TLCTC Framework Glossary V2.0

Comprehensive definitions and concepts for the Top Level Cyber Threat Clusters framework. Including Attack Velocity, Domain Boundaries, and JSON Architecture.

BK Bernhard Kreinz • Reading time varies

A B C D E F G I J K L M N O P R S T U V W

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.

Accessibility (Business 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).

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, typically expressed using TLCTC notation (e.g., `#9→#3→#7`). In V2.0, attack paths can include temporal annotations showing the time between steps (e.g., `#9→[24h]#4→[12m]#1`) and domain boundary markers using the `||` operator.

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 (V2.0), and `|||[context]|||` for domain boundaries (V2.0). 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.

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 defined by the generic vulnerability it initially targets (Axiom II).

Attack Velocity (Δt) V2.0

The temporal dimension of cyber risk representing the time interval between threat cluster transitions in an attack sequence. Denoted as Δt (delta t), it measures how fast an attack progresses from one cluster to another. 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).

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.

Axioms

The foundational principles that must be accepted to validate and effectively use the TLCTC framework. These include ten key axioms that establish relationships between threats, vulnerabilities, attack vectors, and the overall structure of the framework . See the White Paper "Assumptions - Axioms" section for complete details.

B

Bow-Tie Model

A risk management visualization tool adopted by the TLCTC framework that separates causes (threats - left side) from effects (consequences - right side), with the risk event (system compromise / loss of control) positioned at the center. The model enforces temporal causality, prevents confusion between threats and outcomes, enables precise control placement, and reveals attack sequences as causal chains. Critical: The central event "Loss of Control" serves as the pivot point between threat realization and potential consequences.

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

Client-Server Relationship

In the TLCTC framework, a fundamental principle (Axiom VII) 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

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.

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 the control objective which can allow threats to materialize and impact assets. Distinguished from the actual risk event itself (Axiom IV).

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).

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 "Loss of Control" or "System Compromise".

Cyber Threat Radar

A visualization tool based on the TLCTC framework that displays threat distribution across different domains (organizational, state, or sector levels). Uses radar chart format to show impact levels (High/Red, Medium/Orange, Low/Gray, Latent) and movement indicators (▲ increasing, ▼ decreasing) for each of the 10 threat clusters . Enables strategic overview, comparative analysis, and standardized threat communication across organizations and borders.

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 Events

The three primary consequences of system compromise in the TLCTC framework, defined from the attacker's outcome perspective:

- Loss of Confidentiality (C) - Data stolen / unauthorized access to data
- Loss of Integrity (I) - Data modified / unauthorized changes to data
- Loss of Availability (A) - Data gone or unreachable / data no longer exists or cannot be technically accessed by the infrastructure

These outcomes are separated from the mechanisms used to achieve them, enabling clearer risk definition and more effective control mapping. Note: Loss of Availability is distinct from Loss of Accessibility—encrypted data remains available (exists on disk) but is not accessible (unusable for business purpose).

Delta t (Δt) V2.0

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

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 Boundary Operator (||) V2.0

A notation symbol introduced in TLCTC V2.0 to explicitly mark trust domain transitions in attack paths. Format: `||[context][@Source→@Target]||` where 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.

Dual-Use Tools

Legitimate software tools or utilities 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 the TLCTC framework, the invocation mechanism typically maps to #1 Abuse of Functions, while the actual execution of foreign code/scripts maps to #7 Malware, resulting in a #1→#7 sequence.

E

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

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.

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.

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.

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

A fundamental, root-level weakness common across IT systems that defines a threat cluster. Each of the 10 Top Level Cyber Threat Clusters is associated with one unique generic vulnerability (Axiom I). Generic vulnerabilities persist regardless of specific IT system types, software implementations, or evolving attack techniques. Examples include: "The scope of software and functions" (#1), "Server-side code flaws" (#2), "Client-side code flaws" (#3), "Weak identity management" (#4), "Communication path control" (#5), "Finite capacity" (#6), "Code execution capabilities" (#7), "Physical accessibility" (#8), "Human psychological factors" (#9), "Third-party trust dependencies" (#10).

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.

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.

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.

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)

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.

Kxl 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)

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. In the TLCTC framework, LOLBAS attacks typically map as sequences: the invocation/abuse of system capability (#1) followed by execution of foreign code/scripts (#7), resulting in #1→#7 sequences. Examples: Using cmd.exe, PowerShell, WMI, or Task Scheduler to execute attacker-controlled scripts .

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 becomes inaccessible to legitimate users. From the attacker's perspective: "Data inaccessible". This outcome can result from various mechanisms including deletion (a mechanism, not an outcome itself), encryption (ransomware), or system disruption (flooding). Not to be confused with the threat mechanisms that cause this outcome.

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).

Loss of Control

The central event in the Cyber Bow-Tie model, representing system compromise. This serves as the pivot point between threat realization (cause) and potential consequences (effect). Critical: 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.

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

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.

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.

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.

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

Operational Layer

The detailed implementation level in the TLCTC two-tiered approach 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 operational notation (TLCTC-XX.YY) for precise technical implementation.

P

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)

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.

reduce on implementation and maintenance.

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: $RS(E_n) = \{ \text{Response} \} \cup \{ \text{Propagated PR}(E_{n+1}) \} \cup \{ \text{Propagated PR}(E_{n+x}) \}$.

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

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

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.

Responsibility Spheres V2.0

Domain boundary definitions that identify where responsibility and control shift during an attack. Critical for incident response, forensics, and legal responsibility. Defined in `tlctc-responsibility-spheres.json` and customizable per organization. Standard spheres include:

- **Attacker Side:** Infrastructure controlled by threat actor
- **Third-Party/Vendor Side:** External entities in supply chain
- **Victim Side:** Target organization's infrastructure
- **Shared/Transit:** Network path intermediaries

Used in conjunction with the domain boundary operator (||) in attack path notation to precisely document where trust boundaries are crossed.

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.

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.

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: $RS(E_n) = \{ Response \} \cup \{ Propagated\ PR(E_{n+1}) \} \cup \{ Propagated\ PR(E_{n+x}) \}$. 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.

S

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) where the final step involves parallel execution of privilege escalation and ransomware deployment .

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) .

STIX (Structured Threat Information Expression)

A standardized language for representing cyber threat information. TLCTC V2.0 proposes enhancement by introducing TLCTC clusters as a new STIX Domain Object, adding attack path sequence objects, and enhancing attack pattern objects with cluster mappings, enabling standardized high-level threat categorization and attack sequence representation in threat intelligence sharing .

Strategic Management Layer

The high-level planning and governance level in the TLCTC two-tiered approach, focusing on risk management, policy-making, and program governance. Includes threat cluster categorization, generic vulnerability

identification, risk appetite and tolerance definition, security program management, compliance and governance, and resource allocation. Uses strategic notation (#X) for executive communication.

Sub-Threat

Specific, detailed attack techniques or methods that fall within a broader Top Level Cyber Threat Clusters. 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). When #10 appears in an attack sequence, it identifies the step where the attacker exploits a supply-chain channel to traverse a trust/domain boundary.

In Version 2.0 attack-path notation, supply-chain bridges are shown in sequence and may be combined with explicit domain boundary markers, for example:

- `#9 → #4 → #1@Vendor → #10.2 |||[dev][@Vendor→@Org]|| → #7@Org`
- `#9 → #4 → #1 → #10.1 |||[updates][@Vendor→@Org]|| → #3`

Here, #10.x denotes the supply-chain vector being exploited, and `||[channel][@Source→@Target]||` denotes the explicit domain/trust boundary crossing via that channel.

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

- **#10.1 Update Vector:** Post-deployment delivery/update flow compromise (e.g. compromised signed updates, package feeds, or auto-update mechanisms delivered from @Vendor into @Org).
- **#10.2 Development Vector:** Pre-deployment build/dev pipeline, repositories, or package ecosystem compromise (e.g. poisoned build artifacts, compromised CI/CD, dependency confusion) that are later integrated into @Org products or environments.
- **#10.3 Hardware Supply Chain Vector:** Hardware component, firmware, or manufacturing/assembly compromise that is physically integrated into the organization's infrastructure (e.g. backdoored devices, tampered components).

Control-**Level Third-Party Dependencies (Not #10):** Dependencies on third parties for patch delivery, security updates, or managed services (e.g. relying on a vendor to release a patch for a #2 or #3 vulnerability) are treated as *governance/control dependencies* on the right side of the bow-tie. They affect the effectiveness and timeliness of controls against other threat clusters (#2 Exploiting Server, #3 Exploiting Client, #7 Malware, etc.), but they are **not themselves** a #10 Supply Chain Attack. According to Axiom IV, failures here are control failures, not new threats. These risks are managed via Third-Party Risk Management (TPRM) and related governance controls unless the trusted integration channel is directly abused as an attack vector, in which case #10 applies.

System Compromise

Alternative term for "Loss of Control" in the Cyber Bow-Tie model, representing the central cyber risk event where an attacker gains unauthorized control over a system through exploitation of one or more threat clusters.

T

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: In the TLCTC framework's two-tiered approach, MITRE techniques represent the operational layer that maps to the **strategic layer** of the 10 Top Level Cyber Threat Clusters. 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, enabling organizations to move from "what happened" (technique detection) to "how fast it happened" (velocity-aware defense).

See also: TTP (Tactics, Techniques, and Procedures), 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 .

Threat

A set of tactics, techniques and procedures (TTP) that attackers apply to provoke an event or incident, exploiting vulnerabilities in IT systems or human behaviors. In the TLCTC framework, threats are positioned on the cause side of the Bow-Tie model, distinct from vulnerabilities, events, and consequences (Axiom III).

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) .

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

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.

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

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) .

V

Velocity Classes V2.0

The four operational velocity classifications in TLCTC V2.0 that categorize attack sequences based on their Δt (time interval) characteristics:

- **Latent/Slow:** Days to Months - Requires log retention, threat hunting
- **Medium:** Hours - Enables SIEM alerting, analyst triage
- **Fast:** Minutes - Requires automated containment, EDR blocking
- **Realtime:** Seconds/Milliseconds - Demands architecture, hardening, circuit breakers

breakers

Each velocity class determines appropriate control strategies, as control effectiveness depends on matching response speed to attack velocity.

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.

Vulnerability

An exploitable condition or weakness in a system that can be leveraged by a threat actor to compromise security. In the TLCTC framework, vulnerabilities exist in a hierarchical relationship: specific vulnerabilities (like individual CVEs) are instances of generic vulnerabilities, which in turn define the 10 Top Level Cyber Threat Clusters (Axiom I).

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

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 that all specific vulnerabilities map to, providing a universal framework for threat management regardless of the specific CVEs or underlying weaknesses.

See also: Generic Vulnerability, Weakness, CVE, Threat Cluster

W

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: CWE helps developers eliminate weaknesses during coding; TLCTC helps organizations manage the threats that exploit the vulnerabilities those weaknesses create.

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

> **V2.0 Enhancements**
> This glossary includes significant updates for V2.0, such as the introduction of Attack Velocity (Δt), Detection Coverage Score (DCS), the 4 Velocity Classes, Temporal Notation, Domain Boundary Operators, JSON Architecture, and enhanced integration with CWE and MITRE ATT&CK frameworks.