Work-in-Progress Submission Cover Page

Formal Verification of Resource-Constrained Cyber Strategies across Macro, Meso, and Micro Scales

Assertion of Novelty: I, Jonathan Goohs Jr., assert that this manuscript is unpublished. No version of this work has appeared in any peer-reviewed proceeding, online or in print. This submission describes a work-in-progress, and revisions can be incorporated prior to any future submission for full publication.

Prior Disclosures: "There are no prior non-archival pre-submissions related to this work."

Primary Topic Area: Design principles, tools, and architectures for provably secure systems and continuous monitoring and continuous authorization

Contact Information:
Jonathan Goohs Jr.
Naval Postgraduate School
Monterey, California, USA
jonathan.goohs@nps.edu

WiP: Formal Verification of Resource-Constrained Cyber Strategies across Macro, Meso, and Micro Scales

Jonathan Goohs Jr.*
Naval Postgraduate School
Monterey, California, USA
jonathan.goohs@nps.edu

Adam Pease†
Naval Postgraduate School
Monterey, California, USA
adam.pease@nps.edu

Abstract

Defensive cybersecurity strategies are governed by rigid operational constraints, primarily limited maintenance windows and finite budgetary allocations. Current decision-support tools lack the formal machinery to prove that a generated strategy is logically consistent with these constraints or provide traceability across macro (budgetary), meso (policy), and micro (technical) scales. This work-in-progress paper introduces a verification pipeline utilizing a new cyber-domain ontology extension to the Suggested Upper Merged Ontology (SUMO) [22]. We suggest the combination of game-theoretic strategy generation with automated theorem proving (ATP) to perform proof-by-refutation on candidate strategies, which are verifiably safe under an Adversarial Knapsack constraint model[14].

CCS Concepts

• **Computing methodologies** → **Ontology engineering**; • **Security and privacy** → **Network security**; • **Theory of computation** → **Logic**.

Keywords

Cyber Strategy, Game Theory, SUMO, Temporal Logic, Automated Reasoning, Automated Theorem Proving, Verifiable Software, Resource Allocation

ACM Reference Format:
Jonathan Goohs Jr. and Adam Pease. 2026. WiP: Formal Verification of Resource-Constrained Cyber Strategies across Macro, Meso, and Micro Scales. In *Proceedings of Hot Topics in the Science of Security Symposium (HotSoS) (Hot Topics in the Science of Security Symposium '26).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction and Problem Statement

The optimization of cybersecurity resource allocation is currently hindered by a fundamental verification gap across organizational scales. While a Chief Information Security Officer (CISO) may design a budget at the macro-level and describe operational policies at the meso-level, there is no formal mechanism to ensure that the resulting technical actions at the micro-level are mathematically consistent with those constraints[10]. This strategic fragmentation often results in "meso-layer failures," where individually sound technical processes fail to align with higher-level organizational rules or resource limitations[4].

Current industry standards, such as the MITRE ATT&CK framework[19], function primarily as taxonomies for classification rather than formal ontologies[25] for automated theorem proving systems to reason on[29]. These systems lack the logical axioms required to derive the consequences of technical actions or enforce arithmetic constraints across system states[7]. Without a consistent mathematical specification, defensive actions like 'patching' are treated as opaque symbols in a database rather than standard processes with defined resource costs and temporal requirements[2].

SUMO provides the upper-level semantics required to close this verification gap. Its axiomatized notions of Process, TimeInterval, temporal containment, state/attribute change, and quantitative measures supply the formal machinery needed as the basis to model terms such as "a patch process consumes capacity during an interval" and to allow Vampire to refute strategies that violate them. Our cyber module does not replace SUMO; it specializes it by introducing domain-specific predicates whose meaning is grounded in SUMO's existing process and temporal structure[21].

Within this unified framework, our contribution is two-fold: (1) the development of a cyber-domain ontology module that extends a formal ontology (the SUMO) with a formal theory of cybersecurity resource mapping, and (2) a pipeline that applies this ontology to game-theoretic strategy generation[14]. This framework enables proof-by-refutation verification, ensuring that any proposed defensive strategy is logically consistent within defined budget, personnel, and maintenance window constraints.

1.1 The Necessity of Formal Grounding

Most current cybersecurity standards function as taxonomies or relational schemas rooted in Propositional or Description Logic (DL)[1]. While effective for classification, e.g., asserting that a `Server` is a subclass of `ComputingDevice`, these systems lack the expressivity to model arithmetic constraints or temporal intervals[7] that can be employed in inference processes in formal logic. In a standard knowledge graph, a term like "Patching" is merely an opaque string label linked to a data object[2]. To the system, "Patching" and "Walking" are semantically identical; neither possesses an inherent context of resource consumption or state change.

This lack of semantic grounding prevents the computer from inferring the logical implications of an action in its respective use case. Without formal axioms, the system cannot derive that `Patching` is a `Process` that modifies a `softwareVersion` at a later time, whereas `Walking` is a `BodyMotion` that modifies a spatial `location`. Grounding these terms in SUMO's type hierarchy and formal axioms[22]

*M.S. Candidate, Computer Science.
†Associate Professor, Computer Science.

Hot Topics in the Science of Security Symposium '26, Virtual
2026. ACM ISBN 978-1-4503-XXXX-X
https://doi.org/XXXXXXX.XXXXXXX

transforms these symbols into computable semantics. By extending SUMO with cybersecurity concepts, we provide the shared mathematical meaning necessary for an automated theorem prover to verify resource allocation strategies in cyberspace. This transition from labeling to reasoning is what enables the formal verification of defensive strategies.

2 Formal Ontology

We base our application on the Suggested Upper Merged Ontology (SUMO) [1][21, 22], a formal ontology stated in higher-order logic, consisting of roughly 20,000 named concepts and 100,000 hand-written logical axioms created over the past 25 years [2]. It has been employed on a diverse range of projects such as Naval maneuver decision-making[33] and natural language processing for document curation[32]. While it began as a project to create just an upper ontology of very general concepts, SUMO now includes a range of dozens of domains including the military, medicine, and geography. It allows AI scientists to reuse a vast array of conceptual inventory, greatly mitigating the challenge of formally specifying reasoning problems. SUMO is written in SUO-KIF (Standard Upper Ontology Knowledge Interchange Format) [3], a purely declarative logical language designed for use with automated theorem provers such as Vampire[16]. By formalizing cyber operations as an extension to SUMO's axioms, we provide the shared meaning necessary for a computer to verify strategy feasibility mathematically. Questions posed to the SUMO theory produce mathematical proofs that capture Vampire's reasoning, delivering a level of trust unavailable in procedural code or statistical language models.

SUMO has, from its start, anticipated future developments in logic and reasoning, employing hundreds of axioms that use a logic beyond first-order logic. We implement this approach by translating[3, 23] the contents of SUMO to the languages used by Vampire, specifically, typed first order form with arithmetic (TFF/TFA)[31] and Typed Higher-order Form (THF)[30]. Experimental higher-order extensions are available via THF translation for reasoning in modal logic. We have designed a quantified multi-modal logic with Kripke semantics[13, 17, 18] in typed first-order logic and typed higher-order logic[24].

For our present work in cybersecurity that involves significant reasoning about metric time and quantifiable resources we expect primarily to reason with Vampire over SUMO's translation to TFA. While some ATP systems implement typed first-order form (TFF) without arithmetic, Vampire fully implements typed first-order form with arithmetic (TFA) so we will hereafter just refer to TFA as the language of our target translation.

3 Related Work and Competitive Positioning

Formal verification typically relies on one of two dominant paradigms: Satisfiability Modulo Theories (SMT) and Description Logic (DL). Our approach occupies a unique position by utilizing typed First-Order Logic (FOL) with arithmetic to bridge the gaps between these methods in a multi-scale context.

[1] https://www.ontologyportal.org

[2] see https://sigma.ontologyportal.org:8443/sigma/Browse.jsp?kb=SUMO for current statistics on the number of axioms of different kinds in SUMO

[3] https://github.com/ontologyportal/sigmakee/blob/master/suo-kif.pdf

3.1 SMT and DL in Cybersecurity

SMT solvers, such as Z3[9], are widely utilized for verifying network configurations and protocol correctness due to their efficiency in handling arithmetic constraints within bounded search spaces. For example, SMT-based tools have been successfully applied to verify firewall policies and reachability in software-defined networks[9]. Similarly, Description Logic (DL) reasoners like HermiT[12] serve as the backbone for many security ontologies designed for automated classification and threat intelligence mapping. These systems are highly effective at asserting hierarchical relationships (e.g., classifying a specific malware sample as a subtype of a known family and its inherit characteristics).

3.2 The Verification Gap

Despite their strengths, these paradigms face expressivity limits for multi-scale strategy verification. Description-logic systems lack the expressive power to represent complex state transitions and the precise arithmetic constraints needed to verify budget-constrained schedules. SMT solvers excel at arithmetic but struggle with the deeply quantified conceptual hierarchies required to ground technical actions in organizational policies derived from natural-language strategy. Current frameworks often compensate by delegating parts of the reasoning to external procedural code (e.g., Python or Java), which severs the connection between the ontology's intended semantics and the portion of reasoning that can be proved in-house.

3.3 Game-Theoretic Resource Allocation

Game theory is frequently employed to model adversarial interactions in cybersecurity, yet many models remain theoretical abstractions that lack grounding in specific asset inventories[28]. We differentiate our framework by using the Gordon-Loeb model[15] to define the defender's utility function (U_D). This approach ensures that resource allocation is driven by economic value and business realities rather than technical coverage alone.

By framing cybersecurity as an investment problem, we penalize "over-security" when the cost of a defensive process $C(S_D)$ exceeds the potential loss $L(S_A)$. We also adopt the investment perspective discussed by Gordon-Loeb, which shows that organizations that are extremely valuable focused on protecting extremely valuable information can face diminishing returns from concentrating resources on a single asset and often are better off addressing a broader set of medium-risk vulnerabilities across their organization[15]. Within our pipeline, the Gordon-Loeb parameters provide the quantitative basis for the macro-scale axioms in our cyber ontology, enabling the system to verify that a proposed strategy is both technically valid and economically optimal for the organization.

4 Approach Overview

The objective of this work is to develop a formal pipeline that generates and verifies cybersecurity strategies under multi-scale resource constraints. Specifically, we aim to produce a proof-of-concept that includes: (1) formalizes organizational assets and policies using a new cyber resources ontology that extends SUMO; (2) generates candidate strategies through a game-theoretic simulation; and (3) utilizes the Vampire theorem prover[16] to provide a mathematical proof of correctness for a candidate strategy. The primary artifact

is a list of verifiably safe strategies, where every action is logically consistent with the organization's budget, maintenance windows, and operational rules.

4.1 Phase I: Ontological Engineering

While SUMO provides the upper-level axioms for processes and resources[22], it lacks some of the domain-specific predicates required for cybersecurity strategy verification. We aim to develop a module that extends SUMO with formal definitions for cyber-physical resource allocation. Key additions include:

- `capacity`: A `BinaryPredicate` mapping a `ComputingDevice` to its maximum instruction or bandwidth throughput
- `resourceCost`: A `TernaryPredicate` defining the load a specific `ComputerProcess` places on a component
- `financialCost`: A predicate for mapping `Process` instances to `CurrencyMeasures` for macro-scale budget tracking

By grounding these terms in SUMO's temporal reasoning framework, specifically the `holdsDuring` relation, we enable the system to reason about overlapping resource consumption during specific time intervals[24, 26].

4.2 Phase II: Strategy Generation via Stochastic Simulation

To generate candidate strategies, we plan to employ a stochastic simulation engine that models defender and attacker interactions. Defensive utility are derived from a game-theoretic framing of the Gordon-Loeb model[15], while resource competition is governed by the Adversarial Knapsack constraint[14]. The engine generates thousands of potential engagement timelines ($T_1 \dots T_n$). Each timeline represents a potential strategy S that satisfies the economic utility function U_D but requires formal verification to ensure no logical *safety axioms* are violated.

4.3 Phase III: ATP Verification via Proof-by-Refutation

The final phase utilizes the Vampire theorem prover to verify the logical consistency of candidate strategies. We employ *proof-by-refutation*: for a proposed strategy S, we assume its negation $\neg S$ (i.e., "that there exists a state in which S is active and a safety constraint is violated").

Vampire attempts to derive a contradiction from this negated conjecture. If a contradiction is found, the strategy is proven valid relative to the axioms in our cyber ontology and SUMO, ensuring it is both game-theoretically optimal and verifiably safe within the organization's operational bounds.

5 Technical Methodology

This research is structured into three integrated phases that leverage the formal axioms of the SUMO to constrain and verify a stochastic strategy simulation engine.

5.1 Phase I: Ontological Rigor & Temporal Formalism

5.1.1 Application to Cybersecurity via QoS. To define the domain, we utilize the Quality of Service (QoS) Ontology[4], a domain extension of SUMO containing concepts related to system performance, reliability, and computational state of computers. This allows for granular distinctions between static attributes and dynamic states.

For example, we distinguish between a static inventory fact and a dynamic state that consumes computing capacity over a time interval. We represent execution by a process `Proc_Agent_001` such that `(programRunning Proc_Agent_001 SecurityAgent)` and `(computerRunning Proc_Agent_001 Server1)`. This distinction enables the use of the `programRunning` term to axiomatically prevent the allocation of security tools to overloaded nodes.

5.2 Phase II: Scenario Generation (Stochastic Simulation)

Leveraging the formal rules defined in Phase I, we will employ a stochastic simulation engine to stress-test candidate defensive strategies against an adversarial environment. This engine will generate tens of thousands of potential engagement timelines ($T_1 \dots T_n$) by simulating defensive actions across a specific asset profile while simultaneously modeling adversarial counter-moves.

5.2.1 Probabilistic Modeling. To capture operational variance in the environment, we plan to employ a hybrid probabilistic model that distinguishes between machine-speed automation and human-centric decision cycles typical in cybersecurity operations:

(1) **Automated Events (Poisson Processes):** High-frequency events, such as exploit scanning or initial packet arrivals, are modeled via Poisson processes to reflect the background threat noise of cyberspace.
(2) **Adversarial and Defensive Processes (PERT Distributions):** Meso-layer processes, such as *Time-to-Patch* or *Lateral Movement* duration, can be modeled using *PERT distributions*[5]. By defining Optimistic, Pessimistic, and Most Likely parameters, we aim to prevent the simulation from generating unrealistic "instantaneous" mitigation actions.

5.2.2 Parameter Validation and Formalization. The validity of these models depends on accurate parameter estimation calibrated through empirical incident data. These durations will be formalized as functional terms in SUO-KIF (e.g., `patchTime`) to denote specific temporal attributes of defensive or adversarial processes. The simulation supplies the descriptive facts that the Vampire inference applies to the formal axioms in order to detect logical contradictions.

5.3 Phase III: Strategic Optimization via Theorem Proving

The final phase utilizes the Vampire Automated Theorem Prover (ATP)[16] to evaluate the large number of engagement timelines generated in Phase II.

[4] https://github.com/ontologyportal/sumo/blob/master/QoSontology.kif

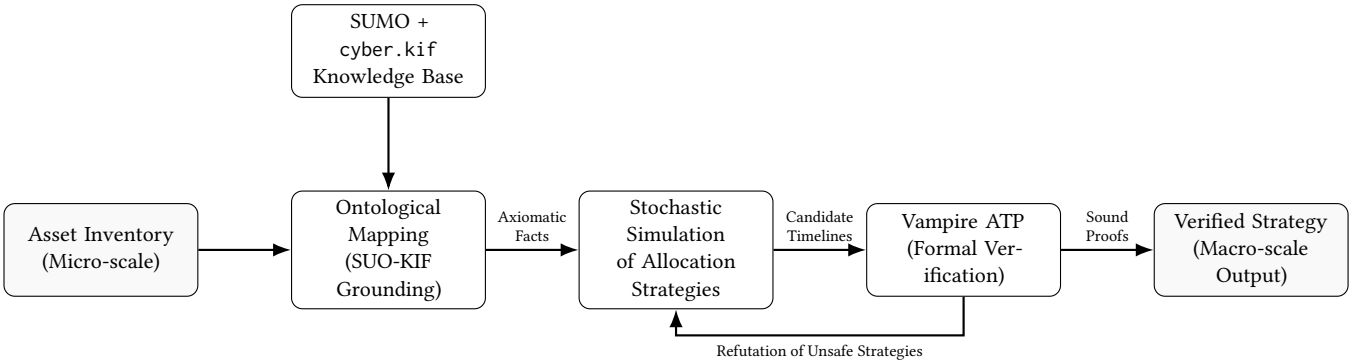

Figure 1: System Architecture: Pipeline for multi-scale strategy verification. Assets are grounded in SUMO axioms[22]; candidate engagement timelines are generated via stochastic simulation to model adversarial interactions and resource competition. These timelines are then formally verified via proof-by-refutation using the Vampire ATP[16] to determine if a strategy remains valid under the Adversarial Knapsack and other constraints as applicable.[14].

5.3.1 Automated Reasoning with Vampire. To verify the validity of a strategy S, the system uses proof-by-refutation[27]. The automated theorem prover (ATP) Vampire begins by assuming the negation of the safety property, $\neg S$, which asserts the existence of a reachable state where the strategy is active but safety constraints are violated. Vampire then attempts to derive a contradiction; if it succeeds, the original claim S is established.

Vampire employs saturation-based theorem proving[11] to search for a contradiction from this assumption. A contradiction arises when two mutually inconsistent assertions—for example, that the combined resource load both exceeds and does not exceed the host's defined capacity—are derived in the same context. Because Vampire's inference steps are sound, deriving such a contradiction implies the initial assumption of non-safety was false, and therefore the strategy S is safe.

6 Case Studies

6.1 The Equifax Breach: A Meso-Scale Failure

The 2017 Equifax breach[34] represents a failure of the meso-layer. While the technical patch (micro) was available, the organizational rules governing its deployment across the enterprise (meso) lacked formal visibility. In an ontologically grounded system, an "unpatched state" for a critical asset would be flagged as a logical contradiction to the organization's safety axioms. The failure was not a lack of data, but a lack of a formal link between asset state and policy enforcement.

6.2 The CrowdStrike Incident: Logic Verification Failure

The 2024 CrowdStrike incident[8] illustrates a failure of formal precondition verification. A content update (Process A) was deployed to a production environment without verifying compatibility with the kernel state of the host Operating System (State B). In our framework, this is modeled as a violation of a *meso-layer* safety rule: no process may be initiated in a production environment unless its dependencies have been validated in an isomorphic test environment.

Listing 1: Meso-layer rule for test environment validation

```
(=>
  (and
    (instance ?PATCH SoftwarePatch)
    (attribute ?PROD_COMP ProductionEnvironment)
    (holdsDuring ?T_NOW
      (computerRunning ?PROC ?PROD_COMP))
    (programRunning ?PROC ?PATCH))
  (exists (?T_PAST ?TEST_COMP)
    (and
      (attribute ?TEST_COMP TestEnvironment)
      (earlier ?T_PAST ?T_NOW)
      (holdsDuring ?T_PAST
        (programValidated ?PATCH ?TEST_COMP)))))
```

Above is the formalization of this meso-rule in the SUO-KIF language and SUMO concepts. Note in particular the use of the temporal modal operator `holdsDuring` to state that a formula is true during a time interval:

This formula may be paraphrased as "If there is a patch running on a production computer, then in the past there must exist that same patch applied to a test environment computer."

When the above is translated into logic and combined with the operational facts describing the deployment, this meso-layer rule is treated as a safety axiom. The Vampire ATP[16] is then used to check the satisfiability of the resulting theory. In the CrowdStrike case, the antecedent (the 'if' part) of the implication is satisfied by the observed deployment of the update in a production environment, while no facts are available to witness the existential requirement for prior validation at T_{past}, formalized using an existential quantifier.

7 Game Formulation

To rigorously model the adversarial dynamics, we define the cybersecurity engagement as a non-cooperative, non-zero-sum game tuple $G = (P, S, U)$.

7.1 Players (P)

The game consists of two primary agents:

- **The Defender (D):** Constrained by budget, regulatory compliance (e.g., NIST 800-53), and system availability requirements.
- **The Attacker (A):** Constrained by time, technical capability (exploit inventory), and the risk of detection.

7.2 Strategies (S)

We define the strategy space S_D (Defender) and S_A (Attacker) across the three Dopferian[10] scales. Defensive moves are presented as concrete controls and investments that can be mapped to the NIST Cybersecurity Framework (CSF) to show which CSF outcomes they implement[20].

- **Micro (Tactical):**
  - S_D: Atomic actions such as *Apply Patch KB4023*, *Close Port 80*, or *Reset User Password*.
  - S_A: Atomic exploit techniques such as *SQL Injection* or *Phishing Email*.
- **Meso (Operational):**
  - S_D: Rule adoption and Standard Operating Procedures (SOPs), such as "Patching Window: Every Tuesday 0200-0400" or "Zero Trust Access Policy."
  - S_A: Campaign tactics, such as *Lateral Movement* or *Persistence Establishment*.
- **Macro (Strategic):**
  - S_D: Systemic allocation, such as Capital Expenditure (CAPEX) on new architecture or purchasing Cyber Insurance.
  - S_A: Strategic objectives, such as *Advanced Persistent Threat (APT)* positioning on target systems or *Ransomware Deployment*.

7.3 Payoff Function (U)

We adopt the "Gordon-Loeb" model of[15] to define the Defender's utility U_D. The goal is not "perfect security" (which is infinite cost and infinite knowledge about system vulnerability), but optimized net value:

$$U_D = (V \times A_{sys}) - C(S_D) - L(S_A)$$

Where:

- V: `(valuation ?ASSET (MeasureFn ?VAL USDollar))`
- A_{sys}: System Availability (Uptime %).
- $C(S_D)$: Sum of `financialCost` across all defensive `Process` instances in the strategy. Vampire verifies budget constraints mathematically by querying the ATP for the `totalValue` of all defensive `Process` instances occurring within the strategy's `TimeInterval`.
- $L(S_A)$: The expected loss from a successful attack.

Listing 2: Type restrictions on the `valuation` predicate

```
(instance valuation BinaryPredicate)
(domain valuation 1 Physical)
(domain valuation 2 CurrencyMeasure)
```

Listing 3: Type restrictions on the `financialCost` predicate

```
(instance financialCost BinaryPredicate)
(domain financialCost 1 Process)
(domain financialCost 2 CurrencyMeasure)
```

Listing 4: Rule that defines part of the meaning of `financialCost`

```
(=>
  (and
    (instance ?STRATEGY DefensiveStrategy)
    (instance ?BUDGET CurrencyMeasure)
    (subProcess ?PROC ?STRATEGY)
    (financialCost ?PROC ?COST)
    (inList ?COST ?COSTLIST)
    (equal ?TOTAL_COST
      (ListSumFn ?COSTLIST)))
  (not
    (greaterThan ?TOTAL_COST ?BUDGET)))
```

This function explicitly penalizes "over-security" where $C(S_D)$ exceeds the potential loss $L(S_A)$, validating the business requirement for efficiency over paranoia.

To enable ATP verification of budget constraints, we ground these economic terms in SUMO, as shown in listings 2, 3 and 4.

8 Discussion: The Meso-Gap and Resource-Constrained Strategy

To provide actionable decision support, we apply Dopfer's framework of Evolutionary Economics to the domain of cyber strategy[10]. Organizations typically focus on two scales: the Micro (individual patches, exploits, and alerts) and the Macro (yearly budgets, compliance audits, and aggregate risk).

However, strategies often fail due to the "Meso-Layer failure": the lack of analysis regarding the "trajectory of rule adoption"[10], which defines the Meso scale as a population of generic rules that are adopted and retained. The framework was recently applied in[4] to demonstrate how overlooking this meso-level leads to fragmented policy interventions in critical infrastructure. In our model, a "Security Policy" (e.g., "Patch every Tuesday") is a Meso-trajectory where there is emergence of that policy, diffusion/adoption across the organization, and retention of the policy as a standard practice.

(1) **Micro (Origination):** The atomic act (e.g., a single server update).
(2) **Meso (Adoption):** The operational rule that governs the frequency and conditions of those acts.

(3) **Macro (Retention):** The systemic stability resulting from the Meso-rules.

Our proposed tool targets this Meso-layer. By simulating thousands of timelines (Phase II) and validating them via ATP (Phase III), we allow leaders to optimize the rules of engagement rather than just reacting to individual Micro-events.

8.1 The Adversarial Knapsack Axiom

To resolve multi-scale resource conflicts, we ground our formal verification in the mathematical principles of the Weighted Knapsack problem and its adversarial extension, the Dueling Knapsack (DK) [14].

Weighted Knapsack. In a standard Weighted Knapsack problem, an agent selects a subset of N objects with associated weights w_i and rewards μ_i to maximize total reward subject to a resource constraint:

$$\max_{\boldsymbol{\theta}\in\{0,1\}^N} \sum_{i=1}^{N} \theta_i \mu_i \quad \text{subject to} \quad \sum_{i=1}^{N} \theta_i w_i \le W, \tag{1}$$

where θ_i are binary decision variables[14].

Dueling Knapsack. In the Dueling Knapsack framework, this optimization becomes a zero-sum game between an attacker (a) and a defender (d)[14]. The attacker selects exploits $\xi_i^a \in \{0,1\}$, while the defender selects patches $\zeta_i^d \in \{0,1\}$[14]. The attacker receives reward μ_i from exploiting vulnerability i only if it is attacked and not patched[14].

Given a fixed attacker strategy $\boldsymbol{\xi}^a$, the defender's best-response problem is defined as:

$$\min_{\boldsymbol{\zeta}^d\in\{0,1\}^N} \sum_{i=1}^{N} \xi_i^a (1-\zeta_i^d)\mu_i \quad \text{subject to} \quad \sum_{i=1}^{N} \zeta_i^d w_i^d \le W_d. \tag{2}$$

Since $\sum_i \xi_i^a \mu_i$ is constant with respect to $\boldsymbol{\zeta}^d$, this minimization is equivalent to the Weighted Knapsack problem:

$$\max_{\boldsymbol{\zeta}^d\in\{0,1\}^N} \sum_{i=1}^{N} \xi_i^a \zeta_i^d \mu_i \quad \text{subject to} \quad \sum_{i=1}^{N} \zeta_i^d w_i^d \le W_d, \tag{3}$$

which selects the subset of attacked vulnerabilities whose mitigation maximally reduces the attacker's expected gain under a limited defense budget W_d[14].

Axiomatic Enforcement. The Vampire ATP enforces the defender's resource constraint, $\sum \zeta_i^d w_i^d \le W_d$, as a logical safety axiom. In SUO-KIF/SUMO, the binary decision variable ($\zeta_i = 1$) is grounded in the `computerRunning` relation; if a `ComputerProcess` (the patch action) is active during a specific `TimeInterval`, it is treated as "packed" into the knapsack. When the combined weights (w_i, formalized as `resourceCost`) of the selected patches exceed the per-time-step budget (W_d, formalized as `capacity`), Vampire derives a contradiction, thereby refuting the strategy as operationally, and thus logically, impossible.

Listing 5 shows the SUMO formalization of the problem, where:

- `?Wd` is the fixed budget for this time-step
- `?P1` is patch action 1 (Item in knapsack)
- `?P2` is patch action 2 (Item in knapsack)
- `(not (equal ?P1 ?P2))` ensures that the items are distinct

Listing 5: SUMO formalization of a cyber budget constraint

```
(=>
  (and
    (instance ?H Computer)
    (capacity ?H ?Wd)
    (instance ?P1 ComputerProcess)
    (instance ?P2 ComputerProcess)
    (not (equal ?P1 ?P2))
    (holdsDuring ?T
      (computerRunning ?P1 ?H))
    (holdsDuring ?T
      (computerRunning ?P2 ?H))
    (resourceCost ?P1 ?H ?W1)
    (resourceCost ?P2 ?H ?W2))
  (not
    (greaterThan
      (AdditionFn ?W1 ?W2)
      ?Wd)))
```

- `?W1` is the cost to apply patch 1
- `?W2` is the cost to apply patch 2

Note that `capacity` and `resourceCost` are proposed predicates in a work-in-progress cyber-domain extension to SUMO. This WiP uses these predicates to demonstrate how arithmetic resource constraints can be validated by ATP; the ontology extension to SUMO will appear in future work at the source code repository for SUMO[5].

8.2 Formal Verification Example: Refuting Over-Budget Tactical Actions

To illustrate the refutation process, we provide a simplified manual translation of SUMO axioms into Typed First Order Arithmetic (TFA), in order to demonstrate how Vampire detects a resource violation (Listing 6). In our current system, we have an entirely automated translation of SUMO to TFA[23]. In this scenario, the simulation proposes a strategy where two patches are applied simultaneously, exceeding the per-time-step budget (W_d). For ease of understanding in this simplified example, we are using an encoding of time as an extra argument similar to[6] rather than our actual method of Kripke semantics. We also omit here the type definitions required in TFA in the interest of space. The arithmetic operations of `$sum` and `$greater` are part of the TFA language but are also fully axiomatized to enforce logical consistency and valid proof checking.

Statements in the TFA language have the keyword `tff` to specify the TPTP sub-language being used, then a name for the axiom, then whether it is an `axiom` - a truth stated by the file author, or a `conjecture` - a question posted by the file author, possibly with existentially quantified variables for which the prover is directed to find a binding. A full description of the language and other resources can be found online[6].

[5] https://github.com/ontologyportal/sumo
[6] https://tptp.org

Listing 6: TFA Translation of the SUMO axioms

```
tff(ax1,axiom,
  patch_A != patch_B).

tff(axiom_budget,axiom,
    capacity(server1,100) ).

tff(patch1_cost,axiom,
    resourceCost(patch_A,server1,85) ).

tff(patch2_cost,axiom,
    resourceCost(patch_B,server1,25) ).

tff(p1_run,axiom,
    computerRunning(patch_A,server1,t1)).

tff(p2_run,axiom,
    computerRunning(patch_B,server1,t1)).

tff(violated_knapsack,axiom,
  (! [P1:$i,P2:$i,H:$i,T:$i,W1:$int,W2:$int,Max:$int] :
    (( P1 != P2
    & computerRunning(P1,H,T)
    & computerRunning(P2,H,T)
    & resourceCost(P1,H,W1)
    & resourceCost(P2,H,W2)
    & capacity(H,Max)
    & $greater($sum(W1,W2),Max) ) =>
    attribute(H,violatedKnapsack) ))).

tff(strategy_unsafe,conjecture,
    ?[H:$i]:attribute(H,violatedKnapsack) ).
```

Vampire Refutation Logic: The `violated_knapsack` axiom states that violation occurs if two distinct tasks run simultaneously on the same host and their combined costs exceed that host's capacity. Vampire 5.0.0[7] derives a contradiction by negating the conjecture. It finds the expected result of `?H=server1` in 0.1 seconds on a modern Linux laptop. The prover instantiates the axiom `violated_knapsack` with $?W1 = 85, ?W2 = 25, ?Max = 100$. It performs the addition ($85 + 25 = 110$) and evaluates the predicate `greaterThan(110, 100)`, which returns `True`. Since the axiom asserts the *negation* of this, Vampire finds a contradiction, derives the empty clause, and refutes the safety of the strategy.

9 Strategic Decision-Support: The CISO Dashboard

The primary deliverable of this research is a proof-of-concept decision-support application that demonstrates how formal logical proofs can be translated into actionable organizational strategy. While not production-ready, this prototype aims to validate and visualize the feasibility of using automated theorem proving for verifiable cyber strategy generation.

9.1 Strategy Generation and Verification

Upon the ingestion of an asset profile and organizational constraints, the system does not search for vulnerabilities; it generates a *sound* cyber strategy across the Micro, Meso, and Macro scales. The Vampire theorem prover acts as a "logical compiler," verifying a set of foundational conjectures (safety axioms) against the proposed strategy at each respective level.

- **The Micro-Scale Output:** A prioritized schedule of technical actions (e.g., specific patching, firewall configurations, and credential rotations) validated against device-level capacity.
- **The Meso-Scale Output:** Optimized operational policies and Standard Operating Procedures (SOPs) that govern the "trajectories" of defense, ensuring that team-level rules do not create systemic resource bottlenecks.
- **The Macro-Scale Output:** Strategic resource allocation schemas that align budget and personnel capacity with long-term organizational risk tolerance.

9.2 Implementation Scope and Limitations

This work proposes a proof-of-concept implementation designed to demonstrate the viability of ontology-based formal verification for cybersecurity strategy. The current prototype focuses on:

- **Verification Capability:** Demonstrating that SUMO axioms can detect resource conflicts (e.g., the Adversarial Knapsack Constraint) through ATP, providing provable guarantees of strategy soundness.
- **Temporal Reasoning:** Showing that `holdsDuring` relations enable interval-based inference (e.g., rootkit blast radius analysis) that static taxonomies cannot perform without external procedural code.
- **Multi-Scale Integration:** Validating that a single logical framework can reason across Micro (device-level capacity), Meso (operational policies), and Macro (budget constraints) simultaneously .

Production Deployment Considerations: Transitioning this prototype to production would require: (1) performance optimization for real-time enterprise-scale reasoning, (2) integration with existing Security Information and Event Management (SIEM) systems, (3) user interface design for non-technical stakeholders, and (4) extensive validation against industry-standard attack scenarios beyond the synthetic simulations presented here. This research establishes the logical foundations and demonstrates the feasibility of a new class of formally verified cybersecurity decision-support systems.

Future Work: While the proof-of-concept handles interval overlap and state transitions, *metric temporal reasoning*—calculating exact durations in calendar units (hours, days of the week, business hours) or determining "what day did the breach occur?" requires complete axiomatization of Gregorian date arithmetic [26]. This ongoing work will enable queries like "If a breach occurred on Tuesday at 14:00 and lasted 6 hours, did it extend into the maintenance window?" to be answered within the ATP. Once complete, this will

[7] https://github.com/vprover/vampire

provide a unified solution that static schemas/taxonomies combined with external code (e.g. Python scripts) cannot match—because any conclusions reached through procedural code would be *logically disconnected* from the ontology's axioms.

10 Conclusion and Future Work

This WiP paper presents a proof-of-concept demonstrating how formal ontologies and automated theorem proving can bridge the semantic and data gaps in cyber strategy. The prototype validates that SUMO-based reasoning provides verifiable guarantees that other approaches cannot achieve. Future work focuses on: (1) completing the cyber-domain SUMO mapping, (2) integrating the stochastic simulation engine with live threat intelligence feeds, and (3) evaluating scalability for enterprise deployment. This research establishes the logical foundations for a new class of formally verified cybersecurity decision-support systems.

Acknowledgments

This work is supported by the Naval Postgraduate School.

References

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (Eds.). 2007. *The Description Logic Handbook* (2 ed.). Cambridge University Press, Cambridge, UK.

[2] Sotiris Batsakis, Euripides Petrakis, Ilias Tachmazidis, and Grigoris Antoniou. 2016. Temporal representation and reasoning in OWL 2. *Semantic Web* 8 (11 2016), 1–20. doi:10.3233/SW-160248

[3] Christoph Benzmüller and Adam Pease. 2010. Progress in Automating Higher-Order Ontology Reasoning. In *PAAR-2010: Practical Aspects of Automated Reasoning*, Boris Konev, Renate Schmidt, and Stephan Schulz (Eds.). Edinburgh, UK, 23–32. https://wwwlehre.dhbw-stuttgart.de/~sschulz/PAPERS/KSS-PAAR-2010.pdf Workshop proceedings.

[4] Dimos Chatzinikolaou and Charis Vlados. 2025. Macro–Meso–Micro: An Integrative Framework for Evolutionary Economics and Sustainable Transitions. *Sustainability* 17, 21 (2025), 9480. doi:10.3390/su17219480

[5] Charles E Clark. 1962. The PERT model for the distribution of an activity time. *Operations Research* 10, 3 (1962).

[6] James Clifford and David S. Warren. 1983. Formal semantics for time in databases. *ACM Trans. Database Syst.* 8, 2 (June 1983), 214–254. doi:10.1145/319983.319986

[7] Anne M. Cregan. 2007. Symbol Grounding for the Semantic Web. In *The Semantic Web: Research and Applications*, Enrico Franconi, Michael Kifer, and Wolfgang May (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 429–442.

[8] Cybersecurity and Infrastructure Security Agency (CISA). 2024. CrowdStrike 2024 Software Update Outage. https://www.cisa.gov/news-events/alerts/2024/07/19/widespread-it-outage-due-crowdstrike-update Alert regarding the widespread impact of a faulty CrowdStrike Falcon sensor update on Windows systems.

[9] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: an efficient SMT solver. *Tools and Algorithms for the Construction and Analysis of Systems* 4963, 337–340. doi:10.1007/978-3-540-78800-3_24

[10] Kurt Dopfer, John Foster, and Jason Potts. 2004. Micro-meso-macro. *Journal of Evolutionary Economics* 14, 3 (2004), 263–279.

[11] Harald Ganzinger. 1996. Saturation-based theorem proving: Past successes and future potential. In *Automated Deduction – CADE-13 (Lecture Notes in Computer Science, Vol. 1104)*. Springer, 3–17. doi:10.1007/3-540-61511-3_64

[12] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. 2014. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning* 53 (2014), 245–269. https://api.semanticscholar.org/CorpusID:15513227

[13] Robert Goldblatt. 2003. Mathematical modal logic: A view of its evolution. *Journal of Applied Logic* 1, 5 (2003), 309–392. doi:10.1016/S1570-8683(03)00008-9

[14] Jon Goohs, Georgel Savin, Lucas Starks, Josiah Dykstra, and William Casey. 2024. Adversarial Knapsack and Secondary Effects of Common Information for Cyber Operations. arXiv:2403.10789 [cs.CR] doi:10.48550/arXiv.2403.10789 arXiv:2403.10789v1, submitted 16 Mar 2024. License: CC BY 4.0.

[15] Lawrence A. Gordon and Martin P. Loeb. 2002. The economics of information security investment. *ACM Trans. Inf. Syst. Secur.* 5, 4 (Nov. 2002), 438–457. doi:10.1145/581271.581274

[16] Laura Kovács and Andrei Voronkov. 2013. First-Order Theorem Proving and Vampire. In *Proceedings of the 25th International Conference on Computer Aided Verification - Volume 8044* (Saint Petersburg, Russia) *(CAV 2013)*. Springer-Verlag, Berlin, Heidelberg, 1–35.

[17] Saul A. Kripke. 1959. A completeness theorem in modal logic. *Journal of Symbolic Logic* 24, 1 (1959), 1–14. doi:10.2307/2964568

[18] Saul A. Kripke. 1963. Semantical Analysis of Modal Logic I Normal Modal Propositional Calculi. *Mathematical Logic Quarterly* 9 (1963), 67–96. Issue 5-6. doi:10.1002/malq.19630090502

[19] MITRE Corporation. 2024. *MITRE ATT&CK Framework*. https://attack.mitre.org

[20] National Institute of Standards and Technology. 2018. *Framework for Improving Critical Infrastructure Cybersecurity (Version 1.1)*. NIST Framework. U.S. Department of Commerce. https://www.nist.gov/cyberframework

[21] Ian Niles and Adam Pease. 2001. Towards a Standard Upper Ontology. *Formal Ontology in Information Systems: Collected Papers from the Second International Conference*, 2–9. doi:10.1145/505168.505170

[22] Adam Pease. 2011. *Ontology: A Practical Guide*. Articulate Software Press, Angwin, CA.

[23] Adam Pease. 2023. Converting the Suggested Upper Merged Ontology to Typed First-order Form. arXiv:2303.04148 [cs.AI] https://arxiv.org/abs/2303.04148

[24] Adam Pease. 2025. Modal and Higher Order Logical Reasoning with SUMO. In *Semantics at the Crossroads: From Theoretical Explorations to Implementations*, Aikaterini-Lida Kalouli, Tracy Holloway King, Stephen Pulman, and Annie Zaenen (Eds.). Konstanz: PubliKon, 165–186.

[25] Adam Pease. 2026. Ontology and Information Systems. In *The Stanford Encyclopedia of Philosophy* (Spring 2026 ed.), Edward N. Zalta and Uri Nodelman (Eds.). Metaphysics Research Lab, Stanford University.

[26] Adam Pease and Pantelimon Stanica. 2025. Bridging the Gap: A Complete Axiomatization of Gregorian Date Arithmetic and Scheduling Logic for Automated Theorem Provers. (2025). In preparation.

[27] J. A. Robinson. 1965. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM* 12, 1 (1965), 23–41. doi:10.1145/321250.321253

[28] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu. 2010. A Survey of Game Theory as Applied to Network Security. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*. 1–10.

[29] Tripwire Security. 2019. ATT&CK Structure Part II: From Taxonomy to Ontology. (July 2019). https://www.tripwire.com/state-of-security/attck-structure-ontology

[30] Geoff Sutcliffe and Christoph Benzmüller. 2010. Automated Reasoning in Higher-Order Logic using the TPTP THF Infrastructure. *Journal of Formalized Reasoning* 3, 1 (2010), 1–27.

[31] Geoff Sutcliffe, Stephan Schulz, Koen Claessen, and Peter Baumgartner. 2012. The TPTP Typed First-Order Form with Arithmetic. In *Logic for Programming, Artificial Intelligence, and Reasoning*, Nikolaj Bjørner and Andrei Voronkov (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 406–419.

[32] Richard Thompson, Adam Pease, Roberto Milanese Jr., Jarrad Singley, and Angelos Toutsios. 2025. Formalizing Natural Language: Cultivating LLM Translations Using Automated Theorem Proving. In *Theorem Proving and Machine Learning in the age of LLMs: SoA and Future Perspectives* (Edinburgh, Scotland, UK), Ekaterina Komendantskaya, Elizabeth Polgreen, Christian Saemann, Kathrin Stark, and Michael Rawson (Eds.). European Research Network on Formal Proofs.

[33] James Timberlake and Adam Pease. 2025. "Rules of the Road" Navigation with Logic, Theorem Proving and a Large Ontology. In *Proceedings of the 2025 International Conference on Scientific Computing (CSCI)*.

[34] U.S. House of Representatives. 2018. The Equifax Data Breach. https://oversight.house.gov/sites/democrats.oversight.house.gov/files/Equifax%20Report.pdf Investigation into the 2017 Equifax data breach and failure to patch Apache Struts vulnerability.

A Research Proposal Note

As per submission requirements, we assert that this manuscript is unpublished. Revisions will be incorporated prior to the next submission for publication.