TLCTC Framework Glossary

Version 2.0 / 2.1

BK
Bernhard Kreinz
• Last updated: 19 Apr 2026 • Loading read time...

A

Abuse of Functions (#1)

A threat cluster where an attacker misuses the logic, scope, or configuration of existing, legitimate software functions for malicious purposes. This manipulation occurs through standard interfaces using expected input types (data, parameters, configurations, sequence of actions), but in a way that subverts the intended purpose or security controls. Crucially, inputs remain data; no foreign code is introduced or executed. The generic vulnerability is the scope, complexity, or inherent trust placed in legitimate software functions. Classification is governed by the R-ABUSE mapping rule: if the attacker's success does not require any implementation flaw and instead abuses intended functionality, scope, or configuration via standard interfaces using expected input types, the step MUST be classified as `#1 Abuse of Functions`.

Reference: R-ABUSE (§2.2.4)

Accessibility (Data Risk Event)

The operational state in which data or resources can be used for their intended purpose by authorized processes (Facility, IT, or Business processes). Loss of Accessibility occurs when data exists but cannot be used—such as encrypted files (ransomware), corrupted data, or permission lockouts. This is distinct from Loss of Availability: ransomware causes Loss of Accessibility (data present but unusable), not Loss of Availability (data gone or unreachable). DRE notation: `Ac` when distinguished from Availability, or `A` as general shorthand covering both.

Actor Archetype

A coarse categorization of threat actors used in TLCTC overlays (Attacker Profiles, Tech Enablers Overlay) to group actors by motivation and resourcing level rather than by identity. The five archetypes are: **Nation State**, **Extortion**, **Fraud**, **Hacktivist**, and **Amateur**. Archetypes are a communication and trend-watching device — they are never used for cluster classification (Axiom IV). An archetype describes *which clusters an actor class tends to prefer* and *which emerging tech enablers that class is likely to adopt*, not the cluster of any specific attack step.

Reference: §17.3 (Attacker Profiles), §17.4 (Tech Enablers Overlay)

See also: Attacker Profile, Tech Enablers Overlay, Cyber Threat Radar, Axiom IV

Attacker Profile *(V2.1)*

An **informative overlay** on the Cyber Threat Radar that describes a threat actor's (or Actor Archetype's) observed preferences across the 10 clusters: per-cluster capability scores, preferred cluster sequences, and typical boundary crossings. Profiles may be wrapped in Diamond-Model framing (adversary / capability / infrastructure / victim). Used for hypothesis generation ("likely next steps"), comparative radar views, and targeting analysis against the organizational radar. **Normative guardrails (R-RADAR-6…9):** actors are not clusters, they *use* clusters (Axiom IV); profiles MUST NOT redefine cluster meanings or introduce actor-based taxonomy; profiles SHOULD be published as probability distributions or pattern frequencies, not deterministic rules; profile scores SHOULD be derived from classified Layer 3 attack path instances so provenance traces back to evidence.

Reference: §17.3

See also: Actor Archetype, Cyber Threat Radar, Axiom IV

Attacker's View

A perspective included in each TLCTC threat cluster definition that describes how the attacker perceives or approaches the exploitation of the specific generic vulnerability. It helps distinguish between clusters by focusing on the attacker's methodology rather than technical implementation details.

Attack Path

The sequence of applied Attack Vectors in a cyber incident, representing an ordered sequence of Attack Steps describing a complete attack scenario. Basic notation uses `#X → #Y → #Z` (e.g., `#9→#3→#7`). Attack paths may include velocity annotations showing the time between steps (e.g., `#9→[24h]#4→[12m]#1`), domain boundary markers using the `||` operator, parallel steps, and Data Risk Event tags.

Reference: §2.2.2 (Global Definitions), §3.0 (Path Semantics)

Attack Path Notation

The standardized format for describing cyber attack sequences using TLCTC clusters. Format uses: `→` for sequential steps, `+` for parallel execution, `[time]` for temporal intervals, and `|||[context][@Source→@Target]||` for domain boundaries. Example: `#9→[24h]#4→[12m]#1 |||[dev][@Vendor→@Org]|| →[weeks]#10.2→[0s]#7`.

Attack Sequence Schema *(V2.0)*

The JSON schema that defines the required structure for documenting attack path instances. The schema ensures all documented attacks follow a consistent format including metadata (sequence_id, attack_title, framework_version), temporal transitions (delta_t_value, delta_t_unit, velocity_class), responsibility spheres, and cluster mappings. File format: `tlctc-attack-sequence-schema.json`.

Attack Step *(V2.0)*

A single attacker action or event that exploits exactly **one generic vulnerability** in a specific context. Each Attack Step MUST map to exactly one TLCTC cluster (per Axiom VI).

Reference: §2.2.2 (Global Definitions), §2.2.7 (Minimal Classification Procedure)

Attack Vector

The specific path or method used by an attacker to gain unauthorized access to a target system. In the TLCTC framework, each distinct attack vector is a distinct initiating method defined by the **initial generic vulnerability targeted** (per Axiom VII). The vector label MUST be based on cause, not outcome.

Reference: §2.2.2 (Global Definitions), Axiom VII (§1.2)

Attack Velocity (Δt) *(V2.0)*

The temporal dimension of cyber risk representing the **time interval** between two adjacent Attack Steps in an attack path. For an edge `#X → #Y`, the value `Δt(X→Y)` represents the elapsed time between step `#X` and step `#Y` in the described scenario. Δt is an edge property attached to the sequence operator, not to steps. Attack velocity is the single most accurate predictor of attacker sophistication and the only metric that truthfully measures control effectiveness. Categorized into four velocity classes: Latent/Slow (days to months), Medium (hours), Fast (minutes), and Realtime (seconds/milliseconds).

Reference: §4.0 (Definitions), §4.1 (Measurement Model), §4.2 (Notation)

Availability (Data Risk Event)

The technical state in which data or resources exist and can be reached by the infrastructure. Availability answers the question: "Does the data exist and can the system access it?" A resource is available if it is present, can be enumerated, and is technically accessible to the system—regardless of whether it can be meaningfully used. For example, encrypted files remain available (they exist on disk) even when rendered unusable by ransomware. Loss of Availability occurs when data is deleted, storage fails, or systems go offline. DRE notation: `Av` when distinguished from Accessibility, or `A` as general shorthand covering both.

Axiom

A foundational premise that defines what terms mean and what kinds of statements are allowed in TLCTC. Axioms are non-negotiable constraints on interpretation that force methodological consequence and prevent logical shortcuts and category errors. These foundational principles must be accepted to validate and effectively use the TLCTC framework. TLCTC defines ten axioms organized into four groups: Scope (I–II), Separation (III–V), Classification (VI–VIII), and Sequence (IX–X).

Reference: §1.2 (Axioms and Assumptions)

A (continued — Industry Terms)

Adware *(Industry Term)*

A type of malware that delivers unwanted advertisements, often bundled with legitimate software. In TLCTC: maps to `#7 Malware` — the software environment's designed execution capability is abused to run foreign advertising code. Adware that arrives via a compromised software package may involve a `#10 → #7` sequence.

See also: Malware (#7), Supply Chain Attack (#10)

AI / AGI / ASI (Positioning in TLCTC)

Artificial Intelligence, Artificial General Intelligence, and Artificial Super Intelligence occupy three distinct roles in the TLCTC framework:

- **As an IT system:** AI is exposed to the same 10 threat clusters as any other IT system (software + hardware).
- **As a tool:** AI enhances the capabilities of both threat actors AND defenders.
- **As AGI/ASI:** Would become a powerful threat actor OR defender in its own right.

AI systems do not create new threat clusters — they are subject to the existing 10 clusters and may amplify their exploitation.

Reference: V1.9.1 Clarifications

Amplification Attack *(Industry Term)*

A flooding technique where an attacker sends small requests to third-party services (e.g., NTP, DNS, memcached) that respond with disproportionately large replies directed at the victim. In TLCTC: maps to `#6 Flooding Attack` — the primary mechanism is volume exceeding finite capacity. The abuse of the amplification service itself may additionally involve `#1 Abuse of Functions`.

See also: Flooding Attack (#6), DDoS

ARP Spoofing *(Industry Term)*

A technique where an attacker sends falsified ARP (Address Resolution Protocol) messages on a local network, linking the attacker's MAC address to a legitimate IP address. In TLCTC: the ARP spoofing itself is `#1 Abuse of Functions` (abusing a legitimate network protocol without exploiting an implementation flaw). ARP spoofing typically leads to a Man in the Middle position, making the full sequence `#1 → #5`.

Reference: V1.9.1 Buzz-Word Refinement (#1)

See also: Man in the Middle (#5), DNS Spoofing, SSL Stripping

B

Bounded Δt

A minimum or maximum bound for Δt derived from known constraints when precise timestamps are unavailable. Notation: `Δt<15m` (upper bound), `Δt>15m` (lower bound), `Δt=10m..20m` (range).

Reference: §4.0.3, §4.2.3

Bow-Tie Model

A risk model that represents risk as a structure with five elements: Threats (left side), Preventive Controls (left side), Central Event (knot), Mitigating Controls (right side), and Consequences (right side). TLCTC is anchored in the Bow-Tie model to enforce strict separation between cause and effect in cyber risk analysis. The model enforces temporal causality, prevents confusion between threats and outcomes, enables precise control placement, and reveals attack sequences as causal chains. The central event "Loss of Control" serves as the pivot point between threat realization and potential consequences.

Reference: §1.4 (The Bow-Tie Anchor), §1.4.1 (Structure and Vocabulary)

Bridge Cluster

A TLCTC cluster whose generic vulnerability **inherently** enables crossing into (or leveraging over) a different domain's control regime. Bridge clusters are: `#8 Physical Attack`, `#9 Social Engineering`, and `#10 Supply Chain Attack`.

Reference: §2.2.2 (Global Definitions), §5.1.4, §5.2 (Topology Classification)

Bridge Step

A step-level instance of a bridge cluster that crosses a specific domain boundary. When a bridge step crosses responsibility spheres, the boundary SHOULD be recorded in path notation via the domain boundary operator `|||[context][@Source→@Target]||`.

Reference: §2.2.2 (Global Definitions), §5.1.6

BGP Hijacking *(Industry Term)*

A technique where an attacker manipulates Border Gateway Protocol routing tables to redirect internet traffic through attacker-controlled infrastructure. In TLCTC: the BGP manipulation itself is `#1 Abuse of Functions` (abusing legitimate protocol functionality, not an implementation flaw). BGP hijacking typically leads to a Man in the Middle position, making the full sequence `#1 → #5`.

Reference: V1.9.1 Buzz-Word Refinement (#1)

See also: ARP Spoofing, DNS Spoofing, Man in the Middle (#5)

Botnet *(Industry Term)*

A network of compromised devices ("bots" or "zombies") controlled by an attacker, typically used to amplify attack capabilities. In TLCTC: the botnet itself is the result of prior `#7 Malware` infections on each device. When a botnet is used for flooding, the attack maps to `#6 Flooding Attack`. The full lifecycle may involve: `#9 → #7` (social engineering delivering malware to build the botnet), then `#6` (coordinated flooding from the botnet).

See also: Flooding Attack (#6), Malware (#7), DDoS

Brute-Force Attack *(Industry Term)*

A method of systematically trying all possible credential combinations (passwords, PINs, encryption keys) to gain unauthorized access. In TLCTC: maps to `#4 Identity Theft` — the attacker is attempting to derive and use credentials to impersonate a legitimate identity. The generic vulnerability exploited is weak credential protection (e.g., lack of account lockout, short password requirements).

Reference: V1.9.1 Buzz-Word Refinement (#4)

See also: Identity Theft (#4), Password Spraying, Credential / Identity Artifact

Buffer Overflow *(Industry Term)*

A class of implementation flaw where a program writes data beyond the boundaries of allocated memory, potentially allowing an attacker to execute arbitrary code or crash the application. In TLCTC: maps to `#2 Exploiting Server` or `#3 Exploiting Client` depending on whether the vulnerable component is in a server role (accepting inbound requests) or client role (consuming external responses) per R-ROLE. Buffer overflows create an unintended data→code transition.

Reference: V1.9.1 §Definitions (#2, #3), Buzz-Word Refinement

See also: Exploiting Server (#2), Exploiting Client (#3), Implementation Flaw

Business Risk Event (BRE)

A discrete, observable business-level event on the consequence side of the Bow-Tie model, triggered by a Data Risk Event or by a preceding BRE. Examples include regulatory notification obligations, service outage declarations, media coverage, customer churn, and regulatory fines. BREs may **chain**: each BRE can trigger subsequent BREs, forming a variable-length consequence sequence (`SRE → DRE → BRE₁ → BRE₂ → ... → BREₙ`). Each BRE→BRE transition has its own Δt representing a detection and intervention window. An organization's Risk Appetite determines at which point a BRE is designated as the terminal **Business Impact (BI)** — BI is a role a BRE can hold, not a separate event category.

Reference: §1.4.3.1 (The Consequence Chain), V1.9.1 §The Anatomy of Risk

See also: System Risk Event (SRE), Data Risk Event (DRE), Business Impact (BI), Event Chain, Consequences

Business Impact (BI)

A **role** assigned to the terminal Business Risk Event in a consequence chain — the BRE beyond which further causal decomposition is no longer operationally useful for a given organization. BI is not a separate event category; it is the point in the BRE chain where an organization's **Risk Appetite** boundary is reached. What constitutes BI for one organization may be a mid-chain BRE for another: a €50K regulatory fine may be terminal impact for a startup but a mid-chain event for a multinational. The BI designation is therefore context-dependent and organization-specific.

Reference: §1.4.3.1 (The Consequence Chain)

See also: Business Risk Event (BRE), Risk Appetite, Event Chain

BxIs (Base Level Indicators)

The lowest level of indicators that still make operational sense, representing metrics at the operational level directly translated into measurable values. Part of the hierarchical KxI framework (KRIs, KCIs, KPIs).

C

BEC (Business Email Compromise) / CEO Fraud *(Industry Term)*

A social engineering attack where an adversary impersonates a senior executive or trusted business partner (often via compromised or spoofed email) to trick employees into transferring funds, revealing sensitive information, or taking other harmful actions. In TLCTC: the manipulation of the human target maps to `#9 Social Engineering`. If the attacker uses compromised email credentials, the email access maps to `#4 Identity`

`Theft`. Typical sequences: `#9 → #1` (social engineering leading to function abuse, e.g., wire transfer) or `#4 → #9 → #1` (stolen credentials enabling impersonation for social engineering leading to function abuse).

Reference: V1.9.1 Buzz-Word Refinement (#9 — "CEO Fraud", "Invoice Manipulation Fraud (BEC Fraud)")

See also: Social Engineering (#9), Whaling, Phishing

Call-Level Mapping Rule

A TLCTC classification principle for function-call-level threat analysis:

- **Parameter tampering**, unauthorized function selection, or misuse of valid functions without executing foreign code → always `#1 Abuse of Functions`.
- **Presentation of identity artifacts** at call time (e.g., stolen API keys, session tokens, cookies, Kerberos tickets) to impersonate a subject → always `#4 Identity Theft`.

This rule prevents overlap between logic misuse (`#1`) and identity presentation/use (`#4`) at the function call level, where the caller acts as "client" and the called function as "server".

Reference: V1.9.1 §Concept Applicability (At Function Call Level)

See also: Abuse of Functions (#1), Identity Theft (#4)

CAPEC (Common Attack Pattern Enumeration and Classification) *(Industry Term)*

A MITRE-maintained dictionary of known attack patterns, each describing a method of exploiting known weaknesses. In TLCTC: CAPEC patterns are operational-level detail that map to the strategic-level TLCTC clusters. CAPEC complements CWE (weaknesses) and CVE (specific vulnerabilities) in the conceptual hierarchy, and TLCTC proposes bridging from NIST to the extended MITRE world (ATT&CK, CWE, CAPEC, CVE) through the 10 clusters.

Reference: V1.9.1 §Standardizing Strategic Cybersecurity

See also: CWE, CVE, MITRE ATT&CK, Techniques (TTPs)

Capacity Exhaustion

Degradation or denial of service caused **primarily** by volume or intensity exceeding finite resources. Resources include: bandwidth, CPU cycles, memory, storage, database connections, API quotas, thread/process pools, file handles. Maps to `#6 Flooding Attack`.

Reference: §2.2.2 (Global Definitions), R-FLOOD (§2.2.4)

Central Event

In the TLCTC Bow-Tie model: **Loss of Control / System Compromise** — the point at which the attacker achieves unauthorized control over the system's behavior, privileges, data, or trust relationships—sufficient to pursue attack objectives. This central event is positioned before outcomes.

Reference: §1.4.3 (Central Event)

Client-Role Component

A component that **consumes external responses, content, or state** relative to the attacker. The component is in "client role" for the specific interaction being classified.

Reference: §2.2.2 (Global Definitions), R-ROLE (§2.2.4)

Client-Server Relationship

A fundamental principle (Axiom II) stating that every networked software system is based on client-server or caller-called function interaction at various levels. The relationship is contextual: the entity requesting a service is the "client," and the entity providing that service is the "server". Roles can be dynamic and change depending on interaction context, particularly across protection ring boundaries.

Coder

A development role focused on implementation and craftsmanship, responsible for writing functional, efficient code according to established patterns, implementing specific security controls at the code level, and following secure coding practices. Primary responsibility for addressing threat clusters #2, #3, and implementation details of #4, #5, and #7. Contrasts with the Programmer role which focuses on architecture and strategy.

Consequences

In the Bow-Tie model: what results after the central event, including technical and business impact (event chains). Consequences are on the right (effect) side of the Bow-Tie and are recorded as Data Risk Events. Consequences are NOT threat categories.

Reference: §1.4.1 (Bow-Tie Structure), §1.4.4 (What TLCTC Does NOT Classify)

Control

A security measure implemented to mitigate threats, reduce vulnerabilities, or minimize the impact of security incidents. In the TLCTC framework, controls are organized using NIST CSF functions (Identify, Protect, Detect, Respond, Recover) and mapped to specific threat clusters. Controls are categorized as Local Controls (protecting specific systems) or Umbrella Controls (protecting groups of systems).

Control Design Effectiveness

An evaluation of whether a control, as conceived and structured, is theoretically capable of achieving its objective if it operates as intended. Assesses the control's capability to address the identified risk within its specific threat cluster.

Control Failure

A deviation from a control objective or lack of effectiveness. Control failure is control-risk and MUST NOT be treated as a threat category (Axiom V). Risk structure remains: Threat → Event/Incident → Consequences; controls influence likelihood and impact but do not define the threat cluster. Distinguished from the actual risk event itself (Axiom IV).

Reference: Axiom V (§1.2), §1.4.2 (Rule 3)

Control Objective

The specific aim or purpose that a control is intended to achieve, defining what the control should accomplish in terms of risk mitigation for a particular threat cluster. Each control aligns with a single, clear objective.

Control Operational Effectiveness

An evaluation of whether a control is actually working as designed in practice, examining if the control is being executed correctly and consistently over time to meet its objective. May vary depending on the nature of the threat cluster (e.g., controls for Malware #7 may never achieve 100% due to detection latencies).

Credential / Identity Artifact

Any secret, token, key, or session artifact that enables authentication or authorization decisions. Examples include: passwords, PINs, passphrases, API keys, bearer tokens, OAuth/OIDC tokens, SAML assertions, session cookies, session identifiers, private keys, client certificate keys, Kerberos tickets, SSH keys, hardware token seeds/OTPs, biometric templates (when used as authenticators).

Reference: §2.2.2 (Global Definitions), Axiom X (§1.2)

Credential Acquisition

The act of obtaining, capturing, exposing, deriving, or forging a credential/identity artifact. Credential acquisition maps to the **enabling cluster**—the generic vulnerability that made the acquisition possible.

Reference: §2.2.2 (Global Definitions), R-CRED (§2.2.4), Axiom X (§1.2)

Credential Application

The act of presenting, using, replaying, or leveraging a credential to authenticate and operate as an identity. Credential application MUST always map to `#4 Identity Theft`.

Reference: §2.2.2 (Global Definitions), R-CRED (§2.2.4), Axiom X (§1.2)

Credential Forgery

The act of creating a credential without possessing the legitimate secret. If forgery succeeds due to an implementation flaw (e.g., weak signing algorithm, missing validation, predictable tokens), the forgery step maps to `#2` or `#3` per R-ROLE. The subsequent use of the forged credential maps to `#4`.

Reference: §2.2.2 (Global Definitions), R-CRED (§2.2.4)

CVE (Common Vulnerabilities and Exposures)

A standardized identifier for publicly known cybersecurity vulnerabilities. In the TLCTC framework, CVEs are mapped to generic vulnerabilities and their corresponding threat clusters to enable consistent threat classification and control implementation.

Cyber Bow-Tie

The specific application of the Bow-Tie Model to cyber risk management, with the 10 Top Level Cyber Threat Clusters on the cause side, "Loss of Control" or "System Compromise" as the central event, and Data Risk Events and Business Risk Events on the consequence side. Enables structured cyber risk register development and event chain analysis.

Cyber Incident

An actual security breach or system compromise that has occurred, representing the materialization of a cyber risk event where control over IT systems or persons has been lost due to one or more of the 10 Top Level Cyber Threat Clusters.

Cyber Risk

The probability of occurrence of a cyber event in which control over IT systems or persons is lost due to one or more of the 10 Top Level Cyber Threat Clusters, leading (via event chains) to consequential damage (impact). Cyber risks are a subset of operational risks (OpRisk).

Cyber Risk Event

A potential occurrence that could lead to a system breach or compromise. Distinguished from Cyber Incidents (which have already occurred) and Data Risk Events (which are consequences). The central event in the Cyber Bow-Tie model is the **System Risk Event**

(SRE) — "Loss of Control" or "System Compromise".

See also: System Risk Event (SRE)

CWE (Common Weakness Enumeration) *(Industry Term)*

A community-developed list of common software and hardware weakness types maintained by MITRE. CWE categorizes the underlying flaws, bugs, or errors (weaknesses) that enable vulnerabilities to exist — e.g., CWE-89 for SQL Injection weakness, CWE-119 for buffer overflow weakness. In the TLCTC conceptual hierarchy: **Weakness (CWE) → Specific Vulnerability (CVE) → Generic Vulnerability (TLCTC) → Threat Cluster (#1–#10)**. CWE provides granular weakness taxonomy at the code level for developers; TLCTC operates at the strategic level by grouping all resulting vulnerabilities into 10 generic vulnerability categories.

See also: Vulnerability, Weakness, CVE, Generic Vulnerability

Cyber Threat Radar

A standard visualization methodology for communicating threat posture, change over time, and comparative exposure across the 10 TLCTC clusters. The radar is a **communication surface, not a classification device** — every position on it is driven by evidence classified under the Section 4 grammar.

Structure:

- **Spokes (10):** one per cluster (`#1`–`#10`), fixed order; the invariant layout is what makes radars from different organizations, sectors, or snapshots directly comparable.
- **Zones (4, outer → inner):** `Latent` → `Low` → `Medium` → `High`. Radial distance encodes current exposure or impact.
- **Sectors (configurable):** angular subdivisions within each spoke representing responsibility scopes. Default alignment is to Layer 2 responsibility spheres (`@Org`, `@Customers`, `@Vendor` / `@External`). At state/national scope, sectors may encode critical-infrastructure sectors (energy, finance, healthcare, etc.).
- **Bubbles:** individual threat instances, sized by significance.
- **Movement indicators:** ▲ rising / ▼ falling exposure vs. the previous snapshot.

Scales: organizational view (enterprise-internal), sector / line-of-business view, and state-level / national view. The spoke layout is invariant across all three — only the sector axis changes, which is what enables cross-view comparison.

Normative rules (R-RADAR-1…5): spoke assignment MUST use the Section 4 grammar; multi-cluster attack paths MUST be rendered as multiple bubbles or a single bubble plus a separate Layer 3 path; zone placement SHOULD follow a disclosed scoring method; snapshots SHOULD be dated and use the same method for movement indicators to be meaningful; aggregated sector / national radars MUST use identical spoke definitions and SHOULD disclose the combination rule (max / average / weighted).

Reference: §17.1–17.2

See also: Attacker Profile, Tech Enablers Overlay, Responsibility Sphere, Layer 2

Tool: `/tools/radar-tlctc-app.html`

Command Injection *(Industry Term)*

An attack where an attacker injects operating system commands into an application that passes user input to a system shell. In TLCTC: maps to `#2 Exploiting Server` or `#3 Exploiting Client` per R-ROLE — this is an implementation flaw (failure to sanitize input) that creates an unintended data→code transition. The execution of the injected commands constitutes Foreign Executable Content, so the full sequence is typically `#2 → #7` or `#3 → #7`.

See also: Exploiting Server (#2), SQL Injection, Foreign Executable Content (FEC)

Cross-Site Scripting (XSS) *(Industry Term)*

A class of implementation flaw where an application includes untrusted data in web output without proper validation or encoding, allowing attacker-controlled scripts to execute in a victim's browser. In TLCTC:

- **Stored/Reflected XSS** (server fails to encode output): `#2 Exploiting Server` — the flaw is in server-side code.
- **DOM-Based XSS** (client script processes data unsafely): `#3 Exploiting Client` — the flaw is in client-side code.

In both cases, the XSS creates an unintended data→code transition via an implementation flaw.

Reference: V1.9.1 §Definitions (#2, #3), Buzz-Word Refinement

See also: Exploiting Server (#2), Exploiting Client (#3), Implementation Flaw

D

Data Processing Pathways

The four distinct paths that data can follow during an attack, each mapping to specific TLCTC clusters:

1. Data → Data (#1 only) — Pure data manipulation without code execution
2. Data → Function Invocation → Foreign Code Execution (#1 → #7) — Function abuse enabling code execution
3. Data → Exploit Code via Implementation Flaw (#2 or #3) — Unintended data→code transition

4. Data → Foreign Code via Designed Execution Capability (#7 only) — Intended execution capability

Data Risk Event (DRE)

An outcome event describing **Loss of Confidentiality (C)** (data stolen / unauthorized access), **Loss of Integrity (I)** (data modified / unauthorized changes), or **Loss of Availability/Accessibility (A)** (data gone or unreachable, or data present but unusable). Data Risk Events MUST be recorded separately from cluster steps, MUST NOT be used as threat categories, and MUST NOT change the cluster classification of the step that preceded them. Notation: `[DRE: C]`, `[DRE: I]`, `[DRE: A]`, or combinations. When the distinction between Availability and Accessibility is operationally relevant, the general code `A` MAY be refined into **`Av`** (Availability — data gone or unreachable) or **`Ac`** (Accessibility — data present but unusable). Example: ransomware encryption = `[DRE: Ac]`; data deletion = `[DRE: Av]`; distinction unknown = `[DRE: A]`.

Reference: §2.2.2 (Global Definitions), §1.4.2 (Rule 2), §3.5.3

Data vs Code Boundary

A normative classification principle: Domain-specific expressions (e.g., SQL, LDAP, XPath, GraphQL, template syntax, configuration languages) are treated as **data** unless they directly cause **FEC execution** via a general-purpose execution engine.

Reference: §2.2.2 (Global Definitions)

Delta t (Δt) *(V2.0)*

Symbol representing the time interval between threat cluster transitions in an attack sequence. See Attack Velocity.

Designed Execution Capability

The environment's **intended** capability to load, interpret, or execute program content. This is the generic vulnerability exploited by `#7 Malware`. Examples: OS loaders, script interpreters, macro engines, browser JS engines, module loaders, container/virtualization runtimes.

Reference: §2.2.2 (Global Definitions), §2.1 (#7 Definition)

Detection Coverage Score (DCS) *(V2.0)*

A strategic Key Performance Indicator (KPI) for measuring security effectiveness derived from Attack Velocity. Formula: `DCS = (Mean Time to Detect) / (Attack Velocity Δt)`.

- **Score < 1.0:** Organization is faster than the adversary (Winning)
- **Score > 1.0:** Adversary completes the step before detection (Losing)

Example: If a ransomware group moves from #4 to #1 in 10 minutes and your SIEM alerts in 15 minutes, DCS = 15/10 = 1.5, indicating systematic blindness requiring automation rather than analyst intervention.

Developer's View

A perspective included in each TLCTC threat cluster definition that provides guidance on secure development practices specific to preventing that cluster. Encompasses both Programmer (architectural) and Coder (implementation) responsibilities.

Domain

A set of assets governed by a coherent control regime (policies, monitoring, enforcement, and accountability). Domains may be technical, organizational, or socio-technical. Examples: cyber/IT domain, physical security domain, human decision domain, vendor development domain, cloud provider control-plane domain.

Reference: §2.2.2 (Global Definitions), §5.1.1

Domain Boundary

A point where responsibility spheres or control regimes change. Crossing a domain boundary means the attack moves from one set of applicable controls to a different set.

Reference: §2.2.2 (Global Definitions), §5.1.3

Domain Boundary Operator (||) *(V2.0)*

Notation: `||[context][@Source→@Target]||`. Used to explicitly mark where an attack path crosses responsibility spheres. The operator SHOULD accompany bridge cluster steps (`#8`, `#9`, `#10`) and MAY be used with any step that crosses a domain boundary. The context describes the transition type (e.g., [dev], [idp], [update]) and the arrow shows the direction of trust crossing. The boundary test: "If removing the third-party trust link would stop the step from succeeding, #10 belongs there". Enables precise mapping of responsibility shifts and supply chain attack analysis.

Reference: §2.2.2 (Global Definitions), §3.3 (Domain Boundary Operator), §5.3

DAST (Dynamic Application Security Testing) *(Industry Term)*

A testing methodology that analyzes a running application by simulating attacks against it to identify security vulnerabilities. In TLCTC: DAST is a **preventive control** (IDENTIFY/PROTECT) primarily targeting `#2 Exploiting Server` and `#3 Exploiting Client` by discovering implementation flaws in deployed applications.

See also: SAST, Control, Exploiting Server (#2), Exploiting Client (#3)

DDoS (Distributed Denial of Service) *(Industry Term)*

An attack where multiple compromised systems (typically a botnet) simultaneously flood a target with traffic or requests. In TLCTC: maps to `#6 Flooding Attack` — the primary mechanism is volume or intensity exceeding finite capacity. The "distributed" aspect describes the attack infrastructure, not the generic vulnerability. Sub-types include SYN Flood, UDP Flood, HTTP Flood, ICMP Flooding, NTP/DNS Amplification, and Slowloris. The underlying botnet is the result of prior `#7 Malware` infections.

Reference: V1.9.1 Buzz-Word Refinement (#6)

See also: Flooding Attack (#6), Botnet, SYN Flood, Amplification Attack

Defense-in-Depth *(Industry Term)*

A security strategy employing multiple layers of controls so that if one layer fails, another provides protection. In TLCTC: defense-in-depth means implementing controls at multiple points along potential attack paths — both Local Controls (specific systems) and Umbrella Controls (groups of systems) — and across NIST CSF functions (Identify, Protect, Detect, Respond, Recover) for each relevant threat cluster. The interplay between different threat clusters (e.g., `#9` potentially circumventing `#4` controls) necessitates a holistic, defense-in-depth approach.

Reference: V1.9.1 §The Anatomy of Risk

See also: Local Controls, Umbrella Controls, Control

Directory Traversal *(Industry Term)*

An attack where an attacker manipulates file path references (e.g., using `../` sequences) to access files or directories outside the intended scope. In TLCTC: maps to `#2 Exploiting Server` — an implementation flaw in how the server-side code handles file path input, enabling unauthorized access. The flaw is in server-side source code (failure to validate/sanitize path references).

Reference: V1.9.1 Buzz-Word Refinement (#2)

See also: Exploiting Server (#2), Implementation Flaw

DNS Spoofing *(Industry Term)*

A technique where an attacker corrupts DNS resolution to redirect traffic to attacker-controlled infrastructure. In TLCTC: the DNS spoofing itself is `#1 Abuse of Functions` (abusing legitimate DNS protocol functionality without exploiting a code flaw). DNS spoofing typically leads to a Man in the Middle position, making the full sequence `#1 → #5`. Note: if the attacker exploits an implementation flaw in a DNS server (e.g., cache poisoning via a code bug), the initial step maps to `#2 Exploiting Server` instead.

Reference: V1.9.1 Buzz-Word Refinement (#1)

See also: ARP Spoofing, BGP Hijacking, Man in the Middle (#5)

Domain Squatting *(Industry Term)*

Registering domain names similar to legitimate ones (typosquatting, homograph attacks) to deceive users into visiting attacker-controlled websites. In TLCTC: the domain registration is infrastructure setup (not itself a cluster step). When used to harvest credentials via fake login pages, the attack maps to `#9 Social Engineering` (luring the user) → `#4 Identity Theft` (using the stolen credentials). When used to deliver malware, the sequence may be `#9 → #3` or `#9 → #7`.

Reference: V1.9.1 Buzz-Word Refinement (#4)

See also: Phishing, Social Engineering (#9), Typosquatting

DORA (Digital Operational Resilience Act) *(Industry Term)*

An EU regulation establishing ICT risk management requirements for financial entities. Like NIS2, DORA emphasizes incident reporting but lacks a unified threat categorization system. The TLCTC framework addresses this gap by providing a standardized taxonomy that supports compliance with DORA's requirements for threat classification and incident reporting. DORA requirements can be mapped as Regulatory Trigger Points in the TLCTC event chain model.

See also: NIS2, Regulatory Trigger Point, E_n Event Notation

Drive-By Download *(Industry Term)*

An attack where malware is automatically downloaded and potentially executed when a user visits a compromised or malicious website, typically by exploiting a browser or plugin vulnerability. In TLCTC: maps to `#3 Exploiting Client` (browser vulnerability exploited) → `#7 Malware` (payload execution). If the user was lured to the site via social engineering, the full sequence is `#9 → #3 → #7`.

Reference: V1.9.1 §F (Edge-Case Resolution)

See also: Exploiting Client (#3), Malware (#7), Watering Hole Attack

Dual-Use Tool

A legitimate administrative utility that can be used for both legitimate administrative purposes and malicious activities when invoked by an attacker. Examples include PowerShell, PsExec, WMI, and remote administration tools. In TLCTC: invocation/abuse of the tool may be `#1` (if no implementation flaw is exploited), while the actual execution of attacker-controlled FEC through that tool is `#7`, resulting in a `#1 → #7` sequence.

Reference: §2.2.4 (R-EXEC, LOLBAS Clarification)

E

Edge (in attack path)

A transition between two adjacent Attack Steps, represented by the sequence operator `→`. Δt (Attack Velocity) is an edge property.

Reference: §3.1 (Sequence Operator), §4.0.2

Estimated Δt

An approximate Δt value derived from partial evidence when precise timestamps are unavailable. Notation: `Δt~15m`.

Reference: §4.0.3, §4.2.3

E_n Event Notation (Regulatory) *(V2.0)*

A numbered event sequence notation used to map attack chains to regulatory compliance triggers:

- **E1:** System Compromise / Loss of Control (the central Bow-Tie event)
- **E2:** Data Risk Event (e.g., PII exposure — GDPR trigger)
- **E3a, E3b, ...:** Compliance violation events (e.g., GDPR breach notification, NIS2 incident report)

The subscript (a, b, etc.) distinguishes parallel regulatory branches triggered by the same upstream event. Different regulations trigger at different points: GDPR Art. 33 triggers at E2 (Data Risk Event involving PII), while NIS2 Art. 23 triggers at E1 (Significant Incident). See also: Event Chain Length, RS Container.

Event Chain Length *(V2.0)*

The number of causal events between the initial incident (E1) and a regulatory trigger point (E3x). Shorter chains mean compliance clocks start sooner. Example: NIS2 path (E1→E3b) = 2 events with 24h early warning requirement; GDPR path (E1→E2→E3a) = 3 events with 72h notification timeline. Understanding chain length helps CISOs structure incident response playbooks to meet multiple regulatory timelines from a single incident.

Exploit Code

Foreign code that targets specific vulnerabilities to modify software behavior, creating an UNINTENDED data→code transition. Used in #2 Exploiting Server and #3 Exploiting Client. Distinguished from Malware Code which operates within expected execution paths.

Exploiting Client (#3)

A threat cluster where an attacker targets and leverages flaws originating directly within the source code implementation of any software acting in a client role (requesting/processing data from a server or resource). These vulnerabilities allow manipulation of client behavior or unauthorized access using Exploit Code, often when the client interacts with malicious content. The generic vulnerability is the presence of exploitable flaws within client-side source code stemming from insecure coding practices.

Exploiting Server (#2)

A threat cluster where an attacker targets and leverages flaws originating directly within the server-side application's source code implementation. These vulnerabilities allow manipulation of server behavior or unauthorized access using Exploit Code, forcing a data→code transition where exploit code executes as new, foreign code in the server context. The generic vulnerability is the presence of exploitable flaws within server-side source code implementation stemming from insecure coding practices.

EDR (Endpoint Detection and Response) *(Industry Term)*

A category of security tools that monitor endpoint devices for suspicious activity and provide automated response capabilities. In the TLCTC velocity model, EDR is a critical control for the **Fast Velocity Class** (minutes) where automated containment is necessary because human analyst response times are insufficient. EDR is particularly relevant for controlling `#3 Exploiting Client` and `#7 Malware` at the endpoint level.

See also: Fast Velocity Class, SIEM, SOAR

Event Chain

A causal sequence where one outcome event triggers subsequent events, following the consequence chain **SRE → DRE → BRE***. The chain cascades from the System Risk Event (central event) through Data Risk Events to one or more Business Risk Events. BREs may themselves chain ($BRE_1 \rightarrow BRE_2 \rightarrow ... \rightarrow BRE_n$), with each transition having its own Δt representing a detection and intervention window where all six NIST CSF functions apply. Example: System Compromise (SRE) → Data Breach involving PII (DRE [C]) → GDPR notification obligation (BRE_1) + NIS2 incident report (BRE_2) → Regulatory fine (BRE_3). Understanding event chains is critical for designing Respond/Recover controls and regulatory compliance workflows.

Reference: §1.4.3.1 (The Consequence Chain), V1.9.1 §Data Risk Event Types, §Clarification on Central Event Position

See also: System Risk Event (SRE), Data Risk Event (DRE), Business Risk Event (BRE), Business Impact (BI), E_n Event Notation, RS Container, Propagated PR

Evil Maid Attack *(Industry Term)*

A physical attack where an adversary with brief unsupervised access to a device (e.g., left in a hotel room) tampers with it — installing hardware keyloggers, modifying boot loaders, or extracting encryption keys. In TLCTC: the physical access step maps to `#8 Physical Attack`. Subsequent technical steps map to their respective clusters: installing a keylogger = `#8 → #7`, extracting credentials from the device = `#8 → #4`.

Reference: V1.9.1 Buzz-Word Refinement (#8)

See also: Physical Attack (#8), USB Baiting

F

Fast Velocity Class *(V2.0)*

A velocity classification where attack progression occurs within minutes. Typical threat clusters: #3 (Exploiting Client), #2 (Exploiting Server). Control strategy requires automated containment and EDR blocking, as human analyst response times are insufficient. Example: Drive-by downloads, browser exploits, RCE exploitation.

Flooding Attack (#6)

A threat cluster where an attacker intentionally overwhelms system resources or exceeds capacity limits through a high volume of requests, data, or operations, leading to disruption, degradation, or denial of service for legitimate users. The generic vulnerability is the finite capacity limitations inherent in any system component (network bandwidth, CPU, memory, storage, database limits, application quotas, API rate limits, process/thread pools). Outcome is typically Loss of Availability.

Fileless Execution / Fileless Malware *(Industry Term)*

An attack technique where malicious code executes entirely in memory without writing traditional files to disk, often using legitimate system tools (PowerShell, WMI, .NET reflection) as execution vehicles. In TLCTC: fileless execution still maps to `#7 Malware` — the FEC definition explicitly includes in-memory execution, interpreted code, and reflective loading with no "on-disk" requirement. The invocation of the legitimate tool to enable fileless execution may be `#1 Abuse of Functions`, making the typical sequence `#1 → #7`.

See also: Foreign Executable Content (FEC), Living Off the Land / LOLBAS, Dual-Use Tool

Foreign Executable Content (FEC)

Attacker-controlled (or otherwise untrusted) program text or bytes that are **interpreted, loaded, or executed** by a **general-purpose execution engine** in the target environment. Includes attacker-controlled commands fed into interpreters. FEC execution includes in-memory (fileless) execution, interpreted code, macro execution, and reflective loading—no "on-disk" requirement exists.

Reference: §2.2.2 (Global Definitions)

Framework Layer *(V2.0)*

The static, universal component of the TLCTC JSON architecture containing threat cluster definitions, generic vulnerabilities, data risk events, bow-tie model principles, attack path notation rules, and framework axioms. Defined in `tlctc-framework.json`. Changes rarely (only during framework evolution) and serves as the common language that all organizations reference. Contrasts with the Intelligence Layer which contains dynamic, incident-specific data.

G

Generic Vulnerability

The single root-level vulnerability category defining a cluster — the **strategic-level attack surface** towards a specific class of threats. For every generic vulnerability, there is exactly one TLCTC cluster (per Axiom VI). The generic vulnerability is what the threat cluster targets: it is the exposed surface that enables the attack vector and through which the attack path proceeds. Generic vulnerabilities are stable across technologies and implementations, persisting regardless of specific IT system types, software implementations, or evolving attack techniques. All specific vulnerabilities (CVEs) are instances of a generic vulnerability; all generic vulnerabilities map to exactly one threat cluster.

The 10 generic vulnerabilities (attack surfaces) are: functional scope/trust (#1), server-side implementation flaws (#2), client-side implementation flaws (#3), identity-artifact binding (#4), lack of end-to-end communication protection (#5), finite capacity limitations (#6), designed execution capability (#7), physical accessibility (#8), human psychological factors (#9), and third-party trust dependencies (#10).

Reference: §2.2.2 (Global Definitions), §2.2.7 (Step 2), Axiom VI (§1.2)

See also: Vulnerability, Attack Vector, Attack Surface Analysis

GOVERN (GV)

The governance function in NIST CSF 2.0, operating at a strategic level to establish the overall cybersecurity risk management framework. GOVERN controls are "assurance controls" that create structure and context for other functions, including setting risk

appetite, defining roles and responsibilities, and establishing policies. While GOVERN oversees threat categorization in the risk register, GV controls themselves don't directly counter specific threats but provide the strategic foundation for comprehensive risk management.

H

HTTP Flood *(Industry Term)*

An application-layer denial of service attack that overwhelms a web server with seemingly legitimate HTTP requests. In TLCTC: maps to `#6 Flooding Attack` — the primary mechanism is volume exceeding finite capacity at the application layer. Distinguished from implementation-flaw-based DoS (which maps to `#2` or `#3` per R-FLOOD).

See also: Flooding Attack (#6), DDoS, Slowloris, SYN Flood

I

ICMP Flooding *(Industry Term)*

A network-layer denial of service attack that overwhelms a target with ICMP echo request (ping) packets. In TLCTC: maps to `#6 Flooding Attack` — volume exceeding finite network capacity.

See also: Flooding Attack (#6), DDoS

Identity Theft (#4)

A threat cluster where an attacker targets weaknesses in identity and access management processes or credential protection mechanisms to illegitimately misuse authentication credentials (passwords, tokens, keys, session identifiers, biometrics) to impersonate a legitimate identity (human or technical). The generic vulnerability is weak Identity Management Processes and/or inadequate credential protection mechanisms throughout the identity lifecycle.

Critical distinction: Credentials have dual operational nature:

- **Acquisition/Exposure:** When credentials are obtained through another cluster (e.g., #2 SQL injection, #5 MitM, #7 keylogger, #9 Phishing), map to the enabling cluster (Loss of Confidentiality consequence)
- **Use/Application:** The subsequent *use* of acquired credentials—regardless of acquisition method—always maps to #4 Identity Theft (Loss of Control / system compromise event)

Non-Overlap Rule: Credential acquisition maps to the enabling threat cluster; credential use always maps to #4.

Implementation Defect (Availability Context)

A flaw in code logic, parsing, memory handling, or resource handling that causes crash, hang, or degradation when triggered—**without** requiring volume/intensity to exceed normal capacity. Includes algorithmic complexity weaknesses (e.g., ReDoS). Maps to `#2` or `#3` per R-ROLE, not `#6`.

Reference: §2.2.2 (Global Definitions), R-FLOOD (§2.2.4)

Implementation Flaw

A defect in source code implementation (logic, parsing, memory handling, resource handling) enabling unintended behavior when triggered. Implementation flaws are exploited by `#2 Exploiting Server` (server-role) or `#3 Exploiting Client` (client-role).

Reference: §2.2.2 (Global Definitions), §2.1 (#2 and #3 Definitions)

Intelligence Layer *(V2.0)*

The dynamic component of the TLCTC JSON architecture containing specific attack instances, software versions & CVEs, timeline & actor TTPs, domain boundaries, and impact assessments. Changes constantly as new incidents occur. Each incident is documented in its own JSON file following the attack sequence schema format: `[incident-id]-attack-path.json`. Enables worldwide threat intelligence sharing while maintaining consistency through reference to the static Framework Layer.

Internal Cluster

A TLCTC cluster that operates primarily **within the software domain's** attack surfaces, without inherently crossing to a different responsibility sphere. Internal clusters are: `#1` through `#7`.

Reference: §2.2.2 (Global Definitions), §5.1.5, §5.2 (Topology Classification)

Insecure Deserialization *(Industry Term)*

A class of implementation flaw where an application deserializes untrusted data without proper validation, potentially allowing arbitrary code execution or object manipulation. In TLCTC: maps to `#2 Exploiting Server` or `#3 Exploiting Client` depending on the role of the vulnerable component per R-ROLE. The deserialization flaw creates an unintended data→code transition via an implementation defect.

Reference: V1.9.1 Buzz-Word Refinement (#2, #3)

See also: Exploiting Server (#2), Exploiting Client (#3), Implementation Flaw

Intra-System Boundary Operator (|...|) *(V2.1)*

Notation: `|[type][@from→@to]|`. Used to annotate boundary crossings **within a single host or system**, such as sandbox escapes, privilege escalations, process boundary violations, and VM escapes. Uses single pipe delimiters to distinguish from the inter-sphere Domain Boundary Operator (`||...||`). Defined boundary types: `sandbox`, `privilege`, `process`, `hypervisor`. The `memory` type is reserved and MUST NOT be used (R-INTRA-9). Intra-system boundaries are observability annotations and never change cluster classification (R-INTRA-7). Example: `#3 |[sandbox][@renderer→@os]|` — browser exploit escaping renderer sandbox.

Reference: §3.3.6 (Intra-System Boundary Operator)

J

JSON Architecture *(V2.0)*

The standardized data structure for threat intelligence sharing in TLCTC V2.0, consisting of four complementary JSON files:

1. **tlctc-framework.json:** Core framework definitions (universal, rarely updated)
2. **tlctc-responsibility-spheres.json:** Domain boundary definitions (customizable, occasionally updated)
3. **tlctc-attack-sequence-schema.json:** Validation schema for attack instances (universal, rarely updated)
4. **[incident]-attack-path.json:** Specific attack instances (per-incident, constantly updated)

This architecture separates universal framework definitions from specific attack instances, enabling machine-readable, validated, consistent threat intelligence exchange worldwide.

K

KCI (Key Control Indicator)

A metric that measures the operational performance of security controls, verifying that intended actions are taken at the appropriate frequency. KCIs provide insights on the ability to apply correct controls correctly, highlighting process weaknesses and tool effectiveness. Example: "Frequency of patch deployments per day" or "Scan verification of implemented patches" for a control requiring critical systems to be patched within 24 hours.

KPI (Key Performance Indicator)

A measurable value demonstrating the outcome and performance of security processes in reaching security objectives. KPIs must be time-based and reflect effectiveness over time. Example: "Average time to restore critical services to full operation within a 4-hour window." In TLCTC V2.0, the Detection Coverage Score (DCS) is introduced as a strategic KPI.

KRI (Key Risk Indicator)

A leading indicator demonstrating the potential for a future cyber threat. KRIs show possible risks before a threat occurs and must be observed in a meaningful timeframe. Example: "Number of unpatched critical vulnerabilities older than 7 days" indicates how processes handle critical vulnerabilities, helping identify, understand, and prioritize security efforts to prevent incidents.

Keylogger *(Industry Term)*

Malware (software) or a hardware device that records keystrokes to capture sensitive information such as passwords, credit card numbers, or other data. In TLCTC: a software keylogger maps to `#7 Malware` (foreign code executing via the environment's designed execution capability). A hardware keylogger maps to `#8 Physical Attack` (physical device installation). Captured credentials that are subsequently used map to `#4 Identity Theft`. Typical sequence: `#7 → #4` (software keylogger) or `#8 → #4` (hardware keylogger).

Reference: V1.9.1 Buzz-Word Refinement (#7, #8)

See also: Malware (#7), Physical Attack (#8), Identity Theft (#4)

Kill Chain *(Industry Term)*

A model describing the stages of a cyber attack from reconnaissance through exploitation to objective completion (originally Lockheed Martin Cyber Kill Chain). In TLCTC: the kill chain concept is complementary to — but not a classification criterion for — the TLCTC framework. TLCTC classifies each step by the generic vulnerability exploited (cause-oriented), while kill chain models describe the phase of the attack (process-oriented). Attack paths in TLCTC notation (`#X → #Y → #Z`) naturally encode the kill chain progression without requiring a separate phase model.

Reference: V1.9.1 §F (Oversimplification)

See also: Attack Path, Sequence, MITRE ATT&CK

KxI Framework

The integrated hierarchical framework of Key Risk Indicators (KRIs), Key Control Indicators (KCIs), and Key Performance Indicators (KPIs), providing a practical mechanism to operationalize the 10 Top Level Cyber Threat Clusters. Each threat cluster has associated KRI, KCI, and KPI values for managing cyber risk and measuring overall cybersecurity program performance. Base Level Indicators (BxIs) represent the lowest operational level that still makes sense.

L

Latent/Slow Velocity Class *(V2.0)*

A velocity classification where attack progression occurs over days to months. Typical threat clusters: #10 (Supply Chain), #7 (APT Implants). Control strategy focuses on log retention and threat hunting, as detection windows are extended. Example: Supply chain compromises with long dwell times, persistent APT campaigns prioritizing stealth over speed.

Living Off the Land / LOLBAS (Living Off the Land Binaries and Scripts)

An attack technique using only software functions and binaries already present on a (potentially compromised) system, invoked with legitimate inputs/parameters, without introducing foreign code initially. Legitimate system binaries are used to execute attacker-controlled content. In TLCTC: the invocation of the legitimate binary may be `#1` (if no implementation flaw is exploited), while the execution of attacker-controlled content through it is `#7`. The sequence `#1 → #7` applies. Examples: Using cmd.exe, PowerShell, WMI, or Task Scheduler to execute attacker-controlled scripts.

Reference: §2.2.4 (R-EXEC, LOLBAS Clarification)

Lateral Movement *(Industry Term)*

The techniques an attacker uses to progressively move through a network after initial compromise, seeking higher-value targets and expanded access. In TLCTC: lateral movement is not a single cluster — it is an attack path composed of multiple sequential steps. Typical lateral movement sequences include: `#4 → #1` (using stolen credentials to access another system's functions), `#4 → #7` (deploying malware on additional systems using stolen credentials), or `#1 → #7` (abusing legitimate remote tools to execute code on other systems). Each step in lateral movement maps to its own cluster based on the generic vulnerability exploited.

Reference: V1.9.1 §Bridging Strategy and Operations, §E (Real World Examples)

See also: Attack Path, Sequence, Identity Theft (#4)

Local Controls

Security measures implemented directly on or for specific IT systems. Distinguished from Umbrella Controls which protect groups of systems. Local controls are essential for systems that cannot be fully protected by umbrella controls (e.g., exposed systems, "Patient Zero" entry points).

Loss of Availability (LoA)

A Data Risk Event outcome where data or resources are gone or unreachable — the resource no longer exists or cannot be technically accessed by the infrastructure. From the attacker's perspective: "Data gone / system down". Examples: deletion, storage failure, system offline, network unreachable. DRE notation: `Av` (refined) or `A` (general). Not to be confused with Loss of Accessibility (data present but unusable, e.g., ransomware encryption → `Ac`).

Loss of Accessibility (LoAc)

A Data Risk Event outcome where data or resources exist and can be reached by the infrastructure, but cannot be used for their intended purpose by authorized processes. From the attacker's perspective: "Data locked / unusable". Examples: ransomware encryption, data corruption, permission lockout. DRE notation: `Ac` (refined) or `A` (general). Not to be confused with Loss of Availability (data gone or unreachable → `Av`).

Loss of Confidentiality (LoC)

A Data Risk Event outcome where an attacker gains unauthorized access to data. From the attacker's perspective: "Data stolen". This describes what bad thing happens, not how it happens. Various threat clusters can lead to this outcome depending on the mechanism used (e.g., #2 via SQL injection, #5 via MitM eavesdropping).

> **Disambiguation:** In TLCTC, **LoC** always means Loss of Confidentiality — a *consequence*-side Data Risk Event. "Loss of Control" is a distinct concept: the Bow-Tie *central event*, always abbreviated **SRE** (System Risk Event), never "LoC". See **System Risk Event (SRE)**.

Loss of Control / System Compromise

The central event in the Cyber Bow-Tie model, abbreviated **SRE** (System Risk Event), representing the point at which the attacker achieves unauthorized control over a system's behavior, privileges, data, or trust relationships. This serves as the pivot point between threat realization (cause) and potential consequences (effect). The SRE is the first event in the consequence chain: **SRE → DRE → BRE***. Some attacks may have delayed data risk events (creating a detection window), while others lead to immediate data risk events. Examples: A server exploit (#2) enabling remote code execution leading to malware (#7) represents loss of control before any data breach occurs. In contrast, successful SQL injection (#2) can immediately result in Loss of Confidentiality.

Reference: §1.4.3 (Central Event), §1.4.3.1 (The Consequence Chain)

See also: System Risk Event (SRE), Data Risk Event (DRE), Business Risk Event (BRE)

Loss of Integrity (LoI)

A Data Risk Event outcome where an attacker successfully makes unauthorized changes to data. From the attacker's perspective: "Data modified". This refined definition focuses on the outcome rather than the mechanism (e.g., "tampering" is a mechanism that produces the LoI outcome).

M

Malicious Code

Code written with harmful intent, distinguished in TLCTC between:

- **Exploit Code:** Targets specific vulnerabilities to modify software behavior (#2/#3)
- **Malware Code:** Operates within expected execution paths for harmful purposes (#7)
- **Malware Software:** Comprehensive suite of tools (foreign code) that may incorporate multiple techniques, including exploit capabilities

Malvertising *(Industry Term)*

The use of online advertising to distribute malware or redirect users to malicious websites. In TLCTC: malvertising is a **delivery vector**, not a distinct threat category. It can deploy either exploits or malware depending on the attacker's strategy:

- If it exploits a browser/plugin vulnerability: `#3 Exploiting Client` → `#7 Malware`
- If it delivers malware directly (e.g., fake download): `#9 Social Engineering` → `#7 Malware`
- If it redirects to a credential-harvesting site: `#9 Social Engineering` → `#4 Identity Theft`

Reference: V1.9.1 Clarifications, Buzz-Word Refinement (#3)

See also: Drive-By Download, Watering Hole Attack, Phishing

Malware (#7)

A threat cluster where an attacker abuses the inherent ability of a software environment to execute foreign executable content, including inherently malicious Malware Code or legitimate tools/scripts when they execute attacker-controlled or otherwise foreign code ("dual-use"). The generic vulnerability is the software environment's designed capability to

execute potentially untrusted 'foreign' code, scripts, or binaries. Distinguished from #2/#3 which use Exploit Code targeting implementation flaws, and from #1 which manipulates existing functions without executing foreign code/scripts/binaries.

Man in the Middle (#5)

A threat cluster where an attacker intercepts, eavesdrops on, modifies, or relays communication between two parties without their knowledge or consent, by exploiting a privileged position on the communication path. The generic vulnerability is the lack of sufficient control, integrity protection, or confidentiality over the communication channel/path, including the implicit trust placed in local networks and intermediate network infrastructure in standard IP networking. Position might be gained locally (shared Wi-Fi) or by leveraging control over existing network intermediaries.

Medium Velocity Class *(V2.0)*

A velocity classification where attack progression occurs within hours. Typical threat clusters: #9 (Phishing), #4 (Manual Credential Abuse). Control strategy utilizes SIEM alerting and analyst triage, as human response times are sufficient for detection and initial response. Example: Phishing campaigns followed by manual reconnaissance and lateral movement.

Mitigating Controls

In the Bow-Tie model: barriers on the right (effect) side that detect, contain, reduce impact, or enable recovery after the central event occurs. Corresponds to NIST CSF functions: RESPOND, RECOVER.

Reference: §1.4.1 (Bow-Tie Structure)

MITRE ATT&CK

A globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. In the TLCTC framework, MITRE techniques are considered operational-level detail that map to the strategic-level threat clusters. TLCTC V2.0 proposes enhancement through adding cluster mappings and typical velocity attributes to techniques.

MFA Bombing / MFA Fatigue *(Industry Term)*

An authentication bypass technique where an attacker, having obtained valid credentials, repeatedly triggers MFA push notifications to overwhelm the user into accidentally approving one. In TLCTC, this is a four-step sequence: `#4 → #1 → #9 → #4`:

1. **#4 Identity Theft:** Attacker uses stolen userID/password
2. **#1 Abuse of Functions:** Repeatedly triggering legitimate MFA challenge requests (abusing intended functionality, not a code flaw)

3. **#9 Social Engineering:** Psychologically manipulating the user through fatigue/annoyance into approving a request
4. **#4 Identity Theft:** Successfully obtaining and using the MFA token to complete authentication

This example demonstrates how TLCTC decomposes a single "buzzword attack" into its constituent generic vulnerabilities.

Reference: V1.9.1 §Attack Path Notation (MFA Bombing Example)

See also: Identity Theft (#4), Abuse of Functions (#1), Social Engineering (#9)

MitM Position

A controlled point on a communication path that enables interception, observation, modification, injection, replay, or protocol downgrade/stripping. The attacker has achieved the ability to influence communication between two endpoints.

Reference: §2.2.2 (Global Definitions), R-MITM (§2.2.4)

N

NIST CSF (Cybersecurity Framework)

The National Institute of Standards and Technology Cybersecurity Framework providing guidelines for managing cybersecurity risk. The TLCTC framework integrates with NIST CSF by mapping the 10 threat clusters to the five core functions (Identify, Protect, Detect, Respond, Recover) and the GOVERN function in CSF 2.0. TLCTC proposes formal adoption of the 10 clusters as the standard taxonomy for Threat Identification in the ID.RA (Risk Assessment) category.

NIS2 (Network and Information Security Directive 2) *(Industry Term)*

The EU directive establishing cybersecurity risk management and incident reporting obligations for essential and important entities. In TLCTC: NIS2 is an **incident-triggered regulation** — its reporting obligations activate at E1 (Significant Incident / System Compromise) regardless of whether personal data is involved. This contrasts with GDPR, which is **data-triggered** (activates at E2 when PII is affected). NIS2 Art. 23 requires a 24-hour early warning, creating a shorter event chain length (E1→E3b = 2 events) compared to GDPR's 72-hour notification timeline (E1→E2→E3a = 3 events).

Reference: V1.9.1 §Cyber Threat Radars

See also: DORA, Regulatory Trigger Point, Event Chain Length, E_n Event Notation

Normative Keywords

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in the TLCTC specification are interpreted as described in RFC 2119 / RFC 8174. When these keywords appear in lowercase, they carry their ordinary English meaning.

Reference: §2.2

Notation Systems

The TLCTC framework employs two complementary notation systems:

- **Strategic Notation:** Human-readable format using `#X` (e.g., #1, #10) for executive communication and risk assessment
- **Operational Notation:** Machine-readable format using `TLCTC-XX.YY` (e.g., TLCTC-01.00) for tool integration, automation, and SIEM

Both notations remain fully compatible and can be used interchangeably based on context.

O

Observed Δt

A Δt value computed from two concrete time observations.

Reference: §4.0.3

Operational Layer

The detailed implementation level where security controls are implemented, monitored, and adjusted. Includes specific vulnerability management, threat intelligence (using frameworks like MITRE ATT&CK), TTP mapping, attack path analysis, vulnerability management (CVE reports), incident response, security testing, and monitoring. Uses the machine-first naming convention `TLCTC-XX.YY`, where XX is the two-digit cluster number (01–10) and YY is the two-digit sub-cluster number (00–99), for tool integration, SIEM rules, automation, threat intelligence exchange, and detailed documentation.

Reference: §2.2.1 (Two-Layer Naming Convention)

Operational Risk (OpRisk) *(Industry Term)*

The broader category of risks arising from inadequate or failed internal processes, people, and systems, or from external events. In TLCTC: cyber risks are explicitly defined as a **subset** of operational risks. While cyber risk management focuses on threats from unauthorized or unknown entities (covered by the 10 TLCTC clusters), comprehensive risk management must also consider traditional IT risks (e.g., "software failure", "error in use",

"abuse of rights" by authorized actors), compliance risks, and third-party risks. Actions of authorized actors should be managed under separate OpRisk categories unless they attempt to breach authorization boundaries, which then falls within cyber risk scope.

Reference: V1.9.1 §Introduction

See also: Cyber Risk, Business Risk Event

Operational Security Layer

The layer of TLCTC that addresses specific vulnerabilities, techniques, and procedures. Contains concrete vulnerabilities (CVEs), operational techniques (TTPs), and indicators used in detection, response, and engineering. Corresponds to `TLCTC-XX.YY` where YY ≠ 00.

Reference: Axiom VIII (§1.2), §2.2.1

OAuth Attack *(Industry Term — Decomposition Required)*

A commonly used but imprecise term that conflates multiple distinct attack mechanisms targeting OAuth implementations. In TLCTC, the framework's precision requirement reveals that "OAuth attack" must be decomposed into its specific mechanism:

- **Misconfigured redirect_uri:** `#1 Abuse of Functions` (abusing legitimate OAuth functionality via configuration)
- **XSS stealing OAuth tokens:** `#3 Exploiting Client` → `#4 Identity Theft`
- **Phishing OAuth consent:** `#9 Social Engineering` → `#4 Identity Theft`
- **Authorization code injection:** `#1` (if design abuse) or `#2` (if server-side validation flaw)

This decomposition is not creating ambiguity — it is exposing the security industry's imprecise terminology and enabling more targeted control implementation.

Reference: V1.9.1 §F (Industry Term Decomposition)

See also: Identity Theft (#4), Abuse of Functions (#1), Exploiting Client (#3)

OWASP (Open Worldwide Application Security Project) *(Industry Term)*

A nonprofit foundation providing freely available resources for web application security, including the OWASP Top 10 list of critical web application security risks. In TLCTC: OWASP risks and testing methodologies are considered operational-level detail. The OWASP Top 10 categories map to TLCTC clusters — e.g., "Injection" maps to `#2`/`#3`, "Broken Authentication" maps to `#4`, "Security Misconfiguration" maps to `#1`. TLCTC notes that OWASP (like STRIDE) is "per se incomplete" and recommends always starting threat assessment with the 10 TLCTC clusters.

Reference: V1.9.1 §Operational Layer

See also: STRIDE, Sub-Threat, Operational Layer

P

Parallel Operator (+)

Notation that denotes **concurrent** (or effectively concurrent) steps—actions that occur in the same phase where their ordering is not meaningful. Parallel steps MUST be grouped using parentheses: `(#X + #Y)`. Each element inside a parallel group is a separate Attack Step mapped to exactly one cluster.

Reference: §2.2.2 (Global Definitions), §3.2 (Parallel Operator)

Parallel Steps

Two or more clusters occurring simultaneously or in tight coordination within the same attack phase. Use when distinct generic vulnerabilities are exploited concurrently rather than sequentially.

Reference: §2.2.2 (Global Definitions), §3.2 (Parallel Operator)

Pass-the-Hash / Pass-the-Ticket *(Industry Term)*

Attack techniques where an attacker uses captured NTLM hashes (Pass-the-Hash) or Kerberos tickets (Pass-the-Ticket) to authenticate as a legitimate user without knowing the actual password. In TLCTC: the **acquisition** of the hash/ticket maps to the enabling cluster (e.g., `#7` if extracted by malware, `#1` if via lsass dump using a legitimate tool). The **use** of the hash/ticket to authenticate always maps to `#4 Identity Theft` per R-CRED.

Reference: V1.9.1 Buzz-Word Refinement (#4)

See also: Identity Theft (#4), Credential Acquisition, Credential Application

Password Spraying *(Industry Term)*

An attack that tries a small number of commonly used passwords against many accounts simultaneously, avoiding account lockout thresholds. In TLCTC: maps to `#4 Identity Theft` — the attacker is attempting to derive valid credentials to impersonate a legitimate identity. The generic vulnerability is weak credential protection (predictable passwords, lack of MFA).

Reference: V1.9.1 Buzz-Word Refinement (#4)

See also: Identity Theft (#4), Brute-Force Attack

Patient Zero *(Industry Term)*

In TLCTC context: the first system compromised in an attack, representing the initial entry point before lateral movement occurs. Patient Zero systems are critical because they are typically exposed systems that cannot be fully protected by Umbrella Controls alone and require robust Local Controls. After Patient Zero compromise, attacks follow the lateral movement paradigm through the internal network.

Reference: V1.9.1 §A (Umbrella Controls)

See also: Local Controls, Umbrella Controls, Lateral Movement

Phishing *(Industry Term)*

A social engineering technique using deceptive communications (email, SMS, voice) to trick individuals into taking actions that compromise security. In TLCTC: phishing is the **delivery vector**, not a distinct threat category. The phishing lure/deception phase maps to `#9 Social Engineering`. Subsequent steps map to their respective clusters depending on the payload:

- Credential harvesting: `#9 → #4` (phishing → identity theft)
- Malware delivery: `#9 → #7` (phishing → malware execution)
- Browser exploit: `#9 → #3` (phishing → client exploitation)
- Feature misconfiguration: `#9 → #1` (phishing → abuse of functions)

Variants include Spear Phishing, Whaling, Vishing, and Smishing.

Reference: V1.9.1 Clarifications, Buzz-Word Refinement (#9)

See also: Social Engineering (#9), Spear Phishing, Vishing, Smishing

Physical Attack (#8)

A threat cluster where an attacker gains unauthorized physical interaction with or causes physical interference to hardware, devices, facilities, or data transmission media (including wireless signals). The generic vulnerability is the physical accessibility of hardware, facilities, and communication media, and the exploitability of Layer 1 (Physical Layer) communications and hardware interfaces. Encompasses two main types:

- **Direct Physical Access Attacks (#8.1):** Require physical touch or direct interaction (tampering, theft, physical intrusion, unauthorized device connection)
- **Indirect Physical Access Attacks (#8.2):** Exploit physical properties without direct contact (TEMPEST, signal jamming, acoustic attacks, environmental disruption)

Pineapple Attack *(Industry Term)*

An attack using a Wi-Fi Pineapple (or similar rogue access point device) to create fake wireless networks that victims connect to, enabling traffic interception. In TLCTC: the physical deployment of the device maps to `#8 Physical Attack` (requiring physical proximity). Once victims connect, the attacker gains a MitM position — exploitation of that position maps to `#5 Man in the Middle`. The full sequence is `#8 → #5`. Some variants also involve `#4` (faking SSID identity).

Reference: V1.9.1 Buzz-Word Refinement (#5)

See also: Physical Attack (#8), Man in the Middle (#5), Rogue Hotspot

Ping of Death *(Industry Term)*

A denial-of-service attack that sends malformed or oversized ICMP packets to crash or destabilize a target system. In TLCTC: if the crash results from an implementation flaw (buffer overflow in ICMP handling), maps to `#2 Exploiting Server` or `#3 Exploiting Client` per R-ROLE and R-FLOOD. If the primary mechanism is volume-based, maps to `#6 Flooding Attack`.

Reference: V1.9.1 Buzz-Word Refinement (#2)

See also: Flooding Attack (#6), Implementation Defect (Availability Context)

Position Acquisition vs Position Exploitation

For `#5 Man in the Middle`: **Gaining** a MitM position maps to another cluster (`#1`, `#8`, `#9`, `#10`, or `#2/#3` depending on initial generic vulnerability). **Exploiting** a MitM position (intercept, modify, relay, inject, replay, downgrade actions) maps to `#5`.

Reference: §2.2.2 (Global Definitions), R-MITM (§2.2.4)

Preventive Controls

In the Bow-Tie model: barriers on the left (cause) side that reduce likelihood of threats reaching the central event. Corresponds to NIST CSF functions: IDENTIFY, PROTECT.

Reference: §1.4.1 (Bow-Tie Structure)

Pretexting *(Industry Term)*

A social engineering technique where an attacker creates a fabricated scenario (pretext) to manipulate a target into divulging information or performing actions. In TLCTC: maps to `#9 Social Engineering` — the generic vulnerability is human psychological factors (trust, authority bias). Pretexting is a sub-threat of `#9`; any subsequent technical steps map to their own clusters.

Reference: V1.9.1 Buzz-Word Refinement (#9)

See also: Social Engineering (#9), Phishing

Privilege Escalation *(Industry Term)*

A commonly used but multi-faceted term describing an attacker gaining higher-level permissions than initially authorized. In TLCTC: privilege escalation is **not a single cluster** — it maps to different clusters depending on the technique employed:

- **Via software vulnerability** (buffer overflow, injection flaw): `#2 Exploiting Server` or `#3 Exploiting Client` per R-ROLE
- **Via misuse of legitimate features** (misconfiguration, overly permissive defaults): `#1 Abuse of Functions`
- **Via stolen credentials** (using admin credentials): `#4 Identity Theft`
- **Via social engineering** (manipulating users into granting access): `#9 Social Engineering`

Distinguishing the underlying technique allows for more targeted control implementation. The Intra-System Boundary Operator (`||[privilege][@from→@to]|`) can annotate privilege escalation in attack paths without changing cluster classification.

Reference: V1.9.1 Clarifications

See also: Abuse of Functions (#1), Exploiting Server (#2), Identity Theft (#4), Intra-System Boundary Operator

Process Injection *(Industry Term)*

A technique where an attacker inserts code into the address space of another running process. In TLCTC: process injection maps to different clusters depending on the mechanism:

- **Via designed features** (debugging APIs, DLL injection via legitimate Windows functionality): `#1 Abuse of Functions` — the injection capability was intentionally designed.
- **Via implementation flaws** (buffer overflows enabling code injection): `#2 Exploiting Server` or `#3 Exploiting Client` — the injection was never intended.

The key distinction is whether the injection vector was a designed feature being misused versus an underlying software vulnerability being exploited.

Reference: V1.9.1 Clarifications

See also: Abuse of Functions (#1), Exploiting Server (#2), Implementation Flaw

Programmer

A development role focused on architecture and strategy, responsible for designing overall software architecture and component interactions, making strategic decisions about frameworks and protocols, establishing secure coding standards and security requirements, and considering system-wide security implications. Primary responsibility for addressing threat clusters #1, #4, #5, #10 at an architectural level. Contrasts with the Coder role which focuses on implementation and craftsmanship.

Propagated PR *(V2.0)*

A Protection Requirement that "propagates backward" from a downstream event into the RS (Respond) container of an earlier event due to regulatory or policy requirements. Notation: `RS(E`$_n$`) = { Response } ∪ { Propagated PR(E`$_{n+1}$`) } ∪ { Propagated PR(E`$_{n+x}$`) }`.

This mechanism explains how multiple regulatory obligations stack into a single incident response workflow. Example: A ransomware attack triggers E1 (System Compromise). If PII is affected, E2 (Data Risk Event) occurs, propagating GDPR notification requirements back into RS(E1). Simultaneously, if the organization is NIS2-scoped, the incident itself propagates NIS2 reporting into RS(E1). The result: two separate Propagated PR controls in the same RS container, with different timelines (72h vs 24h+72h) and different authorities. See also: RS Container, E_n Event Notation.

Protection Ring Architecture

The layered privilege model in computing systems (Ring 0 through Ring 3) where each ring represents a different privilege level. TLCTC framework analyzes threats at ring boundary interactions: Ring 0 (Kernel Mode), Ring 1 (HAL/Driver Level), Ring 2 (OS Services), Ring 3 (User Mode). Nine threat clusters (excluding #9 Social Engineering) apply at each boundary, with roles (client/server) determined by the direction of interaction across rings.

R

R-ABUSE (Function Misuse Determination)

Global mapping rule: If the attacker's success does not require any implementation flaw and instead abuses intended functionality, scope, or configuration via standard interfaces using expected input types, the step MUST be classified as `#1 Abuse of Functions`.

Reference: §2.2.4 (R-ABUSE)

R-CRED (Credential Lifecycle Non-Overlap)

Global mapping rule: Credential acquisition maps to the enabling cluster; credential application MUST always map to `#4 Identity Theft`. If both occur, they MUST be represented as at least two steps: `(enabling cluster) → #4`.

Reference: §2.2.4 (R-CRED)

R-EXEC (Foreign Execution Recording Rule)

Global mapping rule: Whenever Foreign Executable Content (FEC) is interpreted, loaded, or executed, a `#7 Malware` step MUST be recorded at the moment of execution, independent of how execution was enabled. `#7` is additive (does not replace the enabling cluster).

Reference: §2.2.4 (R-EXEC)

R-FLOOD (Capacity Exhaustion vs Implementation Defect)

Global mapping rule: If the primary mechanism is volume or intensity exhausting finite resources, classify as `#6 Flooding Attack`. If the primary mechanism is an implementation defect that causes crash/hang/degradation (including algorithmic complexity), classify as `#2` or `#3` per R-ROLE.

Reference: §2.2.4 (R-FLOOD)

R-HUMAN (Human Manipulation Isolation)

Global mapping rule: If the attacker's advantage comes from psychological manipulation of a human, that manipulation step MUST be classified as `#9 Social Engineering`, and any subsequent technical steps MUST be classified separately.

Reference: §2.2.4 (R-HUMAN)

R-INTRA (Intra-System Boundary Rules) *(V2.1)*

The complete intra-system boundary rule set governing use of the intra-system operator (`|...|`):

Rule	Name	Summary
R-INTRA-1	Single-System Scope	Operator MUST be used only for boundaries within a single system instance
R-INTRA-2	Cluster Attachment	Operator MUST be attached to the cluster step that accomplishes the crossing

Rule	Name	Summary
R-INTRA-3	No Standalone Use	Operator MUST NOT appear without an associated cluster step
R-INTRA-4	No Cluster Change	Operator MUST NOT change cluster classification
R-INTRA-5	Optional Precision	Operator is OPTIONAL; mainly recommended for forensic or vendor-facing use
R-INTRA-6	Multiple Crossings	Multiple annotations MAY follow one step when compressed form is justified
R-INTRA-7	Distinct Vulnerabilities	If crossing requires a separately evidenced vulnerability, a new cluster step MUST be added
R-INTRA-8	Compressed Form	If evidence does not distinguish separate exploit causes, compressed single-step form MAY be used
R-INTRA-9	Anti-Effect Rule / Memory Deferral	Effects (privilege escalation, sandbox escape, etc.) are NOT independent threat categories; `memory` boundary type is deferred and MUST NOT be used

Reference: §4.2.5 (R-INTRA), §11.3.6 (Intra-System Boundary Operator)

R-MITM (Position vs Action)

Global mapping rule: The method of gaining a privileged communication-path position maps to another cluster. `#5 Man in the Middle` begins only once the attacker controls a point on the communication path and performs MitM actions.

Reference: §2.2.4 (R-MITM)

R-PHYSICAL (Physical Domain Isolation)

Global mapping rule: If the attacker's advantage comes from unauthorized physical interaction or interference with hardware, facilities, media, or signals, that step MUST be classified as `#8 Physical Attack`, and subsequent technical steps MUST be classified separately.

Reference: §2.2.4 (R-PHYSICAL)

R-ROLE (Server vs Client Determination)

Global mapping rule: If the vulnerable component accepts and handles inbound requests relative to the attacker, classify as `#2 Exploiting Server`. If the vulnerable component consumes external responses/content relative to the attacker, classify as `#3 Exploiting Client`.

Reference: §2.2.4 (R-ROLE)

R-SUPPLY (Trust Acceptance Event Placement)

Global mapping rule: `#10 Supply Chain Attack` MUST be placed at the Trust Acceptance Event (TAE)—the moment where the third-party trust link is honored and the trust artifact becomes authoritative inside the organization's domain.

Reference: §2.2.4 (R-SUPPLY)

R-TRANSIT (Transit Boundary Rules) *(V2.1)*

The complete transit boundary rule set governing use of the transit operator (⇒):

Rule	Name	Summary
R-TRANSIT-1	Distinct Parties	`@Transit` MUST be distinct from both `@Source` and `@Target`
R-TRANSIT-2	True Intermediary Topology	Operator MUST be used only when the intermediary sits between source and target in the delivery path
R-TRANSIT-3	Vendor Code on Target Device	Vendor code running on the target device is NOT transit — it is the attack surface and MUST be classified by R-ROLE
R-TRANSIT-4	Control Relevance	Operator SHOULD be used when the intermediary has meaningful control responsibility; MAY be omitted when analytically incidental
R-TRANSIT-5	Pure Conduit Fallback	If the intermediary adds no useful control surface, the analyst MAY use the binary v2.0 boundary or omit the transit annotation
R-TRANSIT-6	Compromise or Coercion Is Separate	If transit is enabled by compromise or coercion of the intermediary, that enabling condition MUST be modeled as a preceding cluster step
R-TRANSIT-7	Cluster Independence	Transit annotation MUST NOT change cluster classification
R-TRANSIT-8	Multiple Transit Parties	Chained transit MAY be used when each intermediary has independent analytical relevance

Reference: §4.2.4 (R-TRANSIT), §11.3.5 (Transit Boundary Operator)

Realtime Velocity Class *(V2.0)*

A velocity classification where attack progression occurs within seconds or milliseconds. Typical threat clusters: #6 (Flooding), #2 (Wormable Exploits). Control strategy must rely on architecture, hardening, and circuit breakers, as detection and response times are insufficient. Example: DDoS attacks, wormable exploits like EternalBlue, automated exploitation frameworks.

Regulatory Trigger Point *(V2.0)*

The specific event type in a TLCTC event chain that activates a regulatory notification or compliance obligation. Different regulations have different trigger points within the same attack sequence:

- **Data-triggered regulations** (e.g., GDPR Art. 33): Activate at E2 (Data Risk Event involving PII) — no PII affected means no GDPR notification obligation
- **Incident-triggered regulations** (e.g., NIS2 Art. 23): Activate at E1 (Significant Incident / System Compromise) — regardless of whether PII is involved

Understanding regulatory trigger points enables CISOs to build precise IR playbooks mapping specific RS container actions to logical triggers rather than generic "reporting checklists". See also: Propagated PR, Event Chain Length.

Responsibility Sphere

The organizational owner of a domain, denoted as `@Entity`. Examples: `@Org`, `@Vendor`, `@Facilities`, `@HR`, `@CloudProvider`, `@MSP`. Different spheres have different policies, teams, governance structures, and potentially different legal boundaries. Domain boundary definitions identify where responsibility and control shift during an attack, which is critical for incident response, forensics, and legal responsibility. Defined in `tlctc-responsibility-spheres.json` and customizable per organization. Standard spheres include: Attacker Side, Third-Party/Vendor Side, Victim Side, Shared/Transit. Used in conjunction with the domain boundary operator (||) in attack path notation.

Reference: §2.2.2 (Global Definitions), §3.4, §5.1.2

Risk Event

In the TLCTC Bow-Tie model, the central occurrence that represents the materialization of a threat, positioned between causes (threats) and effects (consequences). For cyber risk, this is "Loss of Control" or "System Compromise". Can trigger event chains where one outcome becomes the event triggering subsequent events.

Role Determination

Classification of a component as server-role or client-role based on its behavior in the specific interaction being classified. The same software product MAY appear as server-role in one interaction and client-role in another. Classification MUST follow the role of the component being exploited in the step.

Reference: §2.2.2 (Global Definitions), R-ROLE (§2.2.4)

RS Container (Respond Container) *(V2.0)*

The logical collection of RESPOND-function controls and actions for a specific event (E_n) in the TLCTC event chain. An RS Container holds:

- **Direct Response Actions:** Containment, eradication, forensics for the event itself
- **Propagated PR Controls:** Protection requirements inherited from downstream compliance events (E3a, E3b, etc.)

Notation: `RS(E_n) = { Response } ∪ { Propagated PR(E_n+1) } ∪ { Propagated PR(E_n+x) }`. Example: RS(E1) for a ransomware incident may contain both incident containment actions AND propagated GDPR/NIS2 notification controls, each with distinct timelines and reporting authorities. Aligns with NIST CSF RESPOND function. See also: Propagated PR, Regulatory Trigger Point.

Ransomware *(Industry Term)*

Malware that encrypts a victim's data and demands payment for the decryption key. In TLCTC: the execution of the ransomware binary maps to `#7 Malware` (foreign code executed via designed execution capability). The resulting data encryption is a Data Risk Event: `[DRE: Ac]` (Loss of Accessibility — data exists but is unusable). Note: this is Loss of **Accessibility**, not Loss of Availability (the encrypted files still exist on disk). A full ransomware attack path typically involves multiple clusters, e.g., `#9 → #7 → #7 → #4 → (#1 + #7)` as seen in the Emotet/Ryuk case study.

Reference: V1.9.1 §Definitions (#7), §Data Risk Event Types, §E (Emotet@Heise)

See also: Malware (#7), Accessibility (Data Risk Event), Loss of Accessibility

RCE (Remote Code Execution) *(Industry Term — Decomposition Required)*

A commonly used but imprecise term describing CVEs that enable an attacker to execute arbitrary code on a remote target. In TLCTC, "RCE" always conflates the vulnerability with its exploitation and MUST be decomposed:

- The **vulnerability** that enables execution: `#2 Exploiting Server` (server-side flaw) or `#3 Exploiting Client` (client-side flaw)
- The **actual execution** of foreign code: `#7 Malware`

- Correct notation: `#2 → #7` or `#3 → #7`

This decomposition is essential because it identifies two distinct generic vulnerabilities being exploited, each requiring different controls.

Reference: V1.9.1 §F (Industry Term Decomposition)

See also: Exploiting Server (#2), Exploiting Client (#3), Malware (#7)

RFID Skimming *(Industry Term)*

The unauthorized reading of RFID (Radio-Frequency Identification) chips from proximity, typically to clone access cards or extract stored data. In TLCTC: maps to `#8 Physical Attack` (specifically `#8.2 Indirect Physical Access`) — the attacker exploits physical properties (radio frequency emanations) without direct contact with the target device.

Reference: V1.9.1 Buzz-Word Refinement (#8)

See also: Physical Attack (#8), TEMPEST

Risk Appetite / Risk Tolerance *(Industry Term)*

Risk Appetite: The level and type of cyber risk an organization is willing to accept in pursuit of its objectives. **Risk Tolerance:** The acceptable deviation from the defined risk appetite. In TLCTC: risk appetite and tolerance should be defined per threat cluster, enabling differentiated risk treatment. For example, an organization may have zero tolerance for `#10 Supply Chain` risks in critical infrastructure but moderate tolerance for `#6 Flooding` risks with adequate DDoS mitigation in place. Risk appetite is a Strategic Management Layer concern that informs operational priorities.

Reference: V1.9.1 §Strategic Management Layer

See also: Strategic Management Layer, GOVERN (GV)

Rogue Hotspot *(Industry Term)*

A fraudulent Wi-Fi access point set up by an attacker to intercept traffic from unsuspecting users. In TLCTC: the physical deployment of the rogue access point maps to `#8 Physical Attack` (physical proximity required). Interception of traffic through the rogue AP maps to `#5 Man in the Middle`. If the attacker fakes an SSID to impersonate a legitimate network, that may additionally involve `#4` (identity impersonation). Typical sequence: `#8 → #5`.

Reference: V1.9.1 Buzz-Word Refinement (#5)

See also: Pineapple Attack, Man in the Middle (#5), Physical Attack (#8)

Rootkit *(Industry Term)*

Malware designed to provide continued privileged access to a system while actively hiding its presence. In TLCTC: maps to `#7 Malware` — the rootkit is foreign executable content running via the system's designed execution capability. The persistence and stealth mechanisms are operational details within the `#7` cluster.

Reference: V1.9.1 Buzz-Word Refinement (#7)

See also: Malware (#7), Spyware

S

SAST (Static Application Security Testing) *(Industry Term)*

A testing methodology that analyzes application source code, bytecode, or binary code for security vulnerabilities without executing the program. In TLCTC: SAST is a **preventive control** (IDENTIFY/PROTECT) primarily targeting `#2 Exploiting Server` and `#3 Exploiting Client` by identifying implementation flaws before deployment.

See also: DAST, Control, Exploiting Server (#2), Exploiting Client (#3)

SBOM (Software Bill of Materials) *(Industry Term)*

A formal, machine-readable inventory of all software components, libraries, and dependencies used in a software product. In TLCTC: SBOM is a critical control for `#10 Supply Chain Attack` — it enables organizations to identify and track third-party components, verify their integrity, and rapidly assess exposure when vulnerabilities are discovered in dependencies.

See also: Supply Chain Attack (#10), SCA, Third-Party Trust Link

SCA (Software Composition Analysis) *(Industry Term)*

Automated tools that identify open-source and third-party components in a codebase, flagging known vulnerabilities and license compliance issues. In TLCTC: SCA is a preventive control for `#10 Supply Chain Attack`, enabling detection of vulnerable or malicious dependencies before they are integrated into production systems.

See also: Supply Chain Attack (#10), SBOM

Semantic Guardrails (SG-1 through SG-7) *(V2.1)*

Normative rules that prevent V2.1 operators (transit boundary, intra-system boundary, unresolved-step) from drifting the classification model. They enforce that: classification is cause-first (SG-1), topology is not classification (SG-2), annotations are subordinate to

clusters (SG-3), effects are not threats (SG-4), actors are not threats (SG-5), distinct exploits require distinct steps (SG-6), and stripping V2.1 annotations must leave a valid cluster sequence (SG-7).

Reference: §4.2.4 (Semantic Guardrails)

Scope of Server Software

In TLCTC: includes Server APIs, incorporated Library APIs, Socket APIs, and Local APIs that run on server-side systems to provide services and resources to clients. The scope defines the attack surface for `#2 Exploiting Server`.

Reference: V1.9.1 Clarifications

See also: Exploiting Server (#2), Scope of Client Software

Scope of Client Software

In TLCTC: encompasses Client APIs, incorporated Library APIs, Socket APIs, and Local APIs that operate on the client side of a communication. The scope defines the attack surface for `#3 Exploiting Client`. Client software includes web browsers, mobile apps, desktop applications, document readers, command-line clients, client libraries, API consumers, and background services acting as clients.

Reference: V1.9.1 Clarifications

See also: Exploiting Client (#3), Scope of Server Software

Secure Software Development Life Cycle (SSDLC)

A structured approach to embedding security throughout the software development process. The TLCTC framework integrates into each SSDLC phase, with programmer-level decisions during Requirements and Design, coder-level implementation during the Implementation phase, and both roles contributing to verification during Testing and ongoing vigilance during Maintenance.

Sequence

The ordered progression of threat clusters in an attack. The TLCTC framework recognizes that identified Top-Level Threats must also be seen as sequence components in the attack scenario, with attackers using these components in varying orders depending on their script (Axiom VIII). Example: A phishing attack might follow #9→#3→#7, or a more complex attack might be #9→#7→#4→(#1+#7).

Sequence Operator (→)

The operator meaning: the right-hand step occurs after the left-hand step, and the left-hand step enables or makes possible the right-hand step in the described scenario. ASCII alternative: `->`.

Reference: §3.1 (Sequence Operator)

Server-Role Component

A component that **accepts and handles inbound requests or stimuli** relative to the attacker. The component is in "server role" for the specific interaction being classified.

Reference: §2.2.2 (Global Definitions), R-ROLE (§2.2.4)

Session Hijacking *(Industry Term)*

An attack where an adversary takes over an active session by stealing or predicting session tokens/cookies. In TLCTC: the **acquisition** of the session token maps to the enabling cluster (e.g., `#5` if intercepted via MitM, `#3` if stolen via XSS, `#7` if captured by malware). The **use** of the stolen session to impersonate the legitimate user always maps to `#4 Identity Theft` per R-CRED.

Reference: V1.9.1 Buzz-Word Refinement (#4)

See also: Identity Theft (#4), Credential Acquisition, Credential Application

SIEM (Security Information and Event Management) *(Industry Term)*

A category of security tools that aggregate and analyze log data from across an organization's infrastructure to detect security events. In the TLCTC velocity model, SIEM is the primary detection tool for the **Medium Velocity Class** (hours) where analyst triage is feasible. SIEM uses operational notation (`TLCTC-XX.YY`) for correlation rules, while dashboards may display strategic notation (`#X`) for SOC managers.

See also: Medium Velocity Class, EDR, SOAR, Notation Systems

Slowloris *(Industry Term)*

An application-layer denial of service attack that holds many connections to the target web server open by sending partial HTTP requests, slowly exhausting server connection resources. In TLCTC: maps to `#6 Flooding Attack` — the primary mechanism is exhausting finite connection/thread pool capacity. Although each individual request is small, the aggregate effect overwhelms the server's connection handling capacity.

Reference: V1.9.1 Buzz-Word Refinement (#6)

See also: Flooding Attack (#6), DDoS, HTTP Flood

Smishing *(Industry Term)*

Social engineering attacks delivered via SMS text messages. In TLCTC: maps to `#9 Social Engineering` — the delivery channel (SMS) does not change the cluster classification. Subsequent steps map to their respective clusters (e.g., `#9 → #4` for credential harvesting, `#9 → #7` for malware delivery).

Reference: V1.9.1 Buzz-Word Refinement (#9)

See also: Social Engineering (#9), Phishing, Vishing

SOAR (Security Orchestration, Automation, and Response) *(Industry Term)*

A category of security tools that enable automated incident response through predefined playbooks. In the TLCTC velocity model, SOAR is critical for the **Fast Velocity Class** (minutes) where automated containment is necessary because human analyst response times are insufficient. SOAR playbooks can be structured around TLCTC attack paths to automate response actions specific to detected cluster sequences.

See also: Fast Velocity Class, EDR, SIEM

System Risk Event (SRE)

The central event in the TLCTC Cyber Bow-Tie model: **Loss of Control / System Compromise**. The SRE is the pivot point between the cause side (threat clusters exploiting generic vulnerabilities) and the consequence side (data and business risk events). It is the first event in the consequence chain **SRE → DRE → BRE***, where each transition has its own Δt representing a detection and intervention window. Not every SRE leads to a DRE — detection and containment at the central event can break the chain before data-level consequences materialize.

> **Disambiguation:** "Loss of Control" is always abbreviated **SRE**, never "LoC". The abbreviation **LoC** is reserved exclusively for Loss of Confidentiality, a *consequence*-side Data Risk Event. See **Loss of Confidentiality (LoC)**.

Reference: §1.4.3 (Central Event), §1.4.3.1 (The Consequence Chain)

See also: Loss of Control / System Compromise, Data Risk Event (DRE), Business Risk Event (BRE)

Social Engineering (#9)

A threat cluster where an attacker psychologically manipulates individuals into performing actions counter to their or their organization's best interests, such as divulging confidential information, granting access, executing code, or bypassing security procedures. The generic vulnerability is human psychological factors: gullibility, trust, ignorance, fear,

urgency, authority bias, curiosity, or general compromisability. Often serves as the initial vector enabling other threat clusters (e.g., #9→#4 for credential harvesting, #9→#7 for malware installation, #9→#1 for feature misconfiguration).

Spear Phishing *(Industry Term)*

A targeted social engineering attack directed at specific individuals or organizations, using personalized information to increase credibility. Variants include **Whaling** (targeting senior executives). In TLCTC: maps to `#9 Social Engineering` — the targeting precision is an operational detail but does not change the cluster. Subsequent steps map to their respective clusters based on the payload delivered.

Reference: V1.9.1 Buzz-Word Refinement (#9)

See also: Social Engineering (#9), Phishing, Whaling

Spyware *(Industry Term)*

Malware that covertly monitors user activity, collects data (keystrokes, browsing history, credentials), and transmits it to the attacker. In TLCTC: maps to `#7 Malware` — foreign code executing via the system's designed execution capability. The data collection constitutes a Data Risk Event: `[DRE: C]` (Loss of Confidentiality). If collected credentials are subsequently used, an additional `#4 Identity Theft` step applies.

Reference: V1.9.1 Buzz-Word Refinement (#7)

See also: Malware (#7), Keylogger, Trojan

SQL Injection *(Industry Term)*

An implementation flaw where an attacker inserts malicious SQL statements into application queries through unvalidated input, enabling unauthorized database access. In TLCTC: maps to `#2 Exploiting Server` — a coding flaw in server-side query building that creates an unintended data→code transition. SQL injection can lead to immediate Data Risk Events: `[DRE: C]` (data exfiltration), `[DRE: I]` (data modification), or `[DRE: Av]` (data deletion).

Reference: V1.9.1 §Definitions (#2), Buzz-Word Refinement

See also: Exploiting Server (#2), Implementation Flaw, Command Injection

SSRF (Server-Side Request Forgery) *(Industry Term)*

An implementation flaw where an attacker induces the server to make requests to unintended locations, potentially accessing internal resources or services. In TLCTC: maps to `#2 Exploiting Server` — a coding flaw in how server-side code processes URL input. The server is in "server role" (accepting inbound requests) and the flaw is in its implementation.

Reference: V1.9.1 Buzz-Word Refinement (#2)

See also: Exploiting Server (#2), Implementation Flaw

SSL Stripping *(Industry Term)*

A MitM technique where an attacker downgrades HTTPS connections to HTTP by intercepting and modifying communication between client and server. In TLCTC: SSL stripping is an action performed from a MitM position and maps to `#5 Man in the Middle` (exploiting the controlled communication path to perform a protocol downgrade). The attacker must already have the MitM position (gained via `#1` ARP spoofing, `#8` physical access, etc.), making the full sequence e.g., `#1 → #5`.

Reference: V1.9.1 Buzz-Word Refinement (#1)

See also: Man in the Middle (#5), ARP Spoofing, Position Acquisition vs Position Exploitation

STIX (Structured Threat Information Expression)

A standardized language for representing cyber threat information. TLCTC integrates with STIX at the **classification layer**: cluster identifiers can be carried as custom properties (e.g., `x_tlctc_primary_cluster`) or controlled-vocabulary tags on existing STIX objects such as `attack-pattern` and `indicator`. **Layer 3 attack path instances do not round-trip to STIX 2.1** — Δt annotations, parallel groups, boundary operators (`||...||`, ⇒, `|...|`), DRE tags, and unresolved operators (`?`, `…`) have no native STIX 2.1 equivalents and cannot be reconstructed from `attack-pattern` + SRO decomposition. Full Layer 3 exchange requires a custom STIX extension or transporting the TLCTC JSON instance as an out-of-band artifact.

STRIDE *(Industry Term)*

A threat modeling methodology developed by Microsoft that categorizes threats into six types: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. In TLCTC: STRIDE is considered "per se incomplete" — it conflates causes with outcomes (e.g., "Information Disclosure" is an outcome/DRE, not a cause) and lacks coverage of human and physical threat vectors. TLCTC's cause-oriented 10-cluster model provides a more logically consistent foundation, and TLCTC recommends always starting threat assessment with its clusters rather than STRIDE alone.

Reference: V1.9.1 §Standardizing Strategic Cybersecurity, §Operational Layer

See also: OWASP, TLCTC

SYN Flood *(Industry Term)*

A network-layer denial of service attack that exploits the TCP three-way handshake by sending many SYN requests without completing the handshake, exhausting the target's connection table. In TLCTC: maps to `#6 Flooding Attack` — volume exceeding finite connection capacity.

Reference: V1.9.1 Buzz-Word Refinement (#6)

See also: Flooding Attack (#6), DDoS, UDP Flood

Strategic Layer (Human-First)

A naming convention for TLCTC clusters using the format `#X` where X ∈ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Used for executive communication, risk registers, board reporting, strategic planning, and high-level attack path discussion.

Reference: §2.2.1 (Two-Layer Naming Convention)

Strategic Management Layer

The stable top-level layer of TLCTC containing the 10 clusters and their generic vulnerabilities. Used for governance, control mapping, and comparable incident documentation. Focuses on risk management, policy-making, and program governance, including threat cluster categorization, generic vulnerability identification, risk appetite and tolerance definition, security program management, compliance and governance, and resource allocation. Corresponds to cluster identifiers `#1`–`#10` or `TLCTC-XX.00`. Uses strategic notation (#X) for executive communication.

Reference: Axiom VIII (§1.2), §2.2.1

Sub-Threat

Specific, detailed attack techniques or methods that fall within a broader Top Level Cyber Threat Cluster. Sub-threats represent the operational-level detail beneath the strategic-level clusters. Example: Under #2 Exploiting Server, sub-threats include SQL Injection, Buffer Overflows, RCE via Deserialization, SSRF, and XXE Injection.

Supply Chain Attack (#10)

A top-level threat cluster on the cause side of the bow-tie, where an attacker compromises systems by abusing the trust relationship within an organization's supply chain. The attacker targets vulnerabilities in third-party software components, hardware, services, or distribution/update mechanisms that are **trusted and integrated** into the organization's own environment or products. The generic vulnerability is the necessary reliance on, and implicit trust placed in, external suppliers, vendors, components, and their associated development or distribution processes.

Supply Chain as "Bridge Not Bucket": #10 is a *bridge* threat cluster that marks the use of a trusted supply-chain channel as an attack vector to cross from one domain/trust boundary into another (e.g. @Vendor → @Org). It does *not* absorb the semantics of other clusters (#1–#9).

Three Key Supply-Chain Vectors (#10.x):

- **#10.1 Update Vector:** Post-deployment delivery/update flow compromise
- **#10.2 Development Vector:** Pre-deployment build/dev pipeline, repositories, or package ecosystem compromise
- **#10.3 Hardware Supply Chain Vector:** Hardware component, firmware, or manufacturing/assembly compromise

Control-Level Third-Party Dependencies (Not #10): Dependencies on third parties for patch delivery, security updates, or managed services are treated as *governance/control dependencies* on the right side of the bow-tie. They are **not themselves** a #10 Supply Chain Attack unless the trusted integration channel is directly abused as an attack vector.

System Compromise

Alternative term for "Loss of Control" in the Cyber Bow-Tie model. See **System Risk Event (SRE)** and **Loss of Control / System Compromise**.

T

Tailgating *(Industry Term)*

A physical social engineering technique where an unauthorized person follows an authorized person through a secured entrance. In TLCTC: maps to `#9 Social Engineering` when it relies on human psychology (politeness, trust) to gain access. If the entry is forced without social manipulation (e.g., slipping through a door), it maps to `#8 Physical Attack`. Subsequent technical exploitation maps to its own cluster.

Reference: V1.9.1 Buzz-Word Refinement (#9)

See also: Social Engineering (#9), Physical Attack (#8)

Tech Enablers Overlay *(V2.1)*

A forward-looking **informative overlay** on the Cyber Threat Radar that maps emerging technologies (e.g., agentic AI, quantum-resistant crypto, deepfake toolchains, commodity exploit kits) against two axes: the **cluster axis** (which generic vulnerability the technology amplifies, `#1`–`#10`) and the **Actor Archetype axis** (Nation State / Extortion / Fraud / Hacktivist / Amateur). Complements the Attacker Profile overlay: where profiles describe which clusters actors favor *today*, the Tech Enablers Overlay tracks which new capabilities are entering the ecosystem, for whom, and which clusters they are likely to amplify next. Entries carry adoption-level indicators (observed / emerging / hypothesized) and snapshot

dates; they MUST map to clusters via the generic vulnerability amplified, not by vendor or product category. Shifts between snapshots SHOULD trigger a review of the organizational radar and relevant Attacker Profiles.

Reference: §17.4

See also: Cyber Threat Radar, Attacker Profile, Actor Archetype

Tool: `/tools/tech-enablers-radar.html`

Techniques (TTPs)

Specific methods, procedures, and tactics that attackers use to exploit vulnerabilities and achieve their objectives. In the MITRE ATT&CK framework, techniques represent the operational "how" of attacks—the concrete actions adversaries take (e.g., T1190 "Exploit Public-Facing Application", T1566 "Phishing"). Techniques often reference specific vulnerabilities (CVEs) they exploit and the platforms/systems they target.

Relationship to TLCTC: Each MITRE technique can be mapped to one or more TLCTC clusters based on the generic vulnerability being exploited:

- T1190 (Exploit Public-Facing Application) → #2 Exploiting Server
- T1566.001 (Spearphishing Attachment) → #9 Social Engineering → #3 or #7 (depending on payload)
- T1078 (Valid Accounts) → #4 Identity Theft

Key distinction: Techniques describe attacker actions and behaviors (operational detail), while TLCTC clusters categorize the fundamental vulnerabilities being exploited (strategic framework). TLCTC V2.0 proposes enhancing MITRE ATT&CK by adding cluster mappings and typical velocity attributes to each technique.

See also: TTP, Sub-Threat, MITRE ATT&CK, Operational Layer, Weakness

Temporal Notation *(V2.0)*

The V2.0 extension to standard attack path notation that explicitly annotates time intervals between threat cluster transitions (Δt) using the format `→[time]`. Time units include seconds (s), minutes (m), hours (h), days, weeks, months. Examples:

- Basic: `#9→[24h]#4→[12m]#1`
- With domain boundaries: `#9→[days]#4→[mins]#1 ||[dev][@Vendor→@Org]|| →[weeks]#10.2→[0s]#7`
- With parallel execution: `#9→[30s]#7→[2m]#4→[15m](#1+#7)`

Enables precise velocity analysis, detection coverage score calculation, and realistic assessment of control effectiveness against time-sensitive attacks.

Transit Boundary Operator (⇒) *(V2.1)*

Notation: `|||[context][@Source⇒@Carrier→@Target]|||`. An extension to the Domain Boundary Operator that marks responsibility spheres which **carry or relay** the attack without being the source or the target. The `⇒` symbol denotes transit (relay), while `→` denotes delivery to the final target. Chained transit uses right-to-left relay order: `|| [context][@Source⇒@CarrierB⇒@CarrierA→@Target]||`. Transit is distinct from `#10 Supply Chain Attack`: transit marks a passive relay, while `#10` marks a Trust Acceptance Event. Key rule (R-TRANSIT-3): vendor code running on the target device is NOT transit — it is the attack surface (classify by R-ROLE). Example: `#9 || [human][@Attacker⇒@SMSProvider→@Victim]||` — phishing SMS relayed through carrier.

Reference: §3.3.5 (Transit Boundary Operator)

Third-Party Trust Link (TTL)

Any reliance relationship where a third party can influence your domain. Examples: software components/libraries/dependencies, update/distribution channels, federation relationships (IdP/SP), managed control planes, SaaS admin consoles, signing/attestation/provenance chains, firmware/hardware supply chains, CI/CD pipeline integrations.

Reference: §2.2.2 (Global Definitions), §2.1 (#10 Definition)

Threat (in TLCTC)

An initiating force that exploits a generic vulnerability and can trigger the central event (Loss of Control), implemented as a set of tactics, techniques, and procedures (TTP) that attackers apply to provoke an event or incident. In TLCTC, threats are implemented as the 10 Top Level Cyber Threat Clusters, each defined by exactly one generic vulnerability. Threats are positioned on the cause side of the Bow-Tie model, distinct from vulnerabilities, events, and consequences (Axiom III). Threats are NOT outcomes, actors, or control failures.

Reference: §1.4.1 (Bow-Tie Structure), Axioms III–V (§1.2)

Threat Cluster

An organizational construct that groups a set of threats exploiting a common generic vulnerability related to IT systems and humans. The TLCTC framework defines 10 mutually exclusive threat clusters, each associated with a unique generic vulnerability (Axiom I) and distinct attack vector (Axiom II).

Threat Topology

A structural property of TLCTC describing whether a threat cluster (or a concrete attack step) operates primarily within the software domain's technical attack surfaces (**internal**) or enables crossing domain boundaries (**bridge**).

Reference: §5.0 (Topology in TLCTC), §5.1 (Definitions)

Tie-Breaker Rules

Precedence rules applied when a step appears to fit multiple clusters. Applied in order: (1) classify by initial generic vulnerability, (2) implementation flaw vs legitimate function misuse, (3) credential use always wins for the use step, (4) MitM starts at controlled position, (5) flooding is about capacity, (6) FEC execution must be explicit, (7) human/physical/third-party are not shortcuts, (8) document non-obvious decisions.

Reference: §2.2.5 (Tie-Breaker / Precedence Rules)

TEMPEST *(Industry Term)*

A codename for standards and techniques related to electromagnetic emanation security — both the interception of unintentional electromagnetic emissions from electronic equipment and the shielding against such interception. In TLCTC: TEMPEST attacks map to `#8 Physical Attack` (specifically `#8.2 Indirect Physical Access`) — exploiting physical emanation properties without direct contact with the target device. Related techniques include Van Eck Phreaking.

Reference: V1.9.1 Buzz-Word Refinement (#8)

See also: Physical Attack (#8), Van Eck Phreaking, RFID Skimming

Token Hijacking *(Industry Term)*

The theft or manipulation of authentication tokens (OAuth tokens, session tokens, API keys, bearer tokens) to gain unauthorized access. In TLCTC: the **acquisition** of the token maps to the enabling cluster per R-CRED (e.g., `#3` via XSS, `#5` via MitM interception, `#7` via malware). The **use** of the stolen token to authenticate always maps to `#4 Identity Theft`.

Reference: V1.9.1 Buzz-Word Refinement (#4)

See also: Identity Theft (#4), Session Hijacking, Credential Application

Trojan *(Industry Term)*

Malware disguised as legitimate software to trick users into installing it. In TLCTC: the execution of the trojan maps to `#7 Malware` (foreign code running via designed execution capability). The deception that leads to installation typically involves `#9`

`Social Engineering` (tricking the user). Full sequence: `#9 → #7`. The trojan may then perform various malicious actions that map to additional clusters (e.g., credential stealing → `#4`, data exfiltration → `[DRE: C]`).

Reference: V1.9.1 Buzz-Word Refinement (#7)

See also: Malware (#7), Social Engineering (#9)

Typosquatting *(Industry Term)*

Registering domain names or package names that are slight misspellings of legitimate ones to deceive users or automated systems. In TLCTC: when used for malicious package publication (e.g., npm typosquatting), maps to `#1 Abuse of Functions` (abusing the publish functionality) → `#10 Supply Chain Attack` (trust boundary crossing) → `#7 Malware` (code execution at consumer). When used for phishing domains, see Domain Squatting.

Reference: V1.9.1 §Attack Path Notation (Example 3)

See also: Domain Squatting, Supply Chain Attack (#10), Abuse of Functions (#1)

TLCTC (Top Level Cyber Threat Clusters)

A pragmatic and structured framework for targeted threat identification that provides a universal approach to cybersecurity applicable across diverse IT systems and contexts. The framework consists of 10 distinct, non-overlapping threat clusters based on generic vulnerabilities, each with strategic and operational applications. The 10 Top Level Cyber Threat Clusters are:

1. Abuse of Functions
2. Exploiting Server
3. Exploiting Client
4. Identity Theft
5. Man in the Middle (MitM)
6. Flooding Attack
7. Malware
8. Physical Attack
9. Social Engineering
10. Supply Chain Attack

TLCTC Enumeration

A structured identifier system (`TLCTC-XX.YY`) where:

- `TLCTC-` prefix ensures proper attribution to the model
- `XX` represents the primary cluster number (01-10), zero-padded for consistent formatting

- `.YY` suffix designed for future refinement (`.00` designates current high-level definitions)

This provides machine readability, consistent sorting, and extensibility for sub-categorization.

Trust Acceptance Event (TAE)

The moment your domain **honors** the Third-Party Trust Link and treats a Trust Artifact/Decision as authoritative. Actions at TAE include: validate, accept, install, apply, execute, attach privileges. `#10 Supply Chain Attack` is placed at the TAE.

Reference: §2.2.2 (Global Definitions), R-SUPPLY (§2.2.4), §2.1 (#10 Definition)

Trust Artifact / Trust Decision (TAD)

What crosses the boundary and is accepted as authoritative in a third-party trust relationship. Examples: SAML/OIDC assertions, federated tokens, signed packages/updates/container images, CI build artifacts/release binaries, policy/configuration pushes, admin actions from managed platforms, firmware images.

Reference: §2.2.2 (Global Definitions), §2.1 (#10 Definition)

TTP (Tactics, Techniques, and Procedures)

A detailed description of attacker behavior. In the TLCTC framework, specific TTPs (like those in MITRE ATT&CK) are considered instances or sub-threats that are categorized under the broader, cause-oriented Top Level Cyber Threat Clusters.

Two-Tiered Approach

The TLCTC structure distinguishing between:

- **Strategic Management Layer:** High-level risk management, policy-making, and governance using the 10 Top Level Cyber Threat Clusters
- **Operational Layer:** Detailed implementation of controls, specific vulnerability management, and threat intelligence using sub-threats and TTPs

U

UDP Flood *(Industry Term)*

A network-layer denial of service attack that overwhelms a target with UDP packets, consuming bandwidth and processing resources. In TLCTC: maps to `#6 Flooding Attack` — volume exceeding finite network/processing capacity.

Reference: V1.9.1 Buzz-Word Refinement (#6)

See also: Flooding Attack (#6), DDoS, SYN Flood

Umbrella Controls

Security measures that provide protection for groups of IT systems within their scope, such as firewalls, proxies, network zones, or external network filters. These contrast with Local Controls that protect specific systems directly. Important consideration: Umbrella controls provide protection only for specific 'Groups of IT-Systems' within their scope and cannot effectively protect all exposed systems (e.g., a firewall protects 'inner IT-Systems' but not directly exposed ones).

Unknown Δt

A Δt value where no supported time statement can be made. Notation: `Δt=?`.

Reference: §4.0.3, §4.2.3

Unresolved-Step Operators (`?`, `…`) *(V2.1)*

Notation operators for partially-resolved attack paths where forensic evidence confirms that a step (or gap of steps) exists but the cluster cannot yet be determined. `?` represents exactly one unresolved step; `…` (or ASCII `...`) represents a gap of one or more steps. Governed by R-UNRES-1 through R-UNRES-9. Key constraints: unresolved steps MUST NOT carry DRE tags (R-UNRES-5); if any cluster can be defended even weakly, the step MUST be classified as `#X [conf=low]` rather than left unresolved (R-UNRES-4); every unresolved step MUST be accompanied by a prose annotation (R-UNRES-8).

Reference: §11.5.4 (Unresolved-Step Operators)

USB Baiting *(Industry Term)*

A physical attack where an attacker leaves malicious USB devices in locations where targets are likely to find and connect them (parking lots, lobbies, conference rooms). In TLCTC: the physical placement of the USB device maps to `#8 Physical Attack`. If the victim is induced by curiosity to plug it in, social engineering is also involved (`#9`). The execution of malicious code from the device maps to `#7 Malware`. Typical sequence: `#8 → #7` or `#9 → #8 → #7`.

Reference: V1.9.1 Buzz-Word Refinement (#8)

See also: Physical Attack (#8), Malware (#7), Evil Maid Attack

V

Velocity Annotation

Notation: `→[Δt=value]` or `→[Δt=Xh]`, `→[Δt=Xm]`, `→[Δt=Xs]`. Indicates the observed or estimated time interval between one Attack Step and the next. Velocity annotations are OPTIONAL but RECOMMENDED for operational analysis and threat intelligence sharing.

Reference: §2.2.2 (Global Definitions), §3.5.2, §4.2 (Δt Notation)

Velocity Class

Categorical labels for Δt ranges that describe the defender's feasible response mode and determine appropriate control strategies. Four classes are defined:

- **VC-1: Strategic / Latent/Slow** (days → months): Log retention, threat hunting, strategic monitoring
- **VC-2: Tactical / Medium** (hours): SIEM alerting, analyst triage, guided response
- **VC-3: Operational / Fast** (minutes): Automation (SOAR/EDR), rapid containment, prebuilt playbooks
- **VC-4: Real-Time** (seconds → milliseconds): Architecture & circuit breakers, rate-limits, hardening, automatic isolation

Reference: §4.4 (Operational Velocity Classes)

Van Eck Phreaking *(Industry Term)*

A technique for eavesdropping on the contents of a CRT or LCD display by detecting and decoding the electromagnetic emissions produced by the display. In TLCTC: maps to `#8 Physical Attack` (specifically `#8.2 Indirect Physical Access`) — exploiting electromagnetic emanation properties without direct physical contact.

Reference: V1.9.1 Buzz-Word Refinement (#8)

See also: Physical Attack (#8), TEMPEST

Vertical Stack Application

The implementation of TLCTC across the layered architecture of IT systems (from application level to hardware), analyzing client/server roles at each protection ring boundary (e.g., Ring 3 to Ring 0) and directional vulnerabilities.

Vishing *(Industry Term)*

Social engineering attacks delivered via voice calls (phone). In TLCTC: maps to `#9 Social Engineering` — the delivery channel (voice) does not change the cluster classification. Subsequent steps map to their respective clusters.

Reference: V1.9.1 Buzz-Word Refinement (#9)

See also: Social Engineering (#9), Phishing, Smishing

Vulnerability

An exploitable condition in a system that constitutes the **attack surface towards a threat**. A vulnerability is what the threat "sees" and targets — it is the exposed surface that enables the attack vector and, through it, the attack path. Without the vulnerability, the threat has no point of entry; with it, the threat has a viable route from cause to compromise.

In the TLCTC Bow-Tie model this relationship is structural: each of the 10 threat clusters is defined by exactly one **generic vulnerability** (Axiom I). The generic vulnerability is the attack surface that the threat cluster exploits. It is what makes the attack vector possible and what the attack path traverses. Controls exist to reduce or eliminate this exposed surface — preventive controls on the left side of the Bow-Tie shrink the attack surface; the vulnerability that remains is the residual exposure the threat can still reach.

Conceptual hierarchy: Weakness (CWE) → Specific Vulnerability (CVE) → Generic Vulnerability (TLCTC) → Threat Cluster (#1–#10).

- A **weakness** (CWE) is the underlying flaw, bug, or error that creates the condition.
- A **specific vulnerability** (CVE) is a concrete, exploitable instance of that condition in a particular product or version.
- A **generic vulnerability** (TLCTC) is the universal, technology-independent category of attack surface that all specific vulnerabilities of the same nature map to.
- A **threat cluster** is the set of threats that target that generic attack surface.

Example: A coding error that fails to validate input (weakness / CWE-89) creates a SQL injection vulnerability (specific vulnerability / CVE-xxxx-yyyy) in a web application. That vulnerability is the attack surface towards `#2 Exploiting Server` — the exposed surface through which the attacker's exploit code enters. The attack vector is defined by this initial generic vulnerability (Axiom VII), and the attack path proceeds from there (e.g., `#2 → #7 → #4`).

Critical distinction: A vulnerability is an exploitable condition that exists in a system, while a weakness is the underlying flaw, bug, or error that enables that vulnerability to exist. TLCTC focuses on categorizing the generic vulnerabilities — the fundamental attack surfaces — that all specific vulnerabilities map to.

See also: Generic Vulnerability, Weakness, CVE, Threat Cluster, Attack Vector, Attack Path, Bow-Tie Model

W

WAF (Web Application Firewall) *(Industry Term)*

A security control that monitors, filters, and blocks HTTP traffic to and from a web application. In TLCTC: WAF is an **Umbrella Control** primarily targeting `#2 Exploiting Server` — it provides a protective layer against common server-side exploitation techniques (SQL injection, XSS, etc.) for web applications within its scope. WAFs cannot protect against all threat clusters and should be part of a defense-in-depth strategy.

See also: Umbrella Controls, Exploiting Server (#2), Defense-in-Depth

Watering Hole Attack *(Industry Term)*

An attack where an adversary compromises a website frequently visited by the target group, then uses the compromised site to deliver exploits or malware to visitors. In TLCTC: the attack decomposes into multiple steps depending on the method:

- Compromising the website: maps to its own cluster (e.g., `#2 Exploiting Server`, `#4 Identity Theft`)
- Luring users to visit: `#9 Social Engineering` (if actively lured) or implicit via the site's normal traffic
- Exploiting visitor's browser: `#3 Exploiting Client`
- Delivering malware: `#7 Malware`

Typical sequence: `#3 → #7` (for the victim's perspective) or the full chain from attacker's perspective.

Reference: V1.9.1 Buzz-Word Refinement (#3)

See also: Exploiting Client (#3), Drive-By Download, Malvertising

Weakness

A flaw, bug, or error in software, hardware, or processes that enables vulnerabilities to exist. In the Common Weakness Enumeration (CWE) framework, weaknesses are categorized as the root causes of software security problems (e.g., CWE-89 for SQL Injection weakness, CWE-119 for buffer overflow weakness).

Critical distinction in TLCTC context: CWE categorizes weaknesses (the flaws themselves), not vulnerabilities (the exploitable conditions those flaws create). In the TLCTC framework, the conceptual hierarchy flows: **Weakness → Specific Vulnerability (CVE) → Generic Vulnerability → Threat Cluster**.

Example: A coding error that fails to validate input (weakness) creates a SQL injection vulnerability (specific vulnerability), which exploits the generic vulnerability of "server-side code flaws" (#2 Exploiting Server). TLCTC's 10 generic vulnerabilities represent the

universal categories that all specific vulnerabilities ultimately map to, regardless of their underlying weaknesses.

Relationship to TLCTC: While CWE provides granular weakness taxonomy at the code level for developers, TLCTC operates at the strategic level by grouping all resulting vulnerabilities into 10 generic vulnerability categories that define the threat clusters. Both frameworks are complementary.

See also: Vulnerability, Generic Vulnerability, CVE, CWE, Threat Cluster, Coder, Programmer

Whaling *(Industry Term)*

A targeted phishing attack aimed specifically at senior executives or high-value targets within an organization. In TLCTC: maps to `#9 Social Engineering` — the seniority of the target is an operational detail that does not change the cluster classification. Often used in BEC (Business Email Compromise) / CEO Fraud scenarios. Subsequent steps map to their respective clusters.

Reference: V1.9.1 Buzz-Word Refinement (#9)

See also: Social Engineering (#9), Spear Phishing, Phishing

Worm *(Industry Term)*

Self-replicating malware that spreads across networks without requiring user interaction, typically by exploiting vulnerabilities in network-accessible services. In TLCTC: the exploitation of the vulnerability maps to `#2 Exploiting Server` (or `#3 Exploiting Client`), and the execution of the worm payload maps to `#7 Malware`. Wormable exploits are characteristic of the **Realtime Velocity Class** (seconds/milliseconds) — e.g., EternalBlue — where architecture and hardening are the only effective controls because detection/response cannot keep pace.

Reference: V1.9.1 Buzz-Word Refinement (#7), Realtime Velocity Class

See also: Malware (#7), Exploiting Server (#2), Realtime Velocity Class

X

XXE (XML External Entity) Injection *(Industry Term)*

An implementation flaw where an application processes XML input containing references to external entities, potentially leading to data disclosure, SSRF, or denial of service. In TLCTC: maps to `#2 Exploiting Server` (or `#3 Exploiting Client` per R-ROLE) — a coding flaw in XML parsing that creates an unintended data→code transition.

Reference: V1.9.1 Buzz-Word Refinement (#2)

See also: Exploiting Server (#2), SSRF, Implementation Flaw

Quick Reference Tables

Cluster Quick Reference

#	Name	Generic Vulnerability	Topology
#1	Abuse of Functions	Functional scope/trust (designed capabilities abused)	Internal
#2	Exploiting Server	Server-side code implementation flaws	Internal
#3	Exploiting Client	Client-side code implementation flaws	Internal
#4	Identity Theft	Identity-artifact binding / credential lifecycle (use)	Internal
#5	Man in the Middle	Lack of end-to-end communication protection	Internal
#6	Flooding Attack	Finite capacity limitations	Internal
#7	Malware	Designed execution capability for untrusted content	Internal
#8	Physical Attack	Physical accessibility/interference	Bridge
#9	Social Engineering	Human psychological factors	Bridge
#10	Supply Chain Attack	Third-party trust dependencies	Bridge

Reference: §2.1 (Cluster Definitions), §5.2 (Topology Classification)

Axiom Quick Reference

#	Name	Group	Core Statement
I	No System-Type Differentiation	Scope	Generic IT assets; sector labels don't create threat classes
II	Client–Server Model	Scope	Universal interaction abstraction
III	Causes, Not Outcomes	Separation	Threats ≠ data risk events
IV	Not Threat Actors	Separation	Threats ≠ actor identity
V	Not Control Failure	Separation	Control-risk ≠ threat category
VI	Single-Cluster Rule	Classification	One step = one vulnerability = one cluster
VII	Initial-Vulnerability Rule	Classification	Vector defined by initial generic vulnerability
VIII	Strategic–Operational Layering	Classification	Clusters → sub-threats
IX	Sequence + Velocity	Sequences	Clusters chain; Δt measures velocity
X	Credential Duality	Sequences	Acquisition vs application

Reference: §1.2 (Axioms and Assumptions)

R-* Rules Quick Reference

Rule	Distinguishes	Key Decision
R-ROLE	`#2` vs `#3`	Server-role (accepts inbound) → `#2`; Client-role (consumes external) → `#3`
R-CRED	Acquisition vs Use	Acquisition → enabling cluster; Use → always `#4`
R-MITM	Gaining vs Exploiting	Gaining position → enabling cluster; Exploiting position → `#5`
R-FLOOD	Capacity vs Defect	Volume exhaustion → `#6`; Implementation defect → `#2/#3`
R-EXEC	FEC Execution	If FEC executes → `#7` MUST be recorded (plus enabling cluster)
R-SUPPLY	TAE Placement	`#10` at Trust Acceptance Event where third-party trust is honored
R-HUMAN	Human Manipulation	Psychological manipulation → `#9`; subsequent tech steps separate
R-PHYSICAL	Physical Access	Physical interaction → `#8`; subsequent tech steps separate
R-ABUSE	Function Misuse	No flaw required, legitimate capability abused → `#1`
R-TRANSIT-1 *(V2.1)*	Distinct Parties	`@Transit` MUST be distinct from `@Source` and `@Target`
R-TRANSIT-2 *(V2.1)*	True Intermediary Topology	Operator only when intermediary sits between source and target
R-TRANSIT-3 *(V2.1)*	Transit vs Attack Surface	Vendor code on target device → classify by R-ROLE, not transit
R-TRANSIT-4 *(V2.1)*	Control Relevance	SHOULD annotate when intermediary has control responsibility
R-TRANSIT-5 *(V2.1)*	Pure Conduit Fallback	MAY omit transit if intermediary adds no useful control surface
R-TRANSIT-6 *(V2.1)*	Compromise Separation	Intermediary compromise → preceding cluster step; transit alone insufficient
R-TRANSIT-7 *(V2.1)*	Cluster Independence	Transit annotation MUST NOT change cluster classification
R-TRANSIT-8 *(V2.1)*	Multiple Transit Parties	Chained transit MAY be used when each party has independent relevance
R-INTRA-1 *(V2.1)*	Single-System Scope	Operator only for boundaries within a single system instance
R-INTRA-2 *(V2.1)*	Cluster Attachment	Operator MUST be attached to the cluster step
R-INTRA-3 *(V2.1)*	No Standalone Use	Operator MUST NOT appear without an associated cluster step

Rule	Distinguishes	Key Decision
R-INTRA-4 *(V2.1)*	No Cluster Change	Operator MUST NOT change cluster classification
R-INTRA-5 *(V2.1)*	Optional Precision	Operator is OPTIONAL; recommended for forensic/vendor-facing use
R-INTRA-6 *(V2.1)*	Multiple Crossings	Multiple annotations MAY follow one step when compressed form justified
R-INTRA-7 *(V2.1)*	Distinct Vulnerabilities	Separately evidenced vulnerability → new cluster step required
R-INTRA-8 *(V2.1)*	Compressed Form	Compressed single-step MAY be used when evidence doesn't distinguish causes
R-INTRA-9 *(V2.1)*	Anti-Effect / Memory Deferral	Effects are not threats; `memory` boundary type deferred → MUST NOT use
R-UNRES-1 *(V2.1)*	Semantic Constraint	`?` and `...` represent real attack steps, not noise or speculation
R-UNRES-4 *(V2.1)*	Classification Threshold	If any cluster can be defended → use `#X [conf=low]`, not `?`
R-UNRES-5 *(V2.1)*	No DRE on Unresolved	DRE tags MUST NOT be appended to `?` or `...`
R-UNRES-7 *(V2.1)*	Resolution Obligation	Every `?`/`...` is an open analytical task → resolve when evidence arrives
R-UNRES-8 *(V2.1)*	Prose Required	Paths containing `?`/`...` MUST have prose annotation explaining gap
R-UNRES-9 *(V2.1)*	Binary Classification	No partial-confidence operators (`?#4`, `#4?`, `#{2|7}`)

Semantic Guardrails Quick Reference *(V2.1)*

ID	Rule	Key Constraint
SG-1	Cause First	Classify by generic vulnerability exploited, not topology or effect
SG-2	Topology ≠ Classification	Transit/intra-system annotations MUST NOT define or imply a cluster
SG-3	Annotations Subordinate	Annotations MUST NOT appear as independent path elements or replace clusters
SG-4	Effects ≠ Threats	Sandbox escape, privilege escalation, etc. are not TLCTC clusters
SG-5	Actors ≠ Threats	Transit parties, vendors, carriers are spheres, not threat categories
SG-6	Distinct Exploit Rule	Separately evidenced exploit → new cluster step required
SG-7	Backward Recoverability	Stripping V2.1 annotations MUST leave a valid cluster sequence

Reference: §4.2.4 (Semantic Guardrails), §4.2.5 (Global Mapping Rules)