Evaluating LLMs for Impact-Faithful Translation of Adversary Behavior Across Operating Systems

Authors:

Fujitsu Research of Europe: Ahmed Elmesiry, Ofir Manor, Inderjeet Singh, Andrés Murillo

MITRE: Mark Perry, Fionna McCrae, Caleb Bynum, Hunter Pittman, Chris Lenk, Zoe Cheuvront, Ivy Oeltjenbruns

June 30, 2026

MITRE FUJITSU

Table of Contents

1 Executive Summary

Adversary behavior is rarely independent of its operating environment. Threat intelligence and adversary emulation plans are frequently authored for a narrow set of platforms, whereas real-world defensive environments vary widely in operating system composition, telemetry sources, and security tooling. The translation of adversary procedures across environments, such as the adaptation of Windows-centric attack chains for Linux servers or mixed operating system fleets, remains a routine yet non-trivial task that depends heavily on individual expertise and is difficult to evaluate objectively. Large Language Models (LLMs) offer promise for assisting this translation; however, they introduce risks of hallucination, ontology mismatch, and defender-irrelevant output that standard metrics fail to capture. MITRE partnered with Fujitsu Research of Europe (FRE) to engage in a six-week effort to develop a framework to evaluate an LLM's ability to faithfully translate adversary behavior across operating systems.

The framework comprises a scoring taxonomy that includes Behavior Chain Fidelity (BCF), Technical Realism (TR), and Defensive Value (DV); automated and manual evaluation tiers; a pre-generation comprehension assessment; a graph-based structural evaluation layer with layered semantic enrichment; and an evaluator-standardization protocol.

An initial trial applied the framework to procedures derived from the ALPHV/BlackCat ransomware adversary, translating a 29-step Windows attack chain to Linux. The automated tier returned a pass rate of 91 percent, but this figure alone does not indicate that the translation succeeded. When the same checks are re-scored under stricter, defender-relevant definitions of telemetry equivalence, the rate falls to 67 percent. The graph-based structural layer is more revealing still: although the variant preserves ATT&CK technique and tactic structure almost perfectly and reaches a telemetry-layer (L2) structural similarity of 0.90, an illustrative composite fidelity score, computed under hand-assigned technical-realism and defensive-value inputs, sits at just 0.59. This gap exposes four failure modes, observed in this single trial, that boolean scoring would not surface.

Two qualifications frame these results. The variant evaluated here is itself an incomplete translation: at nine of its 29 steps it still executes Windows .exe binaries through the wine compatibility layer rather than native Linux tradecraft, which both demonstrates the framework's sensitivity and bounds the fidelity it can report. Separately, the manual evaluation tier proved strongly rater-dependent — five subject-matter experts returned an aggregate pass rate of 46 percent, with per-evaluator rates spanning 22 to 100 percent — so inter-rater reliability, rather than any headline pass rate, is the most important result of the manual tier and is treated accordingly.

Key contributions of the framework include:

- A defender-relevant procedure graph formalism with normalized GED similarity as a continuous fidelity measure.

- Three progressive semantic enrichment layers (tactic overlap, telemetry-class overlap, and Sigma logsource overlap) that localize where a translation departs from defender-relevant equivalence.
- Eight production-ready Linux Sigma rules, six of which are entirely novel, delivered to the open-source SigmaHQ repository as a direct output of the graph-based detection-gap analysis.
- An integrated pre-generation assessment methodology, a replication-procedure control condition, prompt-engineering guidance, an evaluator-standardization protocol, and a multi-adversary research roadmap.

The framework is presented explicitly as a prototype methodological specification, exercised on a single adversary and a single OS pair. The fidelity thresholds and overlap requirements, together with the technical-realism and defensive-value inputs to the composite score, are declared as initial analytic values that require future empirical calibration rather than validated measurements.

2 Introduction

Adversary tradecraft is commonly documented in forms that reflect the specific environments in which it was observed. A given threat actor may pursue identical objectives across organizations; nevertheless, the concrete procedures employed, including commands, tools, execution paths, and system interactions, are shaped by the operating systems, configurations, and controls present at the time of execution. This environment-specific dependency creates a persistent challenge for defenders whose operational reality may differ substantially from the environments reflected in public intelligence reporting.

Detection engineering, coverage assessment, and validation exercises depend on whether a behavior can be executed on a defender's systems and whether that behavior produces telemetry that is representative of documented adversary activity and identifiable by existing detection logic. Defenders require the ability to anticipate how adversary behavior would manifest, and whether it can be detected, within their own environments, not only within the environments in which the activity was originally observed.

LLMs present a compelling opportunity to accelerate cross-environment translation; however, their use introduces additional challenges. LLMs may misunderstand or hallucinate tasks, and the resulting translations may inadvertently alter adversary behavior or its observable characteristics. Furthermore, procedural similarity or syntactic

correctness alone, even when achieved, is insufficient to deliver defender value. Translation can preserve surface form while altering behavior, execution constraints, or defender-relevant telemetry in ways that undermine defensive analysis.

This research attempts to address that gap through a structured set of evaluation measures grounded in defender-relevant indicators rather than direct procedural equivalence. LLM-generated procedure variants are evaluated through behavioral correspondence based on step structure and dependencies, technical plausibility on the target operating system, and defender-relevant utility reflected in telemetry equivalence and differentiation. The methodology integrates pre-generation comprehension assessment, automated evaluation, expert manual evaluation, and graph-based structural evaluation into a unified framework.

An initial trial applied the framework to procedures derived from the ALPHV/BlackCat adversary, translating a 29-step Windows attack chain to Linux. The automated tier returned a pass rate of 91 percent, which falls to 67 percent once the checks are re-scored under stricter, defender-relevant definitions of telemetry equivalence. Manual evaluation by five subject-matter experts produced an aggregate PASS rate of 46 percent and a FAIL rate of 39 percent, with per-evaluator PASS rates ranging from 22 to 100 percent — a spread that underscores the subjective nature of unaided expert assessment and motivates the treatment of inter-rater reliability in Section 5.1. Under the graph-based structural layer the same variant preserves technique and tactic structure almost perfectly and reaches a telemetry-layer (L2) structural similarity of 0.90, while an illustrative composite fidelity score, computed under hand-assigned realism and defensive-value inputs, sits at 0.59. These observations motivate the integrated methodology described in the sections that follow.

3 Methods

The methods for this research are designed to evaluate whether adversary procedure translations by an LLM preserve defender-relevant structure, technical plausibility, and telemetry characteristics across operating environments. Specifically, the methods have the following goals.

1. **Use a defender-centric taxonomy:** The main motivation of evaluating the LLMs capabilities for procedure translation is to provide defenders with better detection engineering tools. For this reason, the taxonomy of criteria used to evaluate the translations is scoped to published detection strategy components like those found in MITRE ATT&CK Detection Strategies.

2. **Use complementary evaluation modalities to maintain model-agnosticism:** This research evaluation combines automated evaluation using deterministic questions, manual evaluation with nine expert questions, and a graph based structural evaluation to try to offer a more robust evaluation of the LLM capabilities.

Documentation and code for this experiment are maintained in the fre-research githab repository: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis

3.1 Test Data Categories and Formats

All procedures are represented in a structured, machine-readable format to enable explicit comparison of step ordering, technique usage, execution context, and derived telemetry abstractions. The data considered in the test consists of the following listed files processed through manual and automated test processes.

1. **Original procedures (Markdown)** human-readable documentation of adversary emulation procedures.
    a. The initial experiment uses test procedures based on the **ALPHV BlackCat** adversary: https://github.com/attackevals/ael/blob/main/ManagedServices/alphv_blackcat/Emulation_Plan/ALPHV_BlackCat_Scenario.md
2. **Control procedures (JSON)** created by (1) using an LLM to convert the markdown-formatted procedures into our test JSON schema (to save time associated with manually typing out the JSON) and (2) adding human experts in the loop for manual verification of equivalence to the original procedure.
    a. The control procedures are the “gold copy” against which replication and variant procedures are evaluated.
    b. Link to control procedures: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis
3. **Replication procedures (JSON)** created by prompting an LLM to translate the original procedure into our machine-readable JSON schema while maintaining the same target environment as the original procedure (i.e. target hosts are running Windows OS versions). The replication scenario serves as a control condition, isolating representation and prompting effects from cross-environment translation effects.
    a. The experimental value of the replication scenario is to help with grading the quality of the research team’s prompting of the LLM, and to capture a low baseline of the LLM’s ability to execute requested tasking on the same target environment. If the LLM struggles with translation onto the same base OS, we would expect additional challenges when adding complexity to translate to a new OS.

   b. Link to folder of replication procedures: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis
4. **Variant procedures (JSON)** created by prompting an LLM to generate a variation on an adversary's attack chain for execution for a different combination of operating systems represented in a variant test environment (ex. the prompt requests abilities targeting Linux for an attack chain originally on Windows).
   a. The experimental value of the variant scenario is to complete the full testing of an LLM's ability to translate an attack chain to fit the specified target operating system(s).
   b. Link to folder of variant procedures: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis
5. **Enriched procedures (JSON)** for control, replication, and variant procedures which contain additional derived D3FEND observables, execution observables, telemetry classes, and execution level for each procedural step.
   a. This automated enrichment aids automated and manual evaluation.
   b. Link to folder of enriched procedures: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis
6. **Scoring Result (CSV, Excel)** generated during the manual and automated evaluation process.
   a. Evaluation Questions and Manual Evaluation Artifacts: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis/LLMEnvTranslationScoringQuestions.xlsx
7. **Sigma rules (YAML),** obtained from Graph Based Structural Evaluation
   a. https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis/
8. **Graph Based Structural Evaluation Results (JSON)**
   a. https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis

Figure 1 depicts the flow of data through various processes to generate result data.

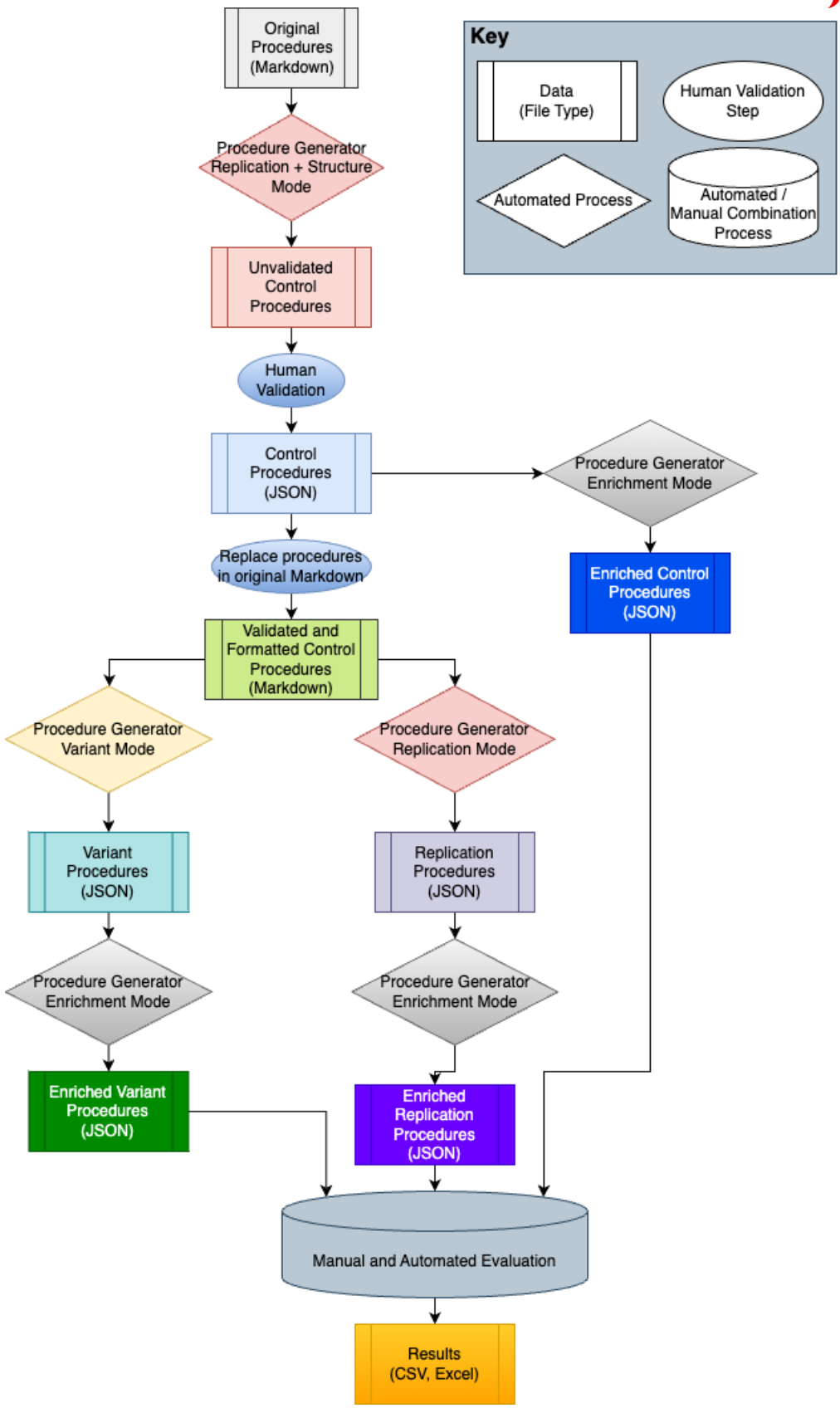

Figure 1: Process for Generating Test Data and Results

3.2 Model Selection

This research evaluates the framework using [GPT-OSS-120b], an open-weight OpenAI GPT-family model, selected based on availability on local hosting infrastructure (Section 5.7). [Nemotron 3 Nano, an NVIDIA model, was used for code development only and did not generate any of the procedures under evaluation. / Both GPT-OSS-120b and Nemotron 3 Nano were used to generate procedures during testing.] The framework is model-agnostic by design; future work should apply it across additional models, including frontier systems (Section 5.7).

3.3 Prompt Engineering and Assessment

Prompt engineering is "the process of designing and refining input queries, or "prompts," to elicit desired responses from LLMs."[1] Prompt engineering in this work serves to standardize procedure generation and is not itself a primary subject of evaluation.

3.3.1 Prompt Template

A prompt template was created to standardize the experiment and enable a baseline for modification in response to pre-generation assessment. There are many prompt templates used at all stages of the pipeline focusing on replicating, translating, enriching, and converting existing procedures. Prompt templates follow the Role-Task-Format (RTF) structure or another structured prompting technique, selected based on effectiveness assessments for each scenario. An example of the prompt templates used in the generation pipeline are included in Appendix A.

The original markdown procedures were augmented as part of prompt development to include an Attack Chain Overview, Target Environment Specification, and tweaks to the Attack Chain Procedure.

3.3.2 Assessment for Prompt Improvement

Manual human validation was used to improve the prompt for the LLM generation process.

3.4 Evaluation Framework

This research used three complementary modalities on its evaluation framework:

1. **Automated evaluation:** Using eleven deterministic questions on the enriched JSON files

[1] Marvin, G., Hellen, N., Jjingo, D., Nakatumba-Nabende, J. (2024). Prompt Engineering in Large Language Models. In: Jacob, I.J., Piramuthu, S., Falkowski-Gilski, P. (eds) Data Intelligence and Cognitive Informatics. ICDICI 2023. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-99-7962-2_30

2. **Manual evaluation:** Using nine expert questions with the evaluator protocol reference. These questions were obtained after evaluating originally 21 questions
3. **Graph-based structural evaluation:** We used a graph nodes and edges to evaluate the preservation of dependency and key attributes in the translated procedures.

For clarity on the differing counts cited throughout this report: the full candidate bank contains 40 scoring questions (Table 13). Of these, 11 were implemented as automated checks for the question-based tier — 12 in the GBSE pass, which adds the os_constructs_valid check and 21 were trialed for manual scoring, of which after subject matter expert review and analysis 9 survived to the final manual questionnaire used to produce the results in Section 4.1.2.

3.5 LLM OS Translation Capability Question Based Evaluation

The research team formulated a set of questions to meaningfully assess LLM capability for generating variant procedures. These questions attempt to capture functional equivalency rather than any sort of direct comparison between the original procedure and the transformed one. Table 1 lists an overview of high-level scoring categories for these questions. The full list of questions related to these scoring categories is listed in https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis/LLMEnvTranslationScoringQuestions.xslx.

Table 1: Variant Adversary Emulation Scoring Category Purpose Overview

Scoring Category	Scoring Sub-Category (if Applicable)	ID	Purpose
Behavior Chain Fidelity (BCF)	N/A	1	Measure whether the variant represents the ***same adversary story, intent, and flow.***
Technical Realism (TR)	N/A	2	Measure whether the procedure is ***technically plausible*** and executable on the target platform.
Defensive Value (DV)	**Defensive Value Score (DVs)**	3.0	(Subset of DV) Measure whether the output is ***useful to defenders***. DVS is downstream of Telemetry Equivalence Score (TES) and Telemetry Richness and Differentiation (TRD).
	Telemetry Equivalence Score(TES)	3.1	(Subset of DV) Measure whether the variant ***enables defender-equivalent detection opportunities.***
	Telemetry Richness & Differentiation (TRD)	3.2	(Subset of DV) Measure whether the behavior produces ***multi-source, defensible telemetry*** sufficient to ***evaluate detection capability***.

3.5.1 Automated Evaluation

Evaluation of the variant procedures can be partially automated by custom scripts aligned to the novel adversary emulation scoring metrics. Table 2 summarizes questions aligned to these automated measures. Interpreting the automated output (CSV) into a pass or fail outcome for each question requires minimal human evaluation.

Table 2: Selected Automated Evaluation Questions

Question ID	Question	Measure	Pass Condition
1.1	Does the variant preserve the same ordered sequence of techniques as the control?	technique_sequence_match	All techniques match in order
1.4	Are any steps in control missing from the variant?	missing_steps	No missing steps
1.5	Are any new steps present in the variant that are not in the control?	extra_steps	No extra steps
1.7	Does the variant preserve the same privilege context progression as the control?	privilege_context	Privilege context flow matches between control and variant
3.1.1	Do the variant and the control map to the same parent ATT&CK techniques?	parent_technique_ids_match	All parent techniques match
3.1.2	Do the variant and the control map to the same ATT&CK Technique IDs (exact)?	technique_ids_match	All technique IDs match exactly
3.1.3	Does the variant generate the same telemetry classes as the control?	telemetry_classes_match	All control classes present
3.2.1	Does execution generate telemetry in more than one telemetry class?	telemetry_multiclass	Variant has >= 2 classes
3.2.2	Are different stages observable via different telemetry classes?	telemetry_consecutive	Telemetry classes match between 2 steps
3.2.7	Does the variant have the same level of telemetry diversity as the control?	telemetry_diversity	Variant count >= control count
3.2.9	Are any telemetry classes from the control missing?	missing_telemetry_classes	No missing control classes
3.2.10	Are additional telemetry classes present in the variant?	extra_telemetry_classes	No extra classes

3.5.2 Manual Evaluation

Human evaluators conduct manual evaluation by filling out a questionnaire spreadsheet. For each question, evaluators indicate that the variant output passed, failed, or the question was not applicable (n/a). Evaluators are also encouraged to leave notes providing context for their response, and feedback on the scoring question. Future work may attempt to automate some manual evaluation questions using data and

enrichment of procedures after generation. However, human evaluators will likely continue to be required to verify the data and evaluate questions that do not have a binary answer.

Table 3 summarizes questions identified for manual evaluation.

Table 3: Selected Manual Evaluation Questions

Question ID	Question	Measure	Pass Condition
1.2	Does the variant preserve the same logical dependencies between steps?	Logical dependencies	Dependencies match
2.1	Do all steps reference real, identifiable commands or tools available on the target platform?	Commands/Tools Used	Commands/Tools exist
2.2	Are all command arguments, flags, and options valid for the referenced tool and execution context?	Command Arguments Used	Command Arguments are valid
2.3	Does the procedure respect OS-specific constraints and permissions?	OS-specific constraints	Constraints are followed
2.4	Does the procedure avoid original OS only constructs on variant systems?	Original OS only constructs	The variant did not use original OS only constructs
2.5	Are privilege requirements stated or implied correctly?	Privilege requirements	Privilege requirements followed
3.1.6	Would a detection written for the control have a logically equivalent form on the variant?	Detection Core Actions	Detection Core Actions match
3.1.8	Does the variant preserve the semantic meaning of what the detection observes?	Detection Observations	Detection Observations match
3.2.11	If additional telemetry exists, does it improve detection capability?	Telemetry Classes	Additional Telemetry Classes Add Value

3.5.3 Graph-Based Structural Evaluation

The presented automated Boolean and expert manual review provide a good baseline evaluation of the translation capabilities of an LLM. Nevertheless, we identified that question-based approaches might overestimate fidelity at procedure level. For this reason, this research complements the analysis using graph-based evaluation. Using graph-based evaluation, this research proposes formalizing each procedure as a

directed attributed graph rather than a flat list of steps. This analysis is explained in Section 4.2.

3.5.4 Scoring

While a methodology to aggregate automated and manual scoring metrics into a final "score" is not yet standardized, discussion of future considerations for scoring can be found in Section 5.10.1. The metrics generated for automated and manual scoring data generated by the trial are described in Sections 4.1.1 and 4.1.2, respectively.

3.6 Use case study: ALPHV/BlackCat Adversary

The research team completed a single trial run of the evaluation methodology using BlackCat procedures and the OpenAI GPT LLM model to generate a control JSON and variant JSON for scoring. The research team is aware that a single trial run is not an exhaustive study, but the scope of this work is aimed at presenting the methodology, using a well-documented adversary and discussing the limitations with the methodology. In this sense, the results and discussion are presented as a first step and an invitation to other research teams to build on the foundational information presented in this research. As for the Graph-based analysis, the research team used the ATT&CK taxonomy version 15, we selected this version because it is the closest version chronologically to the original description of the BlackCat Adversary. Nevertheless, future work could explore the impact on some of the results described here with more recent versions of ATT&CK.

4 Results and Discussion

For the results, this section first analyses the results of the fully automated and manual evaluation questionnaires described in the Section 3.5.1 and 3.5.2; then, the research discusses some empirical observations and identifies some possible failure modes. Following this, the research presents the Graph Based Structured Evaluation (GBSE) results.

Testing results include both the resulting metrics from a trial run of this methodology, as well as products for future research on evaluating LLM capabilities to assist with adversary emulation across diverse operational environments.

4.1 LLM Output Evaluation Metrics

This report refers to the trial run output as metrics since time did not permit creating a scoring methodology (see discussion in Section 5.10).

4.1.1 Automated Evaluation Metrics

Documentation for generating automated evaluation metrics is contained within the README, https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis.

The team calculated an unweighted pass rate for the automated evaluation of 91%. Table 4 shows a summary of the metrics for the automated evaluation of the variant procedures during the trial test. The high pass rate should not be read as evidence that the LLM succeeded at translation for a real emulation. The automatable questions check countable, easily derived properties of ATT&CK techniques, D3FEND countermeasures, and telemetry sources, none of which were verified through an actual emulation. At best the metric flags where the LLM may have fallen short and surfaces discrepancies between the control and variant procedures, easing manual evaluation.

Table 4: Automated Evaluation Metrics Summary

Question ID	Question	Pass Condition	Evaluation (PASS/FAIL)
1.4	Are any steps in control missing from the variant?	No missing steps	PASS
1.5	Are any new steps present in the variant that are not in the control?	No extra steps	PASS
1.7	Does the variant preserve the same privilege context progression as the control?	Privilege context flow matches between control and variant	FAIL
3.1.1	Do the variant and the control map to the same parent ATT&CK techniques?	All parent techniques match	PASS
3.1.2	Do the variant and the control map to the same ATT&CK Technique IDs (exact)?	All technique IDs match exactly	PASS
3.1.3	Does the variant generate the same telemetry classes as the control?	All control classes present	PASS
3.2.1	Does execution generate telemetry in more than one telemetry class?	Variant has >= 2 classes	PASS
3.2.2	Are different stages observable via different telemetry classes?	Telemetry classes match between 2 steps	PASS
3.2.7	Does the variant have the same level of telemetry diversity as the control?	Variant count >= control count	PASS
3.2.9	Are any telemetry classes from the control missing?	No missing control classes	PASS
3.2.10	Are additional telemetry classes present in the variant?	No extra classes	PASS
Aggregate Automated PASS Rate			**10 / 11 = 91%**

4.1.2 Manual Evaluation Metrics

We conducted a trial of manual evaluation to provide further context and clarity on the relevance and usability of questions selected for manual evaluation. Five subject-matter expert reviewers completed manual scoring questionnaires to determine whether the variant procedures passed the criteria for selected manual evaluation questions. The scoring sheets can be found in the https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis/LLMEnvTranslationScoringQuestions.xlsx file, with the original “Manual Questionnaire Template” tab provided to the evaluators, and each evaluator response as a tab “Manual Questionnaire ([scorer ID]).” The collected sheets were aggregated along with calculated metrics into the “AggregatedManualEval” tab. Table 5 shows a summary of the per-question and aggregate metrics for the manual evaluation of the variant procedures during our trial.

The PASS Rate and FAIL Rate of each question across evaluators (Total PASS, Total FAIL, respectively) provide a baseline manual evaluation metric: did evaluators tend to think the LLM output passed or failed each question.

Some evaluators found questions not applicable (N/A). Therefore, the N/A count (Total N/A) and the N/A Rate per question are included. This shows that evaluators did not have consensus on the applicability of questions 3.1.6, 3.1.8, and 3.2.11 for the variant procedures. Question 3.2.11 is dependent on the existence of additional telemetry (Automated Question 3.2.10), of which there was none, explaining why it was ruled N/A by multiple evaluators.

Not all evaluators had sufficient time to evaluate all questions, so the total number of evaluators who recorded an evaluation response (Total Responses Received) is also included. This is important when interpreting the metrics because it shows the total number of times each question was evaluated. Lack of an evaluation does not indicate that the variant procedures passed or failed a question.

Questions where Total Response Received is less than 5, and N/A Rate paired with 0% may need further development. The individual evaluator response sheets provide additional context as to why a question might not have been answered or was considered N/A. https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis

The Aggregate Manual PASS Rate and Aggregate Manual Fail Rate are the combined average of the Pass and Fail Rates per Question. This measures what percentage of questions the LLM passed or failed in aggregate given the questions were relevant and the evaluator had sufficient time. Overall, the LLM-output variant procedures passed the questions 46% of the time and failed 39% of the time they were evaluated or considered applicable.

Table 5: Manual Evaluation Per-Question Metrics Summary

Question ID	Question	Total PASS	Total FAIL	Total N/A	Total Responses Received	PASS Rate	FAIL Rate	N/A Rate
1.2	Does the variant preserve the same logical dependencies between steps?	1	2	0	3	**33%**	**67%**	0%
2.1	Do all steps reference real, identifiable commands or tools available on the target platform?	4	0	0	5	**100%**	**0%**	0%
2.2	Are all command arguments, flags, and options valid for the referenced tool and execution context?	2	3	0	5	**40%**	**60%**	0%
2.3	Does the procedure respect OS-specific constraints and permissions?	1	2	0	3	**33%**	**67%**	0%
2.4	Does the procedure avoid Windows-only constructs on non-Windows systems?	2	2	0	5	**40%**	**40%**	0%
2.5	Are privilege requirements stated or implied correctly?	1	2	0	4	**50%**	**50%**	0%
3.1.6	Would a detection written for the control have a logically equivalent form on the variant?	3	0	1	5	**80%**	**0%**	20%
3.1.8	Does the variant preserve the semantic meaning of what the detection observes?	1	1	1	3	**33%**	**33%**	20%
3.2.11	If additional telemetry exists, does it improve detection capability?	0	1	2	3	**0%**	**33%**	40%
Aggregate Manual PASS/FAIL Rates						**46%**	**39%**	

Table 6 summarizes evaluator metrics. The number of questions the variant procedures passed/failed per evaluator, based on the number of questions answered by the evaluator, provides additional insight into the need for further development. There is wide variation in the rate of questions passed/failed per evaluator, highlighting the subjective nature of manual evaluation and calling for further standardization. Further development of evaluation procedure guidance could also reduce the mental load required to answer each question and reveal opportunities for automation. This will make evaluating many LLMs' translation ability easier and more repeatable.

The answer rate per evaluator and pass/fail rate per-evaluator in general out of all nine manual questions (as opposed to out of the questions answered) provides further context about the completeness of each evaluation.

Additionally, evaluators had varying levels of experience and familiarity with the procedures and question set. Section 5.11 discusses other factors which should be considered when selecting evaluators in the future and could provide more consistent, repeatable, and accurate results

Table 6: Manual Evaluation Per-Evaluator Metrics Summary

Evaluator ID	Total PASS	Total FAIL	Answer Rate	PASS Rate of Answered	PASS Rate of All	FAIL Rate of Answered	Fail Rate Out of All
1	2	4	**100%**	**22%**	22%	**44%**	44%
2	5	0	**56%**	**100%**	56%	**0%**	0%
3	2	7	**100%**	**22%**	22%	**78%**	78%
4	6	0	**89%**	**75%**	67%	**0%**	0%
5	3	2	**56%**	**60%**	33%	**40%**	22%

4.2 Graph-Based Structured Evaluation (GBSE)

Automated binary evaluation shows some limitations because the checks are static counts and do not fully validate structural, semantic, or runtime equivalence. Two concrete observations are the motivation for this work:

Empirical Observation 1 (The measurement discrepancy). Binary evaluation instruments can overestimate cross-OS translation fidelity. A translation can record a high automated pass rate while, under telemetry-aware structural analysis, exhibiting low defender-relevant fidelity. Binary scoring might mask at least three failure modes: step insertion (a technique-valid step that shifts the expected telemetry profile and tactical flow); edge reordering (a dependency change that leaves the step count intact but renders sequence-correlation rules inert); and telemetry class drift (a tool substitution that drops an observable class, for example replacing process telemetry with purely network telemetry). In addition, we noticed that using the automated and manual translation evaluation, there was a Sigma logsource gap (a step that keeps its technique identifier while shifting its logsource to a platform where no equivalent native rule exists).

Empirical Observation 2 (Concrete instance). In the ALPHV/BlackCat trial a lateral-movement step illustrates the discrepancy directly: a Windows PsExec-style action that produces both process and network observables is rendered on Linux as an scp/ssh loop dominated by network observables. Process-based lateral-movement detection becomes ineffective even though the high-level technique correspondence appears preserved.

These failures are invisible to a flat list of steps but explicit once the procedure is treated as a graph. The remainder of this section develops that treatment, applies it to the full

29-step ALPHV/BlackCat plan, and reports a verified set of layers and composite scores.

4.2.1 Assumptions

The framework rests on two assumptions:

- **A1 (OS-independence of labels).** ATT&CK technique and tactic identifiers are deliberately platform neutral. A faithful translation preserves them across an OS boundary, so agreement at technique level (L0) and tactic level (L1) is a genuine measurement of technique preservation, not an artefact of comparing a procedure to itself.
- **A2 (OS-agnostic observable intent).** The telemetry a step is expected to produce can be expressed independently of the host: a step that touches the filesystem, opens a socket, and spawns a process produces {file, network, process} regardless of whether it runs wget or bitsadmin. Telemetry drift is therefore a property of the translation, not of how the source command happens to be spelled. We assign level 2 (L2) to telemetry comparisons.

A1 and A2 predict that a faithful cross-OS translation will agree at L0/L1 and at L2 except where the variant genuinely drifts. The results in Section 5 are consistent with the assumptions. Besides these assumptions, we identified that GBSE can provide insights into detection equivalence mismatches between the original and variant procedures. These insights were used to generate new sigma rules that can assist detecting adversarial behavior for the variant procedure. We label these detection comparisons L3.

4.2.2 The GBSE Framework

In this framework, each control and variant procedure is represented as a directed attributed graph. This captures step-level semantics, dependency order, and global topology at once, and gives a formal substrate for asking whether a translated procedure instantiates the same adversary chain rather than merely reusing the same technique labels.

Definition 1 (Procedure graph). A procedure is a directed attributed graph $G = (V, E, \varphi_v, \varphi^e)$, where V is the set of procedural-step nodes and $E \subseteq V \times V$ is the set of directed dependencies. The map φ_v assigns each node its attributes,

$$\varphi_v(v) = \langle technique\_id, tactic[], privilege\_context, telemetry\_classes[], sigma\_rules[], \dots \rangle,$$

and the map φ^E assigns each edge its type: sequential precedence (w=1.0), conditional-privilege dependency (w=1.5), or dataflow / telemetry-trigger relation (w=0.5).

In the trial both G_c (Windows) and G_v (Linux) carry 29 nodes and 28 sequential edges. Because a faithful translation preserves the per-step technique and tactic, the step-aligned technique identifiers match even though the underlying commands change from Windows to Linux. The discriminating signal therefore lives below the label layer, at telemetry (L2), Sigma logsource (L3), and the out-of-band os_constructs check.

Structural Similarity via normalized GED: Fidelity is defined by how much typed editing is needed to turn the control graph into the variant graph. The edit operations are chosen so that each carries a procedural meaning.

Proposition 1 (Structural similarity as a continuous fidelity measure). Let G_c be the control graph and G_v the variant graph. If procedure fidelity is the amount of semantic and topological editing required to transform G_c into G_v, then

$$Sim_{struct}(G_c, G_v) = 1 - \frac{GED(G_c, G_v)}{\max(|V_c|, |V_v|)} \quad (1)$$

is a continuous measure of structural correspondence on [0, 1]. When node and edge edits are semantically typed, the score decomposes into procedurally meaningful deviations: node insertion = an added operational prerequisite, node substitution = altered execution or observability semantics, and edge substitution = changed dependency logic.

Using the structural similarity value and the automated tests explained in Section 4.1.1, we can provide a unified value for the Behaviour Chain Fidelity (BCF)

$$BCF = 0.5 * auto_{pass} + 0.5 * Sim_{struct}, \quad (2)$$

4.2.2.1 Layered Semantic Enrichment

The node-matching relation is refined in three steps. Each layer subsumes the previous one and adds exactly one defender-relevant constraint, so the layer at which similarity first drops localizes the kind of deviation. All overlap tests use the Szymkiewicz–Simpson coefficient (szss).

L0 (Technique). Nodes match if technique_id is equal. This is the ontology layer.

L1 (Tactic). L0 holds and the tactic sets overlap, szss(T1, T2) ≥ 0.50. A drop here means strategic drift: an inserted execution-oriented step inside a privilege-escalation chain can stay locally plausible while changing the tactical role of that segment.

L2 (Telemetry). L1 holds and the telemetry classes overlap strictly, szss(C1, C2) > 0.50. A drop here means observational drift. This is the critical layer for the trial: a Windows-to-Linux conversion that approximates a Windows PE through a compatibility layer can replace native Windows API effects with Linux process or identity telemetry—which binary scoring misses but telemetry-aware comparison exposes. The threshold is strict, so a boundary overlap of exactly 0.50 fails.

L3 (Sigma logsource). L2 holds and the Sigma logsource categories overlap, szss(L1, L2) ≥ 0.50. A drop here means detectability drift: a step that stays ATT&CK-aligned but no longer maps to an equivalent logsource-bearing detection is, from a defender's view, degraded. As a by-product, this layer is useful for designing Sigma rules to detect variant procedures.

A separate independent L3 relation (L3-i) drops the L2 prerequisite and requires only L0 plus logsource overlap. Section 4.2.4.5 shows why this matters: chained L3 makes detection coverage invisible whenever L2 has already failed, whereas independent L3 measures detectability on its own terms.

Algorithm 1: telemetry-intent parser

For L2 to be a fair cross-platform comparison, the control's telemetry must be expressed in the same vocabulary as the variants. Control steps carry free-text telemetry_expected annotations; Algorithm 1 derives structured telemetry_classes from them using an OS-agnostic keyword map over the four standard classes {file, network, process, identity}.

```
Algorithm 1: parse_telemetry_expected -- structured observable classes from text
Input : free-text annotation t; keyword map K : class -> keyword set
Output: sorted list of observable classes
1  t   <- lower(t)
2  out <- {}
3  foreach (cls, kws) in K do
4      if exists k in kws such that k is a substring of t then
5          out <- out union {cls}
6      end
7  end
8  return sorted(out)
```

Algorithm 2: bipartite GED

Exact GED is NP-hard, so the framework uses the Riesen–Bunke bipartite approximation: build a cost matrix over node substitutions, insertions, and deletions, then solve the assignment with the Hungarian algorithm in $O(n^3)$. The matching predicate is the layer relation of Section 4.2.2.1, so the same GED routine computes every layer's score by swapping the predicate.

```
Algorithm 2: bipartite_ged( C-steps, V-steps, match )
1   nc <- |C|,  nv <- |V|
2   C  <- INF * 1_(nc+nv)x(nc+nv)            // sentinel init (the fix)
3   for i in [0,nc), j in [0,nv) do
4       C[i][j] <- 0 if match(Ci,Vj) else 1  // substitution
5   end
6   for i in [0,nc) do
7       C[i][nv+i] <- 1                       // deletion (diagonal only)
8   end
9   for j in [0,nv) do
10      C[nc+j][j] <- 1                       // insertion (diagonal only)
11  end
12  for i in [0,nv), j in [0,nc) do
13      C[nc+i][nv+j] <- 0                    // null-to-null
14  end
15  (row, col) <- hungarian(C)
16  GED <- sum( C[row, col] )
17  return GED,  max( 0, 1 - GED / max(nc, nv) )
```

The layer predicates stack as follows. match at L0 tests technique equality; L1 adds the tactic overlap; L2 adds the strict telemetry overlap; chained L3 adds the logsource overlap on top of L2; independent L3 requires only L0 and the logsource overlap

Algorithm 3: Sigma injection

Before L3 can be scored, each node must carry the Sigma rules that apply to it. Algorithm 3 attaches a rule to a step when the rule's ATT&CK tag matches the step's technique or when the rule's telemetry overlaps the step's by at least the threshold. The logsource category used for L3 is the final path segment of the rule's logsource, which is process_creation for command-line tradecraft on both platforms.

```
Algorithm 3: inject_sigma( step v, library )
1  out <- {}
2  foreach rule r in library do
3      tm <- ( v.technique_id in r.attack_tags )
4      ov <- szss( v.telemetry_classes, r.telemetry )
5      if tm or ov >= 0.50 then
6          attach r to out with logsource_category = lastsegment(r.logsource)
7      end
8  end
9  return out
```

Using this layered approach, we propose a composite score that keeps adversary-intent preservation primary while treating technical realism and defensive value as secondary dimensions of equal weight:

$$S = 0.4 * BCF + 0.3 * TR + 0.3 * DV \quad (3)$$

The weights in Equation 3 are initial analytic settings pending empirical calibration, not an experimentally validated scoring law.

4.2.3 Experimental setup and workflow

The evaluation compares a native-Windows control procedure against LLM's Linux generated variant procedure. Both graphs are drawn from the same pipeline run. The Linux variant G_v is the full_output stage of the pipeline, used unmodified. The Windows control G_c is reconstructed step-for-step from the pipeline's own conversion record, so that each control step carries the genuine Windows command on which the documented Windows-to-Linux translation acted. Every reconstructed command is grounded in recorded evidence, and the grounding source is stored per step in a win_command_provenance field. Table 7 gives the provenance taxonomy and its distribution over the 29 steps.

Table 7. Provenance of the reconstructed Windows control commands. Each of the 29 control steps is grounded in one recorded source, in strict priority order.

Tag	Priority	Steps	Grounding
conversion_note	1	13	Exact or prefix match against conversion_notes[], original_action (direct ground truth).
cross_platform	4	8	Dual-platform binary (rclone, scp, native PE) retained with the Linux-only sudo prefix stripped.
bitsadmin_map	3	4	wget download of a PE mapped to the documented bitsadmin /transfer form.
de_wine	2	2	Provable inverse of a wine-wrapped PE: wine <pe> <args> → <pe> <args>.
technique_pattern	2b	2	Generalisation of the two documented mstsc↔ssh and netsh↔systemctl conversion patterns to residual steps of the same technique.
Total		29	

Because A1 holds, agreement at L0/L1 is now a genuine measurement that the LLM preserved the technique and tactic sequence across the OS boundary, not a procedure compared to itself. The os_constructs check is likewise a real defect measure: nine wine/.exe Windows constructs survive into the Linux G_v at steps {8, 9, 11, 12, 18, 20, 21, 26, 27}, scored against a true Windows baseline. Two of the 29 control steps are grounded only by technique_pattern; that is, by generalizing the two documented conversion patterns (mstsc↔ssh and netsh↔systemctl) to residual steps of the same technique, rather than by a recorded conversion. These two steps are inferred

reconstructions, not direct ground truth, so the perfect L0/L1 preservation result is contingent on those inferences and would require revisiting if either is incorrect.

The ALPHV/BlackCat procedure is the 29-step ALPHV/BlackCat Windows-to-Linux emulation plan. Its technique spine, shared by both graphs at L0, is T1048.003, T1105, T1021.001×3, T1087.002, T1087.001, T1018, T1003.001×3, T1562.001×2, T1112, T1046, T1077, T1059.001×2, T1021.004, T1059.004, T1083, and T1048.001, covering 16 unique techniques. Table 8 shows representative steps with the control command and the telemetry classes on each side.

Table 8. Representative steps: reconstructed Windows control command and the telemetry classes of G_c (Algorithm 1 parsed) versus G_v. The trio S13/S14/S27 is the source of the L2 loss; all other steps overlap at ≥ 0.667.

Step	Technique	Control command (abridged)	G_c telemetry	G_v telemetry
S1	T1048.003	rclone serve webdav -- addr :8080	file, network, process	file, network, process
S2	T1105	wget http://c2.host/... -O /tmp/cs	file, network, process	file, network, process
S3	T1021.001	ssh user@10.30.10.4 "bash -l"	identity, network, process	identity, network, process
S13	T1562.001	netsh advfirewall set ... state off	file, process	other, process
S14	T1562.001	netsh advfirewall set ... state off	file	other, process
S27	T1059.001	/tmp/collector1.exe /remote ...	file, network, process	identity, process

4.2.3.1 The Sigma Rule Library

Detection content is supplied by a single validated library, sigma_rules_gbse.yml, comprising 49 rules: 19 Linux rules (L01–L19, injected into G_v) and 30 Windows rules (W01–W30, injected into G_c). Every rule targets the process_creation logsource category and is derived from an actual ALPHV/BlackCat command. The library gives complete ATT&CK technique coverage of the procedure on both platforms, so under Algorithm 3 every step receives at least one rule on each side, and the shared process_creation category yields logsource overlap 1.0 for all 29 steps under independent L3. A validation pass corrected four rule-condition defects (an undefined selection identifier in W02; bare selection blocks in L11, W05, and W13 orphaned by a “1 of selection_*” condition) and added eleven Windows rules W20–W30 to close per-step firing gaps, including W29 (T1133) and W30 (T1555) for the gold-control divergence of Section 4.2.4.6. The present paper verified the full 49-rule library against the Sigma specification with zero findings.

4.2.3.2 Pipeline Workflow

The reference build executes the following stages, each of which maps to a component above. For step 6 of this workflow, we assigned manual values to the constants TR, DV, and DV_I (the Defensive Value assigned to the independent-L3 state), to experiment with the composite scoring proposal.

1. Load the OS-agnostic step spine, the Linux variant, the conversion notes, the evaluation result, and the OS-violation list from the pipeline output.
2. Reconstruct G_C: derive each Windows command via the priority order of Table 7, set os=Windows, parse telemetry_expected into structured classes with Algorithm 1, map privilege, and inject the Windows rules.
3. Build G_V: normalize the Linux variant's telemetry, privilege, and tactic fields, and inject the Linux rules.
4. Automated evaluation: run the twelve checks (Section 5.1) including the os_constructs defect count against the Windows baseline.
5. Layer evaluation: run Algorithm 2 once per layer (L0, L1, L2, L3 pre-sigma, L3 chained, L3 independent), recording GED, similarity, and the failing pairs with a per-failure diagnosis.
6. Composite scoring: combine the automated pass rate, the structural similarity, and the manual TR/DV via Equation (3), and test the deployment gate.
7. Gold-control divergence: recover the 7-step technique substitutions from the recorded evaluation result.
8. Persist the enriched control graph, the enriched variant graph, and the full results document as JSON.

4.2.4 Results

4.2.4.1 Automated Evaluation

The automated checks from Section 4.1.1 are re-run for the GBSE layer under two differences. First, a twelfth check, os_constructs_valid, is added to verify that no Windows-only constructs survive into the Linux variant. Second, the two telemetry checks (telemetry_classes_match and extra_telemetry_classes) are evaluated against the control's Algorithm 1–parsed telemetry classes under the strict Szymkiewicz–Simpson threshold of Section 4.2.2.1, rather than the looser class-presence test used in Section 4.1.1. Under these stricter, defender-relevant definitions, eight of twelve checks pass, giving an automated pass rate of 0.6667.

The four failures are: os_constructs_valid (nine steps of the Linux variant still invoke wine for Windows .exe execution, a translation defect against the Windows baseline);

telemetry_classes_match (three steps S13, S14, and S27 whose per-step control/variant telemetry overlap falls at or below the strict 0.50 threshold); extra_telemetry_classes (the variant emits an out-of-vocabulary class, *other*, outside the standard set {file, network, process, identity}); and privilege_progression_match (six of 29 steps mismatch on privilege). All technique-identifier, tactic, diversity, and multi-class checks pass. The pass rate feeds BCF.

This 0.6667 should be read alongside, not in place of, the 91 percent (10 of 11) reported in Section 4.1.1, as both describe the same variant. The privilege check fails in both runs; the difference is that the two telemetry checks that pass in Section 4.1.1 fail here once telemetry equivalence is judged strictly and against Algorithm 1–parsed control classes, and the added os_constructs_valid check fails outright. The drop from 91 percent to 67 percent is therefore not run-to-run noise but the direct effect of replacing permissive presence tests with defender-relevant equivalence tests; which is itself evidence for the central claim of this paper, that binary checks under permissive definitions overestimate translation fidelity.

4.2.4.2 Layered structural results

Table 9 reports GED, similarity, and BCF at each layer. Technique and tactic structure is preserved perfectly; telemetry costs three edits; chained L3 cannot recover because it requires L2 to pass first; independent L3 recovers fully.

Table 9. GBSE layer results (29-step, normaliser 29).

Layer	GED	Sim	BCF	Signal
Baseline (L0)	0.0	1.000	0.8334	Perfect technique preservation; all 29 identifiers match.
L1 + tactic	0.0	1.000	0.8334	All 29 tactic fields match step-for-step.
L2 + telemetry	3.0	0.897	0.7816	3 of 29 fail. S13 szss=0.50 fails strict; S14 szss=0.00; S27 szss=0.333. Root cause: variant other class.
L3 pre-sigma	3.0	0.897	0.7816	Identical to L2; no Sigma rules in source files.
L3 post (chained)	3.0	0.897	0.7816	S13/S14/S27 still fail; chained formulation requires L2 first, so injection cannot bypass the telemetry gate.
L3 post (independent)	0.0	1.000	0.8334	All three failures resolved. S13/S14 via W09/L09; S27 via W15/L15; logsource overlap 1.0 everywhere.

4.2.4.3 L2 Failure root cause

The three L2 failures originate on the variant side. At S13 and S14 (T1562.001) the variant emits an unmapped other telemetry class, and at S27 (T1059.001) the variant substitutes {identity, process} for the control's {file, network, process} while the tactic stays unchanged. Because the control telemetry is fully structured by Algorithm 1 in this run, the failures reflect variant output quality rather than gaps in the gold annotation.

Table 10. L2 analysis. Three of 29 steps fail; all are variant-side telemetry issues.

Step	Technique	Telemetry (control → variant, overlap)	L2
S13	T1562.001	[file, process] → [other, process], szss=0.50	FAIL
S14	T1562.001	[file] → [other, process], szss=0.00	FAIL
S27	T1059.001	[file, net, proc] → [identity, proc], szss=0.333	FAIL
S2	T1105	[file, net, proc] → [file, net, proc], szss=1.00	PASS
S10	T1021.001	[id, net, proc] → [id, net, proc], szss=1.00	PASS
S† (24 others)	various	all szss ≥ 0.667	PASS

The classification of S13 turns on a strict inequality at exactly the threshold value: its overlap of szss = 0.50 fails the strict test (> 0.50) by the smallest possible margin. Under a non-strict threshold (≥ 0.50), S13 instead passes; the L2 edit distance falls from 3.0 to 2.0, L2 structural similarity rises from 0.897 to 0.931, and the L2 BCF rises from 0.782 to roughly 0.799. (S14 and S27, with overlaps of 0.00 and 0.333, fail under either convention, so the binary telemetry_classes_match check remains a failure regardless.) The telemetry-layer result is therefore sensitive to this single uncalibrated cutoff, whose value future work should set empirically rather than by convention.

4.2.4.4 Composite scores and the deployment gate

Using illustrative hand-assigned constants of TR (0.43) and DV (0.51) and Equation (3) we obtain the results shown in Table 11. The best state classifies Medium Fidelity, while the three telemetry-degraded states (L2 and chained L3) fall to Low Fidelity. Under the assigned DV_I = 0.65, independent L3 ranks highest; this margin over baseline equals DV_I − DV and is not independently derived.

Table 11. Composite scores (29-step run, automated pass rate 0.667). States span Medium and Low Fidelity.

State	Sim	BCF	TR/DV	S	Classification
Baseline / L1	1.000	0.8334	0.43 / 0.51	0.6154	Medium Fidelity
L2 telemetry	0.897	0.7816	0.43 / 0.51	0.5946	Low Fidelity
L3 pre-sigma	0.897	0.7816	0.43 / 0.51	0.5946	Low Fidelity
L3 post (chained)	0.897	0.7816	0.43 / 0.51	0.5946	Low Fidelity
L3 post (indep.)	1.000	0.8334	0.43 / 0.65	0.6574	Medium Fidelity

Under these illustrative TR and DV values, no state reaches the S ≥ 0.80 deployment gate. The gate is included here only to show how the composite would route a translation, not as a calibrated pass/fail threshold. For the best state (BCF = 0.8334, DV = 0.65), clearing the gate would require TR ≈ 0.906, well above the 0.43 assigned in this study. Because TR and DV are manual inputs rather than derived quantities, this result illustrates the gate's sensitivity to the realism and defensive-value inputs, not a measured deployment verdict.

4.2.4.5 Sigma Injection: chained versus independent

The distinction between the two L3 relations is operationally decisive. Under chained L3, the three steps that already failed L2 stay failed regardless of detection coverage, because the chained predicate evaluates the telemetry gate first. Under independent L3, all three are resolved through technique-tag matches: S13 and S14 attach W09/L09 (T1562.001) and S27 attaches W15/L15 (T1059.001), each giving logsource-category overlap 1.0. All 29 steps pass independent L3, and the composite rises from 0.595 to 0.657 under the assigned DV_I, a relative improvement of about 11%.

4.2.4.6 Gold-control technique divergence

The 29-step reconstruction aligns each control step to its documented Windows pre-image, so techniques agree by construction and L0 reports zero mismatches. That is correct for measuring procedure-level fidelity, but it suppresses a coarser signal that the pipeline recorded separately. The pipeline's evaluation result references a 7-step Windows gold control scored against the 29-step Linux variant. Recovering that residue surfaces a genuine technique-level divergence at six of the seven gold steps, shown in Table 12.

Table 12. Technique-level divergence between the 7-step Windows gold control and the 29-step Linux variant. Six of seven gold steps are reassigned to a different ATT&CK technique.

Gold step	Windows gold technique	Linux variant technique
1	T1133 External Remote Services / RDP	T1048.003 Exfiltration over unencrypted non-C2
3	T1087.002 Account Discovery: Domain	T1021.001 Remote Services: RDP
4	T1105 Ingress Tool Transfer	T1021.001 Remote Services: RDP
5	T1021.001 Remote Services: RDP	T1087.002 Account Discovery: Domain
6	T1105 Ingress Tool Transfer	T1087.001 Account Discovery: Local
7	T1555 Credentials from Stores	T1018 Remote System Discovery

Two substitutions are operationally significant. At gold step 1 the Windows external-remote-services vector T1133 (RDP exposure) maps to T1048.003 (exfiltration over an unencrypted non-C2 protocol), and at gold step 7 the Windows credential-store access T1555 maps to T1018 (remote-system discovery). These are technique reassignments, not faithful preservations, and a procedure-aligned view cannot expose them. At the granularity of the pipeline's own gold control encodes, the Windows-to-Linux translation does not preserve all ATT&CK techniques. The two coverage rules W29 (T1133) and

W30 (T1555) are motivated precisely by these two substitutions and are present in the validated library.

4.2.4.7 Guidance from the Graph Based Analysis

The following five items follow directly from the results and are stated as remediation guidance, with the layer each one moves.

G1. Remove the other telemetry class from variant output. The variant emits an unmapped other class at S13 and S14, which fails the strict L2 threshold against the control's {file, process}. Post-processing that maps other to a concrete class via D3FEND resolves two of the three L2 failures.

G2. Re-translate S27 with native tradecraft. At S27 the variant drifts to {identity, process}. Re-running the step with a native Linux interpreter for T1059.001 restores telemetry that matches the source and closes the third L2 failure.

G3. Use independent L3 as the primary detectability metric. Chained L3 hides detection coverage whenever L2 has failed. Independent L3 measures detectability on its own terms, is more reliable in sparse-control settings, and reaches Sim_struct=1.000 on all 29 steps, raising the composite from 0.595 to 0.657.

G4. Re-run the trial with a native Linux translation. Because the deployed variant retains wine/.exe at nine steps, the four-layer evaluation cannot yet exercise genuine technique and telemetry differences for those steps. A native re-translation would make the full evaluation meaningful and would likely surface deviations the compatibility layer currently masks.

G5. Recalibrate the gate and the manual instrument. The gate S ≥ 0.80 demands TR ≥ 0.906, which is unachievable at the present TR = 0.43. A two-tier gate, with S ≥ 0.60 for conditional deployment (achievable now) and S ≥ 0.70 for full deployment, is more useful.

4.3 Products for Future LLM Evaluation

A significant outcome of this research is the development of products for future LLM evaluation, these products are described in Table 13.

Table 13: List of Products for Future LLM Evaluation

Product	Description	Location
Scoring Question List	The complete list of questions generated and evaluated by our team to include in the LLM-output-evaluating methodology. Includes 40 questions.	https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis "Final Scoring Questions List" tab

Final Manual Questionnaire	The manual questionnaire finalized including questions our team used to provide results for the sample LLM output. Does not include questions which were eliminated during manual evaluation (specified in Scoring Question List). Includes 9 of the original Scoring Question List questions.	https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis "Final Manual Questionnaire" tab
LLM-Environment-Translation and Formatting Codebase	The codebase containing the repeatable automated procedures to generate formatted procedures in JSON, variant environment output, and enrich that output with data derived from ATT&CK and D3FEND.	Documentation and Code: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis
JSON Procedure Format	The machine-readable JSON format used to format adversary procedures for LLM generation, with fields tailored for manual and automated evaluation. This provides a repeatable standard for future evaluation and refinement.	pydantic models used to specify format are located in MITRE-FRE-ability-translation-analysis src/procedure_generator/models.py
Automated Evaluation Code	The code used to conduct automated evaluation of the variant procedures. The logic is documented in the README of the fre-research github. Evaluates 11 of the original Scoring Questions List questions.	Documentation: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis Core: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis
Graph Based Structural Analysis	The code used to conduct the graph based structural analysis	Documentation and Code: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis
Additional sigma rule	The new Sigma rules obtained after preforming graph based structural analysis	Documentation and Code: https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis

5 Areas for Further Development

We present the methodology, use case, and results on this research as an initial step in evaluation of LLM automatic adversary translation capabilities. Considering this, we have identified the following areas for further development.

5.1 Inter-Rater Reliability as a First-Class Concern

Manual evaluation is the only modality in this framework that captures the properties of a translation that depend on judgment and resist a simple binary answer. Even so, the trial showed that its outputs are presently governed more by the identity of the evaluator than by the artifact under review. Across five reviewers, the aggregate PASS rate was

46 percent. When PASS rates were instead computed for each evaluator across all nine questions, however, they ranged from 22 percent to 67 percent, a spread of 45 percentage points, and when they were computed across only the questions each reviewer actually answered, the spread widened to 78 points, ranging from 22 percent to 100 percent. Reviewers also disagreed about whether questions 3.1.6, 3.1.8, and 3.2.11 were applicable at all. Because disagreement of this magnitude is present, the headline manual metrics cannot yet be attributed with any confidence to the translation quality of the LLM, since a different panel could plausibly have returned a materially different verdict. The methodology is meant to be repeatable and independent of the particular model under test, so that scores stay comparable across LLMs and across research teams. For that reason, unquantified disagreement on this scale is the single most serious threat to the validity of the manual tier, and consequently to any composite score that draws on the manual inputs, namely TR and DV in Equation 3.

Future applications of this methodology should therefore treat agreement among raters, reported as inter-rater reliability (IRR), as a result of equal standing to the PASS and FAIL rates themselves, and it should be computed for every trial rather than only when the results happen to look suspect. Raw percent agreement is not enough, because it does not correct for the agreement that would be expected by chance, and that chance agreement is substantial when the scale offers three categories of PASS, FAIL, and N/A. The research therefore prescribes reporting an agreement coefficient that corrects for chance, both for each question and in aggregate. Because the response matrix from the trial is incomplete, since the answer rate for each evaluator ranged from 56 percent to 100 percent, and because it contains a genuine third category, Krippendorff's α is better suited than the more familiar κ variants. In particular, it tolerates missing data and any number of raters without modification, whereas Fleiss' κ assumes a fully crossed design and Cohen's κ is limited to two raters. Where a fully crossed subset of two raters does exist, Cohen's κ can be reported alongside α for ease of interpretation. The coefficients should be read against published benchmarks, such as those of Landis and Koch, while bearing in mind that those bands are only heuristic. As a working bar, the research suggests treating a coefficient below roughly 0.40 on any given question as a signal that the question itself, and not merely the LLM output, needs revision, and reserving firm comparisons across models for questions that reach at least 0.60 to 0.67.

Reporting agreement only surfaces the problem, whereas reducing it calls for changes in how the evaluation is conducted, and three such changes are recommended. The first is calibration. Before scoring independently, evaluators should score a shared reference sample and then reconcile their disagreements in a norming session, so that the operational meaning of PASS, FAIL, and N/A becomes shared rather than assumed. The second is anchored criteria. Each question should carry an explicit definition of what passing and failing mean, together with at least one worked example, and particular care should go to the applicability conditions that the trial showed were read inconsistently. The N/A category in particular should be defined to mean that the question does not apply to the artifact, and that meaning should be kept apart from the separate case in which the evaluator simply could not determine an answer. The present data appear to blur the two, because question 3.2.11 was ruled N/A both when

its precondition of additional telemetry was absent and, plausibly, when reviewers were uncertain. The third is adjudication. Disagreements that survive calibration should be settled through a documented protocol, whether by consensus discussion or by majority vote with a recorded rationale, rather than being averaged silently, so that each final verdict on a question stays traceable.

Treating agreement as a primary concern also closes a loop with the work on refining evaluation questions described in Section 5.10, because questions that show persistently low agreement even after calibration then become candidates for rewording, decomposition, or retirement, and their agreement should be measured again as the question bank evolves. Reviewer selection and experience, discussed in Sections 5.10.3 and 5.11.3, interact with all of the above, since an agreement coefficient is only meaningful among reviewers who are competent to judge the artifact. Reporting the composition and experience of the panel alongside the coefficient therefore lets a reader separate low agreement that stems from an ambiguous question from low agreement that stems from a panel without the necessary expertise. For all of these reasons, every future trial should publish, for the manual tier, the agreement coefficient, the makeup of the rater panel, and the adjudication method that was used, so that a manual PASS rate is never again reported as a bare number.

5.2 Failure Mode Taxonomy

This research detected four failure modes in the automatic tests evaluating the LLM translation capabilities. We expect these modes to grow as more adversaries, and OS pairs are analyzed using our proposed methodology.

5.3 Honest Grounding Statement

Graph evaluation and enrichment layers are prototype-stage, demonstrated on one case study, with unvalidated thresholds. Particularly, the values of TR, DV, and DV_I used in Section 4.2.4.4 can be tuned through more experimentation.

5.4 Procedure Generation Refinements

5.4.1 Improve Pre-Generation Assessment for Prompt Engineering

Time did not permit completing manual or optimization of automated pre-generation assessments. Future experiments could develop more thorough processes for pre-generation assessment to get a better sense of how different LLMs process and develop responses, and to refine the input prompt (Appendix B describes one potential framework).

5.4.2 Improved Prompting Through Further Generation Decomposition

Over the course of the experiment the procedure conversion process evolved from a one-shot generation of the new procedure to a complex, multistep process centered around a process of procedure ingest, translation to the structured format, conversion to the target OS, enrichment, and automated evaluation. This process is mapped to nodes in the LangGraph workflow procedure. As of completion of this research, some of these nodes are overloaded. This prompt congestion is pervasive throughout the generation pipeline and is a symptom of time constraints and informal prompt grading. Given more time this issue could be ameliorated with more granular step partitioning. For example, the translation node handles parsing the unstructured text into steps, identifies commands, maps to ATT&CK TTP's, and determines the OS each of the steps occur in. In this example, the translation agent's context window houses information relevant to multiple disjoint reasoning steps which could be further decomposed into individual steps to minimize irrelevant context.

5.5 Multiple Adversary Emulation Procedures

This experiment tested just one adversary and set of procedures. Future experiments should evaluate more sets of procedures for additional adversaries. Research might also consider querying LLMs about specific adversaries or training LLMs using adversary data to get a better understanding of adversary emulation capability, and if some LLMs are better at generating realistic procedures.

5.6 Improve Human Readability of Output

The LLM outputs procedures in JSON format to optimize machine-compatibility, and because no standard exists for the original input Markdown that an LLM could use to format the output. Automated evaluation also generates a Comma-Separated Values (CSV) output. These formats worked for this use case since they are easily standardized, easily compared against one another, and human readable by technical evaluators. Other, more human readable formats (such as a standardized Markdown template) might assist with broadly communicating emulation plans for presentation.

5.7 Model Selection

Model selection was constrained to the availability of local LLM hosting infrastructure. Nemotron 3 Nano and GPT OSS 120b were used throughout testing but the utilization of frontier models such as GPT-5.2 or Claude Opus 4.8 would have likely yielded better

results. It is also worth evaluating models of more and less advanced capability, depending on the application and infrastructure available to operators.

5.8 Domain Specific Knowledge RAG

The current generation pipeline has no Retrieval Augmented Generation (RAG) system. Integrating with a RAG would improve key areas of the generation:

1. **MITRE ATT&CK mapping.** The LLM often hallucinated technique IDs, assigned an improper ID, or does not assign the proper sub-technique ID. Indexing the ATT&CK STIX data and providing relevant data through RAG retrieval techniques would likely improve the ID assignments.
2. **Command Translation.** The LLM doesn't always pick the most optimal commands for translation and often generates incorrect commands. Further errors can also be introduced when the LLM generates flags and syntax that modify or direct the command's function. These problems may be ameliorated in two ways. One action is to load relevant man pages into the RAG database so that documentation is accessible to the LLM during the conversion and refinement steps. Another action (more difficult to implement) would be to build a dataset of functionally equivalent commands (mapping a Windows command to equivalent Linux commands and providing some context).

5.9 Automated Evaluation Improvements

As the scoring matrices evolved over the course of the project, the team identified questions which could be turned into automated measures in an ad-hoc manner. These automated measures were built into a final step of the automated generation pipeline and then exported along with the converted procedure. Since the evaluation is calculated and exported altogether with the final converted procedures, there was no way to feed results back into the generation pipeline to improve the generated procedures.

Future iterations of the generation pipeline could include an iterative refinement feedback loop (self-refine) and evaluation calculations during the generation process. Adding conditions, such as syntax validation, to initiate re-generation could improve the accuracy of the translated commands. Additionally, re-generations of the procedure with context about the previous generations' results may yield better final generation results.

5.10 Evaluation Questions and Scoring Refinement

5.10.1 Further Development of Evaluation Questions

The research team was unable to complete the scoring methodology for several evaluation questions which required deep subject-matter expertise or instinctual logic jumps. Future research could consider refining the questions and scoring methodology for these more difficult and complex questions.

5.10.2 Metrics and Weighted Scoring

Future development of the test question catalogue should standardize metrics to make scoring more comparable across LLMs and create a weighting methodology considering that some questions attempt to assess more complex or important tasks.

5.10.3 External Review of Questions and Evaluator Familiarity

The small team size and time available for testing required that the same people who developed the questions for evaluation also conducted the final evaluation. Future testing should include evaluators who are totally new to the question set to help understand whether questions are comprehensible, transferrable to multiple different testers, and sufficiently evaluate LLM output.

5.11 Evaluation Process Refinement

5.11.1 Reliance on Static Frameworks and Ontologies

Due to time constraints, both automated and manual evaluation relied on static ATT&CK and D3FEND resources. While these are good starting points, it requires making assumptions about the progressive state of the operating environment when executed (ex. changes to the environment, telemetry generated), as opposed to actual operational execution. Further research could conduct actual operational testing to validate the assumptions made by using these frameworks, and more enrichment of static data could improve automated and manual evaluation capability without operational execution.

5.11.2 Manual Evaluation

The research produced a limited number of automated/standardized means for evaluating questions. Manual evaluation was conducted by a limited team of experts who may exhibit biases based on the order of analysis, operational experience, awareness of the experiment measure development process, etc. Future experiments

could consider taking more time to develop standardized scripts or automated methods to make evaluation repeatable across a larger number of LLMs.

5.11.3 Evaluator Selection and Experience

Some of the small research team members were previously familiar with the BlackCat procedures used in the test or had more/less experience than others with adversary emulation and cyber threat intelligence subject-areas. Future research could decide on an ideal mix/standard of experience levels for evaluation. Future research could consider the utility of outputs for operators with varying levels of experience and familiarity with tested adversaries.

5.11.4 Varying Adversary Procedures in Different Operating Environments

In the real world, adversaries may not only translate commands for execution one-to-one between environments. In some cases, there may be different operators or operating procedures depending on the environment.

6 Conclusion

Significant work remains to create a comprehensive, repeatable, and appropriately lightweight methodology for evaluating LLM-assisted adversary emulation across diverse operational environments. This research contributes a baseline set of evaluation questions, an initial pipeline for converting human-readable procedures into a machine-readable format and translating them to fit a target environment specification, and a methodology for evaluating LLM output.

7 References

MITRE. MITRE ATT&CK: Adversarial Tactics, Techniques, and Common Knowledge. Version 16. MITRE Corporation, 2026. https://attack.mitre.org/

MITRE. MITRE D3FEND: A Knowledge Graph of Cybersecurity Countermeasures. MITRE Corporation, 2026. https://d3fend.mitre.org/

MITRE. ATT&CK Evaluations: ALPHV BlackCat Emulation Plan. MITRE Engenuity. https://github.com/attackevals/ael/blob/main/ManagedServices/alphv_blackcat/Emulation_Plan/

SigmaHQ. Sigma Detection Format: Generic Signature Format for SIEM Systems. SigmaHQ Project, 2026. https://github.com/SigmaHQ/sigma

Weidman, S. et al. LOLBAS: Living Off the Land Binaries, Scripts, and Libraries. https://lolbas-project.github.io/

Takey, M. et al. GTFOBins: Unix Binaries That Can Be Used to Bypass Local Security Restrictions. https://gtfobins.github.io/

Bunke, H. and Shearer, K. A Graph Distance Metric Based on the Maximal Common Subgraph. Pattern Recognition Letters, vol. 19, no. 3-4, 1998, pp. 255-259.

MITRE AI Platform. Internal LLM Hosting Infrastructure. MITRE Corporation, 2026. https://ai-platform.pages.mitre.org/resources/llm-endpoints/

K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing, 27(7), 2009.*

Z. Zeng et al. Comparing Stars: On Approximating Graph Edit Distance. *PVLDB, 2:25–36, 2009.*

Z. Gou et al. CRITIC: LLMs Can Self-Correct with Tool-Interactive Critiquing. *ICLR, 2024.*

T. Roth and T. Patzke. "Sigma: Generic Signature Format for SIEM Systems". *BSides Munich, 2017.*

MITRE Corporation. CALDERA: Cyber Adversary Language and Operations*, v4. 2023.*

8 Appendices

8.1 Appendix A: Structured Prompt Example

Role: You are a technical documentation system that reproduces procedures exactly as provided. You do not interpret, correct, improve, or explain. You function as a precise copying mechanism that preserves all original content including any errors, unconventional formatting, or suboptimal approaches.

Task: Reproduce the following adversary procedure exactly as written. Do not:

- Fix any errors (syntax, logic, or typographical)
- Reorder any steps
- Add explanations, comments, or context
- Remove or omit any content
- Improve or optimize any commands
- Translate or convert any elements

Your only task is verbatim reproduction.

Format: Output the procedure in the exact same format as the input:

- Preserve all line breaks
- Preserve all spacing and indentation
- Preserve all punctuation and special characters
- Do not wrap in code blocks unless the original uses them
- Do not add headers, labels, or section markers
- Do not add any preamble or postamble text

Adversary Procedure: [CONTROL PROCEDURES HERE]

Context: [CONTEXT HERE IF ANY]

8.2 Appendix B: Proposed Pre-Generation Assessment Methodology

The research team chose two categories of assessment to evaluate prompt effectiveness:

1. **Pre-generation assessment** before the LLM processes a procedure to produce a final result (including checkpoints for processing input within the generation workflow). Pre-generation assessment may influence tweaks to the prompt template and/or the generation workflow to create the best chance of an optimal outcome.
2. **Post-generation assessment** after the LLM has generated its output (the replication or variant procedures in the original and target OS, respectively), measuring the quality of the LLM's final output.

Pre-Generation Assessment tests the LLM's comprehension of the task by asking specific questions about the procedure it was given via the prompt. Potential iterations may include changing the level of detail in the prompt or input to get the best outcomes in the replication or variant scenarios, as evaluated by the post-generation assessment metrics.

The LLM's comprehension of the task can be tested by asking specific questions about the task it was given such as the below.

Criterion	Description	Scoring
Procedure Objective Identification	Correctly states the goal of the procedure	0-3 scale
Step Enumeration Accuracy	Identifies key steps	Ratio: identified / actual
Platform Recognition	Correctly identifies source OS/environment	Binary
Task Understanding	Correctly restates what it's being asked to do	0-3 scale
Prerequisite Awareness	Identifies dependencies, privileges, conditions	Count of correct identifications

Alternatively, a decomposition approach can be used where the LLM parses the procedure into structured fields which are then compared against ground truth using semantic similarity, binary matching, or step count accuracy.

Field	Ground Truth Comparison	Metric
Objective	Match against known objective	Semantic similarity or exact match
Platform	Correct or not	Binary
Steps	Correct parsing	Step count match + content accuracy

To evaluate these prompts, populate the procedure and context fields, run comprehension and decomposition analysis, execute the task, and then evaluate the output using the scoring rubric. With this scoring information, it is possible to conduct iterative prompt refinement to tune prompts for specific replication and variant scenarios.

8.3 Appendix C: List of Supplementary Resources

Resource	Link	Description	Notable Artifacts
Research Github Repository	https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis	Github Repository containing code, documentation, and programmatically-generated artifacts.	
Scoring Questions and Results Spreadsheet	https://github.com/FujitsuResearch/MITRE-FRE-ability-translation-analysis/LLMEnvTranslationScoringQuestions.xlsx	Spreadsheet containing the final question bank, manual analysis questionnaires, and manual analysis results.	• FinalScoringQuestionList tab: contains the final list of scoring questions • FinalManualQuestionnaire tab: contains the final list of manual evaluation questions following our trial manual evaluation process • AggregatedManualEval tab: contains the results of the manual evaluation including all evaluators • Manual Questionnaire Template: the template for manual evaluation provided for our trial manual evaluation • Manual Questionnaire 1-5: the filled-out manual questionnaires completed by evaluators during our trial.

8.4 Appendix D: Enriched Procedure-Graph Schema

Each step node carries the thirteen attributes consumed by the matching layers. The schema below is the structure of the enriched control graph (29 nodes, 28 sequential edges, automated 8/12). The variant graph uses the identical schema with the Linux rule family and source os=linux.

```
{
  gbse_schema_version: "2.0-winlin",
  sigma_library: "sigma_rules_gbse.yml -- Windows W01-W30 (30 rules)",
  source: "Windows G_c reconstructed from conversion_notes + de-wine + bitsadmin map",
  metadata: {
    procedure_id: "alphv_blackcat_windows_ctrl",
    source_os: "windows",
    graph_topology: { V: 29, E_sequential: 28 },
    enrichment: "Algorithm 1: telemetry_expected -> telemetry_classes",
    automated_evaluation: { pass: 8, total: 12, auto: 0.6667 }
  },
  procedure: { action_sequence: [
    {
      step_id: 3,
      technique_id: "T1021.001",
      parent_technique_id: "T1021",
      tactic: ["initial_access"],
      telemetry_classes: ["file","identity","network","process"],
      execution_level: "non-elevated",
      privilege_context: "user",
      privilege_delta: "drop",
      sigma_rules: [
        { rule_id: "...", name: "W03", logsource: "windows/process_creation",
          logsource_category: "process_creation", attack_tag: "T1021.001",
          match_type: "technique", tele_overlap: 0.75 }
      ],
      os_mappable: true, data_flow: null, omission_reason: null,
      win_command_provenance: "conversion_note"
```

```
    }
  ] }
}
```

8.5 Appendix E: Sigma Rule Library Catalogue

The complete 49-rule library is delivered in sigma_rules_gbse.yml. Table 7 lists every rule with its OS family, ATT&CK mapping, and severity. All rules use the process_creation logsource category.

The validated 49-rule Sigma library (19 Linux, 30 Windows).

ID	OS	ATT&CK	Title	Level
L01	Linux	T1048.003	ALPHV rclone WebDAV Listener on Attacker Infrastructure	HIGH
L02	Linux	T1105	Attacker C2 Control Server Binary Execution from Build Path	CRITICAL
L03	Linux	T1021.001	SSH Interactive Bash Login Shell Used for Lateral Movement	MEDIUM
L04	Linux	T1087.002	ldapsearch Domain User Account Enumeration via LDAP	MEDIUM
L05	Linux	T1018	ldapsearch Domain Computer Account Discovery	LOW
L06	Linux	T1087.001	Local Account Enumeration via /etc/passwd Read	LOW
L07	Linux	T1105	wget Executable Binary Download to Staging Path	HIGH
L08	Linux	T1003.001, T1204.002	Wine Windows PE Execution for Credential Access	CRITICAL
L09	Linux	T1562.001	systemctl Security Service Stop and Disable	HIGH
L10	Linux	T1112	sysctl Kernel Security Parameter Modification	HIGH
L11	Linux	T1003.001	gcore Process Memory Dump for Credential Extraction	HIGH
L12	Linux	T1048.003, T1071.001	rclone Data Exfiltration to Remote WebDAV Endpoint	HIGH

ID	OS	ATT&CK	Title	Level
L13	Linux	T1046, T1595.001	nmap Ping Sweep Internal Network Discovery	MEDIUM
L14	Linux	T1077	scp Batch Deployment to Multiple Internal Hosts via Loop	HIGH
L15	Linux	T1059.001, T1204.002	Windows PE or Wine Binary Execution from Staging Path	HIGH
L16	Linux	T1021.004, T1105	scp Lateral File Transfer of Tool or Payload	MEDIUM
L17	Linux	T1059.004	SSH Remote Shell with chmod and Ransomware Execution	CRITICAL
L18	Linux	T1083	smbclient Admin Share File Collection via Get	HIGH
L19	Linux	T1048.001	scp Exfiltration to External Attacker-Controlled Host	HIGH
W01	Win	T1048.003	rclone WebDAV Server Started on Windows Host	HIGH
W02	Win	T1105	C2 Framework or RAT Binary Executed from Non-Standard Directory	MEDIUM
W03	Win	T1021.001	mstsc.exe RDP Connection to Internal Host	MEDIUM
W04	Win	T1087.002	PowerShell or nltest Domain User Account Enumeration	MEDIUM
W05	Win	T1018	PowerShell AD Computer Discovery	LOW
W06	Win	T1087.001	net.exe Local User Account Discovery	LOW
W07	Win	T1105, T1197	bitsadmin PE Binary Staging Download	HIGH
W08	Win	T1003.001	InfoStealer LSASS Credential Access via Named Pipe or Direct Read	CRITICAL
W09	Win	T1562.001	Security Software Process Terminate or Service Stop	HIGH
W10	Win	T1112	Registry ASLR Exploit Mitigation Disable	HIGH
W11	Win	T1003.001	ProcDump or Task Manager LSASS Full Memory Dump	CRITICAL
W12	Win	T1048.003	rclone Copy Exfiltration to Named Remote	MEDIUM
W13	Win	T1046, T1595.001	nmap or Network Port Scanner Execution on Windows	MEDIUM

ID	OS	ATT&CK	Title	Level
W14	Win	T1077	PsExec or Admin Share Lateral Tool Deployment	HIGH
W15	Win	T1059.001, T1486	PowerShell or PE Execution from Temp Path	HIGH
W16	Win	T1021.004, T1105	pscp.exe or WinSCP SCP File Transfer	MEDIUM
W17	Win	T1059.004, T1059.003	CMD Inline Command via SSH Channel Remote Execution	MEDIUM
W18	Win	T1083	net.exe Admin Share File Discovery and Collection	MEDIUM
W19	Win	T1048.001	robocopy or xcopy External Host Exfiltration	HIGH
W20	Win	T1021.004, T1105, T1048.001	OpenSSH scp.exe Secure Copy Transfer or Exfiltration	HIGH
W21	Win	T1021.004, T1059.004	OpenSSH ssh.exe Interactive Remote Command Execution	HIGH
W22	Win	T1105	wget.exe or curl.exe LOLbin Ingress Tool Transfer	MEDIUM
W23	Win	T1562.001, T1562.004	Native Firewall Disable or Microsoft Defender Tampering	HIGH
W24	Win	T1046	PowerSploit Invoke-Portscan Network Service Discovery	MEDIUM
W25	Win	T1059.001, T1486	PE Execution from User-Writable Temp Path or BlackCat Token Args	HIGH
W26	Win	T1003.001, T1555	Non-Standard InfoStealer Binary LSASS or Credential Access	HIGH
W27	Win	T1105, T1071.001	Cross-Platform C2 Server Binary Execution from Resource Path	MEDIUM
W28	Win	T1083, T1135	PowerShell Get-ChildItem Admin-Share File Discovery	MEDIUM
W29	Win	T1133	External Remote Services Enablement or Exposure (RDP/SSH)	HIGH
W30	Win	T1555, T1555.003	Credentials from Password Stores Access	HIGH

A representative rule, L01, shown in full to fix the format used throughout the library:

```
title: ALPHV rclone WebDAV Listener on Attacker Infrastructure
id: aa255d9f-995d-41a9-84ba-e4949f564c97
name: L01
status: experimental
description: >
  Detects rclone started in WebDAV server mode, binding to a non-loopback
  address. Procedure S1: rclone serve webdav /srv/http --addr 176.59.1.18:8080
  (T1048.003). Attacker hosts a WebDAV endpoint to receive exfiltrated data.
author: Elmisery, Fujitsu Research of Europe
date: 2026/06/05
tags:
  - attack.exfiltration
  - attack.t1048.003
  - attack.command_and_control
logsource:
  product: linux
  category: process_creation
detection:
  selection_serve:
    Image|endswith: '/rclone'
    CommandLine|contains|all: ['serve', 'webdav']
  filter_loopback:
    CommandLine|contains: ['--addr 127.', '--addr ::1', '--addr localhost']
  condition: selection_serve and not filter_loopback
falsepositives:
  - Legitimate rclone WebDAV mounts for cloud storage workflows
level: high
x-gbse-telemetry: [file, network, process]
x-gbse-steps: [1]
```

Fujitsu Research of Europe Limited, Security Science Research Group. In collaboration with MITRE Research. This document describes a defensive evaluation methodology and the detection content it produced; all attack tradecraft referenced is drawn from public threat-intelligence reporting on ALPHV/BlackCat.