RESEARCH PAPER

What CMMC 20X Cannot Automate

A task-level boundary for software-assisted CMMC evidence work, including the judgments, authorities, and field conditions that remain human.

STATUS	LAST REVIEWED	RESEARCH LANES	CLAIM REGISTER
Current	**August 13, 2026**	**Principles · Evidence · Evaluation · Mechanism**	**V-03 · V-05 · C-07 · C-08 · C-11**

Current CMMC 20X capability-boundary paper as of August 13, 2026. It states proposed uses and exclusions for evaluation; it makes no claim of Government approval, measured deployment performance, or legal advice.

Where should the machine stop?

Automation is useful when the task can be specified, observed, tested, and corrected. CMMC work contains many such tasks: collecting exports, checking required fields, comparing populations, finding stale records, mapping sources to possible objectives, and rendering consistent documents.

CMMC work also contains judgments about scope, implementation, sufficiency, mission consequence, credibility, and authority. Those judgments depend on facts outside a file, professional examination, organizational responsibility, and powers assigned by regulation, policy, or contract.

CMMC 20X draws the line at the task level. It does not label an entire workflow “automated” or “human.” It asks what software may do, what evidence supports the operation, how errors are detected, who reviews exceptions, and who has authority over the resulting decision.

This paper describes a proposal for controlled evaluation. Current CMMC rules and assessment methods govern unless an authorized process changes them.

A capability map

TASK	SUITABLE MACHINE ROLE	REQUIRED HUMAN ROLE	WHY THE BOUNDARY MATTERS
Collect a configuration export	Execute a defined connector and preserve provenance	Authorize access; verify source and scope	The connector sees only wha its permissions and source expose
Check dates, fields, hashes, and counts	Apply deterministic rules	Set rules; investigate exceptions	Structural validity does not establish safeguard effectiveness
Suggest objective mappings	Rank candidates with citations	Confirm relevance and completeness	Similar language can conceal different implementations
Compare claims and sources	Flag contradictions and population gaps	Determine consequence and corrective action	A discrepancy can reflect failure, scope error, or a justified condition
Draft a possible finding	Generate cited candidate text	Examine evidence and author the finding	Assessment judgment belong to the qualified reviewer
Generate an SSP or evidence index	Render reviewed records into a required view	Approve content and resolve open states	Consistent output can still contain unsupported assertions
Assign review depth	Supply facts and scenarios	Government applies authorized risk policy	Evidence quality cannot define mission consequence by itself
Issue status or take program action	No decision role	Authorized person or body decides	Legal and program effect cannot arise from software output

The useful region is substantial. So is the exclusion region.

Software cannot determine the real boundary from files alone

System scope depends on where information flows, which people and assets touch it, what services perform security functions, and how contracts and mission context apply. Inventories, diagrams, tickets, and telemetry can support the decision. They can also be incomplete or internally inconsistent.

A tool may discover an unmanaged device or a provider connection absent from a diagram. It may propose that the boundary be reopened. It cannot interview the right people, observe every informal process, resolve ambiguous contract language, or accept organizational responsibility for the final scope.

The test for a scoping feature should therefore measure discovery and omission. Seed cases with undocumented data flows, shared credentials, local emergency accounts, provider-side administration, and obsolete diagrams. Score what the tool surfaces and what remains invisible. Keep the final scope decision with a named responsible person and preserve the reasons.

Software cannot prove that a safeguard works in practice

A configuration export captures a state through a particular interface at a particular time. A policy describes expected behavior. A ticket records a work item. None alone proves that people follow the process, that an exception path is controlled, or that the safeguard remains effective under operational conditions.

Independent examination can require interviews, demonstrations, sampling, observation, follow-up questions, and professional skepticism. Software can organize those records and highlight where the sources disagree. It cannot substitute a generated narrative for the examination.

This boundary is especially important for procedural objectives and sparse environments such as operational technology, legacy systems, and small organizations with limited telemetry. Lack of machine-readable evidence does not establish failure. Abundant telemetry does not establish effectiveness.

Software cannot own the contractor's assertion

The contractor owns its implementation, evidence, gaps, and organizational affirmation. A provider can contribute service facts. An advisor can prepare the record. A tool can draft or check it. Those contributions do not transfer the contractor's responsibility.

The interface should make affirmation a separate human action, show every open conflict, identify the record version, and prevent a generated score from silently becoming the assertion. Material changes should reopen affected claims for human review.

An evaluation should attempt to induce unsafe affirmation: hide an exception in a long package, insert conflicting evidence after review, or present a confident summary from incomplete sources. The workflow should reveal the conflict and require explicit disposition.

Software cannot author an independent assessor's judgment

AI can offer a candidate mapping or finding. The assessor has to decide what to examine, which samples matter, whether evidence is sufficient, how conflicts affect an objective, and why the finding follows. The assessor also has to remain independent within the applicable program rules.

Authorship requires more than a click. The record should show the evidence the reviewer examined, tests performed, machine suggestions accepted or rejected, reasoning, disagreements, and final disposition. A reviewer needs enough time and access to challenge the output.

The evaluation should compare unassisted and assisted review on the same frozen cases. Measure missed conflicts, unsupported findings, corrections, agreement, review labor, and variation among qualified reviewers. Analyze reliance effects: does a plausible machine error make reviewers more likely to miss the planted problem?

Software cannot allocate mission risk by inspecting evidence quality

A thin package may signal poor preparation. It does not reveal the consequence of compromise to a weapons program, operational mission, sensitive technology, or supply chain. A complete package can describe a high-consequence system.

CMMC 20X proposes that mission consequence, data sensitivity, threat exposure, supplier criticality, and material change inform verification depth. Government must define and govern that policy. Evidence quality should trigger correction, sampling, or escalation within the assigned route.

Conflating these concepts creates a perverse incentive: a supplier with weak evidence might be moved to a "higher-risk" category even though risk should have been determined from mission and data context. The reverse is worse: polished evidence might make a consequential system appear suitable for lighter review.

Software cannot grant authority

A model output, deterministic check, schema-valid package, or vendor badge has no independent authority to certify, affirm, accept, award, enforce, or resolve an appeal. Those acts belong to people and institutions under the governing framework.

Interfaces should reserve official status terms, distinguish machine results from human findings, and record the authority behind every decision. Exports should preserve the distinction. Marketing language should follow the same rule.

This is a technical control as much as a governance statement. If an API allows a machine result to populate an official status field, the architecture has created an authority-escalation path.

Software cannot validate itself

A vendor-selected demonstration can favor known cases and convenient measures. An average score can conceal severe errors in rare or difficult conditions. A model can change after evaluation. Data drift, prompt injection, poisoned context, version changes, and operator workarounds can alter behavior.

Government should control the evaluation question, cases, reference findings, thresholds, operating environment, stop rules, and deployment decision. The vendor can supply the system and instrumentation. Independent participants should attempt to reproduce the results and break the assigned boundaries.

Validation should continue after any permitted deployment. Version changes and material shifts in data or use should trigger regression tests and renewed authorization for the affected task.

A proposed stop-work register

Every automated task should have a published stop-work condition. The following register is a starting point:

- **Source failure:** required source is unavailable, unauthenticated, outside the stated period, or materially transformed without lineage.

- **Coverage failure:** observed population cannot be reconciled with the expected population.
- **Conflict failure:** material sources disagree and no qualified person has resolved the disagreement.
- **Task escape:** the system attempts to issue a finding, status, or action beyond its permitted operation.
- **Version failure:** deployed rules, model, prompt, profile, or connector differ from the evaluated version without approved change control.
- **Safety threshold failure:** a predefined severe-error, false-negative, or unsupported-conclusion limit is exceeded.
- **Audit failure:** the output cannot be traced to inputs, method, version, and reviewer actions.
- **Context failure:** a case falls outside evaluated systems, supplier types, evidence conditions, or operating constraints.

Stopping should send the case to a named human path. It should never convert an unknown state into a passing state.

How to test the boundary

Start with a task inventory. Describe each operation using a verb and object: collect identity export, compare account populations, suggest objective mapping, draft candidate finding. Avoid broad labels such as “perform assessment.”

For each task, publish permitted inputs, expected outputs, excluded decisions, human qualifications, measures, thresholds, and stop conditions. Build frozen cases that include normal, degraded, ambiguous, changed, and adversarial states. Have qualified reviewers establish reference dispositions before exposure to machine output.

Run ordinary and assisted workflows. Preserve every source, rule result, model output, reviewer action, correction, and version. Report results by case group and severity. A tool that saves time on easy cases while increasing severe misses on provider-dependent cases has not earned broad use.

Any deployment decision should name the exact tasks permitted, environments, required controls, monitoring, correction path, and conditions that stop use. Expansion should require new evidence.

The point of the boundary

The objective is better use of scarce human attention. Repetitive collection, comparison, mapping, reconciliation, and rendering can be suitable for automation when the method is inspectable and tested. People can then spend more time on scope, interviews, conflict, implementation, consequence, and judgment.

That benefit remains a hypothesis until measured. CMMC 20X offers working software and a synthetic package as material for evaluation. It does not present them as proof of accuracy, labor savings, Government acceptance, or safe scale.

A credible automation proposal should be clearest about its stopping point. The machine can help make the security work visible and testable. Accountable people still decide what the work proves and what happens next.

Inspect the worked evidence example and review the Government pilot proposal.

TRACE THE CLAIM

Sources and revision history.

PRIMARY AND GOVERNING SOURCES

01 **Cybersecurity Maturity Model Certification Program, 32 CFR part 170**
Electronic Code of Federal Regulations
↗

02 **CMMC Assessment Guide — Level 2, Version 2.13**
Department of War Chief Information Officer
↗

03 **Artificial Intelligence Risk Management Framework (AI RMF 1.0)**
National Institute of Standards and Technology
↗

04 **Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile**
National Institute of Standards and Technology
↗

05 **Secure Software Development Framework, NIST SP 800-218**
National Institute of Standards and Technology
↗

CORRECTIONS AND MATERIAL REVISIONS

No material corrections recorded.

See an error or a source we missed? Send a correction. Material changes are recorded here rather than silently overwritten.