Engineering Safer Content Flows

Content Disarm and Reconstruction Through a Systems Security Engineering Lens

Dr. Ron Ross

CEO, RONROSSECURE, LLC
ISTS FELLOW, DARTMOUTH COLLEGE

Abstract

Content Disarm and Reconstruction (CDR) offers a practical way to reduce dangerous capabilities in files before people or applications use them. It can remove selected active content and reconstruct permitted information without first recognizing a particular attack. CDR illustrates how systems security engineering (SSE) connects a technical mechanism to mission protection needs. Its value, however, depends on the protection needs, policies, architecture, and evidence surrounding the mechanism. Written for security architects, systems security engineers, and program owners, this article examines CDR through Least Functionality, Mediated Access, Domain Separation, and related security design principles; develops an illustrative maintenance-document workflow; identifies failure modes of CDR itself; and proposes assurance claims and a focused pilot. The objective is to build justified confidence that required information reaches its destination while prohibited capabilities and unacceptable consequences remain constrained. File reconstruction alone does not guarantee information accuracy, sender legitimacy, or mission security.

1. The Protection Problem

An organization may need the information in a document without needing every capability packaged inside it. A maintenance instruction can contain text and diagrams that technicians require, together with scripts, embedded objects, or external references they do not need. Allowing the complete original into a sensitive environment can therefore grant more opportunity for harmful behavior than the mission requires.

Many conventional file-security tools examine files for evidence of maliciousness. That remains useful, particularly for investigation and response. CDR adds a different protection opportunity: transform incoming content so that selected capabilities are absent before downstream applications process it. The organization can make that decision because a capability is unnecessary or prohibited, even when no tool identifies a particular malicious payload.

The important engineering question is what the receiving system must be able to do with the information. That question connects content processing to stakeholder protection needs. It also exposes the difference between buying a file-security product and building an enforceable boundary around a consequential workflow.

A mission may need the information in a file without needing the capabilities that travel with it.

---

2. What CDR Contributes

CDR implementations differ, but a common approach identifies the actual file format, parses its structure, applies policy to its constituent elements, and produces a reconstructed output. Depending on the

© 2026 RONROSSECURE, LLC | CC BY 4.0.

implementation and configuration, processing can remove macros, scripts, embedded objects, or other unwanted features and repair or reject nonconforming structures. Some approaches preserve the original format; others produce a simpler representation. These choices affect both protection and usability [1][2]. The NCSC import pattern similarly describes transforming complex formats into simpler ones, removing active content such as macros and scripts, and rebuilding the original format after verification where the destination requires it [11].

The relevant distinction is between recognizing an attack and removing the capability an attack could use. A policy that prohibits document macros need not determine whether each macro is malicious before excluding it. This can reduce dependence on identifying known attack signatures for the capabilities that reconstruction actually removes. It does not eliminate the need for correct parsing, adequate format coverage, or secure downstream applications.

Claims about a reconstructed file should state exactly what has been shown. Structural conformity indicates compliance with specified format rules. Content policy determines which features are permitted. Neither shows that every permitted feature is harmless in every receiving application. The engineering claim must account for the destination, its software, and the operations the user or service will perform.

The following CDR failure modes can undermine the protection claim and should guide design and verification.

- **Format coverage.** A service can enforce content-specific policy only on supported formats. Withholding is the default for unsupported formats, including rare or revised formats outside the supported profile. Any alternative handling follows a separately authorized, controlled process; the original is not automatically passed through.
- **Format identification.** A polyglot is valid in two or more formats. A 2025 study found that existing identification and embedded-file detection tools did not reliably detect polyglots used in the wild [12]. Different interpretations by processing and destination applications can undermine policy. NCSC advises robust, consistent format verification at every step [11]. The study's research CDR tool sanitized all tested image-based polyglots; this result does not establish protection for all formats [12].
- **Parser compromise.** The transformation engine processes complex untrusted input and may itself be vulnerable. NCSC therefore places transformation on the less trusted side and assumes the engine could be compromised [11].
- **Nested content.** NCSC advises unpacking embedded objects, transforming and verifying them, and limiting nesting and recursion [11].

Section 8 links these failure modes to the evidence needed to support the protection claims.

CDR should operate within a broader protection architecture. OWASP recommends combining applicable CDR with file-type validation, authorization, protected storage, processing limits, and antivirus or sandbox analysis where available [9]. The NCSC pattern likewise combines transformation with verification, flow control, patching, and sandboxing rather than relying on any single control [11]. Endpoint protection and monitoring address additional exposure at the destination. The organization should select complementary safeguards against its identified losses and verify their combined behavior.

3. Commercial Availability and Mission Suitability

CDR is commercially established: Menlo Security announced its acquisition of Votiro in February 2025 [3], and OPSWAT documents Deep CDR as a commercial technology [2]. Availability and investment show that products exist; they do not show that a product is effective in a particular organization.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

Suitability is a workflow question. Licensing is only part of the cost: integration, user support, exception handling, policy maintenance, and re-evaluation after software changes determine whether protection stays sustainable, and a control that pushes users toward an uncontrolled transfer path undermines its own purpose.

Evaluation should therefore begin with representative files, run through the destination applications that will use them, because clinical documents, customer uploads, and maintenance instructions differ in fidelity, confidentiality, latency, and permitted features. The decision is whether a defined workflow gains useful protection at acceptable operating cost; the broad claim that CDR makes documents safe is too imprecise to support it.

4. The Strongest Connections to SSE

NIST SP 800-160, Volume 1, Revision 1 provides principles for trustworthy secure design [4]. The following connections are an engineering interpretation of CDR, rather than a claim that NIST prescribes a particular product or reconstruction method.

- **Defense in Depth** provides an overarching architectural context for applying CDR. No individual mechanism can be assumed to provide perfect protection. Engineers select and combine security design principles according to mission protection needs, unacceptable losses, and explicit compromise assumptions. Coordinated mechanisms provide opportunities to prevent loss, constrain its escalation, and contain consequences when individual mechanisms fail or are compromised [4]. Their mutual reinforcement must be demonstrated: bypass paths or shared dependencies can defeat apparently separate layers together.
- **Least Functionality** reduces the capabilities carried by incoming content. The content profile preserves functions required for the receiving task and removes unnecessary capabilities [4][5]. Policies should reflect the task and destination; a finance workflow may require features that a maintenance-reference workflow does not. Product policy controls support such distinctions [1]. Authentication or group membership alone should not justify retaining active content. Exceptions should identify the required capability, safeguards governing its use, and its consequences.
- **Mediated Access** governs content passage through policy enforcement before it can affect the destination [4][6]. Email, supplier portals, removable media, alternate APIs, and emergency transfers must be covered by non-bypassable enforcement of the applicable policy.
- **Domain Separation** isolates hostile parsing and reconstruction from protected use. Separate processing, storage, and administrative domains should prevent compromise on the untrusted side from conferring authority over the protected destination [4].
- **Least Privilege** limits the information, interfaces, and operations available to processing services and administrators. A CDR service should have only the authority needed to receive, transform, and submit content for verification; it should not acquire operational credentials or direct control of the destination [4].
- **Least Persistence** limits how long temporary originals, extracted objects, and processing credentials remain available for their authorized purpose [4][13]. Define when processing artifacts are removed and temporary authority expires. Originals retained for evidence or records requirements require separately protected storage and explicit retention rules. Verify cleanup after successful processing, rejection, failure, and recovery.
- **Distributed Privilege** can prevent one compromised administrator from unilaterally weakening consequential content policies or authorizing exceptional release. Where the consequences warrant it,

© 2026 RONROSSECURE, LLC | CC BY 4.0.

require coordinated authorization from independently controlled entities, with evidence that neither can bypass the required agreement [4].

- **Protective Failure** requires the workflow to preserve its protection constraints when processing is uncertain or fails. Unsupported or ambiguous formats, failed verification, timeouts, and overload should lead to withholding or controlled resolution rather than automatic delivery of the original. Define a governed alternative for urgent mission needs and test the failure behavior [4].
- **Compositional Trustworthiness** connects the processing stages to the system-level claim. Evidence that an individual parser, verifier, or release service works correctly does not establish that their integration preserves the intended protection. Analyze interfaces, bypass paths, policy consistency, and shared dependencies, and verify that the complete workflow satisfies the stated assumptions and constraints [4][8].

These design principles reinforce one another when their implementation preserves protection across the complete workflow. In the Section 5 example, *Least Functionality* permits only the text, approved diagrams, and specified tables technicians need. Requests for live hyperlinks or embedded spreadsheets require justification. For this illustrative architecture, *Domain Separation* is implemented through a separate processing host, credentials, and storage, with shared dependencies analyzed to ensure they do not undermine that separation. *Mediated Access* extends the project beyond an email gateway to every in-scope delivery path. Each design principle changes a concrete design decision. The table summarizes the resulting design obligations and the evidence that would show each one is met.

Table 1. Design principles, obligations, and evidence.

Principle	CDR design obligation	Example evidence
Defense in Depth	Coordinate barriers to prevent or contain loss when a mechanism fails	Loss-scenario, bypass, and common-dependency analysis
Least Functionality	Exclude features unnecessary for the receiving task	Output inspection and required-task testing
Mediated Access	Govern every in-scope content transfer	Transfer inventory and attempted bypass tests
Domain Separation	Isolate original-file processing from protected use	Interface and dependency analysis
Least Privilege	Bound processing and administrative authority	Permission review and hostile-service tests
Least Persistence	Limit artifact retention and temporary authority to their required duration	Cleanup, credential-expiry, and retention-policy tests
Distributed Privilege	Require coordinated authorization for consequential policy changes or exceptions	Approval-path and single-administrator bypass tests
Protective Failure	Withhold delivery when acceptance conditions fail	Timeout, overload, verification-failure, and unsupported-format tests
Compositional Trustworthiness	Preserve protection claims across integrated processing stages	Integration tests and review of cross-stage assumptions

5. A Maintenance Document Entering a Sensitive Environment

Consider an illustrative facility that receives maintenance instructions from equipment suppliers. Technicians need approved text, diagrams, and specified tables on an engineering workstation. The facility has identified unauthorized changes to control logic and materially incorrect maintenance information as unacceptable outcomes. The example is a proposed design exercise, not a report of a tested deployment.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

The first engineering decision is to separate document receipt from operational authority. Suppliers submit files to an intake service outside the protected operational environment. That service has no authority to change controllers, access operational credentials, or write directly to control-system storage. Original files remain in a restricted holding area, accessible only through separately governed procedures.

A processing service evaluates the submitted file against the maintenance workflow policy. The policy defines supported formats and permitted features, including how nested files and external references are handled. Reconstruction produces an approved representation without prohibited active content. Files that cannot be processed within defined limits are withheld for controlled resolution. A change in filename or extension cannot substitute for format identification. Because polyglot files can defeat format identification [12], a file whose format cannot be determined unambiguously is treated as unsupported and withheld.

A separately protected verification service, sharing no parser, credentials, or administrative authority with the processing service, checks each reconstructed output against the policy. The release service delivers only accepted output and records its relationship to the original, the policy version, and the processing result. A hash can identify which bytes were processed or delivered; it cannot show that their meaning is correct. Destination access rules prevent users from retrieving the original through the normal maintenance workflow. An exception uses a separate, explicitly authorized process whose security consequences are assessed.

Where investigation or records requirements warrant retention, preserve an unmodified original in separately protected storage and analyze working copies. NIST forensic guidance recommends examining copies and verifying their integrity [10]. Applied to this workflow, maintain identifiers, cryptographic hashes, processing records, and access history linking the original to its reconstructed output. An authorized request to examine an original should use an isolated environment and should not automatically authorize its release to the protected destination. Define retention and disposal rules in advance.

Technicians still verify that the instruction applies to the correct equipment and revision. Consequential maintenance changes follow independent authorization and engineering checks. These measures address the second unacceptable outcome: misleading instructions can survive reconstruction because their harmfulness lies in their meaning rather than executable features. CDR contributes to the file-processing constraint; other safeguards address content authenticity and operational decisions.

Verification should trace the whole path. Can a supplier place a file directly in the destination share? Can a user retrieve the quarantined original? Does a timeout cause release? Does a file that is valid in two formats reach release? Can the processing service acquire operational credentials? Does the reconstructed diagram retain labels and units? Each test addresses an explicit condition that could defeat protection or required use.

The acceptance test must show both that prohibited capability is absent and that required information remains usable.

6. Engineering for Compromise of the Protection Service

The CDR service is itself part of the system that must be protected. Because it processes untrusted content and influences what reaches the destination, its compromise must be an explicit design assumption. The objective is to constrain the losses that compromise could cause. The following measures apply that objective to processing stages, administrative policy, and failure behavior.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

- **Assume the service is hostile.** *Engineering for Compromise* distinguishes a compromised system element from the unacceptable losses its authority and connections could enable [7]. Applied here, the question is what becomes possible if the CDR service is already hostile. A service that can manufacture arbitrary output and deliver it to every user may create a concentrated source of exposure even while reducing threats from external files. The NCSC import pattern makes the same assumption: it presumes the transformation engine could be compromised and arranges the later stages so that taking over that engine alone does not reach the destination [11].
- **Separate the stages.** The design should distinguish ingestion, parsing, output checking, release, and administration according to their consequences. The Section 5 design places a separately protected verification service between processing and release. Its value depends on its independence, coverage, and susceptibility to common defects. Reusing the same parser and administrative authority may reproduce the same failure in a second stage. The NCSC pattern explains that verification may be limited to one simple, tightly controlled format, reducing the complexity of the verification task. Verification failures can indicate malfunction or compromise and should prompt operator investigation [11].
- **Protect the policy.** Compromise analysis should test whether a hostile processing service can change content policy, enable original-file release, or redirect delivery. Policy administration and exception approval must remain outside that service's authority, including through shared credentials or management interfaces.
- **Fail protectively.** A compromised transformation engine may fabricate a success result. The release decision must depend on independently established acceptance conditions, with failures or inconsistent results withholding delivery and alerting an operator [11].

7. Boundaries on the Protection Claim

A credible protection claim must state what CDR can establish and what remains outside its scope. Evidence that prohibited file capabilities have been removed supports a defined technical claim; broader conclusions require additional safeguards and evidence. The following limitations address the meaning of retained content, its use by AI systems, and the preservation of information and provenance.

- **Content accuracy remains unverified.** A reconstructed document can still contain false statements, fraudulent payment instructions, or harmful operational advice. Removing executable features does not confirm the truth or authority of the remaining information. Source authentication and independent confirmation of consequential requests remain necessary where those losses matter.
- **AI consumers.** The same limitation applies when an AI system consumes documents. Text can attempt to redirect an assistant or induce unauthorized tool use without containing a macro or script. File reconstruction should not be credited with preventing such semantic manipulation unless that capability has been specifically demonstrated. AI authority limits and separation between document content and executable instructions require their own design and evidence.
- **Fidelity and provenance.** Reconstruction can also change document properties important to the business. An altered signed file generally requires separate handling of the original signature and provenance. Visual similarity alone cannot demonstrate preservation of formulas, accessibility, units, annotations, or machine-readable fields. Required information must be defined and checked in the applications that will actually use it.

8. An Assurance Case for the Content Workflow

Assurance should connect *claims* to arguments and evidence [8]. The following pattern is proposed for the illustrative maintenance workflow. It is not a finding that an existing product satisfies these claims. Figure 1

© 2026 RONROSSECURE, LLC | CC BY 4.0.

shows the claim structure, with the evidence that would support each *subclaim*; the evidence for Subclaim 2 includes the CDR failure modes described in Section 2.

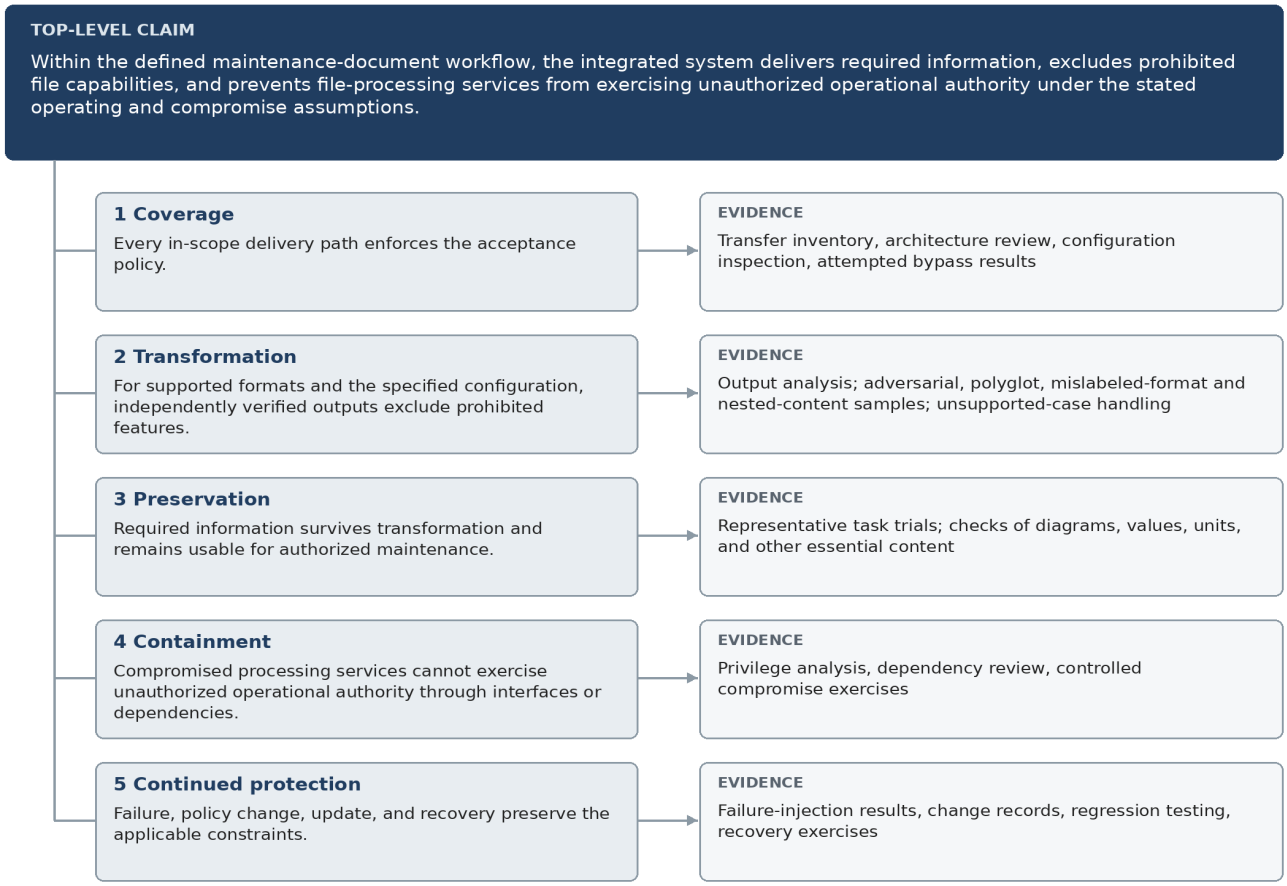


Figure 1. Assurance claim structure for the maintenance-document workflow (illustrative).

Scope and assumptions. This claim covers content constraints and operational authority; it does not establish the correctness of maintenance information or protection against every destination-application defect. Source authentication, technician verification, and independent engineering checks in Section 5 require a separate assurance argument. The case assumes a defined content profile and destination configuration, all in-scope paths under enforcement, and an attacker controlling the intake or transformation service but unable to compromise the independent verifier, release enforcement, policy administration, or operational access controls. Those separation and independence assumptions require evidence.

Top-level claim. Within the defined maintenance-document workflow, the integrated system delivers required information, excludes prohibited file capabilities, and prevents file-processing services from exercising unauthorized operational authority under the stated operating and compromise assumptions.

Argument. Coverage subjects every in-scope transfer to enforcement; transformation and independent verification exclude prohibited capabilities; preservation establishes required usability; containment denies processing services operational authority; and continued protection maintains these conditions through failure and change. Together, these subclaims support the top-level claim only within the stated scope and assumptions, with integration evidence showing that bypass paths and shared dependencies do not invalidate their combination.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

Each *subclaim* needs explicit limitations. Testing a finite collection of files cannot support a claim of universal protection against every possible file or application defect. Reports should identify product versions, policy settings, file formats, a set of test files, destination applications, and unresolved exceptions. Integration evidence is needed because a component result cannot show that the full transfer architecture preserves the same property.

9. A Focused Pilot for Organizations

Start with one consequential content flow and a representative set of organizational users. Define unacceptable losses, required information, supported formats, and prohibited capabilities before selecting acceptance measures. Assign owners for policy, exceptions, and ongoing maintenance. The pilot should include systems security engineers and the people who perform the receiving task.

Measure the pilot against five groups of evidence.

- **Protection.** Bypass resistance and observed policy violations, including the failure modes in Section 2: unsupported formats, mislabeled or polyglot files, and deeply nested content.
- **Utility.** Required task completion and failures of required functionality, checked in the destination applications.
- **Operations.** Percentage of files requiring exceptions, exception resolution time, file rejection rate, and staff effort.
- **Performance.** Latency and throughput with representative file sizes, nesting, and expected peak load.
- **Resilience.** Behavior under overload and loss of service, in addition to normal operation.

Policy-based processing can reduce dependence on malware classification, but rejected legitimate files and damaged functionality still create operational costs. Compare complete operating cost with the protection gained and expand only when the evidence supports the intended use and remaining limitations have accountable owners.

CDR can make an important contribution by removing unneeded file capability before use. Systems security engineering shows where that contribution fits, what else must stay protected, and what evidence justifies confidence in the resulting architecture.

What to do Monday. Choose one inbound file flow. Write down the few capabilities its recipients actually need and list every path by which files reach them. Then run a candidate product against your own files and try to deliver an original through a path you did not plan for.

Acknowledgments

The author gratefully acknowledges the support provided by Dartmouth College's Institute for Security, Technology, and Society (ISTS). The resources provided by the ISTS contributed to the research and development of this paper. The views and conclusions expressed herein are those of the author and do not necessarily represent the views of Dartmouth College.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

References

[1] Bussell, J. (Glasswall). *What is Content Disarm and Reconstruction (CDR)?* July 14, 2026. https://www.glasswall.com/us/blog/what-is-content-disarm-and-reconstruction-cdr

[2] OPSWAT. *Deep CDR Technology*. Product description. https://www.opswat.com/technologies/deep-cdr

[3] DeBolle, P. *Menlo Security Acquires Votiro*. February 19, 2025. https://www.menlosecurity.com/blog/menlo-security-acquires-votiro

[4] Ross, R., McEvilley, M., and Winstead, M. *Engineering Trustworthy Secure Systems*. NIST SP 800-160, Volume 1, Revision 1. November 2022. Appendix E. https://doi.org/10.6028/NIST.SP.800-160v1r1

[5] Ross, R. *Least Functionality*. Engineering with NIST Security Design Principles, Principle 4. RONROSSECURE, LLC, 2026. https://ronrossecure.com/library

[6] Ross, R. *Mediated Access*. Engineering with NIST Security Design Principles, Principle 2. RONROSSECURE, LLC, 2026. https://ronrossecure.com/library

[7] Ross, R. *Engineering for Compromise: Designing Systems That Remain Trustworthy Secure Under Adversity*. RONROSSECURE, LLC, 2026. https://ronrossecure.com/library

[8] Ross, R. *Assurance and Trustworthy Secure Systems*. RONROSSECURE, LLC, 2026. https://ronrossecure.com/library

[9] OWASP Foundation. *File Upload Cheat Sheet*. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

[10] Kent, K., Chevalier, S., Grance, T., and Dang, H. *Guide to Integrating Forensic Techniques into Incident Response*. NIST SP 800-86. August 2006. Sections 4.2 and 4.5. https://doi.org/10.6028/NIST.SP.800-86

[11] National Cyber Security Centre (UK). *Pattern: Safely Importing Data*. Version 1.0, July 15, 2018. https://www.ncsc.gov.uk/guidance/pattern-safely-importing-data

[12] Koch, L., et al. *On the Abuse and Detection of Polyglot Files*. Proceedings of the ACM Web Conference 2025, pp. 4810–4822. https://doi.org/10.1145/3696410.3714814

[13] Ross, R. *Least Persistence: Limiting Availability and Use to the Time Required for the Intended Purpose*. Engineering with NIST Security Design Principles, Principle 5. RONROSSECURE, LLC, 2026. https://ronrossecure.com/library

Source and Scope Note

Web sources accessed October 31, 2026. Commercial sources are used to describe their own products and announcements. Their inclusion does not constitute a comparative product evaluation or endorsement. Independent sources [9], [11], and [12] support the discussion of architecture and failure modes; the polyglot study [12] evaluated a research tool, not a commercial product. The architecture, pilot, and assurance pattern are proposed engineering applications. Their effectiveness requires evaluation in the intended system and operational environment.

© 2026 RONROSSECURE, LLC | CC BY 4.0.