

Paper Type: Original Article

Attack Graph, Attack HyperGraph, and Attack SuperHyperGraph in Security

Takaaki Fujita^{1*} 

¹ Independent Researcher, Tokyo, Japan; Takaaki.fujita060@gmail.com,

Received: 20 Sep 2025

Revised: 17 Nov 2025

Accepted: 15 Jan 2026

Published: 17 Jan 2026

Abstract

Graph theory studies mathematical structures of vertices and edges to model relationships and connectivity. Hypergraphs extend traditional graphs by allowing *hyperedges* that connect any number of vertices simultaneously. Superhypergraphs further generalize this idea through iterated powerset layers, enabling hierarchical and self-referential connections among hyperedges.

An Attack Graph is the subgraph of an Attack Model containing exactly those states from which the attacker can reach a security violation, including its transitions and error states. An *Attack HyperGraph*, which extends the Attack Graph concept from the perspective of hypergraphs, is also well known and has been similarly studied. In this paper, we investigate *Attack Superhypergraphs*, which extend Attack Graphs to the superhypergraph framework. We expect that these developments will advance research in graph theory and hierarchical security models.

Keywords: SuperHyperGraph, HyperGraph, Attack Graph, Attack HyperGraph, Attack SuperHyperGraph

1 | Preliminaries

This section presents the fundamental concepts and definitions that underpin the discussions in this paper. For detailed treatment of specific operations and related notions, the reader is referred to the appropriate literature.

1.1 | SuperHyperGraph

In classical graph theory, a hypergraph extends the ordinary graph by permitting *hyperedges* that connect more than two vertices. This added flexibility makes hypergraphs an ideal tool for modeling complex, multi-way relationships across a variety of fields [1, 2]. A *SuperHyperGraph* further enriches this framework by integrating iterated powerset constructions into the hypergraph structure, a concept that has recently garnered considerable attention in the literature [3, 4, 5].

 **Corresponding Author:** Takaaki.fujita060@gmail.com

 <https://doi.org/10.61356/j.aics.2026.3638>

 Licensee **Artificial Intelligence in Cybersecurity**. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).

Beyond its theoretical appeal, SuperHyperGraphs have been applied in domains such as molecular modeling, network analysis, and signal processing [6, 7]. Throughout this paper, the parameter n in both the n -th powerset and the n -SuperHyperGraph is taken to be a non-negative integer.

Definition 1 (Base Set). A *base set* S is the foundational set from which complex structures such as powersets and hyperstructures are derived. It is formally defined as:

$$S = \{x \mid x \text{ is an element within a specified domain}\}.$$

All elements in constructs like $\mathcal{P}(S)$ or $\mathcal{P}_n(S)$ originate from the elements of S .

Definition 2 (Powerset). The *powerset* of a set S , denoted $\mathcal{P}(S)$, is the collection of all possible subsets of S , including both the empty set and S itself. Formally, it is expressed as:

$$\mathcal{P}(S) = \{A \mid A \subseteq S\}.$$

Definition 3 (Hypergraph). [8, 1] A *hypergraph* $H = (V(H), E(H))$ consists of:

- A nonempty set $V(H)$ of vertices.
- A set $E(H)$ of hyperedges, where each hyperedge is a nonempty subset of $V(H)$, thereby allowing connections among multiple vertices.

Unlike standard graphs, hypergraphs are well-suited to represent higher-order relationships. In this paper, we restrict ourselves to the case where both $V(H)$ and $E(H)$ are finite.

Example 4 (Hypergraph for Coordinated Credential Abuse in an Enterprise). Let the vertex set be

$$V(H) = \{\text{Web, App, DB, IdP}\},$$

representing an Internet-facing web server, an application server, a database server, and an identity provider (IdP). Define hyperedges

$$\begin{aligned} e_1 &= \{\text{Web, App}\} && \text{(a reused deployment key on Web grants access to App),} \\ e_2 &= \{\text{App, DB}\} && \text{(an ORM service account on App permits DB dumps),} \\ e_3 &= \{\text{Web, App, IdP}\} && \text{(token exfiltration on Web + misconfig on App + IdP refresh abuse).} \end{aligned}$$

Then $H = (V(H), E(H))$ with $E(H) = \{e_1, e_2, e_3\}$ is a finite hypergraph. The 3-ary hyperedge e_3 models a higher-order dependency that cannot be captured by pairwise edges alone, reflecting a coordinated precondition across three systems for SSO token forgery.

Definition 5 (n -th Powerset). (cf.[9, 10]) The n -th powerset of a set H , denoted $\mathcal{P}_n(H)$, is defined iteratively, starting with the standard powerset. The recursive construction is given by:

$$\mathcal{P}_1(H) = \mathcal{P}(H), \quad \mathcal{P}_{n+1}(H) = \mathcal{P}(\mathcal{P}_n(H)), \quad \text{for } n \geq 1.$$

Similarly, the n -th non-empty powerset, denoted $\mathcal{P}_n^*(H)$, is defined recursively as:

$$\mathcal{P}_1^*(H) = \mathcal{P}^*(H), \quad \mathcal{P}_{n+1}^*(H) = \mathcal{P}^*(\mathcal{P}_n^*(H)).$$

Here, $\mathcal{P}^*(H)$ represents the powerset of H with the empty set removed.

Example 6 (Interpreting the n -th Powerset in IAM Role Composition). Let $H = \{p_{\text{read}}, p_{\text{write}}, p_{\text{backup}}\}$ be a set of fine-grained permissions. Elements of the first powerset $\mathcal{P}_1(H) = \mathcal{P}(H)$ represent *roles*. For instance,

$$R_{\text{db}} = \{p_{\text{read}}, p_{\text{write}}\}, \quad R_{\text{ops}} = \{p_{\text{backup}}\}.$$

Elements of the second powerset $\mathcal{P}_2(H) = \mathcal{P}(\mathcal{P}(H))$ represent *bundles of roles*. For example,

$$B = \{R_{\text{db}}, R_{\text{ops}}\} \in \mathcal{P}_2(H)$$

is a composite entitlement assignable to a contractor account. Auditing $\mathcal{P}_2^*(H)$ (the non-empty version) helps detect toxic combinations—e.g., a bundle B whose combined permissions enable unintended data exfiltration—by reasoning over sets of roles rather than individual permissions.

Definition 7 (*n*-SuperHyperGraph). (*cf.* [11, 12])

Fix $n \in \mathbb{N}_0$. Let V_0 be a finite ground set of vertices. For any set Y and any $k \in \mathbb{N}_0$, define the iterated powerset by

$$\mathcal{P}^0(Y) = Y, \quad \mathcal{P}^{k+1}(Y) = \mathcal{P}(\mathcal{P}^k(Y)),$$

where $\mathcal{P}(\cdot)$ denotes the usual powerset operator. An *n-SuperHyperGraph* is a pair

$$\text{SHG}^{(n)} = (V, E),$$

with

$$V \subseteq \mathcal{P}^n(V_0) \quad \text{and} \quad E \subseteq \mathcal{P}(V).$$

Elements of V are called *n-supervertices*, and elements of E are called *n-superedges* (i.e., families of *n-supervertices*). If one wishes to exclude the empty superedge, replace $E \subseteq \mathcal{P}(V)$ by $E \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$.

Example 8 (Smart-Home Lateral Movement). Consider a home network with a gateway router, a CCTV camera, and a smart door lock. Let the security property be $\varphi = \neg(\text{UnlockDoor})$ (“the door remains locked”). Define states

$$\begin{aligned} s_0 &= \emptyset, \\ s_1 &= \{\text{CompRouter}\}, \\ s_2 &= \{\text{CompRouter}, \text{CompCamera}\}, \\ s_3 &= \{\text{CompRouter}, \text{CompCamera}, \text{UnlockDoor}\}, \end{aligned}$$

so that $S_A = \{s_0, s_1, s_2, s_3\}$ and $s_3 \in E$ (a violation state for φ).

In the Attack 2-SuperHyperGraph, the contiguous subchain “router $\rightarrow$ camera $\rightarrow$ door-unlock” is bundled as

$$X_0 = \{s_1, s_2, s_3\} \in \mathcal{P}(S_A).$$

Its 2-level lift (the associated 2-supervertex) is

$$\nu^{(2)} = \{X_0\} \in \mathcal{P}^2(S_A),$$

and the corresponding minimal (singleton) 2-superedge is

$$e^{(2)} = \{\nu^{(2)}\} = \{\{X_0\}\} \in \mathcal{P}(V^{(2)}) \quad (V^{(2)} = \mathcal{P}^2(S_A)).$$

This singleton 2-superedge compactly encodes the multi-device kill chain and supports treating the mitigation set (router hardening, CCTV isolation, lock PIN/2FA) as one defensive bundle. A simple schematic overview is shown in Fig. 1.

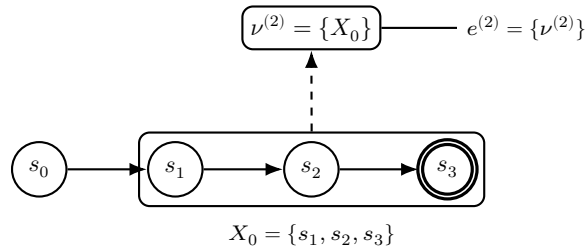


FIGURE 1. Smart-home lateral movement: the bundled subchain X_0 , its 2-level lift $\nu^{(2)} = \{X_0\}$, and the minimal singleton 2-superedge $e^{(2)} = \{\nu^{(2)}\}$.

Example 9 (Small-Business Supply-Chain Compromise). A small firm uses a developer workstation (DevWS), a CI/CD server, and a production VM. Let $\varphi = \neg(\text{PrivEscProd})$ (“no production root compromise”). Model

the attack progression by

$$\begin{aligned} s_0 &= \emptyset, \\ s_1 &= \{CompDevWS\}, \\ s_2 &= \{CompDevWS, CompCICD\}, \\ s_3 &= \{CompDevWS, CompCICD, SignedMalware\}, \\ s_4 &= \{CompDevWS, CompCICD, SignedMalware, PrivEscProd\}, \end{aligned}$$

so that $S_A = \{s_0, s_1, s_2, s_3, s_4\}$ and $s_4 \in E$ (a violation state for φ).

The contiguous multi-stage subchain “DevWS $\rightarrow$ CI/CD $\rightarrow$ signed artifact $\rightarrow$ production root” is bundled as

$$X_0 = \{s_1, s_2, s_3, s_4\} \in \mathcal{P}(S_A).$$

Its 2-level lift (the associated 2-supervertex) is

$$\nu^{(2)} = \{X_0\} \in \mathcal{P}^2(S_A),$$

and the corresponding minimal (singleton) 2-superedge is

$$e^{(2)} = \{\nu^{(2)}\} = \{\{X_0\}\} \in \mathcal{P}(V^{(2)}) \quad (V^{(2)} = \mathcal{P}^2(S_A)).$$

This singleton 2-superedge formalizes a realistic mitigation target: token rotation, attested build signing, and gated deployments, treated as one prioritized defensive bundle. A schematic overview is shown in Fig. 2.

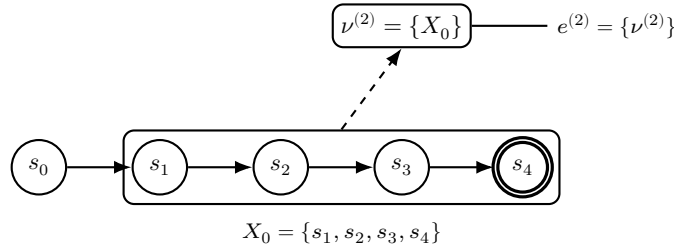


FIGURE 2. Small-business supply-chain compromise: the bundled subchain X_0 , its 2-level lift $\nu^{(2)} = \{X_0\}$, and the minimal singleton 2-superedge $e^{(2)} = \{\nu^{(2)}\}$.

1.2 | Attack Graph

Graph theory is also applied in the context of security [13, 14, 15]. An Attack Model is a finite labeled transition system representing attacker and system states, transitions for exploits, initial compromise, and proposition-labeling contexts. An Attack Graph is the subgraph of an Attack Model containing exactly those states from which the attacker can reach a security violation, including its transitions and error states [16, 17, 18].

Definition 10 (Attack Model). Let AP be a (finite) set of atomic propositions. An *Attack Model* is a finite labeled transition system

$$\mathcal{M} = (S, \tau, s_0, \ell),$$

where:

- S is a finite set of *states*, each state representing a complete valuation of the attacker’s and system’s attributes.
- $\tau \subseteq S \times S$ is the *transition relation*. A transition $(s, s') \in \tau$ corresponds to an elementary attack action (e.g., exploitation of a vulnerability) that changes the system from state s to state s' .
- $s_0 \in S$ is the *initial state*, representing the system before any attack has occurred (only entry-point compromises have happened).
- $\ell : S \rightarrow 2^{AP}$ is a *labeling function* that assigns to each state $s \in S$ the set of atomic propositions (e.g., “attacker has root on host H ”) true in that state.

Definition 11 (Attack Graph). (cf. [16, 17]) Given an Attack Model $\mathcal{M} = (S, \tau, s_0, \ell)$ and a security property φ (over AP), an *Attack Graph* is the sub-structure of $\mathcal{M}$ that contains exactly those states from which the attacker can eventually reach a violation of φ . Formally, the Attack Graph is the tuple

$$\mathcal{G} = (S_A, \tau_A, s_0, E, \ell'),$$

where:

- $S_A \subseteq S$ is the set of all states $s \in S$ for which there exists a finite τ -path starting at s and ending in some violation state. That is,
- $$S_A = \{s \in S \mid \exists k \in \mathbb{N}_0, \exists (s_0^t, \dots, s_k^t) \in S^{k+1} : s_0^t = s, (s_i^t, s_{i+1}^t) \in \tau \ (0 \leq i < k), \ell(s_k^t) \not\models \varphi\}.$$
- $\tau_A = \tau \cap (S_A \times S_A)$ is the restriction of the transition relation to S_A .
 - s_0 is the initial state of $\mathcal{M}$. Note that $s_0 \in S_A$ if and only if the attacker can violate φ starting from s_0 .
 - $E = \{s \in S_A \mid \ell(s) \not\models \varphi\}$ is the set of *error states* (violation states).
 - ℓ' is the labeling function ℓ restricted to S_A .

Example 12 (Real-World Example of an Attack Graph). Consider a small corporate network consisting of three machines:

Host₁ : Internet-facing Web Server, Host₂ : Application Server, Host₃ : Database Server.

The attacker's goal is to escalate privileges on Host₃ (the Database Server) in order to exfiltrate sensitive data. We model each possible compromise as a state in an Attack Model.

Atomic Propositions AP . Let

$$AP = \{CompWeb, CompApp, CompDB, PrivEscApp, PrivEscDB\},$$

where:

- $CompWeb$ = "Attacker has code execution on Web Server (Host₁)."
- $CompApp$ = "Attacker has code execution on Application Server (Host₂)."
- $CompDB$ = "Attacker has code execution on Database Server (Host₃)."
- $PrivEscApp$ = "Attacker has obtained administrative (root) privileges on Application Server."
- $PrivEscDB$ = "Attacker has obtained administrative (root) privileges on Database Server."

States S . We encode each state $s \in S$ by the set of atomic propositions currently true. Thus:

$$S = \{s_0, s_1, s_2, s_3, s_4, s_5\},$$

where:

$s_0 = \emptyset$	(No compromises yet; the attacker is external.)
$s_1 = \{CompWeb\}$	(Attacker has exploited a vulnerability on Web Server.)
$s_2 = \{CompWeb, CompApp\}$	(After pivoting from Web to Application Server.)
$s_3 = \{CompWeb, CompApp, PrivEscApp\}$	(Attacker escalated privileges on Application Server.)
$s_4 = \{CompWeb, CompApp, PrivEscApp, CompDB\}$	(Pivoted to Database Server without full DB privileges.)
$s_5 = \{CompWeb, CompApp, PrivEscApp, CompDB, PrivEscDB\}$	(Attacker has root on DB Server; goal reached.)

Transition Relation τ . Each transition $(s, s') \in \tau$ corresponds to a single exploit or privilege escalation:

- (s_0, s_1) : The attacker exploits a known SQL-injection vulnerability in the Web application to gain code execution on Host₁. Label increment: $\emptyset \rightarrow \{CompWeb\}$.
- (s_1, s_2) : From the Web Server, the attacker discovers valid credentials for Host₂ in a configuration file. Using those credentials, they upload a reverse shell to Host₂, resulting in code execution. Label increment: $\{CompWeb\} \rightarrow \{CompWeb, CompApp\}$.
- (s_2, s_3) : On the Application Server (Host₂), the attacker leverages a local privilege escalation vulnerability (e.g. a misconfigured SUID binary) to gain root on Host₂. Label increment: $\{CompWeb, CompApp\} \rightarrow \{CompWeb, CompApp, PrivEscApp\}$.
- (s_3, s_4) : With root on Host₂, the attacker locates a credential file for Host₃. They use those credentials to obtain a low-privilege shell on Host₃, i.e. “code execution” without full root. Label increment: $\{CompWeb, CompApp, PrivEscApp\} \rightarrow \{CompWeb, CompApp, PrivEscApp, CompDB\}$.
- (s_4, s_5) : On the Database Server (Host₃), the attacker exploits a misconfigured setuid binary (e.g. “mysql” with weak permissions) to escalate from a low-privilege shell to root. Label increment:

$$\begin{aligned} & \{CompWeb, CompApp, PrivEscApp, CompDB\} \rightarrow \\ & \{CompWeb, CompApp, PrivEscApp, CompDB, PrivEscDB\}. \end{aligned}$$

Labeling Function ℓ . By definition,

$$\ell(s_i) = \{s_i, \quad i = 0, 1, 2, 3, 4, 5\}.$$

For example, $\ell(s_3) = \{CompWeb, CompApp, PrivEscApp\}$.

Security Property φ . Let φ be the property “No root on Database Server,” formally

$$\varphi = \neg(PrivEscDB).$$

Any state in which *PrivEscDB* holds is a violation of φ .

Attack Graph $\mathcal{G}$. From the definitions above:

$$S_A = \{s_0, s_1, s_2, s_3, s_4, s_5\}, \quad E = \{s \in S_A \mid \ell(s) \not\models \varphi\} = \{s_5\}.$$

The induced subgraph $\mathcal{G} = (S_A, \tau_A, s_0, E, \ell')$ has:

- $\tau_A = \tau$ (all transitions remain within S_A).
- s_0 as the initial root.
- Error state $E = \{s_5\}$ where the attacker has root on Host₃.

Interpretation.

- $s_0 \rightarrow s_1$: Exploit Web-app SQL injection.
- $s_1 \rightarrow s_2$: Pivot to Application Server using stolen credentials.
- $s_2 \rightarrow s_3$: Privilege-escalate on Application Server.
- $s_3 \rightarrow s_4$: Pivot to Database Server with limited shell.
- $s_4 \rightarrow s_5$: Privilege-escalate to root on Database Server (goal).

Thus the Attack Graph explicitly shows all intermediate states and transitions that lead to the violation $\ell(s_5) \models \neg\varphi$. This concrete example illustrates how an attacker moves step-by-step through multiple hosts, exploiting vulnerabilities and escalating privileges until they compromise the final high-value asset.

2 | Review: Attack HyperGraph

In this section, we present the definitions and foundational concepts related to Attack HyperGraphs (cf. [19, 20]). An Attack HyperGraph represents attack-relevant states as vertices and multi-step exploit sequences as hyperedges, capturing higher-order dependencies and pathways effectively.

Definition 13 (Attack HyperGraph). (cf. [19, 20]) Let $\mathcal{M} = (S, \tau, s_0, \ell)$ be an Attack Model and let φ be a security property over AP . Define the set of *attack-relevant states*

$$S_A = \{ s \in S \mid \exists k \in \mathbb{N}_0, \exists (s_0^t, \dots, s_k^t) \in S^{k+1} : s_0^t = s, (s_i^t, s_{i+1}^t) \in \tau \ (0 \leq i < k), \ell(s_k^t) \not\models \varphi \}.$$

Let $\text{Paths}_{\geq 1}(S_A, \tau)$ denote the set of all finite τ -paths of length at least 1 that stay inside S_A , i.e.

$$\text{Paths}_{\geq 1}(S_A, \tau) = \left\{ t = (s_0^t, \dots, s_k^t) \mid k \geq 1, s_i^t \in S_A \ \forall i, (s_i^t, s_{i+1}^t) \in \tau \ \forall i < k \right\}.$$

An *Attack HyperGraph* is the hypergraph

$$\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H})),$$

where:

- $V(\mathcal{H}) := S_A$.
- $E(\mathcal{H})$ is the collection of (nonempty) hyperedges induced by finite attack paths:

$$E(\mathcal{H}) := \{ e_t \mid t \in \text{Paths}_{\geq 1}(S_A, \tau) \}, \quad e_t := \{ s_0^t, s_1^t, \dots, s_k^t \} \subseteq V(\mathcal{H}).$$

Example 14 (Real-World Example of an Attack HyperGraph). Consider a small enterprise network consisting of four machines:

Host_{Web} : Internet-facing Web Server, Host_{App} : Application Server,

Host_{DB} : Database Server, Host_{Adm} : Domain Controller.

An attacker's ultimate goal is to obtain domain-administrator privileges on Host_{Adm}. We model each identifiable compromise as a state in an Attack Model.

Atomic Propositions AP . Let

$$AP = \{ \text{CompWeb}, \text{CompApp}, \text{CompDB}, \text{CompAdm}, \text{PrivEscApp}, \text{PrivEscDB}, \text{PrivEscAdm} \},$$

where each proposition means:

- *CompWeb*: “Attacker has code execution on Host_{Web}.”
- *CompApp*: “Attacker has code execution on Host_{App}.”
- *CompDB*: “Attacker has code execution on Host_{DB}.”
- *CompAdm*: “Attacker has code execution on Host_{Adm}.”
- *PrivEscApp*: “Attacker has obtained administrative (root) privileges on Host_{App}.”
- *PrivEscDB*: “Attacker has obtained administrative privileges on Host_{DB}.”
- *PrivEscAdm*: “Attacker has obtained domain-administrator privileges on Host_{Adm}.”

States S . Each state $s \in S$ is identified by the set of atomic propositions true in that moment. We write:

$$S = \{ s_0, s_1, s_2, s_3, s_4, s_5, s_6 \},$$

with:

$s_0 = \emptyset$	(No compromises yet; attacker is external.)
$s_1 = \{CompWeb\}$	(Exploit on Web Server.)
$s_2 = \{CompWeb, CompApp\}$	(Pivot from Web to App Server.)
$s_3 = \{CompWeb, CompApp, PrivEscApp\}$	(Escalated to root on App Server.)
$s_4 = \{CompWeb, CompApp, PrivEscApp, CompDB\}$	(Pivoted to Database Server with code execution.)
$s_5 = \{CompWeb, CompApp, PrivEscApp, CompDB, PrivEscDB\}$	(Escalated to root on DB Server.)
$s_6 = \{CompWeb, CompApp, PrivEscApp, CompDB, PrivEscDB, CompAdm, PrivEscAdm\}$	(Obtained domain-admin on Domain Controller.)

Transition Relation τ . Each transition $(s, s') \in \tau$ corresponds to a single exploit or privilege escalation:

- (s_0, s_1) : The attacker exploits a remote code execution (RCE) vulnerability in $Host_{Web}$. Label: $\emptyset \rightarrow \{CompWeb\}$.
- (s_1, s_2) : From Web Server, attacker steals stored credentials for $Host_{App}$ and gains code execution. Label: $\{CompWeb\} \rightarrow \{CompWeb, CompApp\}$.
- (s_2, s_3) : On App Server, attacker exploits a local privilege escalation (e.g. a vulnerable sudo). Label: $\{CompWeb, CompApp\} \rightarrow \{CompWeb, CompApp, PrivEscApp\}$.
- (s_3, s_4) : With root on App Server, attacker obtains credentials for $Host_{DB}$ and gains low-privilege shell. Label: $\{CompWeb, CompApp, PrivEscApp\} \rightarrow \{CompWeb, CompApp, PrivEscApp, CompDB\}$.
- (s_4, s_5) : On DB Server, attacker exploits misconfiguration (e.g. weak MySQL privileges) to escalate to root. Label:

$$\{CompWeb, CompApp, PrivEscApp, CompDB\} \rightarrow \{CompWeb, CompApp, PrivEscApp, CompDB, PrivEscDB\}.$$

- (s_5, s_6) : From DB Server root, attacker reuses Kerberos tickets to compromise Domain Controller, then exploits a known RCE to become domain-admin. Label:

$$\{CompWeb, CompApp, PrivEscApp, CompDB, PrivEscDB\} \rightarrow \{CompWeb, CompApp, PrivEscApp, CompDB, PrivEscDB, CompAdm, PrivEscAdm\}.$$

Labeling Function ℓ . For each state s_i ,

$$\ell(s_i) = s_i.$$

Security Property φ . Let

$$\varphi = \neg(PrivEscAdm),$$

i.e. “The attacker does not have domain-admin on $Host_{Adm}$.” Any state where $PrivEscAdm$ holds violates φ .

Attack-Relevant States S_A . Since from s_0 there is a path to s_6 (which violates φ), all intermediate states are attack-relevant:

$$S_A = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}.$$

Thus $V(\mathcal{H}) = S_A$.

Hyperedges $E(\mathcal{H})$. Each hyperedge $e_t \in E(\mathcal{H})$ corresponds to a *sequence* of one or more consecutive transitions that stay entirely in S_A . Below we list all non-trivial sequences (length ≥ 2) and their corresponding hyperedges:

- Path $t_1 = (s_0, s_1, s_2)$: This sequence covers compromise of Web Server followed by pivot to App Server. The hyperedge is

$$e_{t_1} = \{s_0, s_1, s_2\}.$$

- Path $t_2 = (s_1, s_2, s_3)$: This sequence covers pivot to App and privilege escalation on App. The hyperedge is

$$e_{t_2} = \{s_1, s_2, s_3\}.$$

- Path $t_3 = (s_2, s_3, s_4)$: This sequence covers escalation on App then pivot to DB. The hyperedge is

$$e_{t_3} = \{s_2, s_3, s_4\}.$$

- Path $t_4 = (s_3, s_4, s_5)$: This sequence covers pivot to DB and escalation on DB. The hyperedge is

$$e_{t_4} = \{s_3, s_4, s_5\}.$$

- Path $t_5 = (s_4, s_5, s_6)$: This sequence covers escalation on DB then compromise of Domain Controller. The hyperedge is

$$e_{t_5} = \{s_4, s_5, s_6\}.$$

- Path $t_6 = (s_0, s_1, s_2, s_3)$: Entire sequence from initial compromise to privilege escalation on App. The hyperedge is

$$e_{t_6} = \{s_0, s_1, s_2, s_3\}.$$

- Path $t_7 = (s_1, s_2, s_3, s_4)$: Privilege escalation on App then pivot to DB. The hyperedge is

$$e_{t_7} = \{s_1, s_2, s_3, s_4\}.$$

- Path $t_8 = (s_2, s_3, s_4, s_5)$: Pivot and escalation on DB. The hyperedge is

$$e_{t_8} = \{s_2, s_3, s_4, s_5\}.$$

- Path $t_9 = (s_3, s_4, s_5, s_6)$: Pivot and final compromise of Domain Controller. The hyperedge is

$$e_{t_9} = \{s_3, s_4, s_5, s_6\}.$$

- Path $t_{10} = (s_0, s_1, s_2, s_3, s_4)$: Entire sequence through pivot to DB. The hyperedge is

$$e_{t_{10}} = \{s_0, s_1, s_2, s_3, s_4\}.$$

- Path $t_{11} = (s_1, s_2, s_3, s_4, s_5)$: Full attack through root on DB. The hyperedge is

$$e_{t_{11}} = \{s_1, s_2, s_3, s_4, s_5\}.$$

- Path $t_{12} = (s_2, s_3, s_4, s_5, s_6)$: Pivoted App→DB→Domain. The hyperedge is

$$e_{t_{12}} = \{s_2, s_3, s_4, s_5, s_6\}.$$

- Path $t_{13} = (s_0, s_1, s_2, s_3, s_4, s_5)$: Long path through root on DB. The hyperedge is

$$e_{t_{13}} = \{s_0, s_1, s_2, s_3, s_4, s_5\}.$$

- Path $t_{14} = (s_1, s_2, s_3, s_4, s_5, s_6)$: Long path ending at Domain compromise. The hyperedge is

$$e_{t_{14}} = \{s_1, s_2, s_3, s_4, s_5, s_6\}.$$

- Path $t_{15} = (s_0, s_1, s_2, s_3, s_4, s_5, s_6)$: Entire attack from Web compromise to Domain-admin. The hyperedge is

$$e_{t_{15}} = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}.$$

Interpretation.

- $e_{t_1} = \{s_0, s_1, s_2\}$ groups the two-step path “exploit Web→pivot to App.”

- $e_{t_4} = \{s_3, s_4, s_5\}$ groups “App root→DB compromise→DB root.”
- $e_{t_9} = \{s_3, s_4, s_5, s_6\}$ groups “App root→DB compromise→DB root→Domain compromise.”
- The largest hyperedge, $e_{t_{15}} = \{s_0, \dots, s_6\}$, groups the entire multi-stage attack chain in one set.

Thus the Attack HyperGraph

$$\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H})) = \left(\{s_0, \dots, s_6\}, \{e_{t_1}, e_{t_2}, \dots, e_{t_{15}}\} \right)$$

shows how each hyperedge collects a *sequence* of one or more attack transitions—highlighting not only individual steps but also entire subsequences of the attack chain as single hyperedges. This richer structure can be used to identify multi-stage “atomic” attack patterns and to prioritize defense for those combined sequences.

Theorem 15. *Let $\mathcal{M} = (S, \tau, s_0, \ell)$ be an Attack Model with S finite, and let φ be a security property. Let $\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H}))$ be the Attack HyperGraph. Then $\mathcal{H}$ is a (finite-vertex) hypergraph in the sense that*

$$V(\mathcal{H}) \text{ is finite, and } E(\mathcal{H}) \subseteq 2^{V(\mathcal{H})} \setminus \{\emptyset\}.$$

Moreover, $V(\mathcal{H}) \neq \emptyset$ whenever there exists at least one τ -path from some state to a violation of φ .

Proof: By construction, $V(\mathcal{H}) = S_A \subseteq S$, hence $V(\mathcal{H})$ is finite because S is finite.

Next, every hyperedge in $E(\mathcal{H})$ has the form $e_t = \{s_0^t, \dots, s_k^t\}$ for some finite τ -path $t = (s_0^t, \dots, s_k^t)$ of length $k \geq 1$ that stays inside S_A (i.e. $s_i^t \in S_A$ for all i). Therefore $e_t \neq \emptyset$ and $e_t \subseteq S_A = V(\mathcal{H})$. Hence

$$E(\mathcal{H}) \subseteq 2^{V(\mathcal{H})} \setminus \{\emptyset\}.$$

Finally, if there exists at least one τ -path ending in a violation state (some s_E with $\ell(s_E) \not\models \varphi$), then the start state of that path belongs to S_A , so $S_A \neq \emptyset$ and consequently $V(\mathcal{H}) \neq \emptyset$. $\square$

Theorem 16. *The standard Attack Graph $\mathcal{G} = (S_A, \tau_A, s_0, E, \ell')$ is a special case of the Attack HyperGraph $\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H}))$. In particular, if we restrict every hyperedge in $E(\mathcal{H})$ to be of size exactly 2, then we recover $\mathcal{G}$.*

Proof: Recall that the Attack Graph $\mathcal{G}$ has vertex set S_A and directed edge set $\tau_A \subseteq S_A \times S_A$. Each directed edge $(s, s') \in \tau_A$ corresponds to a single-step exploit. We define a mapping

$$f : \tau_A \longrightarrow E(\mathcal{H}), \quad f((s, s')) = \{s, s'\}.$$

We must show that:

- For each $(s, s') \in \tau_A$, the image hyperedge $e = \{s, s'\}$ belongs to $E(\mathcal{H})$. Indeed, the two-state sequence (s, s') is a valid attack sequence in τ with $s, s' \in S_A$, so by definition $\{s, s'\} \in E(\mathcal{H})$.
- Conversely, any hyperedge $e_t \in E(\mathcal{H})$ of size 2 must arise from a sequence of the form $t = (s, s')$ with $(s, s') \in \tau_A$. Hence every size-2 hyperedge corresponds uniquely to a directed edge in τ_A .

Therefore, if we collect exactly those hyperedges in $E(\mathcal{H})$ whose cardinality is 2, then we obtain the set

$$E_2 = \{\{s, s'\} \mid (s, s') \in \tau_A\}.$$

Interpreting each $\{s, s'\} \in E_2$ as an (undirected) 2-ary hyperedge recovers the adjacency relation of τ_A . In fact, the ordered pair (s, s') in the Attack Graph can be recovered by imposing an orientation on $\{s, s'\}$, but at the level of hypergraphs (which are undirected), the pair $\{s, s'\}$ suffices. Thus the directed graph $\mathcal{G}$ is embedded in $\mathcal{H}$ by viewing each directed edge as a 2-element hyperedge. $\square$

3 | Result: Attack n -SuperHyperGraph

In this section we give a precise construction of an *Attack n -SuperHyperGraph* induced by a given attack model. A (finite) τ -path is a state sequence

$$t = (s_0^t, s_1^t, \dots, s_k^t) \quad (k \in \mathbb{N}_0), \quad (s_i^t, s_{i+1}^t) \in \tau \quad \text{for all } 0 \leq i < k,$$

and we write

$$S(t) := \{s_0^t, \dots, s_k^t\}$$

for the set of states visited by t (duplicates ignored).

Definition 17 (Attack n -SuperHyperGraph). Let $\mathcal{M} = (S, \tau, s_0, \ell)$ be an Attack Model and let φ be a security property over AP . Define the set of *attack-relevant states*

$$S_A := \{s \in S \mid \exists k \in \mathbb{N}_0, \exists t = (s_0^t, \dots, s_k^t) \text{ with } s_0^t = s, (s_i^t, s_{i+1}^t) \in \tau \ (0 \leq i < k), \ell(s_k^t) \not\models \varphi\}.$$

Fix $n \in \mathbb{N}$, and define the iterated powerset operator by

$$\mathcal{P}^0(Y) = Y, \quad \mathcal{P}^{m+1}(Y) = \mathcal{P}(\mathcal{P}^m(Y)) \quad (m \in \mathbb{N}_0).$$

Let $\text{Paths}_{\geq 1}(S_A, \tau)$ be the set of all τ -paths of length at least 1 that stay inside S_A , i.e.

$$\text{Paths}_{\geq 1}(S_A, \tau) = \left\{ (s_0, \dots, s_k) \mid k \geq 1, s_i \in S_A \ \forall i, (s_i, s_{i+1}) \in \tau \ \forall i < k \right\}.$$

Choose a (user-specified) family of paths $\mathcal{T} \subseteq \text{Paths}_{\geq 1}(S_A, \tau)$ to be represented (e.g. all simple paths, or all paths of length $\leq L$, or a set of minimal violation-reaching paths).

The *Attack n -SuperHyperGraph* (over $\mathcal{T}$) is the pair

$$\mathcal{H}_{\mathcal{T}}^{(n)} = (V^{(n)}, E^{(n)})$$

defined as follows.

- **n -supervertices (canonical lift of a path).** For each $t \in \mathcal{T}$, define

$$X_1^t := S(t) \in \mathcal{P}(S_A), \quad X_{m+1}^t := \{X_m^t\} \in \mathcal{P}^{m+1}(S_A) \quad (1 \leq m < n).$$

Set

$$\nu_t^{(n)} := X_n^t \in \mathcal{P}^n(S_A).$$

- **Vertices.**

$$V^{(n)} := \{\nu_t^{(n)} \mid t \in \mathcal{T}\} \subseteq \mathcal{P}^n(S_A).$$

- **n -superedges (minimal singleton construction).**

$$E^{(n)} := \{\{\nu_t^{(n)}\} \mid t \in \mathcal{T}\} \subseteq \mathcal{P}(V^{(n)}) \setminus \{\emptyset\}.$$

Hence $\mathcal{H}_{\mathcal{T}}^{(n)}$ is an n -SuperHyperGraph: its vertices lie in $\mathcal{P}^n(S_A)$ and its hyperedges are nonempty subsets of $V^{(n)}$.

Remark 18 (On finiteness and variants). If $\mathcal{T}$ is finite (as in most practical settings, e.g. bounded-length or simple paths), then $V^{(n)}$ and $E^{(n)}$ are finite. More expressive variants may aggregate several lifted paths into one hyperedge, e.g. grouping $\nu_t^{(n)}$ by a shared endpoint, shared asset set, or shared violation class; the singleton choice above is the minimal admissible construction.

Example 19 (Enterprise network, $n = 2$ (minimal singleton hyperedges)). Consider the four-host setting

$$\text{Host}_{\text{Web}}, \text{Host}_{\text{App}}, \text{Host}_{\text{DB}}, \text{Host}_{\text{Adm}}$$

with states $s_0, \dots, s_6$ as specified previously and $S_A = \{s_0, \dots, s_6\}$. Take $\mathcal{T} = \{t_a, t_b, t_c\}$ with

$$t_a = (s_0, s_1, s_2), \quad t_b = (s_3, s_4, s_5), \quad t_c = (s_4, s_5, s_6).$$

For $n = 2$, the 2-supervertex associated with t_a is

$$X_1^{t_a} = S(t_a) = \{s_0, s_1, s_2\} \in \mathcal{P}(S_A), \quad \nu_{t_a}^{(2)} = X_2^{t_a} = \{X_1^{t_a}\} = \{\{s_0, s_1, s_2\}\} \in \mathcal{P}^2(S_A),$$

and the corresponding singleton 2-superedge is

$$e_{t_a}^{(2)} = \{\nu_{t_a}^{(2)}\} = \{\{\{s_0, s_1, s_2\}\}\} \in \mathcal{P}(V^{(2)}).$$

Analogously,

$$e_{t_b}^{(2)} = \{\{\{s_3, s_4, s_5\}\}\}, \quad e_{t_c}^{(2)} = \{\{\{s_4, s_5, s_6\}\}\}.$$

Thus

$$V^{(2)} = \{\nu_{t_a}^{(2)}, \nu_{t_b}^{(2)}, \nu_{t_c}^{(2)}\} \subseteq \mathcal{P}^2(S_A), \quad E^{(2)} = \{\{\nu_t^{(2)}\} \mid t \in \mathcal{T}\} \subseteq \mathcal{P}(V^{(2)}).$$

Each singleton hyperedge $\{\nu_t^{(2)}\}$ represents, as one atomic element, the 2-level lift of the multi-step subsequence t , supporting bundled defensive prioritization for that chained pattern.

Example 20 (Cloud setting, $n = 2$ (minimal singleton hyperedges)). Let the components and states $s_0, \dots, s_6$ be as specified (Workstation, WebApp, DBServer, CloudMgmt) with $S_A = \{s_0, \dots, s_6\}$ and security goal $\varphi = \neg(\text{PrivEscCloud})$. Take $\mathcal{T} = \{t_u, t_v, t_w\}$ with

$$t_u = (s_1, s_2, s_3), \quad t_v = (s_2, s_3, s_4, s_5), \quad t_w = (s_3, s_4, s_5, s_6).$$

For each $t \in \mathcal{T}$,

$$\nu_t^{(2)} = \{S(t)\} \in \mathcal{P}^2(S_A), \quad e_t^{(2)} = \{\nu_t^{(2)}\} \in \mathcal{P}(V^{(2)}).$$

Concretely,

$$e_{t_u}^{(2)} = \{\{\{s_1, s_2, s_3\}\}\}, \quad e_{t_v}^{(2)} = \{\{\{s_2, s_3, s_4, s_5\}\}\}, \quad e_{t_w}^{(2)} = \{\{\{s_3, s_4, s_5, s_6\}\}\}.$$

The singletons $\{\nu_t^{(2)}\}$ act as compact surrogates for contiguous multi-step kill chains, enabling joint prioritization of controls (token hygiene, build attestation, identity hardening) per chain.

Theorem 21. Let $n \in \mathbb{N}$ and let $\mathcal{H}_{\mathcal{T}}^{(n)} = (V^{(n)}, E^{(n)})$ be the Attack n -SuperHyperGraph constructed in Definition 17 over a path family $\mathcal{T} \subseteq \text{Paths}_{\geq 1}(S_A, \tau)$. Then $\mathcal{H}_{\mathcal{T}}^{(n)}$ is an n -SuperHyperGraph in the sense that

$$V^{(n)} \subseteq \mathcal{P}^n(S_A) \quad \text{and} \quad E^{(n)} \subseteq \mathcal{P}(V^{(n)}) \setminus \{\emptyset\}.$$

Proof: By Definition 17, for each $t \in \mathcal{T}$ we define $X_1^t = S(t) \in \mathcal{P}(S_A)$ and $X_{m+1}^t = \{X_m^t\} \in \mathcal{P}^{m+1}(S_A)$ for $1 \leq m < n$. Hence the associated n -supervertex $\nu_t^{(n)} = X_n^t$ satisfies $\nu_t^{(n)} \in \mathcal{P}^n(S_A)$. Since $V^{(n)} = \{\nu_t^{(n)} \mid t \in \mathcal{T}\}$, we obtain $V^{(n)} \subseteq \mathcal{P}^n(S_A)$.

Moreover, each n -superedge in the minimal construction is a singleton of the form $\{\nu_t^{(n)}\}$ with $t \in \mathcal{T}$. Therefore $\{\nu_t^{(n)}\} \subseteq V^{(n)}$, and in particular $\{\nu_t^{(n)}\} \in \mathcal{P}(V^{(n)})$ and $\{\nu_t^{(n)}\} \neq \emptyset$. It follows that

$$E^{(n)} = \{\{\nu_t^{(n)}\} \mid t \in \mathcal{T}\} \subseteq \mathcal{P}(V^{(n)}) \setminus \{\emptyset\}.$$

□

Remark 22 (Relation to Attack HyperGraphs and Attack Graphs). Let $n = 1$. Then $\nu_t^{(1)} = X_1^t = S(t) \subseteq S_A$ for each $t \in \mathcal{T}$, and $V^{(1)} = \{S(t) \mid t \in \mathcal{T}\} \subseteq \mathcal{P}(S_A)$. The associated singleton-edge structure $\mathcal{H}_{\mathcal{T}}^{(1)}$ can be projected to a standard Attack HyperGraph whose vertex set is S_A and whose hyperedge family is $\{S(t) \mid t \in \mathcal{T}\}$ via the map

$$\pi : E^{(1)} \longrightarrow \mathcal{P}(S_A) \setminus \{\emptyset\}, \quad \pi(\{\nu_t^{(1)}\}) = \nu_t^{(1)} = S(t).$$

Thus, each singleton hyperedge of $\mathcal{H}_{\mathcal{T}}^{(1)}$ corresponds to the ordinary hyperedge $S(t)$ in the Attack HyperGraph.

If we further restrict $\mathcal{T}$ to paths of length 1, i.e. $\mathcal{T} \subseteq \{(s, s') \mid (s, s') \in \tau, s, s' \in S_A\}$, then each $S(t)$ has size 2. Interpreting $S(t) = \{s, s'\}$ as an (undirected) 2-hyperedge recovers the adjacency of the Attack Graph; equipping $\{s, s'\}$ with the orientation $s \rightarrow s'$ induced by $(s, s') \in \tau$ recovers the directed edges of the classical Attack Graph.

Theorem 23 (Injectivity of the canonical lift). *Assume that $\mathcal{T}$ contains no two distinct paths with the same visited-state set, i.e.*

$$t, u \in \mathcal{T}, t \neq u \implies S(t) \neq S(u).$$

Then the map

$$\Lambda_n : \mathcal{T} \longrightarrow V^{(n)}, \quad \Lambda_n(t) := \nu_t^{(n)},$$

is injective. Consequently, $|V^{(n)}| = |\mathcal{T}|$.

Proof: Let $t, u \in \mathcal{T}$ and suppose $\Lambda_n(t) = \Lambda_n(u)$, i.e. $\nu_t^{(n)} = \nu_u^{(n)}$. By construction, $\nu_t^{(n)}$ is obtained by iterating the singleton operator on $S(t)$:

$$\nu_t^{(n)} = \underbrace{\{\dots \{S(t)\} \dots\}}_{n-1 \text{ singletons}}.$$

The singleton operator is injective on sets, and so is any finite iterate of it. Hence $\nu_t^{(n)} = \nu_u^{(n)}$ implies $S(t) = S(u)$. By the hypothesis on $\mathcal{T}$, $S(t) = S(u)$ forces $t = u$. Therefore Λ_n is injective. Since $V^{(n)} = \{\nu_t^{(n)} \mid t \in \mathcal{T}\}$, we also get $|V^{(n)}| = |\mathcal{T}|$. $\square$

Theorem 24 (Singleton-edge degree). *In the minimal singleton-edge construction, every vertex of $\mathcal{H}_{\mathcal{T}}^{(n)}$ is incident with exactly one hyperedge. Equivalently, for each $v \in V^{(n)}$ there exists a unique $e \in E^{(n)}$ such that $v \in e$, and that $e = \{v\}$.*

Proof: By definition, $E^{(n)} = \{\{\nu_t^{(n)}\} \mid t \in \mathcal{T}\}$. Fix any $v \in V^{(n)}$. Then $v = \nu_t^{(n)}$ for some $t \in \mathcal{T}$, and the hyperedge $e = \{v\} = \{\nu_t^{(n)}\}$ belongs to $E^{(n)}$ and satisfies $v \in e$. If $e' \in E^{(n)}$ also satisfies $v \in e'$, then $e' = \{\nu_u^{(n)}\}$ for some $u \in \mathcal{T}$ and $v \in \{\nu_u^{(n)}\}$ implies $v = \nu_u^{(n)}$, hence $e' = \{v\} = e$. Thus the incident hyperedge is unique and is the singleton $\{v\}$. $\square$

Theorem 25 (Edge set size). *If $V^{(n)} \neq \emptyset$, then $|E^{(n)}| = |V^{(n)}|$. More precisely, the map*

$$\Phi : V^{(n)} \longrightarrow E^{(n)}, \quad \Phi(v) := \{v\},$$

is a bijection.

Proof: By Theorem 24, for each $v \in V^{(n)}$ the set $\{v\}$ is an element of $E^{(n)}$. Hence Φ is well-defined. It is injective because $\{v\} = \{w\}$ implies $v = w$. It is surjective because each $e \in E^{(n)}$ is, by construction, a singleton $\{v\}$ for some $v \in V^{(n)}$. Therefore Φ is a bijection and $|E^{(n)}| = |V^{(n)}|$. $\square$

Theorem 26 (Compatibility across levels). *For $n \geq 2$, define the level-reduction map*

$$\rho_n : V^{(n)} \longrightarrow \mathcal{P}^{n-1}(S_A), \quad \rho_n(Z) := \bigcup Z.$$

Then for every $t \in \mathcal{T}$,

$$\rho_n(\nu_t^{(n)}) = \nu_t^{(n-1)}.$$

In particular, the image $\rho_n(V^{(n)})$ equals $V^{(n-1)}$ when both levels are built over the same $\mathcal{T}$.

Proof: Fix $t \in \mathcal{T}$. By construction, $\nu_t^{(n)} = \{\nu_t^{(n-1)}\}$ for $n \geq 2$. Therefore

$$\rho_n(\nu_t^{(n)}) = \bigcup \{\nu_t^{(n-1)}\} = \nu_t^{(n-1)},$$

as claimed. If both $V^{(n)}$ and $V^{(n-1)}$ are defined over the same $\mathcal{T}$, then $\rho_n(V^{(n)}) = \{\nu_t^{(n-1)} \mid t \in \mathcal{T}\} = V^{(n-1)}$. $\square$

Theorem 27 (Monotonicity under path-family extension). *Let $\mathcal{T} \subseteq \mathcal{T}' \subseteq \text{Paths}_{\geq 1}(S_A, \tau)$, and let $\mathcal{H}_{\mathcal{T}}^{(n)} = (V^{(n)}, E^{(n)})$ and $\mathcal{H}_{\mathcal{T}'}^{(n)} = (V'^{(n)}, E'^{(n)})$ be the corresponding Attack n -SuperHyperGraphs (using the same minimal singleton-edge rule). Then*

$$V^{(n)} \subseteq V'^{(n)} \quad \text{and} \quad E^{(n)} \subseteq E'^{(n)}.$$

Proof: If $\mathcal{T} \subseteq \mathcal{T}'$, then $\{\nu_t^{(n)} \mid t \in \mathcal{T}\} \subseteq \{\nu_t^{(n)} \mid t \in \mathcal{T}'\}$, hence $V^{(n)} \subseteq V'^{(n)}$. Applying the singleton-edge construction to both sides yields

$$E^{(n)} = \{\{\nu_t^{(n)}\} \mid t \in \mathcal{T}\} \subseteq \{\{\nu_t^{(n)}\} \mid t \in \mathcal{T}'\} = E'^{(n)}.$$

□

4 | Conclusion

In this paper, we investigated *Attack Superhypergraphs*, which extend Attack Graphs within the superhypergraph framework. In future work, we plan to examine extensions based on Fuzzy Graphs [21, 22], Intuitionistic Fuzzy Graphs [23, 24], Neutrosophic Graphs [25, 26, 27], and Plithogenic Graphs [28]. Since the present paper focuses primarily on theoretical development, we also expect that future studies will advance quantitative analyses and effectiveness evaluations, for instance through computer experiments and empirical validation.

Acknowledgments

The author is grateful to the editorial and reviewers, as well as the correspondent author, who offered assistance in the form of advice, assessment, and checking during the study period.

Funding

This research has no funding source.

Data Availability

The datasets generated during and/or analyzed during the current study are not publicly available due to the privacy-preserving nature of the data but are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that there is no conflict of interest in the research.

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

- [1] Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.
- [2] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.
- [3] Takaaki Fujita and Florentin Smarandache. *HyperGraph and SuperHyperGraph Theory with Applications (IV): Uncertain Graph Theory*, volume IV of *HyperGraph and SuperHyperGraph Theory with Applications*. Neutrosophic Science International Association (NSIA) Publishing House, 1.0 edition, 2026.
- [4] Takaaki Fujita and Florentin Smarandache. *HyperGraph and SuperHyperGraph Theory with Applications (III): Intersection Graph and Graph Labeling*, volume III of *HyperGraph and SuperHyperGraph Theory with Applications*. Neutrosophic Science International Association (NSIA) Publishing House, 2026.
- [5] Takaaki Fujita and Florentin Smarandache. *HyperGraph and SuperHyperGraph Theory with Applications (II): Graph Property and Parameter*, volume II of *HyperGraph and SuperHyperGraph Theory with Applications*. Neutrosophic Science International Association (NSIA) Publishing House, 1.0 edition, 2026.
- [6] Takaaki Fujita and Florentin Smarandache. *HyperGraph and SuperHyperGraph Theory with Applications*. Neutrosophic Science International Association (NSIA) Publishing House, 2026.
- [7] Shouxian Zhu. Neutrosophic n-superhypernetwork: A new approach for evaluating short video communication effectiveness in media convergence. *Neutrosophic Sets and Systems*, 85:1004–1017, 2025.
- [8] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering*. Cham: Springer, 1, 2013.
- [9] Florentin Smarandache. Foundation of superhyperstructure & neutrosophic superhyperstructure. *Neutrosophic Sets and Systems*, 63(1):21, 2024.

- [10] F. Smarandache. Introduction to superhyperalgebra and neutrosophic superhyperalgebra. *Journal of Algebraic Hyperstructures and Logical Algebras*, 2022.
- [11] Masoud Ghods, Zahra Rostami, and Florentin Smarandache. Introduction to neutrosophic restricted superhypergraphs and neutrosophic restricted superhypertrees and several of their properties. *Neutrosophic Sets and Systems*, 50:480–487, 2022.
- [12] Florentin Smarandache. *Extension of HyperGraph to n-SuperHyperGraph and to Plithogenic n-SuperHyperGraph, and Extension of HyperAlgebra to n-ary (Classical-/Neutro-/Anti-) HyperAlgebra*. Infinite Study, 2020.
- [13] Harith A Dawood. Graph theory and cyber security. In *2014 3rd International Conference on Advanced Computer Science Applications and Technologies*, pages 90–96. IEEE, 2014.
- [14] Sangeetha Nandigam and Perumali Sundarayya. Graph theory for data cryptography security. *International Journal of Mathematics in Operational Research*, 30(2):167–184, 2025.
- [15] Peter D Zegzhda, Dmitry P Zegzhda, and Alexey V Nikolskiy. Using graph theory for cloud system security modeling. In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 309–318. Springer, 2012.
- [16] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. Ranking attack graphs. In *International Workshop on Recent Advances in Intrusion Detection*, pages 127–144. Springer, 2006.
- [17] Jianping Zeng, Shuang Wu, Yanyu Chen, Rui Zeng, and Chengrong Wu. Survey of attack graph analysis methods from the perspective of data and knowledge processing. *Security and Communication Networks*, 2019(1):2031063, 2019.
- [18] Matej Rabzelj, Ciril Bohak, Leon Štefanič Južnič, Andrej Kos, and Urban Sedlar. Cyberattack graph modeling for visual analytics. *IEEE Access*, 11:86910–86944, 2023.
- [19] Rudra Prasad Chatterjee, Islaur Rahaman Ruku, Mridul Sankar Barik, and Ayan Gain. Attack hypergraph: A framework for modeling multi-stage attacks. In *Analytics Global Conference*, pages 19–28. Springer, 2024.
- [20] Junbo Jia, Li Yang, Yuchen Wang, and Anyuan Sang. Hyper attack graph: Constructing a hypergraph for cyber threat intelligence analysis. *Computers & Security*, 149:104194, 2025.
- [21] John N Mordeson and Premchand S Nair. *Fuzzy graphs and fuzzy hypergraphs*, volume 46. Physica, 2012.
- [22] Azriel Rosenfeld. Fuzzy graphs. In *Fuzzy sets and their applications to cognitive and decision processes*, pages 77–95. Elsevier, 1975.
- [23] Sankar Sahoo and Madhumangal Pal. Product of intuitionistic fuzzy graphs and degree. *Journal of Intelligent & Fuzzy Systems*, 32(1):1059–1067, 2017.
- [24] M. Mohammed Jabarulla and K. Asmiya Banu. Anti intuitionistic fuzzy graph. *Advances and Applications in Mathematical Sciences*, 20(4):597–604, 2021.
- [25] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Single valued neutrosophic graphs. *Journal of New theory*, (10):86–101, 2016.
- [26] Said Broumi, Mohamed Talea, Assia Bakali, Florentin Smarandache, and PK Kishore Kumar. Shortest path problem on single valued neutrosophic graphs. In *2017 international symposium on networks, computers and communications (ISNCC)*, pages 1–6. IEEE, 2017.
- [27] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Interval valued neutrosophic graphs. *Critical Review, XII*, 2016:5–33, 2016.
- [28] Fazeelat Sultana, Muhammad Gulistan, Mumtaz Ali, Naveed Yaqoob, Muhammad Khan, Tabasam Rashid, and Tauseef Ahmed. A study of plithogenic graphs: applications in spreading coronavirus disease (covid-19) globally. *Journal of ambient intelligence and humanized computing*, 14(10):13139–13159, 2023.

Disclaimer/Publisher’s Note

The perspectives, opinions, and data shared in all publications are the sole responsibility of the individual authors and contributors and do not necessarily reflect the views of Sciences Force or the editorial team. Sciences Force and the editorial team disclaim any liability for potential harm to individuals or property resulting from the ideas, methods, instructions, or products referenced in the content.