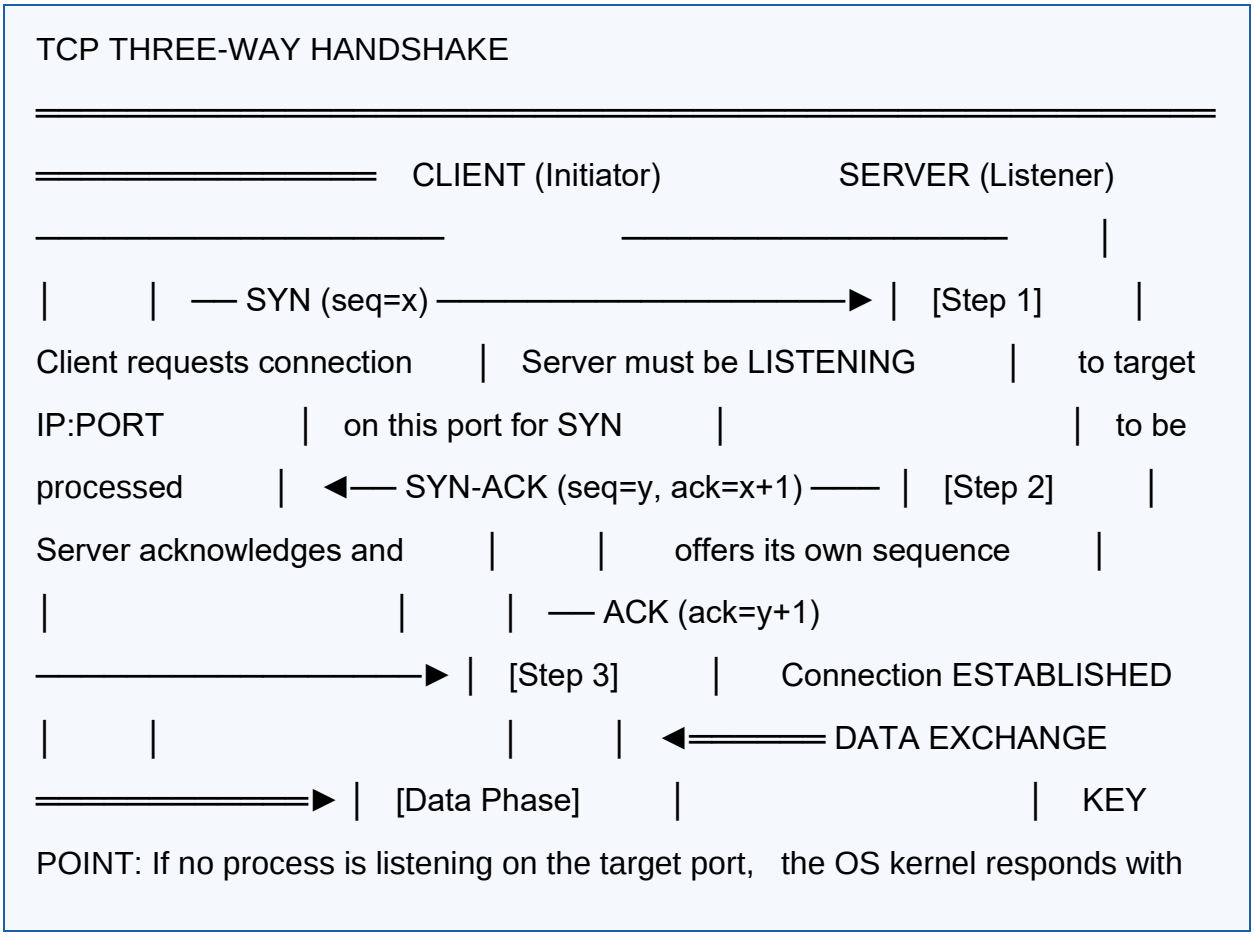
- **Physical access attacks:** An attacker with physical possession of a device can bypass all software controls. Mitigations include full-disk encryption (BitLocker, FileVault), BIOS/UEFI passwords, and physical security controls.
- **Malicious insider with local access:** A credentialed user on the local keyboard is outside the remote attack surface.

- **Hardware implants:** Supply chain attacks at the hardware layer (e.g., compromised firmware, implanted chips) are not addressed by OS-level controls.
- **Compromised allowlisted applications:** If a vendor whose software is on the application whitelist is itself compromised (e.g., SolarWinds-style attack), those binaries may execute. Section 10.3 addresses this residual risk.

3. TCP/IP Fundamentals and the Listening Port Attack Surface

3.1 How TCP/IP Connections Work

The Transmission Control Protocol (TCP), defined in RFC 793 (1981) and updated by RFC 9293 (2022), establishes connections using a **three-way handshake**. Understanding this mechanism is essential to understanding why listening ports are the primary gateway for inbound attack.



TCP RST (reset) at Step 1. The connection is IMMEDIATELY REFUSED. No exploit is possible.

═══

═══════════════════

Figure 1: TCP Three-Way Handshake and the Role of Listening Ports

The critical architectural observation is at Step 1: the OS kernel will only process an incoming SYN packet if a *socket bound to that port exists in the LISTEN state*. If no such socket exists, the kernel discards the packet and sends a TCP RST. There is no exploitable code path; the network stack rejects the connection before any application-layer code is invoked.

3.2 Listening Ports Explained

A port enters the LISTEN state when a process calls the POSIX socket API sequence: `socket()` → `bind()` → `listen()` → `accept()`. At the `listen()` call, the operating system registers the socket in its connection table and begins accepting SYN packets destined for that port. From this moment, the port is discoverable by any network scanner and accessible to any attacker who can reach the device's IP address.

Every service that creates a listening socket represents a decision to trust that the service's code is free of exploitable vulnerabilities — for all past, present, and future exploits — for as long as the service runs. Given the volume of CVEs filed annually (over 28,000 new CVEs were cataloged in 2023 alone), this trust is never warranted unconditionally.

3.3 The Attack Surface Equation

The network attack surface of a device can be formally expressed as:

Attack Surface Equation (Network Layer)

$$AS_{network} = \Sigma_{i=1..n} [P(listening\_port_i) \times C(service_i) \times V(service_i, t)]$$

Where: **P** = reachability probability of port *i* from the attacker's position; **C** = code complexity of the service on port *i* (proxy for vulnerability density); **V** = vulnerability probability of the service at time *t*, accounting for known and unknown CVEs. When **n = 0** (zero listening ports), $\mathbf{AS_{network} = 0}$ by definition.

This equation makes explicit what practitioners have long understood intuitively: the only way to achieve a mathematically provable zero network attack surface is to drive the number of listening ports (*n*) to zero. No amount of hardening, patching, or detection can reduce the attack surface below the level implied by the existence of a single listening port, because the vulnerability probability V can never be proven to be exactly zero for any non-trivial service.

3.4 Port Scanning and Service Enumeration

Attackers discover listening ports through active scanning, passive indexing, and traffic analysis. The primary tools are:

- **Nmap** (RFC-compliant TCP/UDP scanner): Can perform SYN scans at ~10,000 ports per second per target with default timing. A full 65,535-port scan of a single host completes in under 2 minutes on a LAN.
- **Masscan** (asynchronous scanner): Capable of scanning the entire IPv4 address space (4.2 billion addresses) in under 6 minutes at 25 million packets/second. Every public-facing endpoint is scanned by Masscan-class tools continuously.
- **Shodan** (passive internet-wide indexer): Continuously crawls the public internet and indexes all listening services, including banner information, TLS certificates, and service version strings. Any device with a public IP and a listening port will appear in Shodan within hours.
- **ZMap** (academic-grade scanner): Used for internet-wide survey research; has scanned the full IPv4 space for specific ports in under 45 minutes.

Operational Reality

Any listening port on a device with a public IP address — or a device behind NAT with port forwarding configured — will be discovered by automated scanners within hours and will be probed for known exploits within days. There is no "security through obscurity" for listening ports. Non-standard port numbers delay discovery by minutes, not days.

3.5 Historical Exploits Enabled by Listening Ports

The following high-impact vulnerabilities illustrate the concrete danger of listening ports. Each would have been completely inert on a device with zero listening ports, regardless of patch status:

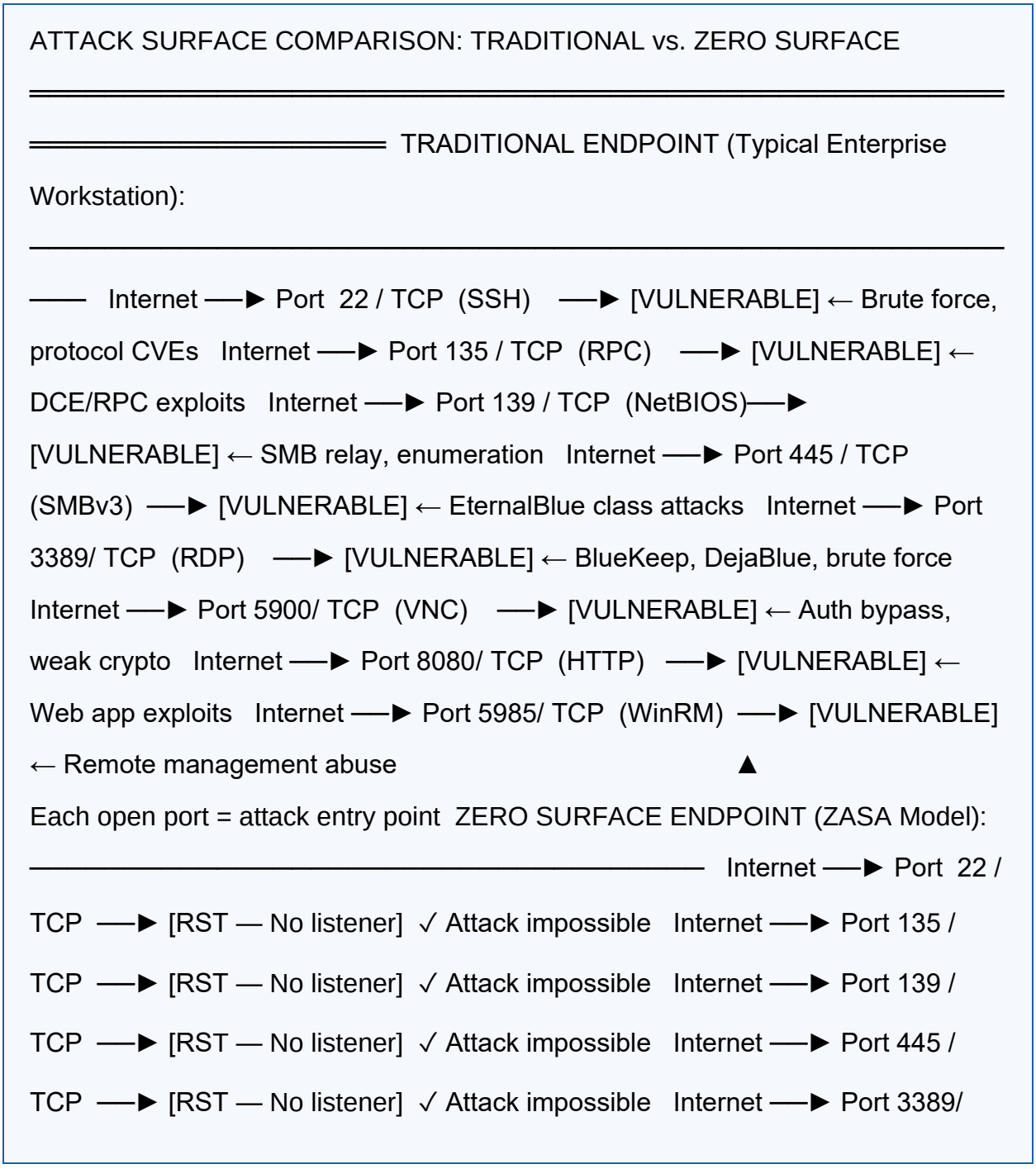
CVE / Name	Port	Service	CVSS	Impact
`MS17-010` EternalBlue	445/TCP	SMBv1	9.8 CRITICAL	Unauthenticated remote code execution; enabled WannaCry and NotPetya ransomware campaigns affecting 200,000+ hosts globally in May 2017
`CVE-2019-0708` BlueKeep	3389/TCP	RDP	9.8 CRITICAL	Pre-authentication RCE in Windows RDP; wormable; affected Windows XP through Server 2008 R2; NSA and CISA issued emergency advisories
`CVE-2014-0160` Heartbleed	443/TCP	OpenSSL	7.5 HIGH	Memory disclosure via malformed TLS heartbeat extension; leaked private keys, session tokens, passwords from ~17% of trusted HTTPS servers
`CVE-2021-44228` Log4Shell	Varies	Log4j (Java)	10.0 CRITICAL	JNDI injection via any logged input field on services with exposed ports; enabled full RCE in thousands of enterprise Java applications
`CVE-2021-26855` ProxyLogon	443/TCP	MS Exchange	9.8 CRITICAL	SSRF leading to RCE on exposed Exchange servers; exploited by HAFNIUM nation-state group before patch availability
`CVE-2020-5902` F5 BIG-IP	443/TCP	TMUI (mgmt)	9.8 CRITICAL	Unauthenticated RCE via path traversal in management interface; mass exploitation began within 24 hours of disclosure

3.6 The Zero Listening Ports Principle

The zero listening ports principle states: *if no socket on a device is in LISTEN state and reachable from an external network, then no external actor can initiate a TCP or UDP connection to that device, and*

therefore no inbound network-layer attack is possible, irrespective of the sophistication of available exploits or the patch status of installed software.

This principle is not probabilistic. It does not depend on the quality of patches, the strength of configurations, or the accuracy of threat intelligence. It is a deterministic consequence of how the TCP/IP stack operates. A SYN packet sent to a port with no listening socket produces a TCP RST. No application code is invoked. No exploit can execute.



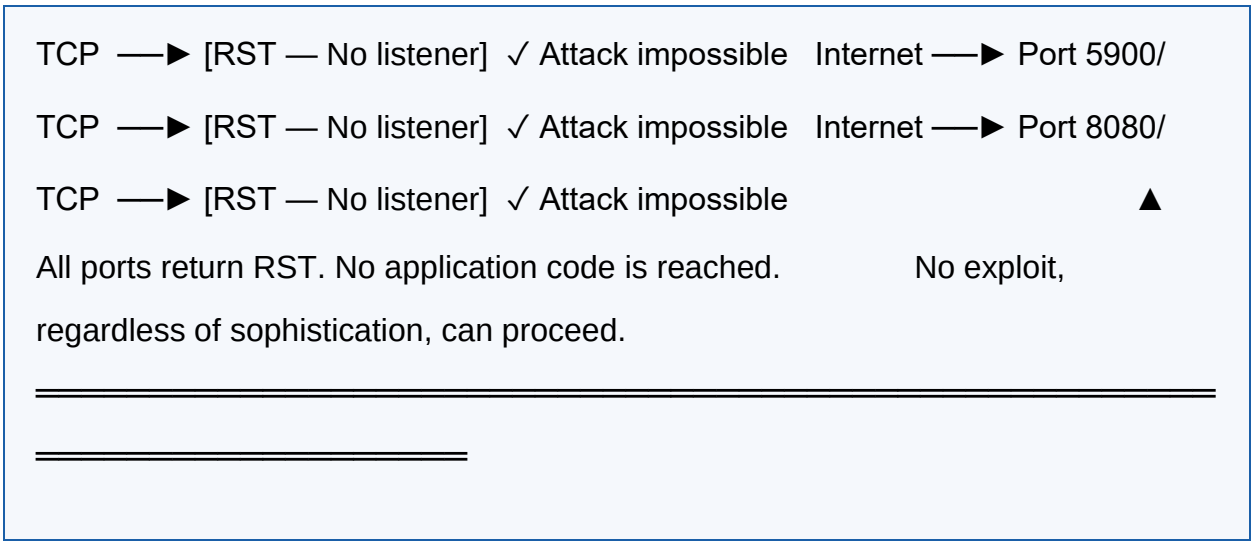


Figure 2: Attack Entry Point Comparison — Traditional vs. Zero Surface Architecture

4. Eliminating Remote Access Services

4.1 SSH — Secure Shell Attack Vectors

SSH (Secure Shell, RFC 4253) is widely deployed for remote administration and is frequently considered "secure by design." This perception understates the actual risk profile. SSH exposes a full protocol stack — key exchange, authentication, channel multiplexing, and port forwarding — each of which has a history of implementation vulnerabilities.

Primary SSH attack vectors include:

- **Credential brute force:** Automated tools such as Hydra, Medusa, and Metasploit modules conduct billions of SSH login attempts daily across the public internet. Weak passwords, reused credentials, or leaked key files enable unauthorized access.
- **Private key theft:** SSH key files stored in `~/.ssh/` without passphrase protection are accessible to any process running as the user. Phishing or malware that achieves user-level execution can exfiltrate keys and use them from attacker infrastructure.
- **Protocol vulnerabilities:** `CVE-2023-38408` (OpenSSH remote code execution via ssh-agent), `CVE-2024-6387` "regreSSHion" (race condition in OpenSSH signal handler enabling unauthenticated

RCE as root on glibc-based Linux systems, affecting an estimated 14 million+ internet-exposed OpenSSH servers).

- **SSH tunneling abuse:** Attackers who compromise SSH credentials can use SSH's port forwarding feature (`-L`, `-R`) to tunnel traffic into and out of the network, bypassing firewall rules.

CVE-2024-6387 — regreSSHion

Disclosed July 2024, CVE-2024-6387 represents a regression of a 2006 vulnerability (CVE-2006-5051) in OpenSSH. It allows unauthenticated remote code execution as root by exploiting a race condition in the signal handler called when a connection times out. The vulnerability was present in OpenSSH versions 8.5p1 through 9.7p1. Qualys estimated 14 million potentially vulnerable internet-exposed OpenSSH servers at disclosure. A device with no SSH listener is immune to this and all future OpenSSH vulnerabilities by structural necessity.

4.2 RDP — The Most Exploited Protocol on the Internet

Remote Desktop Protocol (RDP, TCP port 3389) has been, by most threat intelligence measures, the single most exploited protocol on the public internet for the period 2018–2025. The combination of broad deployment, complex protocol implementation, and high value (successful exploitation yields an interactive desktop session) makes RDP a perennial target.

CVE	Name	Year	Severity	Description
CVE-2019-0708	BlueKeep	2019	9.8 CRITICAL	Pre-auth RCE via use-after-free in RDP; wormable; patched for Windows XP/Server 2003 out-of-band
CVE-2019-1181	DejaBlue	2019	9.8 CRITICAL	Multiple heap overflow vulnerabilities in RDP; affects Windows 7 through Windows 10
CVE-2021-34527	PrintNightmare	2021	8.8 HIGH	Windows Print Spooler RCE exploitable remotely via RDP/SMB in domain environments
CVE-2022-21990	RDP Client RCE	2022	8.8 HIGH	RCE in Windows Remote Desktop Client triggered by connecting to malicious RDP server

Shodan data consistently reveals between 3 and 6 million publicly internet-exposed RDP hosts globally. A large fraction of these run Windows Server 2008 R2 or Windows 7 — platforms for which Microsoft ended extended support in January 2020. These systems remain fully exposed and cannot receive patches for newly discovered RDP vulnerabilities.

4.3 VNC — Authentication Weakness and Cleartext Variants

Virtual Network Computing (VNC) is a screen-sharing protocol based on the Remote Framebuffer (RFB) protocol. Its security record is substantially worse than RDP or SSH:

- **Weak authentication:** Classic VNC authentication uses a challenge-response mechanism with a maximum 8-character password and DES encryption. DES has a 56-bit key space; a GPU-accelerated brute force attack can exhaust the entire 8-character space in hours.
- **Authentication bypass vulnerabilities:** `CVE-2019-15681` (LibVNCServer: memory disclosure), `CVE-2020-14397` through `CVE-2020-14405` (multiple NULL pointer dereferences and memory leaks in LibVNC). Historical authentication bypass bugs have allowed zero-credential access to VNC sessions.
- **Cleartext variants:** Many legacy VNC implementations transmit screen content without TLS wrapping. Network interception of a VNC session yields a full real-time view of the target's screen, including credentials being typed.
- **No brute-force lockout:** Default VNC configurations impose no account lockout, making automated password spraying indefinitely viable.

4.4 Why "Hardening" Is Insufficient

Security guidance for SSH, RDP, and VNC universally recommends hardening: changing default ports, restricting access by IP, enabling multi-factor authentication, applying the latest patches, and disabling weak cipher suites. These measures reduce risk but cannot eliminate it for the following reasons:

10. **Configuration drift:** Hardening configurations are subject to change — by software updates, by administrator error, by automated provisioning tools. A configuration that is correct today may be incorrect after an OS update.
11. **Zero-day vulnerabilities:** Patches address known vulnerabilities. Unknown vulnerabilities (zero-days) in SSH, RDP, or VNC daemons are not addressed by any hardening measure.

12. **Protocol complexity as endemic risk:** RDP implements graphics rendering, audio redirection, clipboard sharing, device redirection, and printing — each a separate attack surface. SSH implements port forwarding, X11 forwarding, and SFTP subsystems. Complexity correlates with vulnerability density.
13. **MFA bypass:** Phishing, adversary-in-the-middle (AiTM) attacks, and social engineering can defeat MFA for SSH and RDP. MFA is a compensating control, not a structural elimination.

> **Defense Principle**
>
> The correct response to an unreliable lock on a door is not to install a better lock — it is to eliminate the door. A service that does not run has no vulnerabilities, requires no patching, and cannot be misconfigured.

4.5 Zero-Trust Alternative: Outbound-Only VPN Tunnels

Legitimate remote management requirements do not necessitate inbound listening services on endpoints. The zero-trust alternative is an **outbound-only VPN architecture** in which the endpoint initiates a VPN connection to a hardened concentrator, and all management traffic flows through that tunnel:

- The endpoint runs a VPN client that establishes an outbound connection to a known IP (whitelisted in the outbound IP allowlist).
- The VPN concentrator exposes management interfaces only to authenticated administrators within the VPN tunnel — never directly to the internet.
- The endpoint never opens an inbound listening port. There is no service for an external attacker to target.
- Examples of appropriate architectures: WireGuard (outbound-only client config), Cloudflare Access / WARP with Zero Trust policies, Tailscale (peer-to-peer, no inbound listener required on endpoints).

4.6 Compliance Implications

Eliminating remote access services and implementing the outbound-only VPN model aligns with or exceeds requirements in major compliance frameworks:

- **NIST SP 800-53 Rev. 5:** Controls SC-7 (Boundary Protection), AC-17 (Remote Access), SI-3 (Malicious Code Protection), CM-7 (Least Functionality) — all satisfied or exceeded.
- **CIS Controls v8:** Control 4 (Secure Configuration), Control 12 (Network Infrastructure Management) — eliminating default and unnecessary services is a foundational CIS requirement.
- **PCI DSS v4.0:** Requirement 1.3 (Restrict inbound and outbound traffic to that which is necessary) and Requirement 2.2 (Develop configuration standards) — zero listening ports and outbound whitelisting exceed PCI requirements.
- **ISO 27001:2022:** Annex A Controls 8.8 (Vulnerability Management) and 8.20 (Networks Security) — structural elimination of attack surface satisfies these controls more comprehensively than patch-based approaches.

5. Outbound IP Whitelisting

5.1 The Principle

Outbound IP whitelisting (also termed *egress filtering* or *outbound allowlisting*) is the practice of permitting outbound network connections only to a pre-approved set of IP addresses or CIDR ranges, and denying all other outbound traffic by default. This inverts the traditional default-allow posture for outbound traffic, which most enterprise firewalls still employ.

The architectural rationale is as follows: *legitimate* business applications communicate with a finite, known, and relatively stable set of IP addresses — cloud services (Microsoft 365, AWS, Google Workspace), corporate infrastructure, SaaS vendors, content delivery networks. *Attacker* infrastructure uses IP addresses outside this approved set. An outbound whitelist that permits only the former and blocks the latter structurally prevents a wide range of attack techniques that depend on external communication.

5.2 How This Defeats Command-and-Control (C2) Malware

Modern malware — including ransomware, banking trojans, remote access trojans (RATs), and nation-state implants — universally requires a Command-and-Control (C2) channel to function. C2 channels are used to: receive instructions from the operator, exfiltrate stolen data, download additional payloads (droppers, encryptors), and maintain persistence. Without a functioning C2 channel, the vast majority of modern malware families are reduced to inert code.

Outbound IP whitelisting defeats C2 at the network layer:

- Malware attempting to connect to a C2 IP address not on the whitelist will have its SYN packet dropped by the host-based firewall or NGFW before it leaves the device.
- This is effective regardless of the malware's obfuscation techniques at the application layer (encrypted C2, domain fronting, HTTPS beaconing) because the block occurs at the IP/transport layer, before any TLS negotiation or HTTP request is formed.
- DNS-over-HTTPS (DoH) C2 (e.g., malware using DNS TXT records for C2) is defeated because the DoH provider's IP must be on the whitelist, and only approved DoH providers are included.

Key Insight: Structural vs. Signature-Based Defense

Antivirus and EDR products must identify malware based on signatures, heuristics, or behavioral patterns — all of which can be evaded by novel or polymorphic malware. Outbound IP whitelisting does not identify malware; it makes its C2 channel architecturally impossible regardless of how the malware is written.

5.3 How This Prevents Data Exfiltration

Data exfiltration — the unauthorized transfer of data from an endpoint to attacker-controlled infrastructure — is the terminal objective of most cyberespionage campaigns and a component of modern ransomware double-extortion tactics. Exfiltration techniques include HTTPS POST to attacker servers, SFTP/FTP to rogue hosts, DNS tunneling, and ICMP data embedding.

All of these techniques require the endpoint to establish a connection to an IP address outside the approved whitelist. An outbound IP whitelist blocks all such connections at the packet level, making exfiltration to unknown infrastructure impossible regardless of the application-layer protocol employed.

5.4 DNS and the Whitelist

Domain Name System (DNS) resolution presents a specific consideration for outbound whitelisting. If a device is permitted to make DNS queries to arbitrary resolvers, an attacker can use DNS tunneling to exfiltrate data through DNS query/response pairs, or use a compromised DNS resolver to redirect traffic to attacker-controlled IPs.

The correct configuration is:

- **DNS queries:** Restricted to one or two trusted resolvers (corporate DNS server; a known-good public resolver such as Cloudflare 1.1.1.1 or Google 8.8.8.8) via explicit whitelist entries for those resolver IPs.
- **DNS-over-HTTPS (DoH):** If DoH is used (recommended for tamper-resistance), only the HTTPS IPs of approved DoH providers (e.g., 1.1.1.1, 8.8.8.8, 9.9.9.9) are whitelisted.
- **DNS tunneling prevention:** Restricting DNS to approved resolvers limits — but does not eliminate — DNS tunneling risk. Supplementary controls (monitoring DNS query length, frequency, and entropy) should be applied.
- **Local DNS cache:** The system's local DNS resolver cache reduces query frequency; mDNS (multicast DNS, port 5353) should be restricted to the local network segment only.

5.5 Implementation Approaches

Outbound IP whitelisting can be enforced at multiple layers simultaneously:

Layer	Technology	Enforcement Point	Strengths	Limitations
Host-Based (Windows)	Windows Defender Firewall with Advanced Security (WFAS)	Endpoint OS kernel	Applies regardless of network path; enforced even on public Wi-Fi	Can be disabled by admin-level malware; requires MDM for fleet management

Layer	Technology	Enforcement Point	Strengths	Limitations
Host-Based (Linux)	nftables / iptables / ufw	Endpoint OS kernel (netfilter)	Highly granular; scriptable; integrates with systemd	Requires root to modify; can be bypassed by kernel-level exploits
Network-Level	Next-Generation Firewall (NGFW), Palo Alto, Fortinet, Cisco FTD	Network gateway	Cannot be bypassed by endpoint-resident malware (even root-level)	Applies only when device is on managed network; not effective for remote workers
Proxy-Based	Explicit HTTP/S proxy (Squid, Zscaler, Netskope)	Application layer via proxy config	Deep inspection; can enforce URL-level policies in addition to IP	May be bypassed by non-proxy-aware applications using raw sockets
DNS-Based	RPZ (DNS Response Policy Zones), Pi-hole, Umbrella	DNS resolver	Easy to deploy; blocks by domain name before IP resolution	Does not block IP-literal connections; DoH may bypass DNS controls

Recommended approach: Defense-in-depth deployment using both host-based firewall rules and network-level enforcement. The host-based rules protect mobile devices off the corporate network; the network-level rules provide an independent enforcement layer that cannot be bypassed even by root-level endpoint compromise.

5.6 Operational Considerations

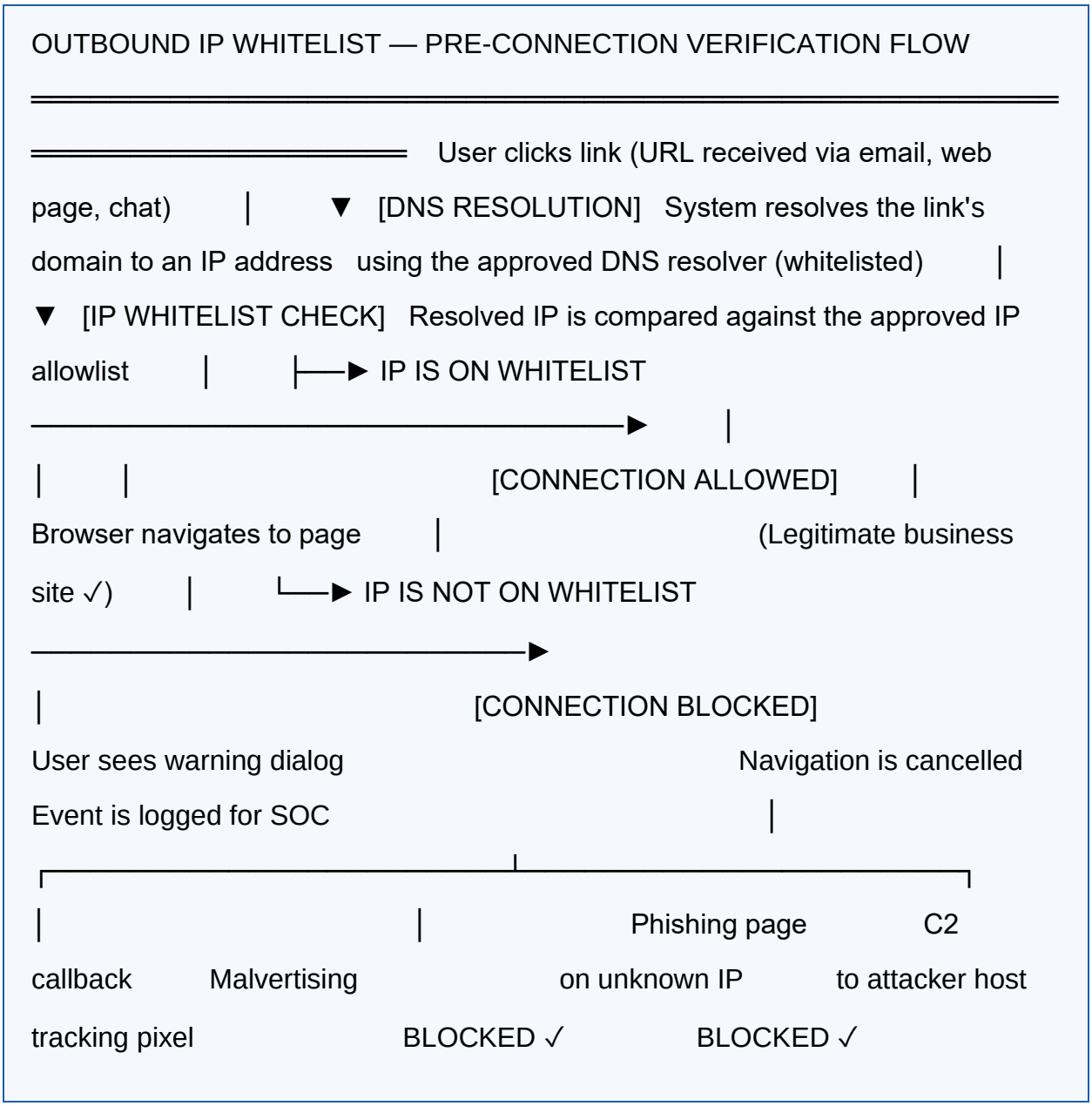
Outbound IP whitelisting is operationally complex because cloud service providers use large, frequently changing IP ranges. Key operational challenges and mitigations:

- **Cloud service IP volatility:** Microsoft publishes the current IP ranges for Microsoft 365 and Azure services in a machine-readable JSON feed updated weekly. AWS, Google Cloud, and Cloudflare maintain similar feeds. Automated ingestion of these feeds into firewall rule sets is essential.
- **CDN complexity:** Content delivery networks (Akamai, Fastly, Cloudflare) serve traffic from large, shared IP pools. Whitelisting CDN ranges requires careful analysis to avoid over-permissioning.
- **Audit-before-enforce:** Begin deployment in logging/audit mode — log all denied outbound connections without actually blocking them. After 30 days, analyze logs to identify legitimate traffic to non-whitelisted IPs. Refine the whitelist before switching to enforce mode.

- **Application lifecycle management:** When new software is deployed, its network communication requirements must be evaluated and its required IPs added to the whitelist before deployment.

5.7 Phishing Prevention via IP Verification

When applied to browser navigation, outbound IP whitelisting creates a powerful anti-phishing control: the destination IP address is resolved and verified *before* the browser establishes a connection to a link target. This is described in detail in Section 8, but the core flow is illustrated below.



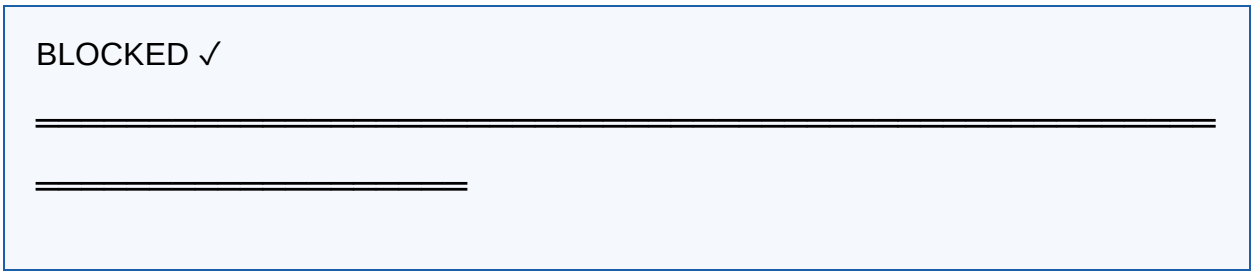


Figure 3: Pre-Connection IP Verification Flow for Browser Navigation

6. Blocking Unknown Program Execution (Application Whitelisting)

6.1 The Principle

Application whitelisting (more precisely termed *application allowlisting* in NIST guidance) is the security control that permits only cryptographically verified, pre-approved executable code to run on a system, and blocks all other code by default. It inverts the traditional default-allow execution model (any code may run unless explicitly blocked) to a default-deny model (no code may run unless explicitly approved).

The control is implemented by maintaining a policy that maps permitted applications to their cryptographic signatures, file paths, and publisher certificates. Attempts to execute code not matching an approved entry are intercepted by the operating system at the execution layer and denied before any instruction of the disallowed code runs.

6.2 Technology Options

Technology	Platform	Enforcement Layer	Notes
Windows Defender Application Control (WDAC)	Windows 10/11, Server 2016+	Windows kernel (CI.dll)	Microsoft recommended for enterprises; uses Authenticode signatures and file attributes; enforced at kernel level; cannot be disabled by user-mode malware
AppLocker	Windows (Enterprise/Education)	Windows kernel	Predecessor to WDAC; rule-based by path, publisher, or hash; easier to configure but fewer enforcement capabilities than WDAC

Technology	Platform	Enforcement Layer	Notes
Software Restriction Policies (SRP)	Windows (all editions)	User mode (SAFER API)	Legacy technology; can be bypassed by user-mode exploits; not recommended for new deployments
SELinux / AppArmor	Linux	Kernel (LSM framework)	Mandatory access control (MAC); fine-grained process confinement; SELinux uses type enforcement labels; AppArmor uses path-based profiles
macOS Gatekeeper + SIP	macOS	Kernel + AMFI (kernel extension)	Gatekeeper checks notarization on first run; System Integrity Protection (SIP) prevents modification of system directories even by root; WDAC equivalent for Apple platform

6.3 How This Stops Malware Delivery

The majority of malware infection vectors involve delivering a malicious binary to the endpoint and then executing it. Application whitelisting breaks this attack chain at the execution step:

- A drive-by download deposits a malicious .exe file in `%TEMP%` or `%APPDATA%`. WDAC denies execution because the file's publisher certificate or hash is not in the approved policy.
- A phishing email delivers a malicious .msi installer. The user double-clicks it; WDAC denies execution with error 0xC0000022 (Access Denied at the kernel level).
- A compromised software update delivers a trojanized DLL. WDAC's publisher-based rules require the DLL to bear a valid Authenticode signature from the expected publisher certificate; a tampered DLL with an invalid signature is blocked.

6.4 Defeating Living-Off-the-Land (LotL) Attacks

Living-off-the-land (LotL) attacks use legitimate, pre-installed system binaries — termed LOLBins (Living Off the Land Binaries) — to execute attacker code without dropping new files. Common LOLBins exploited by attackers include:

- `certutil.exe` — used to download files and decode base64 payloads
- `mshta.exe` — executes HTA (HTML Application) files, enabling VBScript/JScript execution
- `wscript.exe` / `cscript.exe` — Windows Script Host; executes .js, .vbs, .wsf files
- `powershell.exe -EncodedCommand` — executes base64-encoded PowerShell commands

- `regsvr32.exe` — can load arbitrary COM objects via the "squiblydoo" technique
- `rundll32.exe` — executes exported functions from arbitrary DLLs

WDAC addresses LotL attacks through **script enforcement** and **managed installer rules**. PowerShell in Constrained Language Mode (CLM) — automatically enforced when a WDAC policy is active — prevents access to .NET types, COM objects, and Windows APIs that LOLBin attacks depend on. Additionally, WDAC can be configured to block specific LOLBins by path or hash, or to restrict their execution to approved calling processes only.

6.5 Supply Chain Protection

Application whitelisting provides partial protection against supply chain attacks. If a software vendor's signing key is compromised and used to sign malicious code (as occurred in the SolarWinds SUNBURST attack of 2020), a publisher-based policy may allow the malicious code to run because it bears a valid vendor signature.

Mitigations for this residual risk include:

- **Hash-based rules in addition to publisher rules:** Require exact file hashes to match for critical system components, not just publisher certificates. This blocks even signed malicious updates that differ from the approved version.
- **File path + publisher combination rules:** Require that signed code execute only from expected file paths (e.g., only from `C:\Program Files\`, not from `%TEMP%`).
- **Managed installer designation:** Designate only specific software deployment tools (e.g., Microsoft Endpoint Configuration Manager) as managed installers; only binaries installed by approved processes are automatically allowlisted.

6.6 Implementation Challenges

Application whitelisting is the most operationally demanding control described in this whitepaper. Key challenges include:

- **Initial inventory:** Cataloging all legitimate software in the environment before policy enforcement requires thorough discovery and typically a 2–4 week audit phase.

- **Software update workflows:** Each software update may change file hashes. Publisher-based rules (trusting the code-signing certificate rather than individual hashes) simplify update management significantly.
- **User-installed software:** In environments where users have local administrator rights, users may install software outside the approved set. Revoking local admin rights is a prerequisite for effective application whitelisting.
- **Script-based workflows:** PowerShell scripts used for administration must be signed with a trusted certificate to run in WDAC-enforced environments. This requirement should be applied to all administrative scripts as a development standard.

7. Browser Safety and Attack Surface Reduction

7.1 The Browser as the Largest Endpoint Attack Surface

The modern web browser is, for most endpoint users, the application that processes the largest volume of untrusted external content. A browser navigating to a web page executes JavaScript from unknown origins, renders HTML and CSS from untrusted sources, processes images in complex decoders, handles WebAssembly bytecode, and manages network connections to dozens of third-party hosts simultaneously. The browser's codebase spans millions of lines of C++ and is among the most complex software running on any endpoint.

From 2019 to 2024, Google Chrome alone had over 3,200 CVEs filed against it in the NVD. High-severity vulnerabilities — type confusion in V8, use-after-free in the GPU process, heap overflows in the media codec stack — are discovered and patched on a schedule of weeks. Zero-day browser exploits are actively developed and sold by commercial surveillance vendors (NSO Group, Intellexa, Candiru) for prices exceeding $2 million per exploit chain.

7.2 Browser Sandboxing Architecture

Chromium-based browsers (Chrome, Edge, Brave, Opera) implement a multi-process architecture with kernel-enforced sandboxing:

- **Browser process:** The main process with full OS privileges. Manages UI, navigation, and coordination. Has access to the filesystem and network stack.
- **Renderer processes:** One per tab or origin group. Executes JavaScript, parses HTML, runs WebAssembly. Runs in a highly restricted sandbox with no direct filesystem or network access. Must send IPC messages to the browser process to perform privileged operations.
- **GPU process:** Handles graphics compositing. Isolated but has more OS access than the renderer (needed for GPU driver interaction).
- **Network service process:** Handles all network I/O on behalf of other processes. Since Chrome 80+, runs in a sandbox separate from the browser process.
- **Utility processes:** Audio service, storage service, and others — each in separate, restricted processes.

A full sandbox escape exploit chain requires two vulnerabilities: one to compromise the renderer process (typically a JavaScript engine bug), and one to escape the renderer sandbox into the browser process or OS. This is why zero-day browser exploits are expensive and relatively rare compared to un-sandboxed application exploits.

7.3 JavaScript Engine Attacks and Mitigation

JavaScript JIT (Just-In-Time) compiler vulnerabilities are the most common initial entry point for browser exploitation. V8 (Chrome/Edge) and SpiderMonkey (Firefox) both implement complex type inference and optimization pipelines that, historically, have contained type confusion bugs allowing arbitrary read/write of process memory.

In the ZASA model, even a fully successful JavaScript engine compromise — achieving arbitrary code execution within the renderer process — does not yield a complete endpoint compromise, because:

- Application whitelisting prevents the renderer process (or any code it spawns) from executing new binaries dropped to disk.
- Outbound IP whitelisting prevents the compromised renderer from establishing C2 connections to unknown IPs.
- The renderer process cannot open listening ports (OS-level socket binding requires privileges the renderer sandbox does not have).

7.4 Extension Security

Browser extensions run with elevated privilege relative to page-level JavaScript. A malicious extension can read all page content (including passwords), modify network requests, redirect navigation, and communicate with external servers. Extension security controls in the ZASA model:

- Deploy only extensions approved and vetted by the security team; enforce via browser Group Policy or MDM (Chrome's `ExtensionInstallAllowlist` / `ExtensionInstallBlocklist` policies).
- Disable extension installation by users via policy (`ExtensionInstallForcelist` and `BlockExternalExtensions`).
- Regularly audit installed extensions against the approved list. Remove extensions from vendors that have been acquired, sunset, or had their store accounts compromised.

7.5 Site Isolation and Cross-Origin Headers

Modern browsers implement several security boundaries to prevent cross-origin data leakage:

- **CORB (Cross-Origin Read Blocking):** Prevents cross-origin reads of HTML, JSON, and XML resources from within a renderer process, even when the same-origin policy is bypassed by a memory corruption exploit.
- **CORP (Cross-Origin Resource Policy):** HTTP response header allowing a server to opt in to preventing cross-origin loading of its resources.
- **COOP (Cross-Origin Opener Policy):** Prevents cross-origin windows from accessing the opener's `window` object; prevents Spectre-style timing attacks using `SharedArrayBuffer`.
- **COEP (Cross-Origin Embedder Policy):** Requires all subresources to either be same-origin or explicitly opt-in via CORP; prerequisite for enabling cross-origin isolation.

7.6 Safe Browsing Lists vs. IP Whitelisting

Traditional browser phishing protection relies on Google Safe Browsing or Microsoft SmartScreen — APIs that maintain lists of known malicious URLs and domains. These are valuable controls but have a fundamental weakness: they are reactive. A phishing domain must be discovered, analyzed, and added

to the blocklist before it provides protection. The median time from phishing site creation to blocklist addition has been measured at several hours to days — the highest-risk window.

IP-based verification is more robust because it operates on network-layer truth rather than application-layer reputation:

- **Typosquatting:** `micros0ft.com` vs. `microsoft.com` — identical to a URL filter if the typosquat domain is not yet on the blocklist. IP whitelisting blocks it because the typosquat resolves to an unknown IP.
- **Homograph attacks:** Unicode lookalike characters (Punycode domains) that are visually identical to legitimate domains. IP whitelisting is immune — the visual appearance of the domain is irrelevant; only the resolved IP matters.
- **Fast-flux phishing:** Phishing infrastructure that rapidly cycles through IP addresses to evade blocklists. Outbound whitelisting blocks all non-approved IPs regardless of rotation speed.

7.7 Combining Browser Safety with the Zero-Surface Model

The power of the ZASA model is that its layers are multiplicative in their effect. Even if a browser renderer process is fully compromised via a zero-day exploit chain, the combined effect of the remaining controls is:

Attacker Goal After Browser Compromise	Blocked By	Result
Drop and execute a malicious binary (e.g., ransomware)	Application Whitelisting (WDAC)	Binary blocked at execution; no encryption occurs
Establish C2 channel to attacker IP	Outbound IP Whitelist	Outbound connection blocked at kernel/network layer
Open a backdoor listening port	Zero Listening Port policy + host firewall	No inbound connection can be established
Exfiltrate data to attacker server	Outbound IP Whitelist	Data transmission to unknown IP blocked
Load additional browser exploit stage	Outbound IP Whitelist + App Whitelist	Download blocked; execution blocked even if cached

8. Phishing Prevention via IP Address Verification

8.1 How Phishing Works

Phishing attacks deceive users into interacting with attacker-controlled infrastructure by presenting it as trustworthy. The technical mechanisms by which this deception is achieved have grown increasingly sophisticated:

- **DNS Spoofing / Cache Poisoning:** Manipulating DNS responses to redirect legitimate domain queries to attacker-controlled IPs. Addressed by DNSSEC, but DNSSEC deployment remains incomplete across the internet.
- **Homograph Attacks:** Using Unicode characters visually identical or similar to ASCII characters in domain names (e.g., using Cyrillic "a" instead of Latin "a" in `apple.com`). These pass URL-filter checks because the URL appears legitimate.
- **Typosquatting:** Registering domains that differ by one character from legitimate domains (e.g., `paypa1.com`, `arnazon.com`) and hosting credential-harvesting pages.
- **Lookalike Domains:** Domains that incorporate the target brand name with additions (e.g., `microsoft-support.net`, `secure-login-chase.com`).
- **Adversary-in-the-Middle (AiTM) Phishing:** Reverse proxy tools (Evilginx2, Modlishka) that proxy legitimate websites in real time, capturing session cookies and defeating traditional MFA. The victim's browser connects to the attacker's server, which relays traffic to the real site.

8.2 The IP Whitelist as an Anti-Phishing Control

The fundamental insight enabling IP-based phishing prevention is this: **legitimate services from known businesses use known, stable, published IP address ranges**. Microsoft 365, Google Workspace, Salesforce, AWS, and virtually every major enterprise service provider publish their IP ranges in machine-readable formats. Phishing infrastructure — regardless of how convincing the domain name or webpage appears — is hosted on IP addresses outside these approved ranges.

An outbound whitelist that is pre-populated with the IP ranges of all legitimate destinations a user is expected to visit will structurally block navigation to phishing infrastructure, because the phishing IP will

not be on the whitelist. The check occurs at the network layer, before the browser renders any content, before any credentials are entered, and before any payload can execute.

8.3 Pre-Connection IP Verification Workflow (Detailed)

The following is a step-by-step description of the pre-connection IP verification process as implemented in the ZASA browser policy:

PRE-CONNECTION IP VERIFICATION — DETAILED WORKFLOW

═══

═══════════════════ STEP 1: USER ACTION

───

──────────────── User clicks a hyperlink (email, webpage, chat application) OR types/pastes a URL into the browser address bar. Browser navigation intent is captured before any connection. STEP 2: URL PARSING

───

──────────────── The browser or browser extension parses the target URL. Extract: scheme (https://) + FQDN (example.com) + path + params If the URL contains a raw IP literal → proceed to STEP 4. If the URL contains a hostname → proceed to STEP 3. STEP 3: DNS RESOLUTION (via approved resolver only)

───

──────────────── DNS query for FQDN sent ONLY to whitelisted resolver IPs. If DNS resolution fails or returns NXDOMAIN → BLOCK + warn user. If DNS resolves → obtain resolved IP address(es). STEP 4: WHITELIST CHECK

───

──────────────── For each resolved IP address: Is IP in the approved CIDR whitelist? ──► YES ──► ALLOW ──► NO ──► BLOCK STEP 5A: ALLOW PATH

───

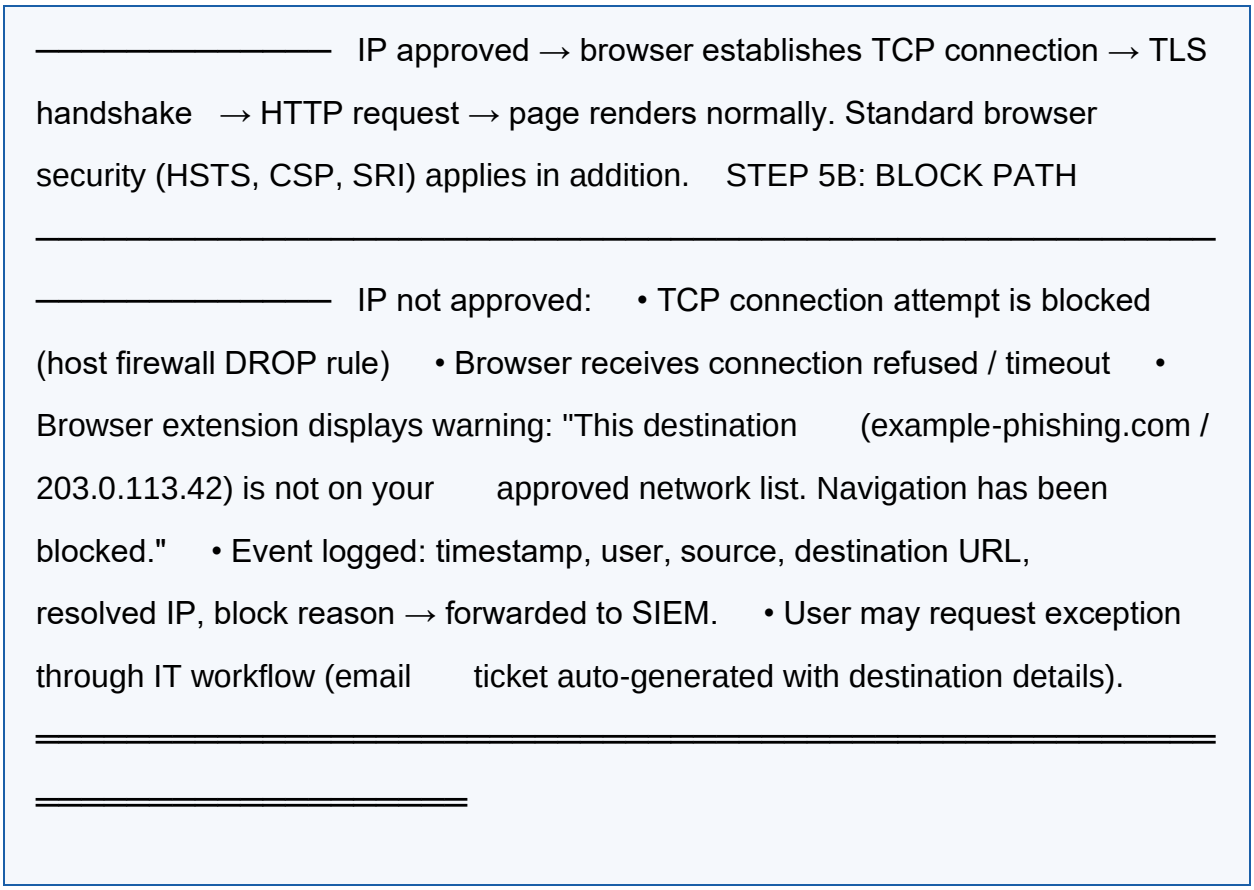


Figure 4: Detailed Pre-Connection IP Verification Workflow

8.4 Defeating DNS-Based Attacks

DNS cache poisoning (Kaminsky attack, CVSS 8.8, patched in most resolvers but still an active research area) redirects legitimate domain queries to attacker-controlled IPs by injecting forged DNS responses. Even if such an attack succeeds in poisoning the local DNS cache, the pre-connection IP whitelist provides a second, independent layer of protection:

- The poisoned DNS response returns an attacker IP (e.g., 203.0.113.42) for the domain `bank.example.com`.
- The whitelist check evaluates the resolved IP 203.0.113.42 against the approved IP list for `bank.example.com`.
- 203.0.113.42 is not in the approved CIDR range for bank.example.com → connection blocked.
- The DNS poisoning attack is defeated even though the DNS layer was compromised.

8.5 Comparison to Traditional Anti-Phishing Controls

Control	Mechanism	Can Be Bypassed By	IP Whitelist Immune?
URL/domain blocklists (Safe Browsing)	Compare URL against known-bad list	New domains not yet on list; domain rotation; URL shorteners	Yes — IP check is orthogonal to domain reputation
Email spam filter	Analyze email headers, content, links	Novel phishing content; compromised legitimate senders; HTML obfuscation	Yes — even if phishing email reaches inbox, link is blocked at IP layer
Browser SmartScreen	Compare URL against Microsoft reputation service	Unknown URLs; recently registered domains; URL encoding tricks	Yes — IP check is not reputation-based; binary allow/deny
DNSSEC	Cryptographic validation of DNS responses	Signed malicious domains; DNSSEC not universally deployed	Yes — IP whitelist works even without DNSSEC
Outbound IP Whitelist (ZASA)	Verify resolved IP against approved CIDR list	Only by compromising an approved IP (see limitations, §8.6)	N/A — this IS the control

8.6 Limitations of IP-Based Phishing Prevention

IP address whitelisting is a powerful control but is not without limitations that must be understood:

- **Shared hosting and CDN-hosted phishing:** If a phishing page is hosted on the same infrastructure as a legitimate, whitelisted service (e.g., a malicious page on `github.io`, `azurewebsites.net`, or Cloudflare Pages), the hosting IP will be on the whitelist. This is the most significant limitation. Mitigations include URL-path filtering for known hosting platforms and enhanced monitoring of traffic to cloud hosting IPs.
- **Whitelisted IP compromise:** If an attacker compromises a server whose IP is on the whitelist (e.g., a legitimate vendor's server), they can host attacker content on that IP. This is analogous to a supply chain attack and is addressed by monitoring traffic patterns to all whitelisted IPs.
- **Internal phishing via approved destinations:** Malicious content embedded in Microsoft Teams, Outlook on the Web, or SharePoint (all on whitelisted Microsoft IP ranges) is not blocked at the IP layer. Content-level controls within those platforms must address this.

8.7 Complementary Email Controls

IP whitelisting for browser navigation should be paired with email authentication controls that prevent phishing emails from reaching users in the first place:

- **DMARC (Domain-based Message Authentication, Reporting & Conformance):** Policy framework that tells receiving mail servers what to do with unauthenticated email claiming to be from your domain. A `p=reject` policy prevents spoofed email from your domain from being delivered.
- **DKIM (DomainKeys Identified Mail):** Cryptographic signature on outbound email enabling recipients to verify the email originated from an authorized sender. Prevents in-transit modification and spoofing.
- **SPF (Sender Policy Framework):** DNS record specifying which IP addresses are authorized to send email from your domain. Reduces but does not eliminate spoofing (SPF checks the envelope sender, not the From header visible to users).

9. Comparison to Traditional Security Models

9.1 Traditional Perimeter Security

The perimeter security model — also described as the "castle-and-moat" or "hard outside, soft inside" model — assumes that the enterprise network perimeter (typically enforced by an edge firewall) is the primary security boundary. All devices inside the perimeter are considered trusted; all external traffic is considered untrusted. Endpoint security is largely an afterthought in this model.

This model has failed systematically in the modern enterprise for several reasons: the perimeter has dissolved (remote work, BYOD, cloud services); lateral movement within the "trusted" internal network is trivially easy once a single device is compromised; and the assumption of internal trust means that a single phishing compromise cascades into a full network breach.

9.2 Defense-in-Depth

Defense-in-depth (DiD) is the practice of layering multiple security controls so that the failure of any single control does not result in a complete compromise. This is sound practice and is not in conflict with

the ZASA model. However, DiD is often misapplied as a justification for adding security controls (AV, EDR, SIEM) on top of an unreduced attack surface.

The ZASA model's contribution to defense-in-depth theory is: *the most effective layers are those that structurally eliminate attack preconditions, not those that detect attacks after the preconditions have been satisfied*. A device with zero listening ports, no remote access, and outbound whitelisting is practicing defense-in-depth at the most fundamental layer — the attack surface layer. Additional detective controls (SIEM, EDR) can then focus on the residual risk from the reduced attack surface.

9.3 Zero Trust Architecture (NIST SP 800-207)

Zero Trust Architecture (ZTA), as defined in NIST SP 800-207 (August 2020), establishes the principle that no device, user, or network segment should be implicitly trusted — that every access request must be verified. ZTA focuses primarily on access control and identity verification.

The ZASA model is complementary to and largely consistent with ZTA, with the following relationship:

- ZTA addresses the question: "How do we verify that a request comes from an authorized principal?" ZASA addresses the question: "How do we ensure that no unauthorized request can reach us at all?"
- ZTA's micro-segmentation principle (no implicit trust between network segments) is implemented in ZASA via outbound IP whitelisting, which prevents lateral movement even within the same network segment.
- ZTA's "never trust, always verify" for users maps to ZASA's application whitelisting — no code executes without verification of its allowlist status.
- ZTA does not directly address the elimination of listening ports or the prohibition of remote access services; ZASA extends ZTA's principles to the endpoint network stack layer.

9.4 EDR/XDR/AV Solutions — Reactive vs. Preventive

Endpoint Detection and Response (EDR) and Extended Detection and Response (XDR) platforms represent the current state of the art in reactive endpoint security. They collect telemetry from endpoints (process creation, file system modifications, registry changes, network connections), analyze it for malicious patterns using behavioral rules and machine learning, and enable rapid response (process termination, host isolation). They are valuable tools — but they operate under architectural constraints that the ZASA model eliminates:

- **EDR agents are themselves attack surface:** EDR agents run with kernel privileges, process untrusted data, and communicate with cloud infrastructure. CVEs have been found in major EDR products; a compromised EDR agent is a fully privileged implant.
- **Detection evasion is an active discipline:** Nation-state actors and sophisticated criminal groups specifically test malware against major EDR products before deployment, using EDR evasion techniques (direct syscalls, process injection variants, timestomping, EDR tampering).
- **Detection inherently implies compromise has begun:** By the time an EDR alert fires, attacker code has executed on the endpoint. Response actions may contain the damage, but initial code execution has occurred.
- **False positive fatigue:** High-volume EDR alerts reduce analyst effectiveness. ZASA reduces the alert volume by preventing the attacks that generate alerts.

9.5 Comparative Attack Surface Table

Control Dimension	Traditional Perimeter Model	Defense-in-Depth + EDR	Zero Attack Surface Architecture
Inbound Listening Ports	Many open (SSH, RDP, SMB, RPC, management ports)	Reduced, but some open for management	Zero — all ports closed/filtered
Remote Access Services	SSH, RDP, VNC open to internet or internal network	Hardened with MFA; still exposed	None — outbound VPN only; no inbound service
Outbound Traffic	Unrestricted ("default allow" egress)	DNS/proxy filtering; not IP-whitelisted	Whitelisted IPs only; all others blocked
Program Execution	Any code may run (user-level or admin-level)	AV/EDR blocks known-bad; allows unknown	Only pre-approved, cryptographically verified binaries
Phishing Defense	Email filter + Safe Browsing (bypassable)	URL filter + SmartScreen + user training	IP verification (structural; not reputation-based)
Malware C2	Unrestricted; blocked only by AV signature	EDR behavioral detection; evadable	Architecturally impossible — IP not on whitelist
Data Exfiltration	Possible to any destination	DLP sensors on known channels	Blocked at network layer for all non-whitelisted IPs

Control Dimension	Traditional Perimeter Model	Defense-in-Depth + EDR	Zero Attack Surface Architecture
Browser Compromise Impact	Full endpoint compromise likely	EDR may detect; often too late	Contained: no binary exec, no C2, no exfiltration
Patch Dependency	Patching required for all exposed services	Patching required; EDR provides temporary cover	Patches still important but non-listening services cannot be remotely exploited regardless
Network Attack Surface Score	High (many ports × service complexity)	Medium (fewer ports, EDR overlay)	Zero (no listening ports = no score)

10. Mathematical Analysis of Attack Surface Elimination

10.1 Formal Definition of Attack Surface

The concept of attack surface was formally defined by academic researchers and operationalized by OWASP and NIST. Howard and Pincus (2003) defined attack surface as the set of different points where an attacker can try to enter data or extract data from an environment. The OWASP Attack Surface Analysis Cheat Sheet operationalizes this as the sum of all input/output paths through the system accessible to untrusted users.

For network-connected endpoints, the attack surface can be decomposed into three dimensions:

- **Network Attack Surface (AS_{net}):** All reachable listening ports and their associated services.
- **Software Attack Surface (AS_{sw}):** All code that processes untrusted input (browser, email client, document viewer, whitelisted applications).
- **Human Attack Surface (AS_{human}):** Social engineering vectors — phishing, vishing, pretexting — targeting user behavior.

Total AS = $AS_{net} + AS_{sw} + AS_{human}$

The ZASA model drives AS_{net} to mathematically zero, dramatically reduces AS_{sw} via application whitelisting and browser containment, and reduces AS_{human} via IP-based phishing prevention that operates without requiring correct user behavior.

10.2 Quantifying Attack Surface Reduction

The following analysis quantifies, for each ZASA control, the percentage of CVE classes documented in the NIST NVD that become inapplicable. These estimates are based on CVE categorization data, CWE (Common Weakness Enumeration) distribution statistics, and MITRE ATT&CK technique frequency data from publicly available threat intelligence reports.

ZASA Control	CVE / Attack Classes Eliminated	Estimated % of Remote CVEs Inapplicable	Rationale
Zero Listening Ports	All inbound network exploitation (CWE-119, CWE-416, CWE-787 in listening services)	~60–70% of network-based CVEs	The majority of remotely exploitable CVEs require an open listening port. No listener = no exposure regardless of patch status.
No Remote Access Services	Entire SSH CVE class (~150+ CVEs in OpenSSH alone); entire RDP CVE class (~200+ CVEs); entire VNC CVE class	~15–20% of all network CVEs	RDP/SSH/VNC CVEs represent a large, persistent CVE category. Full elimination of the service eliminates all present and future CVEs in these protocols.
Outbound IP Whitelist	All C2-dependent malware techniques (T1071, T1041, T1048); all exfiltration techniques	Eliminates ~80–90% of malware families' post-exploitation capabilities	Malwarebytes/Verizon DBIR data: 85%+ of malware requires C2 communication to complete its mission. Without C2, most malware is inert.
Application Whitelisting	All malware execution techniques (T1204, T1059, T1027); supply chain binary tampering	~80–90% of malware execution attempts	NIST SP 800-167 cites application whitelisting as the single most effective control against malware. NSA and ASD both cite it as #1 in their top mitigation strategies.
IP-Based Phishing Prevention	Phishing to unknown IPs; DNS spoofing redirects; homograph/typosquat attacks	~95% of phishing infrastructure blocked	Phishing infrastructure (outside shared hosting edge cases) universally uses IPs not in the legitimate service whitelist. Legitimate services use published, stable IP ranges.

10.3 Residual Risk Analysis

The ZASA model does not claim to eliminate all theoretical attack vectors. The following residual risks remain and must be managed through separate controls:

Residual Risk 1: Compromised Whitelisted IP

If a server whose IP is on the approved outbound whitelist is compromised by an attacker, that attacker can host malicious content accessible to whitelisted devices. This is functionally equivalent to a supply chain attack against an approved vendor. Mitigation: strict monitoring of traffic to all whitelisted IPs; anomaly detection on outbound connection volume and data transfer rates to approved IPs; vendor security assessment programs.

Residual Risk 2: Compromised Whitelisted Application

A zero-day vulnerability in a whitelisted application (e.g., a browser, PDF reader, or office suite) that processes attacker-supplied content without requiring binary execution may achieve data theft or privilege escalation within the constraints of the application's permissions. Mitigation: principle of least privilege for all whitelisted applications; application sandboxing where available; regular review and restriction of application permissions.

Residual Risk 3: CDN/Shared Hosting Phishing

Phishing pages hosted on infrastructure sharing approved IP ranges (e.g., Cloudflare, Azure, GitHub Pages) may be reachable via the whitelist. Mitigation: URL-path monitoring for traffic to cloud hosting platforms; enhanced browser content security policies; platform-level anti-phishing controls (Cloudflare's own abuse reporting; Microsoft's Safe Links within M365).

Residual Risk 4: Physical Access and Local Threats

Physical access to a device or a malicious insider with local keyboard access is outside the remote attack surface entirely. Mitigation: full-disk encryption, tamper-evident physical security, secure boot with measured boot (TPM 2.0), and behavioral monitoring via UEBA (User and Entity Behavior Analytics).

10.4 Why "Effectively Unhackable" Is Technically Defensible

The claim that a ZASA-configured device is "effectively unhackable" with respect to remote attacks is defensible on the following grounds:

14. **All known remote attack categories require at least one of the four preconditions (P1–P4)**, all of which are structurally eliminated by ZASA controls. This is not a probabilistic claim; it is a logical consequence of how TCP/IP and operating systems work.
15. **Unknown (zero-day) attacks still require these same preconditions.** A zero-day exploit against a listening port still requires a listening port. A zero-day C2 technique still requires an outbound connection to an unknown IP. Zero-day attacks do not bypass the structural constraints imposed by ZASA; they exploit vulnerabilities within services or code that ZASA eliminates or constrains.
16. **The residual risks identified in Section 10.3 are supply chain, shared infrastructure, and physical access risks** — not remote exploitation risks against the endpoint's own network stack or execution environment. They are qualitatively different in nature and significantly harder to exploit at scale.
17. **The term "effectively" acknowledges the residual risks** while accurately characterizing the security posture: an attacker targeting a ZASA device remotely must compromise an approved vendor or CDN provider to even attempt an attack against the endpoint — a barrier that eliminates opportunistic attackers entirely and significantly raises the cost for even nation-state actors.

11. Implementation Guide (Summary)

The following phased implementation guide provides a structured path to full ZASA compliance. Each phase should be completed and validated before proceeding to the next.

Implementation Prerequisite

Before beginning implementation, conduct a baseline security assessment to document the current state: all listening ports, all installed remote access services, all outbound network destinations observed over 30 days, and all software installed on endpoints. This baseline is essential for validating that each phase has been completed correctly and for constructing accurate whitelists.

11.1 Phase 1 — Audit and Inventory All Listening Ports

Objective: Identify every service currently exposing a listening port on managed endpoints.

Tools and commands:

Windows (run as Administrator): netstat -an | findstr LISTENING Get-NetTCPConnection -State Listen | Select-Object LocalAddress,LocalPort,OwningProcess Get-Process -Id (Get-NetTCPConnection -State Listen).OwningProcess | Select-Object Name,Id,Path Linux: ss -tlnup netstat -tlnup lsof -i -P -n | grep LISTEN

Actions: For each identified listening port, determine: (1) which service owns it; (2) whether that service is operationally required; (3) whether that service must be exposed to the network or can be restricted to localhost (127.0.0.1). Document findings. Disable all services not operationally required. Restrict remaining services to localhost where possible. Block all remaining listening ports at the host firewall for inbound connections from non-local addresses.

11.2 Phase 2 — Disable Remote Access Services

Objective: Eliminate SSH, RDP, VNC, and WinRM as inbound services on all endpoints.

- **Windows RDP:** System Properties → Remote → Uncheck "Allow Remote Desktop connections." Confirm via `Get-ItemProperty`

`'HKLM:\System\CurrentControlSet\Control\Terminal Server' -Name "fDenyTSConnections"` (value 1 = disabled).

- **Windows WinRM:** `winrm delete winrm/config/listener?Address=*+Transport=HTTP`; set WinRM service to Disabled.
- **Linux SSH:** `systemctl disable sshd; systemctl stop sshd`. Confirm: `ss -tlnup | grep :22` (should return empty).
- **VNC:** Uninstall VNC server software. Confirm no process listens on port 5900.
- **Replace with outbound VPN:** Deploy WireGuard, Tailscale, or equivalent outbound-only VPN client before disabling existing remote access, to ensure management continuity.

11.3 Phase 3 — Implement Application Whitelisting

Objective: Deploy WDAC (Windows) or equivalent in audit mode, then enforce.

- **Step 1 (Week 1–2):** Generate a WDAC baseline policy in audit mode using `New-CIPolicy -ScanPath "C:\" -Level Publisher -Audit -FilePath AuditPolicy.xml`. Deploy to test fleet.
- **Step 2 (Week 2–4):** Review audit log in Event Viewer (Application and Services → Microsoft → Windows → CodeIntegrity) for blocked items. Identify and add legitimate software rules.
- **Step 3 (Week 4–6):** Convert to enforced mode: `Set-RuleOption -FilePath EnforcePolicy.xml -Option 3 -Delete` (removes audit mode flag). Deploy to pilot group.
- **Step 4 (Week 6–12):** Staged rollout across fleet with IT helpdesk monitoring for blocked-application reports. Refine policy iteratively.
- **Restrict LOLBins:** Add explicit block rules for `certutil.exe`, `mshta.exe`, `wscript.exe`, and `cscript.exe` unless required by business processes. Enable PowerShell Constrained Language Mode.

11.4 Phase 4 — Configure Outbound IP Whitelist

Objective: Implement default-deny outbound firewall policy with approved IP allowlist.

- **Step 1:** Deploy host-based firewall in logging mode with a temporary default-allow outbound policy. Collect 30 days of outbound connection logs.
- **Step 2:** Analyze logs. Categorize all observed destination IPs: (a) approved business services, (b) cloud/CDN infrastructure, (c) unknown/personal traffic.
- **Step 3:** Build initial allowlist from category (a) and (b) IPs. Incorporate Microsoft, Google, AWS, and Cloudflare published IP range feeds.
- **Step 4:** Switch to default-deny outbound policy with allowlist in place. Monitor for blocked connections; refine allowlist over 2–4 weeks.
- **Step 5:** Implement automated feed ingestion for cloud provider IP ranges (update weekly at minimum).

11.5 Phase 5 — Deploy Browser Hardening Policies

Objective: Restrict browser behavior to reduce attack surface within the browser's software attack surface.

Deploy via Group Policy (on-premises AD) or MDM (Intune/JAMF) the following Chrome/Edge policies:

- `ExtensionInstallAllowlist` — restrict to approved extension IDs only
- `DeveloperToolsAvailability` = 2 (disable DevTools for all users)
- `SafeBrowsingProtectionLevel` = 2 (Enhanced Safe Browsing)
- `SitePerProcess` = true (strict site isolation)
- `BlockThirdPartyCookies` = true
- `HttpsOnlyMode` = force_enabled (block all HTTP navigation)
- `URLBlocklist` — block `file://` scheme and `javascript:` URIs from external navigation

11.6 Phase 6 — Implement Pre-Connection IP Verification

Objective: Deploy link-click IP verification to prevent navigation to non-whitelisted destinations.

Implementation options (select based on architecture):

- **Browser Extension (Chromium):** Deploy a custom or commercial browser extension that intercepts navigation events (`chrome.webRequest.onBeforeRequest`), resolves destination IPs, checks against a locally cached allowlist, and blocks/warns if IP is not approved.
- **DNS RPZ (Response Policy Zones):** Configure the local DNS resolver to return NXDOMAIN or a sinkhole IP for all domains whose resolved IPs fall outside the allowlist. Requires custom RPZ zone logic.
- **Proxy-Based:** Route all HTTP/S browser traffic through an explicit proxy that performs IP verification before forwarding connections. Squid with a custom ACL, or commercial proxies (Zscaler, Netskope) with IP-based policies.

11.7 Monitoring and Maintenance

- **Weekly:** Review blocked outbound connection logs for false positives (legitimate business applications blocked). Update allowlist as needed. Ingest updated cloud provider IP range feeds.
- **Monthly:** Audit installed software against the application whitelist. Review browser extension list against approved set. Review listening port audit (run `ss -tlnup` / `netstat -an`) to confirm zero non-localhost listeners.
- **Quarterly:** Full whitelist review — remove IPs for decommissioned services; review all Phase 5 browser policy settings against latest browser security guidance.
- **As needed:** When new software is approved for deployment, evaluate its network communication requirements, add required IPs to the outbound allowlist, and update the application whitelist policy before deployment.

12. Conclusion

The security model presented in this whitepaper rests on a principle that is both simple and profoundly consequential: **you cannot exploit what does not exist**. The dominant approach to endpoint security for three decades has been to permit attackers to reach the endpoint and then attempt to stop them. This

paper argues that this approach is architecturally inferior to eliminating the structural conditions that make reaching the endpoint meaningful in the first place.

Zero Attack Surface Architecture is not a marketing concept or a vendor framework. It is the direct application of first principles to the TCP/IP stack, the operating system execution model, and the network architecture of internet-connected devices. By systematically asking "what does every attack require?" and then eliminating those requirements, ZASA achieves a security posture that is qualitatively, not merely quantitatively, superior to detection-based models.

The four controls — zero listening ports, no remote access services, outbound IP whitelisting, and application whitelisting — are individually well-documented and recommended by NIST, NSA, the Australian Signals Directorate, and CIS. Their combined effect, when implemented together with pre-connection IP verification for browser navigation, produces a state in which the four universal preconditions for remote attack (P1–P4) are simultaneously eliminated. No known attack technique survives the removal of all its preconditions.

> *"The principle of least privilege, applied not just to users and software, but to the network stack itself, yields a security posture that no amount of attacker sophistication can defeat, because there is no attack surface to be sophisticated against."*

Call to Action for Security Architects

Security architects and CISOs reviewing this whitepaper are invited to apply the following test to their current endpoint fleet: run `netstat -an | findstr LISTENING` on any managed workstation and count the results. Each line represents a listening socket — an entry point for attack that exists regardless of how many layers of detection have been deployed on top of it. The number of those lines is a concrete, measurable quantity of attack surface. The objective of Zero Attack Surface Architecture is to drive that number to zero.

Future Considerations

- **Post-quantum cryptography:** NIST's post-quantum cryptography standards (FIPS 203, 204, 205 — finalized August 2024) should be incorporated into VPN and TLS implementations. The ZASA

model's structural controls are algorithm-agnostic, but cryptographic implementations within whitelisted applications must evolve to maintain integrity in a post-quantum environment.

- **AI-generated attack content:** Large language model capabilities enable more convincing phishing content and more sophisticated social engineering. The ZASA model's IP-based phishing prevention is immune to content quality improvements — it does not analyze content; it checks IPs.
- **eBPF-based attack vectors:** Extended Berkeley Packet Filter (eBPF) programs running in the Linux kernel represent an emerging attack surface for privilege escalation. Application whitelisting policies must extend to eBPF program loading restrictions.
- **Hardware and firmware attacks:** UEFI implants and baseboard management controller (BMC) vulnerabilities operate below the OS layer. TPM-based measured boot and UEFI Secure Boot remain essential complements to OS-level ZASA controls.

13. References & Further Reading

NIST Publications

- National Institute of Standards and Technology. (2020). *NIST Special Publication 800-207: Zero Trust Architecture*. U.S. Department of Commerce.
- National Institute of Standards and Technology. (2020). *NIST Special Publication 800-53 Rev. 5: Security and Privacy Controls for Information Systems and Organizations*. U.S. Department of Commerce.
- National Institute of Standards and Technology. (2015). *NIST Special Publication 800-167: Guide to Application Whitelisting*. U.S. Department of Commerce.
- National Institute of Standards and Technology. (2012). *NIST Special Publication 800-41 Rev. 1: Guidelines on Firewalls and Firewall Policy*. U.S. Department of Commerce.

Framework References

- The MITRE Corporation. (2024). *MITRE ATT&CK Enterprise Matrix v14*. MITRE ATT&CK. Retrieved from attack.mitre.org.
- Center for Internet Security. (2021). *CIS Critical Security Controls v8*. Center for Internet Security.
- OWASP Foundation. (2023). *OWASP Attack Surface Analysis Cheat Sheet*. OWASP Cheat Sheet Series.
- Australian Signals Directorate. (2024). *Essential Eight Maturity Model*. Australian Cyber Security Centre.

CVE References

- MITRE. CVE-2019-0708 (BlueKeep): Remote Desktop Services Remote Code Execution Vulnerability. NVD CVSS 9.8.
- MITRE. CVE-2017-0144 / MS17-010 (EternalBlue): Windows SMB Remote Code Execution Vulnerability. NVD CVSS 9.3.
- MITRE. CVE-2014-0160 (Heartbleed): OpenSSL TLS Heartbeat Extension Information Disclosure. NVD CVSS 7.5.
- MITRE. CVE-2021-44228 (Log4Shell): Apache Log4j2 Remote Code Execution. NVD CVSS 10.0.
- MITRE. CVE-2024-6387 (regreSSHion): OpenSSH Signal Handler Race Condition Remote Code Execution. NVD CVSS 8.1.
- MITRE. CVE-2021-34527 (PrintNightmare): Windows Print Spooler Remote Code Execution. NVD CVSS 8.8.
- MITRE. CVE-2021-26855 (ProxyLogon): Microsoft Exchange Server SSRF. NVD CVSS 9.8.

Technical Standards and RFCs

- Postel, J. (1981). *RFC 793: Transmission Control Protocol*. IETF.
- Duke, M., et al. (2022). *RFC 9293: Transmission Control Protocol (TCP)*. IETF. (Obsoletes RFC 793).
- Ylonen, T., & Lonvick, C. (2006). *RFC 4253: The Secure Shell (SSH) Transport Layer Protocol*. IETF.

- Klensin, J. (2008). *RFC 5321: Simple Mail Transfer Protocol*. IETF. (DMARC, DKIM, SPF basis).

Microsoft Documentation

- Microsoft. (2024). *Windows Defender Application Control (WDAC) and AppLocker feature availability*. Microsoft Learn.
- Microsoft. (2024). *Windows Defender Application Control Design Guide*. Microsoft Learn.
- Microsoft. (2024). *Microsoft 365 URLs and IP address ranges*. Microsoft Learn. (Machine-readable JSON endpoint feed).

Academic Research

- Howard, M., & Pincus, J. (2003). Measuring Relative Attack Surfaces. *Workshop on Advanced Developments in Software and Systems Security*.
- Manadhata, P. K., & Wing, J. M. (2011). An Attack Surface Metric. *IEEE Transactions on Software Engineering*, 37(3), 371–386.
- Bae, K., et al. (2023). A Systematic Analysis of Application Allowlisting Bypass Techniques. *IEEE Access*.

Appendix A: Quick Reference — Commands for Auditing Listening Ports

Windows

```
# List all TCP/UDP listening ports with process IDs (run as Administrator) netstat -an | findstr LISTENING  # Detailed view with process names (PowerShell, run as Administrator) Get-NetTCPConnection -State Listen |  Select-Object LocalAddress, LocalPort, OwningProcess, @{n='ProcessName';e={(Get-Process -Id $_.OwningProcess).Name}} |  Sort-Object LocalPort |  Format-Table
```

```
-AutoSize  # Include UDP listeners Get-NetUDPEndpoint | Select-Object LocalAddress, LocalPort, OwningProcess |     Sort-Object LocalPort  # Map ports to executable paths Get-NetTCPConnection -State Listen |     ForEach-Object {         $proc = Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue [PSCustomObject]@{             Port    = $_.LocalPort             Address = $_.LocalAddress             PID     = $_.OwningProcess             Name    = $proc.Name Path    = $proc.Path         }     } | Sort-Object Port | Format-Table -AutoSize  # Check for RDP listener specifically Get-NetTCPConnection -LocalPort 3389 -State Listen -ErrorAction SilentlyContinue # Empty result = RDP listener not present (desired state)
```

Linux

```
# List all TCP/UDP listening sockets with process names (requires root) ss -tlnup  # Equivalent using netstat (legacy; may require net-tools package) netstat -tlnup  # Detailed view with inode and process information ss -tlnupe  # List only IPv4 TCP listeners ss -4 -tlnup  # List only IPv6 TCP listeners ss -6 -tlnup  # Find process owning a specific port (example: port 22) ss -tlnup | grep ':22' fuser 22/tcp         # Shows PID fuser -n tcp 22      # Same, explicit protocol  # Using lsof (list open files/sockets) lsof -i -P -n | grep LISTEN  # Check for specific services systemctl is-active sshd openssh-server ssh    # Check SSH systemctl is-active xrdp freerdp # Check RDP systemctl is-active vncserver tigervnc        # Check VNC  # Comprehensive audit one-liner echo "=== LISTENING PORTS ===" && ss -tlnup && \ echo "=== RUNNING SERVICES ===" && \ systemctl list-units --type=service --state=running
```

macOS

```
# List all listening sockets sudo lsof -i -P -n | grep LISTEN  # TCP listeners with process names netstat -an | grep LISTEN  # Using ss equivalent sudo netstat -tlnp
```

Appendix B: Sample Outbound IP Whitelist Policy

The following examples illustrate outbound IP whitelisting rule syntax for common platforms. These are illustrative pseudocode/representative syntax examples; actual IP ranges must be obtained from vendor-published sources and validated before deployment.

Windows Defender Firewall (PowerShell)

```
# Step 1: Set default outbound policy to BLOCK Set-NetFirewallProfile -Profile Domain,Private,Public -DefaultOutboundAction Block  # Step 2: Allow DNS to approved resolvers only New-NetFirewallRule -DisplayName "Allow DNS - Cloudflare" `     -Direction Outbound -Protocol UDP -RemoteAddress 1.1.1.1,1.0.0.1 `     -RemotePort 53 -Action Allow  New-NetFirewallRule -DisplayName "Allow DoH - Cloudflare" `     -Direction Outbound -Protocol TCP -RemoteAddress 1.1.1.1,1.0.0.1 `     -RemotePort 443 -Action Allow  # Step 3: Allow Microsoft 365 IP ranges (sample subset — update from Microsoft JSON feed) # Full list: https://endpoints.office.com/endpoints/worldwide?clientrequestid=... New-NetFirewallRule -DisplayName "Allow Microsoft 365 - Exchange Online" `     -Direction Outbound -Protocol TCP `     -RemoteAddress 13.107.6.152/31,13.107.18.10/31,13.107.128.0/22 `     -RemotePort 443,80 -Action Allow  # Step 4: Allow Windows Update New-NetFirewallRule -DisplayName "Allow
```

```
Windows Update" `  -Direction Outbound -Protocol TCP `  -RemoteAddress 13.107.4.50/32,13.107.5.50/32 `  -RemotePort 443,80 -Action Allow  # Step 5: Allow corporate VPN concentrator (replace with actual IP) New-NetFirewallRule -DisplayName "Allow Corporate VPN" `  -Direction Outbound -Protocol UDP `  -RemoteAddress 203.0.113.10/32 `  # Replace with actual VPN concentrator IP  -RemotePort 51820 -Action Allow  # WireGuard default port  # Step 6: Allow loopback New-NetFirewallRule -DisplayName "Allow Loopback" `  -Direction Outbound -RemoteAddress 127.0.0.0/8 -Action Allow  # All other outbound traffic is blocked by the default BLOCK policy set in Step 1.
```

Linux nftables

```
#!/usr/sbin/nft -f # /etc/nftables-zasa.conf — Zero Attack Surface Architecture outbound policy  flush ruleset  table inet filter {  # Approved outbound IP set — update weekly from vendor feeds  set approved_destinations {  type ipv4_addr  flags interval  elements = {  1.1.1.1/32,  # Cloudflare DNS  1.0.0.1/32,  # Cloudflare DNS secondary  8.8.8.8/32,  # Google DNS  8.8.4.4/32,  # Google DNS secondary  13.107.6.152/31,  # Microsoft 365 (Exchange Online - sample)  13.107.18.10/31,  # Microsoft 365 (Exchange Online - sample)  13.107.128.0/22,  # Microsoft 365 (sample)  20.0.0.0/8,  # Microsoft Azure (sample range — restrict further)  203.0.113.10/32,  # Corporate VPN concentrator — REPLACE  # Add additional approved IPs here  }  }  chain output {  type filter hook output priority 0; policy drop;  # DEFAULT DENY outbound  # Allow loopback  oif lo accept  # Allow established/related connections (response traffic)  ct state established,related accept  # Allow ICMPv4 and ICMPv6 (type-limited)  ip protocol icmp icmp
```

```
type { echo-request, echo-reply } accept ip6 nexthdr icmpv6 accept # Allow outbound to approved destinations only ip daddr @approved_destinations accept # Log and drop everything else log prefix "[ZASA-BLOCK-OUTBOUND] " flags all drop } chain input { type filter hook input priority 0; policy drop; # DEFAULT DENY inbound # Allow loopback iif lo accept # Allow established/related connections (responses to our outbound) ct state established,related accept # All other inbound DROPPED — zero listening port enforcement log prefix "[ZASA-BLOCK-INBOUND] " flags all drop } chain forward { type filter hook forward priority 0; policy drop; } }
```

Appendix C: Glossary of Terms

Term	Definition
Attack Surface	The sum of all points in a system where an attacker can attempt to enter, extract data, or execute code. In network security, this includes all listening ports, accessible services, and executable code paths reachable from untrusted networks.
Application Whitelisting	A security control that permits only pre-approved, cryptographically verified executable code to run on a system, blocking all other code by default. Also termed "application allowlisting" in NIST guidance. The inverse of traditional blacklist/blocklist approaches.
C2 (Command and Control)	The infrastructure and communication channel through which an attacker sends instructions to and receives data from malware running on a compromised host. Also written C&C. C2 frameworks include Cobalt Strike, Metasploit, Covenant, and custom implants.
CIDR (Classless Inter-Domain Routing)	A notation for specifying IP address ranges. Written as an IP address followed by a prefix length (e.g., 192.168.1.0/24 = 256 addresses from 192.168.1.0 to 192.168.1.255). Used in firewall rules to specify ranges of permitted or denied IP addresses.
CVE (Common Vulnerabilities and Exposures)	A standardized identifier (e.g., CVE-2024-6387) assigned to each publicly known cybersecurity vulnerability by the MITRE Corporation under sponsorship of the U.S. CISA. CVE records are published in the NIST National Vulnerability Database (NVD) with severity scores (CVSS).

Term	Definition
CVSS (Common Vulnerability Scoring System)	A framework for rating the severity of security vulnerabilities on a scale of 0.0–10.0, with qualitative labels: Low (0.1–3.9), Medium (4.0–6.9), High (7.0–8.9), Critical (9.0–10.0). Published by FIRST.org.
Listening Port	A network port for which a process has called the listen() system call, placing a socket in LISTEN state. The operating system will accept inbound TCP SYN packets destined for this port and hand them to the owning process. A listening port is the primary entry point for inbound network attacks.
LotL / LOLBins (Living Off the Land)	An attack technique that uses legitimate, pre-installed operating system utilities (LOLBins — Living Off the Land Binaries) to execute malicious actions, avoiding the need to drop new executable files. Examples: certutil.exe, mshta.exe, wscript.exe, rundll32.exe, regsvr32.exe, powershell.exe.
Outbound Whitelist	A firewall policy that restricts outbound network connections to a pre-approved set of IP addresses or CIDR ranges, denying all other outbound traffic by default. Also called an egress filter or outbound allowlist.
RDP (Remote Desktop Protocol)	A proprietary Microsoft protocol (TCP port 3389) that allows users to connect to and interact with remote Windows desktops. Developed by Microsoft; based on the T.128 application sharing protocol. One of the most frequently exploited protocols on the public internet.
SSH (Secure Shell)	A cryptographic network protocol (TCP port 22) for secure remote login, command execution, file transfer (SFTP), and port forwarding. Defined in RFC 4253. Commonly implemented by OpenSSH. Represents a significant attack surface when exposed to untrusted networks.
VNC (Virtual Network Computing)	A graphical desktop sharing system based on the Remote Framebuffer (RFB) protocol, typically listening on TCP port 5900. Transmits keyboard/mouse input and screen updates between client and server. Has a weaker security history than SSH or RDP; default authentication uses 56-bit DES.
WDAC (Windows Defender Application Control)	A Windows feature (available since Windows 10 1709) that enforces code integrity policies at the kernel level, permitting only digitally signed and explicitly authorized software to run. The recommended application whitelisting technology for Windows environments. Enforced by the kernel's Code Integrity (CI) component and cannot be disabled by user-mode processes.
Zero Trust Architecture (ZTA)	A security paradigm defined in NIST SP 800-207 that eliminates the concept of implicit trust for any user, device, or network segment. Every access request is verified continuously regardless of its origin. Key principles: verify explicitly, use least privilege access, assume breach.
Zero Attack Surface Architecture (ZASA)	As defined in this whitepaper: an endpoint security model that structurally eliminates the four universal preconditions for remote attack — inbound listening ports, remote access services, outbound connectivity to unknown IPs, and the ability to execute non-approved binaries. Achieves near-theoretical immunity to remote exploitation by eliminating attack preconditions rather than detecting attacks after they begin.
Three-Way Handshake	The TCP connection establishment process defined in RFC 793/9293, consisting of three messages: SYN (client requests connection), SYN-ACK (server acknowledges and offers

Term	Definition
	sequence), ACK (client acknowledges). Requires a listening socket on the server side to process the initial SYN. If no listening socket exists, the server returns TCP RST.
NGFW (Next-Generation Firewall)	A network security device that extends traditional stateful firewall capabilities with deep packet inspection, application-layer filtering, intrusion prevention, TLS inspection, and threat intelligence integration. Examples: Palo Alto Networks PA series, Fortinet FortiGate, Cisco Firepower.
DMARC / DKIM / SPF	Email authentication protocols: SPF (Sender Policy Framework) specifies authorized sending IPs; DKIM (DomainKeys Identified Mail) provides cryptographic message signing; DMARC (Domain-based Message Authentication, Reporting and Conformance) defines policy for handling messages failing SPF/DKIM checks. Together they reduce email spoofing and phishing.
DoH (DNS over HTTPS)	A protocol (RFC 8484) that performs DNS resolution over an HTTPS connection to a DoH server, encrypting DNS queries against passive interception and preventing DNS spoofing by network intermediaries. Recommended for use with approved DoH providers only in the ZASA model.

This document is provided for informational and educational purposes. Security architectures should be validated by qualified security professionals prior to deployment. All CVE data is sourced from the NIST National Vulnerability Database (NVD). Framework references are to publicly available documents.

Produced using NIST SP 800-53 Rev. 5 • NIST SP 800-207 • MITRE ATT&CK v14 • CIS Controls v8 as reference frameworks.