THE FILENAME AS THE CONTROL PLANE

An Unprotected Trust Surface in Enterprise Systems

Abstract v1.1

Filename Control Plane Security protects the invisible architecture of enterprise systems by cryptographically binding a file's identity directly to its content and designation.

VeraSec Technologies

About VeraSec Technologies

VeraSec Technologies is a pioneer in Artifact Identity Security (AIS), focused on moving the security context from the infrastructure directly to the file itself. While traditional security models rely on centralized systems to establish and maintain trust, VeraSec's patent-pending technologies empower files to carry their own verifiable identity - preparing them for an actual Zero Trust world.

Operating at the forefront of Portable File Identity (PFI) technology, VeraSec invented the Validation Integrity Key (VIK) process. This innovation patches a fundamental vulnerability in traditional PKI that has persisted for decades. By cryptographically binding a file's content and metadata into a compact, human-readable identifier embedded directly within the artifact, VeraSec helps organizations set up durable, portable trust in every digital asset, signed or un-signed - wherever it travels.

About The Author

Brian Hajost is a cybersecurity entrepreneur and technologist with more than 30 years of experience in information security, compliance automation, and trusted computing systems.

As the founder of SteelCloud, he pioneered the automation of DISA STIG and CIS benchmark compliance for the DoD, federal agencies, and enterprise customers. Brian holds 15 patents in information technology and mobile security. In 2022 he was named one of the "Top 10 Most Influential People in Cyber Security" by CIO Views.

His current work focuses on solving what he describes as the Artifact Identity Problem - the absence of durable identity for files as they move across distributed infrastructures and untrusted networks.

Trademarks

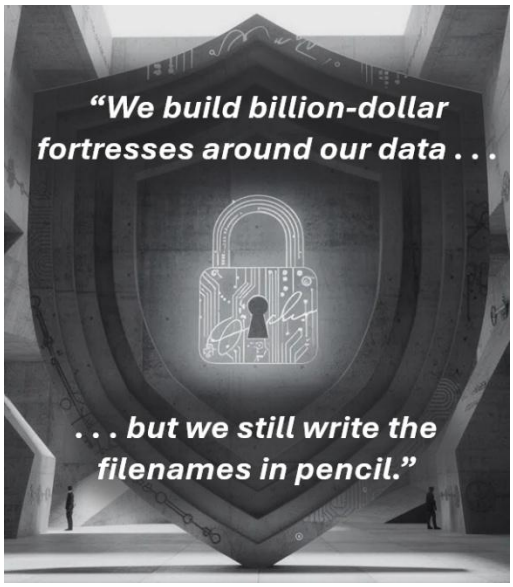
The Filename as the Control Plane

An Unprotected Trust Surface in Enterprise Systems

Executive Summary

Since the dawn of modern computing, an unspoken dependency has quietly taken hold: *the filename*. What originated as a simple, human-readable label has evolved into something far more consequential. In contemporary enterprise systems, filenames are routinely parsed, interpreted, and acted upon as authoritative signals that determine routing, classification, execution, and operational context. Systems frequently make critical decisions about how a file should be handled before its contents are ever inspected - and often without any validation of the file itself.

Despite this elevated role, the filename has never been treated as a security boundary. It remains fully mutable and unauthenticated, yet implicitly trusted. This disconnect, between operational reliance and security treatment has created a structural vulnerability spanning industries, architectures, and workflows.



This paper examines how the filename has effectively become a control plane across enterprise systems, why it has remained unprotected, and what this implies for data integrity. It further explores how **VeraFile™** addresses this gap by transforming the filename from untrusted metadata into a cryptographically bound component of a file's identity. The conclusion is not speculative: the filename is already a primary input to critical systems. The only unresolved question is whether it will remain an unprotected one.

Introduction: *A Dependency Hidden in Plain Sight*

Modern security architectures rigorously focus on systems, identities, and data states. Controls are applied to networks, endpoints, and storage layers, while files themselves are protected through encryption, access control, and digital signatures. These mechanisms ensure data is protected at rest and in motion.

Yet, in the day-to-day operation of enterprise systems, another variable exerts massive influence - one rarely acknowledged in formal security models: ***the filename***.

Filenames are not passive descriptors. They encode meaning, convey intent, and frequently dictate how a file is handled within a workflow. A file labeled as "final" is treated differently from one labeled "draft." A release artifact may be promoted into production pipelines, while another is quarantined, based entirely on its naming convention. These behaviors occur automatically, at scale, and typically without any content inspection.

This reliance was rarely documented as a formal design decision; rather, it emerged organically. The filename has become a lightweight, universally accessible signaling mechanism across platforms and boundaries.

In practice, it functions as a control plane.

The Filename as a Control Plane

In classical systems architecture, the control plane directs how operations occur, determining routing decisions, enforcing policies, and governing execution flow. These control planes are explicit, structured, and strictly secured.

The filename, however, represents an **implicit control plane**. Though never designed for this role, widespread operational use has forced it to assume the responsibility. Across environments, filenames guide files into pipelines, signal completion states, encode metadata *(time, region, customer IDs)*, and trigger automation.

What is striking is the consistency of this behavior: the filename is interpreted *before* the file is validated. This inversion, where control decisions precede trust validation - introduces a structural weakness, allowing unverified inputs to influence critical operations.

The Trust Gap

The filename occupies a unique and problematic position: functionally critical, yet structurally unprotected. Unlike file contents, which may be encrypted or digitally signed, the filename remains fully mutable. It is not cryptographically bound to the data it represents and can be altered without a trace, and without breaking the underlying file's integrity.

> The Core Disconnect: Systems act on filenames as if they are authoritative, while security models treat them as untrusted and irrelevant.

The consequences are pervasive. A file can be renamed to alter its perceived purpose. A malicious artifact can be introduced under a trusted naming convention. A legitimate file can be modified without changing its name, preserving the illusion of continuity. Because these actions occur within normal operational bounds, they bypass traditional security alerts. They do not exploit code vulnerabilities; they exploit our assumption that a filename is a reliable indicator of intent.

Why This Went Unaddressed

The lack of protection around filenames stems from an error in categorization rather than negligence.

Historically, filenames were classified as metadata - useful for casual human interaction, but unsuitable for security decisions. The result is a divergence between how systems are designed to be secure and how they actually operate. Security models assume that filenames are untrusted and therefore irrelevant. Operational systems assume that they are meaningful and actionable. Consequently, filenames were excluded from trust models by design, while security mechanisms were built for file contents.

Simultaneously, operational engineers seeking efficiency began relying on filenames as convenient signals. Parsing a filename is exponentially faster than inspecting file contents. Over time, this convenience calcified into a multi-faceted dependency.

Both assumptions are incomplete, and together, they have created a massive blind spot.

The Operational Reality

In real-world environments, the filename is often the first, and sometimes the only piece of information available upon encountering a file. Whether received via email, downloaded from a portal, or transferred across a supply chain, the filename shapes the initial perception and handling of the data.

This is particularly evident in scenarios where files cross organizational boundaries, where validation infrastructure is not shared, or where users lack specialized tools. In these contexts, the filename becomes the primary interface between the file and its consumer.

If this initial interface is untrusted, the entire subsequent interaction can be compromised.

Use Cases: The Filename in Operational Control

The risk becomes starkly apparent across real enterprise workflows where filenames drive behavior:

- **Software Development:** Filenames communicate artifact states (e.g., stable, production-ready, version-specific) in CI/CD pipelines. Deployment decisions are frequently made based on these signals, inviting substituted artifacts into trusted environments.
- **Supply Chain:** Files move between organizations lacking shared trust infrastructure. Downstream systems or partners accept production artifacts based solely on their names. Trust is inferred, not established.
- **Batch Processing:** Filenames serve as execution triggers. Pipelines depend on naming patterns to initiate jobs, sequence operations, and select data sets. A maliciously renamed file can trigger unintended, cascading execution paths.
- **Data Platforms:** In modern data platforms, such as Amazon S3, Azure Data Lake, or Apache Spark, directory structures and filenames dictate data partitioning. Query engines rely on these names; manipulated filenames compromise the integrity of analytical processes. When filenames are incorrect or manipulated, the integrity of the entire analytical process is compromised.
- **Human Workflows:** At the human level, filenames serve as the primary trust signal in everyday interactions. Users encountering files in email, downloads, or shared drives rely on names to assess legitimacy and intent. This makes filenames a central component of social engineering attacks, where the goal is not to exploit system vulnerabilities, but to manipulate human perception.

- **High Trust Environments:** Even in high-assurance environments, such as cross-domain or air-gapped systems, the filename remains one of the few attributes that persists across boundaries. In the absence of shared validation infrastructure, it becomes a default proxy for trust, despite having no inherent mechanism to justify that trust.

Across all of these scenarios, the pattern is consistent. The filename is treated as authoritative input, yet it lacks any intrinsic means of validation.

Control Plane Gap and MITRE ATT&CK Alignment

The MITRE ATT&CK Framework models adversary operations but largely assumes files are correctly interpreted at the point of interaction. In reality, techniques across Initial Access, Execution, and Defense Evasion routinely rely on manipulating filenames to convey legitimacy or blend into trusted workflows.

Because filenames are unbound to content, attackers bypass controls that validate payloads but ignore designations. Systems act on the name first; validation happens later - at best. Securing the filename control plane closes this gap, removing the ambiguity that ATT&CK techniques implicitly exploit.

(See MITRE ATT&CK Appendix for detailed mapping)

The Solution: *Verifiable Identity at the Control Plane*

As enterprises invest heavily in Zero Trust architectures, they continue to rely on unverified signals to drive system and user behavior. A new category of security technology - **Filename Control Plane Security** - is emerging to address this.

This technology binds file identity to both content and designation, enabling systems to explicitly trust the inputs driving their operations.

VeraFile and the Securing of the Control Plane

VeraFile addresses this architectural blind spot by redefining the filename's role. Rather than treating it as mutable metadata, VeraFile incorporates the filename directly into the file's identity.

It binds the contents and the name into a single cryptographic construct. This is achieved by deriving a compact identifier - a **Validation Integrity Key *(VIK)***, from both elements and embedding that identifier within the filename itself. The result is a composite structure where the filename is inextricably linked to the file it represents.

The implications are immediate:

1. A file can no longer be renamed without detection.
2. File contents cannot be altered without invalidating the identity.
3. The filename becomes both a routing signal and a verifiable security surface.

For the first time, the filename as an implicit control plane, is structurally secured.

Architectural Implications

Recognizing and securing the filename as a control plane challenges long-standing assumptions about enterprise architecture:

- **Reframing the Location of Trust:** Files traditionally inherit trust from their environment *(infrastructure, identity providers, etc.)*. Once they move, trust degrades. VeraFile shifts this dynamic.
- **Aligning Security with Reality:** Preventing untrusted inputs from influencing trusted routing and automation processes aligns security models with how operational workflows actually function.
- **Expanding File Identity:** A file is defined not just by its data, but by its designation. Binding these two creates a true, composite identity.
- **Portable Integrity:** By incorporating validation into the artifact itself, integrity becomes independent of shared infrastructure. Trust follows the artifact – everywhere it goes.

The Archival Challenge: *The Decay of Trust Over Time*

The reliance on infrastructure for file security introduces a severe vulnerability in long-term data retention: **the decay of trust over time**.

Most traditional file protection mechanisms are inherently short-term. Digital signatures rely on Public Key Infrastructure *(PKI)*, where certificates expire, revocation lists vanish, and cryptographic standards deprecate. Consequently, long-term archival systems are forced to rely on "infrastructure trust" - the assumption that the storage environment itself remained perfectly secure, consistent, and unmanipulated over years, or even decades. Validating that historical assumption is practically impossible.

When trust is tied to the environment, moving or archiving a file eventually strips it of its verifiable integrity. VeraFile's approach, binding a cryptographic identifier *(the VIK)* directly into the file and its name, creates **temporal resilience**. Because trust is intrinsic to the artifact rather than dependent on active, external infrastructure, a file stored today can be independently and definitively validated twenty years from now, regardless of the system it resides in.

Conclusion

The filename has evolved into a critical component of enterprise operations, functioning as an implicit control plane across systems and boundaries. Yet it remains unprotected, creating a fundamental misalignment between how systems operate and how they are secured.

VeraFile' patent-pending technology resolves this by binding the filename and file contents into a single, verifiable identity, securing a trust surface that has long been relied upon, but never validated.

Systems already depend on filenames to make critical decisions. That dependency is not going away. The only question that remains is: ***Will those decisions continue to be made on unverified signals - or on trust that can be proven?***

File Security Methods vs. Filename Treatment

Method / Technology	Primary Purpose	How It Secures the File	Treatment of Filename	Gap / Limitation
Hashing (SHA-256/512)	Integrity verification	Computes hash of file content and compares later	❌ Ignored entirely	No linkage between file and name; rename undetectable
Digital Signatures (PKI)	Identity + integrity	Signs file content with private key	❌ Ignored (not in PKI scope)	File can be renamed or repackaged without detection
Code Signing (Software / Firmware)	Trust in executables	Verifies publisher identity and integrity of binary	❌ Ignored	Filename spoofing still possible; relies on user/system context
Checksums - MD5 / CRC	Basic integrity	Lightweight hash comparison	❌ Ignored	Same limitations as hashing, weaker assurance
File Integrity Monitoring (FIM)	Detect system changes	Tracks file changes on monitored systems	⚠️ Indirect (path-based)	Works only within system boundary; not portable
SIEM / Log Monitoring	Event analysis	Aggregates and analyzes logs	⚠️ Context only	Trusts filename as metadata; not validated
Encryption (at rest / in transit)	Confidentiality	Protects file contents from unauthorized access	❌ Ignored	Does not protect integrity of name or identity
Secure Transport (TLS, SFTP)	Secure delivery	Protects file during transmission	❌ Ignored	Protection ends after delivery
ZIP / Container Signing	Package integrity	Signs archive or container	❌ Ignored	Internal filenames can vary (non-canonical)
Content Management Systems	Document governance	Tracks versions and metadata	⚠️ Managed internally	Filename trust lost when file leaves system
SBOM / Manifest Files	Dependency integrity	External list of file hashes	❌ Separate artifact	Linkage between file and manifest can break
Blockchain / Ledger Anchoring	Immutable record	Stores hash externally	❌ Ignored	Still external; no binding to filename
Version Control Systems (Git)	Change tracking	Tracks content changes via hashes	❌ Ignored (filenames)	Rename operations do not affect content hash
VeraFile (Artifact -Centric Integrity)	Portable file integrity	Cryptographically binds file content + filename into a VIK	☑ **Integral part of trust anchor**	None of the above gaps; detects rename + content change

Addendum: MITRE ATT&CK Framework for Files

Tactic	Technique (ID)	How Filenames Are Exploited	Control Plane Gap	How VeraFile Disrupts the Technique
Initial Access	Spearphishing Attachment (T1566.001)	Malicious files use trusted naming (e.g., *invoice_final.pdf*) to induce opening	User/system trusts filename before validation	Filename-content binding exposes mismatch immediately; prevents masquerade
	Spearphishing Link (T1566.002)	Downloaded files inherit trusted or misleading names	Filename becomes first trust signal	Post-download validation ensures file matches its claimed identity
	Supply Chain Compromise (T1195)	Malicious files inserted with legitimate naming conventions	Downstream systems trust designation	Any substitution or rename invalidates identity
Execution	User Execution (T1204)	User decision driven by filename context	Filename acts as psychological trigger	Users can validate identity prior to execution
	Command and Scripting Interpreter (T1059)	Scripts named to appear benign or expected	Automation trusts naming patterns	Scripts cannot masquerade as trusted artifacts
Persistence	Boot or Logon Scripts (T1037)	Malicious scripts adopt expected naming patterns	Systems load scripts based on name/location	Unauthorized script identity detectable immediately
	Scheduled Task/Job (T1053)	Tasks reference files by expected filenames	Execution assumes filename legitimacy	Task input validation ensures file integrity + designation match
Privilege Escalation	Exploitation for Privilege Escalation (T1068)	Malicious payloads disguised as trusted system components	Filename influences trust level	Payload identity mismatch prevents trusted execution context
Defense Evasion	Masquerading (T1036)	Renaming files to appear legitimate (core technique)	Filename fully trusted, unverified	Renaming breaks identity - masquerade fails instantly
	Obfuscated/Compressed Files (T1027)	Filename used to hide true nature of file	Detection relies on filename cues	Identity validation bypasses obfuscation tricks
	Indicator Removal / Renaming (T1070)	Files renamed to evade detection or tracking	Tools rely on name continuity	Any rename invalidates identity; tampering visible
Credential Access	Input Capture / Credential Files (T1056)	Fake files named to collect credentials (e.g., login forms, docs)	User trusts filename context	Identity mismatch reveals malicious artifact
Discovery	File and Directory Discovery (T1083)	Attackers identify naming patterns to blend in	Naming conventions become attack surface	Enforced identity removes ability to blend via naming
Lateral Movement	Remote Services / SMB (T1021)	Files shared across systems with trusted names	Receiving system trusts filename	Cross-system validation ensures file integrity regardless of source
	Replication Through Removable Media (T1091)	Malware spreads using trusted filenames	No shared validation across environments	Portable identity allows validation in disconnected systems
Collection	Data from Local System (T1005)	Files named to influence classification or handling	Filename used for prioritization	Ensures classification signals are accurate
Exfiltration	Exfiltration Over Web Services (T1567)	Files renamed to avoid detection or misclassification	Controls rely on naming/context	Renamed files fail validation, exposing manipulation
Command & Control	Ingress Tool Transfer (T1105)	Tools delivered with benign filenames	Trust based on naming convention	Delivered tools cannot impersonate trusted files
Impact	Data Manipulation (T1565)	Files modified without renaming to preserve trust	Filename assumed to represent original state	Content change invalidates identity immediately
	Defacement / Data Destruction (T1491/T1485)	Files replaced but retain trusted names	Systems continue to trust designation	Replacement detectable via identity mismatch