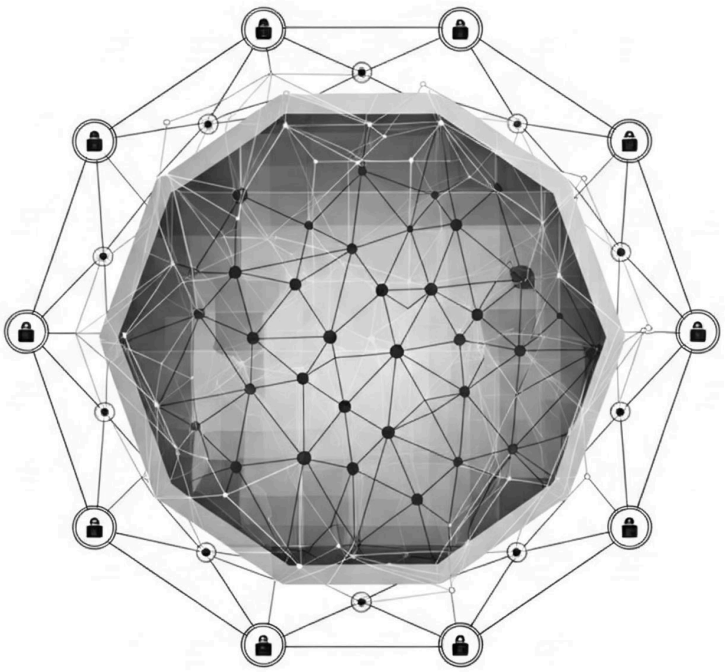
White Paper Version 1.6.3

April 2025, Bernhard Kreinz

Top Level Cyber Threat Clusters



Bridging the Gap: A Unified Approach to identify and categorize Threats in Cyber Risk Management. A Pragmatic Solution for Targeted Cyber Threat Identification and Cyber Risk Management connecting the Operational Level with the Strategic Level.

Executive Summary

The cybersecurity landscape is fragmented. Organizations struggle with inconsistent terminology and siloed approaches to threat identification and risk management, hindering effective defense strategies. Current frameworks often conflate vulnerabilities, attack techniques, and outcomes, leading to confusion and gaps in threat modeling. This white paper introduces the Top Level Cyber Threat Clusters (TLCTC) framework, a novel solution designed to bridge this critical gap and unify strategic planning with operational security. Unlike existing approaches, the TLCTC framework provides a universal, consistent taxonomy of ten distinct threat clusters, each rooted in a fundamental underlying vulnerability rather than observed events or attacker behaviors. This clear, cause-oriented categorization facilitates targeted threat identification, precise mapping of threats to controls, and seamless integration with existing frameworks like NIST CSF, MITRE ATT&CK & CWE, and STIX.

The TLCTC framework employs a unique two-tiered approach, distinguishing between strategic management and operational security. At the strategic level, it empowers leadership to define risk appetite, allocate resources effectively, and communicate cyber risk clearly. Operationally, it enables security teams to implement targeted threat intelligence, enhance incident response, and streamline security operations. This unified approach ensures consistent cybersecurity strategy understanding and execution across all levels of the organization.

This white paper details the derivation of the ten threat clusters through a logical thought experiment, provides clear definitions and real-world examples, and outlines methods for integrating the framework into existing security practices, including secure coding guidelines and the software development lifecycle. Furthermore, it introduces the concept of Cyber Threat Radars, a visualization tool based on the TLCTC framework, for improved threat analysis, communication, and collaboration across organizations and

national borders. By adopting the TLCTC framework, organizations can transition from reactive, fragmented cybersecurity practices to a proactive, unified approach, strengthening their overall security posture and enabling more informed decision-making in the face of evolving cyber threats, leading to more resilient and adaptable cybersecurity postures. I encourage the cybersecurity community to engage with this framework, validate its applicability, and provide feedback to further refine and enhance its effectiveness.

Table of Content

Introduction

Demystifying the Cyber Threat Landscape: A Pragmatic Approach to Threat Identification and Risk Management

Cybersecurity professionals face a critical challenge in effectively identifying and categorizing threats due to the inconsistent and often ambiguous guidance provided by leading standards and frameworks (NIST CSF, ISO 27000, CIS, ENISA, BSI, MITRE, others and all CERT reports I have analyzed). The lack of clear distinctions between threats, threat actors (or their motivation), vulnerabilities, control failures, IT system types, and risk events has led to a semantic blur that hinders the development of effective risk management strategies.

Driven by the need for a more coherent and actionable approach, I embarked on a thought experiment to distill the essence of what constitutes a 'threat' in the cybersecurity domain. The objective was to create a refined conceptual framework that clearly segregates threats from commonly confused elements, providing a universal approach to cybersecurity that can be applied across diverse IT systems and contexts.

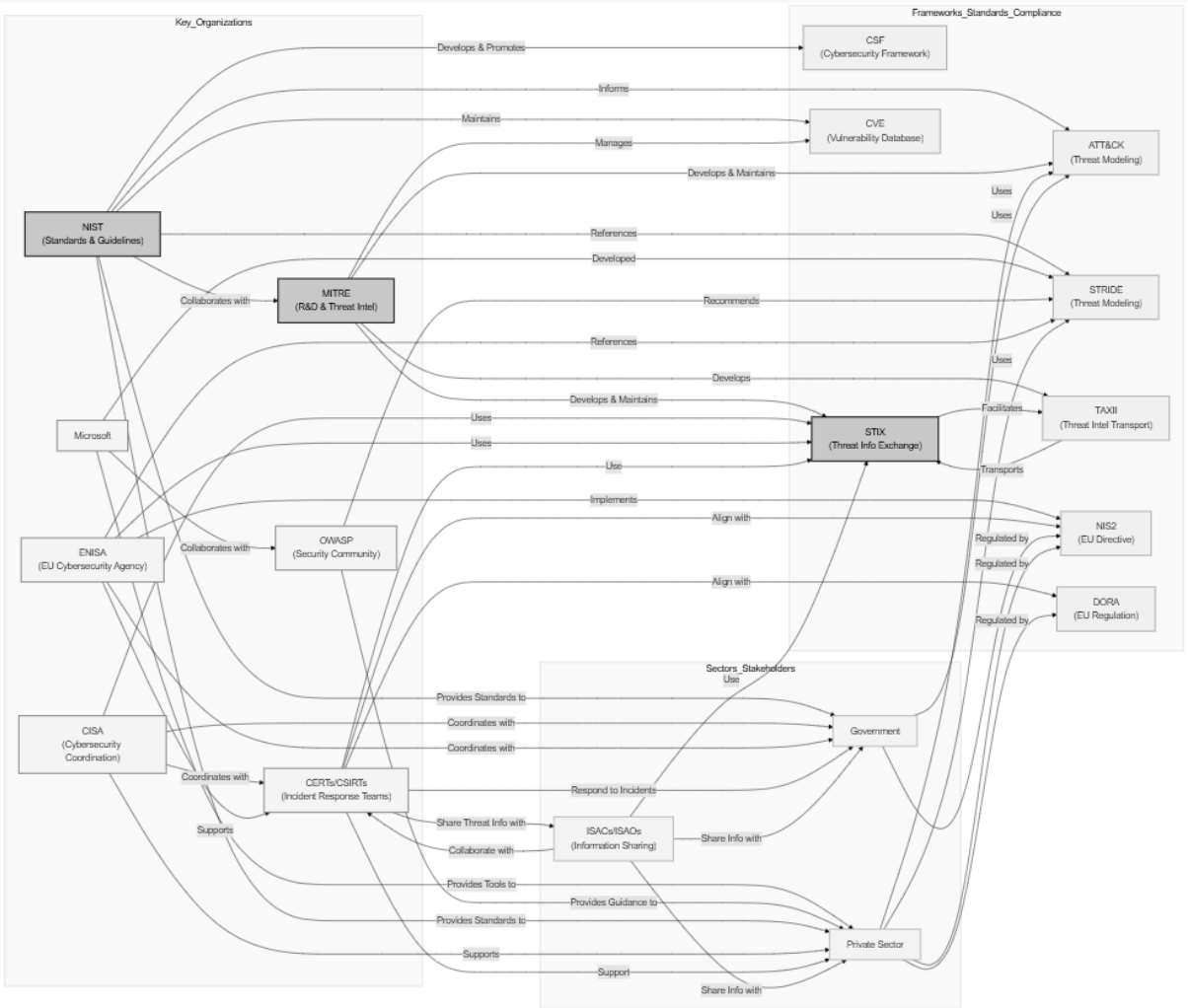
The resulting framework, the "Top Level Cyber Threat Clusters (TLCTC)" provides a pragmatic and structured solution for targeted threat identification. It seamlessly integrates enterprise risk management (ERM) with security operations center (SOC) and threat intelligence processes. By defining distinct, non-overlapping categories, this framework eliminates ambiguity and ensures precise mapping of threats to controls. These clusters are universally applicable both horizontally across various domains (e.g., enterprise IT, cloud environments, IoT) and vertically through the IT stack (e.g., application layer, operating system, hardware). This approach bridges the gap between strategic risk management and operational security, empowering organizations to develop targeted threat intelligence,

implement effective risk mitigation strategies, and address the complexity of the cyber threat landscape with clarity and confidence.

It is crucial to understand that cyber risks are a subset of the broader category of operational risks (OpRisk). While cyber risk management focuses primarily on threats from unauthorized or unknown entities, a comprehensive risk management strategy must consider the full spectrum of operational risks. This includes traditional IT risks (with threats such as e.g. "error in use" and "abuse of rights"), compliance risks, and third-party risks (including their associated cyber risks). Organizations should integrate cyber risk management within a holistic OpRisk framework to gain a consolidated view of their risk landscape. This approach allows for better resource allocation, more effective risk mitigation strategies, and a clearer understanding of how cyber risks interact with other operational risks. It's important to note that while actions of authorized actors (such as employees or customers) are typically managed under separate risk categories, any attempts by these individuals to breach or misuse systems would fall within the scope of cyber risks. This nuanced approach ensures that all potential cyber threats are addressed, regardless of their origin, while maintaining the broader context of operational risk management.

Objectives

The Challenge: A Fragmented Cybersecurity Landscape



Today's cybersecurity landscape is fragmented by disparate frameworks, varying terminology, and siloed approaches to threat management. Organizations struggle to maintain a coherent view of cyber threats across strategic planning, operational security, and global collaboration. Standards bodies, security teams, and threat intelligence providers often speak different languages when describing the same threats, leading to inefficient risk management and delayed response times.

The 10 Top Level Cyber Threat Clusters framework serves as a Rosetta Stone in the fragmented cybersecurity landscape, providing a universal translation layer between

strategic risk management and operational security through three core stakeholder objectives:

For Strategic Leadership & Risk Management:

- Establish a universal standard for cyber threat identification and risk management
- Enable direct mapping of threats to enterprise risk management processes
- Support quantifiable risk assessment through standardized threat categorization

For Security Operations & Technical Teams:

- Provide a systematic foundation for threat hunting and incident response
- Enable precise attack path mapping and root cause analysis
- Create a common taxonomy for threat intelligence sharing and incident classification

For Global Cybersecurity Community:

- Establish a common language for cross-border threat communication
- Enable standardized threat intelligence sharing between sectors
- Support coordinated incident response across organizations

This streamlined framework bridges the gap between strategic cyber risk management and operational security practices, uniting the fragmented landscape of cybersecurity approaches under a common understanding supported by major organizations and frameworks like NIST, CISA, ETSI, CVE, and MITRE.

Assumptions - Axioms

YOU MUST AGREE TO THIS ASSUMPTIONS IN ORDER TO VALIDATE THIS CONCEPT

Why Start With Assumptions and Axioms?

Before diving into the cyber threat clusters, we must establish our foundational principles. In any logical framework, axioms serve as basic truths that we accept without proof, while assumptions define the scope and context of our thinking. Like mathematical proofs that build upon basic axioms, our cyber threat framework requires clear starting points to ensure consistent and logical development.

Agreement Required

The following assumptions and axioms form the essential foundation of the 10 Top Level Cyber Threat Clusters concept. You must agree with these basic principles to validate and effectively use this framework. If any of these foundational elements don't align with your understanding, the subsequent threat categorization may not serve its intended purpose.

Key Axioms and Assumptions

I. Threats and Vulnerabilities of assets (software, hardware, human) have a unique relationship, i.e., for every generic vulnerability (root weakness), there is ONE threat cluster.

II. Each distinct attack vector is defined by the generic vulnerability it initially targets.

III. Threats are on the cause side from a Bow-Tie perspective. That means we do not mix threats with events like data breach (Loss of Confidentiality) or Loss of Availability (eg DDOS) or Loss of Integrity.

IV. The failure of controls is a Control-Risk (deviation from the Control Objective / lack of effectiveness) and should not be confused with the actual Risk

(Threat->Incident/Event->Consequences). Therefore, it is not a structuring element.

V. We separate threats from threat actors because threats can be applied by different actors. On the strategic level, this is sufficient. From a Cyber Defense perspective, it makes sense to further differentiate and "track" these APTs and others.

VI. We stick to the IT assets of the generic software and hardware and do not differentiate by IT system types. Even SCADA systems have software and hardware. Medical devices likewise and network components like switches, routers, and firewalls too - IoT also. Generic refers to the fundamental components and architecture common to all IT systems, regardless of their specific domain or purpose.

VII. Every networked software system, regardless of its complexity or scale, is fundamentally based on the principle of client-server interaction. This interaction occurs at various levels, from basic network communication (such as IP and DHCP) to complex application architectures. The ten cyber threat clusters address the vulnerabilities inherent in these interactions.

VIII. The identified Top-Level Threats can/must also be seen as sequence components in the attack scenario of the cyber actor (attack vector, attack path (inclusive lateral movement)). The sequence in which the attacker uses these components varies from perpetrator to perpetrator and their "script."

IX. Top-Level Threat Clusters have Sub-Threats - It is the separation "Strategic Level" and "Operational Level."

Without these clear starting points, we risk mixing threats with vulnerabilities, confusing causes with effects, and creating overlapping or inconsistent categories that don't serve practical security needs.

The Thought Experiment

Imagine the complex world of information technology as a single object. This object, although robust and seemingly closed, has various attack surfaces – the generic vulnerabilities.

1. We are at asset software. First, we concentrate on the essentials and take care of the functional domain and scope and realize that every function can be abused and that more scope also means more attack surface. Here our first threat cluster arises: ****Abuse of Functions****

2. Every software, although optimized, may contain code flaws that can be exploited, especially if it is directly exposed (Server side). This leads us to the threat cluster: ****Exploiting Server****

3. Even on the client side, there is a risk that existing software code flaws can be exploited. This type of attack, where the client accesses a malicious resource, manifests itself in the threat cluster: ****Exploiting Client****

4. Our software interacts with identities and credentials, both human and technical. When these identities are compromised, they can be abused. This leads to the threat cluster: ****Identity Theft****

5. Communication is crucial in our connected world. Yet, as data is transmitted between points A and B, rogue parties might eavesdrop or inject themselves. This reveals the threat cluster: ****Man in the Middle****

6. This continuous connectivity also makes us susceptible to attacks that want to flood our infrastructure or software (application) and put it out of action. This leads us to the threat cluster: ****Flooding Attack****

7. In the digital landscape, there is a continuous exchange of files and data. Some of these files could contain malware code and thus pose a threat. Here the threat cluster arises: ****Malware****

8. We must not forget that there are physical points of access and interaction through which intruders might come. Therefore, we have the threat cluster: ****Physical Attack****

9. And we should not forget about the human factor. We are susceptible to deception, manipulation, and misconduct. This human element leads us to the threat cluster: ****Social Engineering****

10. Our software or hardware ecosystems are almost always linked with third-party software or hardware. Do we have control over these? This leads to the last threat cluster: ****Supply Chain Attack****

Through this thought experiment and careful examination of vulnerabilities in the IT landscape, I have derived these 10 distinct top level threat clusters. It offers us a clear structure and a deeper understanding of the diverse threats that our IT systems, people, and processes face.

Definitions

The control examples are merely exemplary and intended to facilitate quick understanding. Systematic control selection is discussed in the chapter "Cyber Threat Cluster Control Framework."

#1 Abuse of Functions

Definition: An attacker abuses the logic or scope of existing, legitimate software functions, features, or configurations 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.

Generic Vulnerability: The scope, complexity, or inherent trust placed in legitimate software functions, features, and configurations. More scope/complexity can create a larger attack surface.

Context: This threat addresses the manipulation of the *functional domain* itself – what the software is *designed to do*. The attacker misuses capabilities intentionally built into the system, often exceeding implicit boundaries or leveraging overly permissive designs via standard interfaces using expected input types (data, parameters, configurations, sequence of actions). Crucially, this does not involve executing foreign Malware Code (unlike #7), nor does it rely on exploiting specific implementation flaws/bugs (unlike #2/#3 Exploiting Server/Client). When initiated via Social Engineering (#9), this often involves tricking a user into enabling, disabling, or misconfiguring legitimate, existing features (e.g., enabling RDP access, creating firewall exceptions, changing security settings), rather than installing new software (which would facilitate #7).

Sub-Threats Examples: Data Poisoning, Abuse of document sharing functions, BGP Hijacking, Misuse of API functionalities, Parameter Tampering (exploiting logic), Enabling insecure configurations

Control Examples: Input validation (for logic), Strong configuration management, Least privilege for functions/APIs, Feature usage monitoring, Business logic checks, Multi-step approvals for sensitive configurations

Attacker's View: "I abuse a functionality, not a coding issue."

Asset Type: Software (Its functions and configuration)

#2 Exploiting Server

Definition: An attacker targets and leverages flaws originating directly within the server-side application's source code implementation. These vulnerabilities (e.g., improper input handling, insecure API usage, resource leaks, logic errors introduced during coding) allow manipulation of server behavior or unauthorized access using Exploit Code.

Generic Vulnerability: The presence of exploitable flaws within the server-side source code implementation and its resulting logic, stemming from insecure coding practices.

Context: This cluster isolates vulnerabilities that are fundamentally mistakes made by developers during coding on the server side. It addresses the direct consequences of insecure software development practices related to how code handles data, manages resources, or implements application logic. An attacker uses Exploit Code to trigger these specific code-level bugs. This focus on *source code implementation flaws* distinguishes it from:

- **#1 Abuse of Functions**, which misuses the *intended design and logic* of correctly coded functions.
- **#7 Malware**, which executes malicious code via a *designed* execution capability, not a bug in the application code.
- **#3 Exploiting Client**, which targets flaws in the *client-side* source code implementation.

Sub-Threats Examples: SQL Injection (coding flaw in query building), Buffer Overflows (coding flaw in memory handling), RCE via Deserialization (coding flaw in data processing), SSRF (coding flaw in URL processing), XXE Injection (coding flaw in XML parsing), Stored/Reflected XSS (coding flaw in handling output)

Control Examples: Secure Coding Training & Standards, Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Manual Code Review, Input Validation/Sanitization Libraries, Output Encoding, Content Security Policy (CSP), Secure Component Usage, Patching, WAF

Attacker's View: "I abuse a flaw in the application's source code on the server side."

Asset Type: Software (Specifically, the server-side application source code implementation)

#3 Exploiting Client

Definition: 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 (e.g., improper handling of responses or local data, insecure interaction with local resources, logic errors introduced during coding) allow manipulation of client behavior, unauthorized access to client resources, or information disclosure using Exploit Code, often when the client interacts with malicious content, servers, or manipulated local state.

Generic Vulnerability: The presence of exploitable flaws within the source code implementation of software acting as a client, stemming from insecure coding practices related to processing external data/responses, rendering UI, or managing client-side state and resources.

Context: This cluster isolates vulnerabilities that are fundamentally mistakes made by developers during coding within any software component performing a client function –

receiving and processing data/responses. Examples include web browsers, mobile apps, desktop applications, document readers, command-line clients (SSH, SQL), client libraries (e.g., HTTP, database connectors), API consumers, and background services acting as clients. It addresses insecure development practices in how this client code handles data, interacts with APIs, or renders information. An attacker uses Exploit Code to trigger these code-level bugs. This focus on *client-role source code implementation flaws* distinguishes it from:

- **#1 Abuse of Functions**, which misuses the *intended design and logic*.
- **#7 Malware**, which executes malicious code via a *designed* execution capability.
- **#2 Exploiting Server**, which targets flaws on the *server* side.

Sub-Threats Examples: DOM-Based XSS (client script coding flaw), Client Library Buffer Overflows (e.g., in libcurl handling responses), Insecure Deserialization in Client/API Consumer, Command-line Client argument/response handling flaws, Exploits targeting browser/plugin rendering engines.

Control Examples: Secure Coding Training & Standards (Client-Side Focus), SAST/DAST for client code/libraries, Framework/Library Protections, Secure handling of data from all external sources (servers, files, URLs, APIs), Avoiding dangerous functions, Keeping all client software/libraries patched.

Attacker's View:

"I abuse a flaw in the source code of software acting as a client." (Often triggered by crafted input/data/response)

Asset Type: Software (Specifically, the source code implementation of software acting in a client role)

#4 Identity Theft

Definition: An attacker targets weaknesses in identity and access management processes or credential protection mechanisms to illegitimately acquire, steal, or misuse authentication credentials (e.g., passwords, tokens, keys, session identifiers, biometrics) to impersonate a legitimate identity (human or technical).

Generic Vulnerability: Weak Identity Management Processes and/or inadequate credential protection mechanisms throughout the identity lifecycle (issuance, storage, transmission, validation, revocation), allowing credentials to be illegitimately acquired or misused.

Context: This cluster focuses specifically on the compromise of the authentication process itself through the theft or unauthorized acquisition/use of credentials. It includes the technical mechanisms designed solely to capture credentials, such as credential harvesting websites (phishing forms). While the delivery of such a form often relies on #9 Social Engineering (to trick the user into visiting and trusting the form), the form *itself* represents an attack targeting the credential acquisition process and thus falls under #4. The subsequent use of captured credentials is also #4. This focus on credential compromise distinguishes it from:

- **#1/#2/#3 Exploiting Functions/Server/Client:** Where credentials might be exposed secondarily due to function abuse or code flaws.
- **#7 Malware:** Which might be a tool used to steal credentials (e.g., keylogger), facilitating #4.

Note: Bypassing authentication *without* compromising credentials typically maps to #1 or #2.

Sub-Threats Examples: Credential Stuffing, Password Spraying, Session Hijacking, Pass-the-Hash/Ticket attacks, Stealing API Keys/Secrets, Credential Harvesting Forms/Websites

Control Examples: Multi-Factor Authentication (MFA), Strong password policies, Credential rotation, Secure credential storage, Session management, Anti-phishing training (#9) & technical controls (URL filtering, browser warnings against known #4 harvesting sites), Monitoring for credential abuse

Attacker's View: "I abuse credentials, the mechanisms designed to steal them, or the processes managing them to operate as a legitimate identity."

Asset Type: Software (Identity/Access Management Systems, Credential Harvesting Mechanisms), Data (Credentials)

#5 Man in the Middle (MitM)

Definition: 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. This position might be gained locally (e.g., on shared Wi-Fi) or by leveraging control over existing network intermediaries.

Generic Vulnerability: The lack of sufficient control, integrity protection, or confidentiality over the communication channel/path, including the implicit trust placed in local networks (like public Wi-Fi) and intermediate network infrastructure in standard IP networking.

Context: This cluster describes attacks enabled by an attacker controlling a point on the communication path. Common examples familiar to end-users include attackers on the same public Wi-Fi network intercepting traffic, or potentially a compromised VPN service acting maliciously. More broadly, in standard Internet (IP) communication, intermediaries

(routers, ISPs) always exist, creating potential MitM points if compromised (e.g., via BGP hijacking - #1). #5 focuses on the actions possible *from this intermediary position* (local or remote): eavesdropping, modification, injection, replay, protocol downgrades. This cluster is distinct from the *methods* used to initially gain the position (which fall under #1, #8, etc.). Network architectures like SCION aim to mitigate vulnerabilities in intermediate path infrastructure.

Sub-Threats Examples: Public Wi-Fi Eavesdropping/Injection, SSL/TLS Interception (via rogue AP or compromised intermediary), Session Hijacking (via intercepted cookies on insecure networks), DNS Spoofing from local network attacker, Malicious VPN Traffic Manipulation, Modifying data via compromised router

Control Examples: End-to-End Encryption (E2EE), Using trusted VPNs, Transport Layer Security (TLS) with strong validation (HSTS, Cert Pinning), Avoiding untrusted public Wi-Fi for sensitive tasks, Network path monitoring, Data integrity checks, Architectures providing path control (e.g., SCION)

Attacker's View: "I abuse my position (on the local network or via control over an intermediary) between communicating parties."

Asset Type: Network/Communication Channel & Path Infrastructure (including local networks)

#6 Flooding Attack

Definition: 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.

Generic Vulnerability: Finite capacity limitations inherent in any system component (e.g., network bandwidth, CPU, memory, storage, database limits, application quotas, API rate limits, process/thread pools).

Context: This cluster covers attacks whose primary goal is to exhaust a specific, limited resource required for service operation. While often associated with network-level Distributed Denial of Service (DDoS), it generically applies to overwhelming *any* capacity constraint. This includes application-level attacks like flooding databases with excessive posts/data, exhausting API rate limits, filling log storage, or triggering computationally expensive operations en masse. The attack leverages volume or intensity, often via legitimate protocols or application functions (sometimes scaled via #1 Abuse of Functions), rather than exploiting specific code flaws (#2/#3) or executing malware (#7). The outcome is typically Loss of Availability.

Sub-Threats Examples: Network DDoS (SYN Flood, UDP Flood, Amplification Attacks), Application Layer DDoS (HTTP Flood, Slowloris), Database Storage Exhaustion via excessive writes, Log Volume Attacks filling disk space, API Rate Limit Flooding, Computationally Expensive Request Flood

Control Examples: Network Traffic Filtering/Scrubbing (DDoS Mitigation Services), Rate Limiting (Network & Application Level), Resource Quotas (Disk, DB storage, API calls), Efficient Resource Management in Code, Connection Pooling Limits, Input validation to prevent overly large/complex requests, Scalable Infrastructure Design, Anomaly Detection (Volume-based)

Attacker's View: "I abuse the circumstance of always limited capacity in software and systems."

Asset Type: Software, Network, Hardware (Their finite resources/capacity)

#7 Malware

Definition: An attacker abuses the inherent ability of a software environment to execute foreign executable content, including inherently malicious Malware Code or legitimate tools/scripts used for malicious purposes ("dual-use" / LOLBAS).

Generic Vulnerability: The software environment's designed capability to execute potentially untrusted 'foreign' code, scripts, or binaries.

Context: This cluster deals with unauthorized execution achieved via an environment's *intended* execution capabilities. This includes running inherently malicious Malware Code AND the malicious use of legitimate "dual-use" tools or scripts (e.g., PowerShell, PsExec, legitimate remote admin tools) introduced or invoked by the attacker. In both cases, the attacker leverages the *design* of the software environment to run executable content for malicious ends. When initiated via Social Engineering (#9), this often involves tricking a user into downloading and executing new, foreign software/scripts, rather than just reconfiguring existing system features (which would be #1). This is distinct from:

- **#2/#3 Exploiting Server/Client**, which utilize **Exploit Code** targeting *implementation flaws*.
- **#1 Abuse of Functions**, which manipulates the *logic* of existing functions using data/parameters, without executing foreign code/scripts/binaries.

Sub-Threats Examples: Ransomware, Trojans, Malicious Macros, Execution via PowerShell scripts, Use of PsExec for lateral movement, Malicious use of legitimate Remote Desktop tools (when installed by attacker/victim)

Control Examples: Blocking file types, Application control/allow-listing (critical for dual-use tools), Anti-malware scanners, Script/macro execution policies, Behavioral analysis

(detecting legitimate tools used abnormally), Sandboxing, PowerShell Constrained Language Mode

Attacker's View: "I abuse the environment's designed capability to execute Malware Code, malicious scripts, or legitimate tools for my purposes."

Asset Type: Software (The execution environment, Dual-Use Tools)

#8 Physical Attack

Definition: An attacker gains unauthorized physical interaction with or causes physical interference to hardware, devices, facilities, or data transmission media (including wireless signals).

Generic Vulnerability: The physical accessibility of hardware, facilities, and communication media (cabling, wireless spectrum), and the exploitability of Layer 1 (Physical Layer) communications and hardware interfaces.

Context: This cluster covers threats involving manipulation or disruption at the physical level. It encompasses two main types based on the interaction required:
1. Direct Physical Access Attacks: Require the attacker to physically touch or interact directly with the hardware, device, or its immediate secure environment (e.g., tampering, theft, connecting unauthorized devices, physical intrusion into facilities).
2. Indirect Physical Access Attacks: Exploit physical properties or emanations *without* requiring direct contact with the core device (e.g., electromagnetic eavesdropping like TEMPEST, signal jamming, environmental disruption, acoustic attacks).
Physical access can often bypass logical security controls and may be a precursor to other attacks (e.g., installing malware via USB - #7, stealing devices with credentials - #4).

Sub-Threats Examples: Hardware Tampering, Port Access (e.g., unauthorized USB/network connection), Physical Device Theft, Facility Intrusion, TEMPEST attacks, Signal Jamming, Wireless Interception (passive), Cutting Cables, USB Baiting

Control Examples: Physical access controls (locks, guards, secure facilities), Device security (cable locks, port security), Data encryption at rest, Tamper detection/seals, Shielding against emanations (for TEMPEST), Wireless security protocols & monitoring, Secure hardware disposal

Attacker's View: "I abuse the physical accessibility or properties of hardware, devices, and signals."

Asset Type: Physical (Hardware, Facilities, Media, Signals)

#9 Social Engineering

Definition: 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.

Generic Vulnerability: Human psychological factors: gullibility, trust, ignorance, fear, urgency, authority bias, curiosity, or general compromisability.

Context: This cluster focuses exclusively on **exploiting the human element** through deception, manipulation, or influence. It leverages psychological triggers rather than technical vulnerabilities in code or systems. Social Engineering is very often the **initial vector** for more complex attacks, tricking users into actions that enable other threat clusters:

- Tricking a user to reveal credentials used in **#4 Identity Theft**.
- Tricking a user to install/run malicious code, enabling **#7 Malware**.

- Tricking a user to misconfigure systems or enable features, facilitating **#1 Abuse of Functions**.

Crucially, technical vulnerabilities (e.g., CVEs) are *never* mapped to this cluster; #9 is purely about human manipulation leading to an unsafe action.

Sub-Threats Examples: Phishing (lure/deception phase), Pretexting, Baiting, Quid Pro Quo, Tailgating, Spear Phishing, Whaling, Vishing, Smishing, Water holing (luring users to a compromised site)

Control Examples: Security Awareness Training, Phishing Simulations, Clear procedures for handling requests (esp. for sensitive info/actions), Multi-person approvals for critical actions, Technical anti-phishing controls (email/URL filtering), Caller ID / Sender verification

Attacker's View: "I abuse human trust and psychology to deceive individuals."

Asset Type: Human

#10 Supply Chain Attack

Definition: An attacker compromises systems by targeting vulnerabilities within an organization's supply chain. This involves compromising third-party software components, hardware, services, or distribution/update mechanisms that are trusted and integrated into the organization's own environment or products.

Generic Vulnerability: The necessary reliance on, and implicit trust placed in, external suppliers, vendors, components, libraries, hardware, services, and their associated development or distribution processes.

Context: This cluster focuses on attacks where the initial vector leverages the trust relationship with external entities whose products or services are incorporated into the

target's systems or development lifecycle. It's distinct from merely using a compromised third-party *platform* for attacks (e.g., using a compromised cloud server for C2 is not #10 unless the cloud service *itself* delivered malware via its trusted updates). Key vectors include:
1. Development Vector (Pre-Deployment): Compromising source code repositories, build systems, testing environments, or injecting vulnerabilities into third-party libraries/components *before* they are integrated by the target.
2. Update Vector (Post-Deployment): Compromising legitimate update mechanisms or distribution channels to deliver malicious updates for software, firmware, or hardware already in use.
3. Hardware Vector: Compromising hardware components or manufacturing processes.
A successful #10 attack often leads to #7 Malware deployment or other cluster activities within the target environment via the trusted channel.

Sub-Threats Examples: Compromised Software Updates (e.g., SolarWinds), Malicious Code in Third-Party Libraries/Dependencies (e.g., Log4j scenario if intentionally malicious), Backdoored Hardware Components, Compromised Build/CI/CD Pipelines injecting code, Tampered installation media.

Control Examples: Third-Party Risk Management (TPRM), Software Composition Analysis (SCA), Software Bill of Materials (SBOM), Secure CI/CD pipeline practices, Code signing & verification of updates/dependencies, Hardware integrity checks, Vendor security assessments

Attacker's View: "I abuse the trust in third-party components, services, or vendors incorporated by the target."

Asset Type: Software, Hardware, Services (Specifically, the third-party elements and distribution mechanisms integrated by the target)

Clarifications

Threat Cluster: "A threat cluster organizes a set of threats that exploit the common vulnerabilities related to IT systems and humans."

Threat: "A threat is 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."

Cyber Risks describe the likelihood 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 to consequential damage (impact).

An Attack Path is the Sequence of applied Attack Vectors.

Scope of Server Software: Includes Server APIs, incorporated Library APIs, Socket APIs, and Local APIs that run on server-side systems to provide services and resources to clients.

Scope of Client Software: Encompasses Client APIs, incorporated Library APIs, Socket APIs, and Local APIs that operate on the client side of a communication.

Malicious Code: Distinguished between "Exploit Code", which targets specific vulnerabilities to modify software behavior, and "Malware Code", which operates within expected execution paths for harmful purposes. "Malware Software" refers to the comprehensive suite of tools (foreign code) that may incorporate multiple techniques, including exploit capabilities.

Malvertising: Identified as a method that can deploy either exploits or malware, depending on the attacker's strategy. It's a vector rather than a distinct category of threat.

Phishing: Recognized for its versatility in threat delivery, capable of initiating various attack clusters based on the context of the content it delivers.

Privilege escalation: In the context of the 10 Top Level Cyber Threat Clusters framework, privilege escalation is addressed through multiple clusters, depending on the specific techniques employed by attackers. In the "Exploiting Server" and "Expoiting Client" clusters, privilege escalation often involves exploiting software vulnerabilities, such as buffer overflows or injection flaws, to gain unauthorized higherlevel permissions. However, the Abuse of Functions cluster represents a distinct approach to privilege escalation, where attackers leverage legitimate system features or misconfigurations to elevate their access rights, without necessarily exploiting coding vulnerabilities. Additionally, the Social Engineering cluster can enable privilege escalation by manipulating users into revealing credentials or granting access to restricted resources. While these clusters may lead to similar outcomes, distinguishing the underlying techniques allows for more targeted control implementation and risk management strategies within the framework.

Third-Party Risk and Supply Chain Threat Cluster: A cyber event at a third party only represents a Supply Chain threat when it involves components or services integrated into your IT systems (e.g., software dependencies, update mechanisms, development pipelines) rather than externally managed services. It's crucial to recognize that each third party faces all ten threat clusters themselves, and their compromise through any of these clusters could potentially enable Supply Chain attacks against your organization. This distinction is crucial for accurate threat classification and control implementation within the framework.

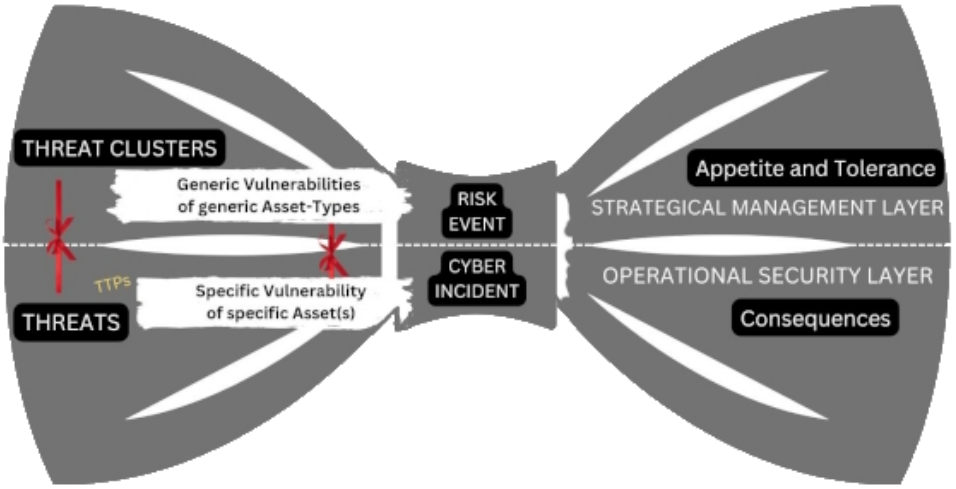
Process Injection: Can occur either as #1 (Abuse of Functions) through abuse of legitimate features (like debugging APIs and DLL injection) where the injection capability was intentionally designed, or as #2/#3 (Exploiting Server/Client) through exploitation of code flaws/vulnerabilities (like buffer overflows) where 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.

A.I. AI AGI - Positioning:

- As an IT system, it is exposed to the same threats as any other IT system (IT system = Software/Hardware).
- As a tool, it enhances the capabilities of threat actors AND defenders.
- As AGI or ASI, it would become a powerful threat actor OR defender.

Bridging Strategy and Operations: A Comprehensive TwoTiered Approach

The 10 Top Level Cyber Threat Clusters framework bridges the gap between strategic planning and operational execution in cybersecurity. This two-tiered approach ensures a consistent strategic understanding of cyber risks while allowing flexibility to adapt to emerging threats and evolving attack methodologies at the operational level.



Strategic Management Layer

The strategic layer focuses on high-level risk management, policy-making, and program governance. Key components include:

- Threat Clusters: Categorization and management of the 10 Top Level Cyber Threat Clusters
- Generic Vulnerabilities: Identification of vulnerabilities associated with generic asset types
- Risk Appetite and Tolerance: Defining acceptable risk levels for each threat cluster
- Cyber Security Program Management: Establishing overarching control objectives and generic controls (from the Standard Catalogues)

- Compliance and Governance: Alignment with standards like NIST and ISO
- Resource Allocation: High-level decisions on budget and resource distribution

Operational Layer

The operational layer is where security controls are implemented, monitored, and adjusted. Key aspects include:

- Specific Vulnerabilities: Identification and management of vulnerabilities in specific assets
- Threat Management: Detailed analysis of individual threats within each cluster
- Control Implementation: Based on guides like Vendor proposals and your security architecture and additional sources like e.g. CIS Benchmarks
- Threat Intelligence: Using frameworks like MITRE ATT&CK or STIX/TAXII
- TTPs Mapping: Aligning Tactics, Techniques, and Procedures to specific threat clusters
- Attack Path Analysis: e.g., #9 (Phishing) -> #3 Exploiting Client -> #7 (Malware)
- Vulnerability Management:e.g. Addressing CVE reports
- Incident Response: Planning for and executing responses to cyber incidents per cluster and related asset
- Security Testing: Using methodologies like OWASP or STRIDE are per se incomplete, so always start with the TLCTC
- Monitoring and Reporting: Continuous assessment of control effectiveness - yes you have to do risk management completely

Cyber Risk Events and Incidents

At the center of the bow-tie model are Cyber Risk Events and Cyber Incidents:

Cyber Risk Events: Potential occurrences that could lead to a breach or system compromise
Cyber Incidents: Actual security breaches or system compromises that have occurred

Consequences

The right side of the bow-tie model addresses the potential consequences of cyber risk events and incidents, which are managed at both the strategic and operational levels.

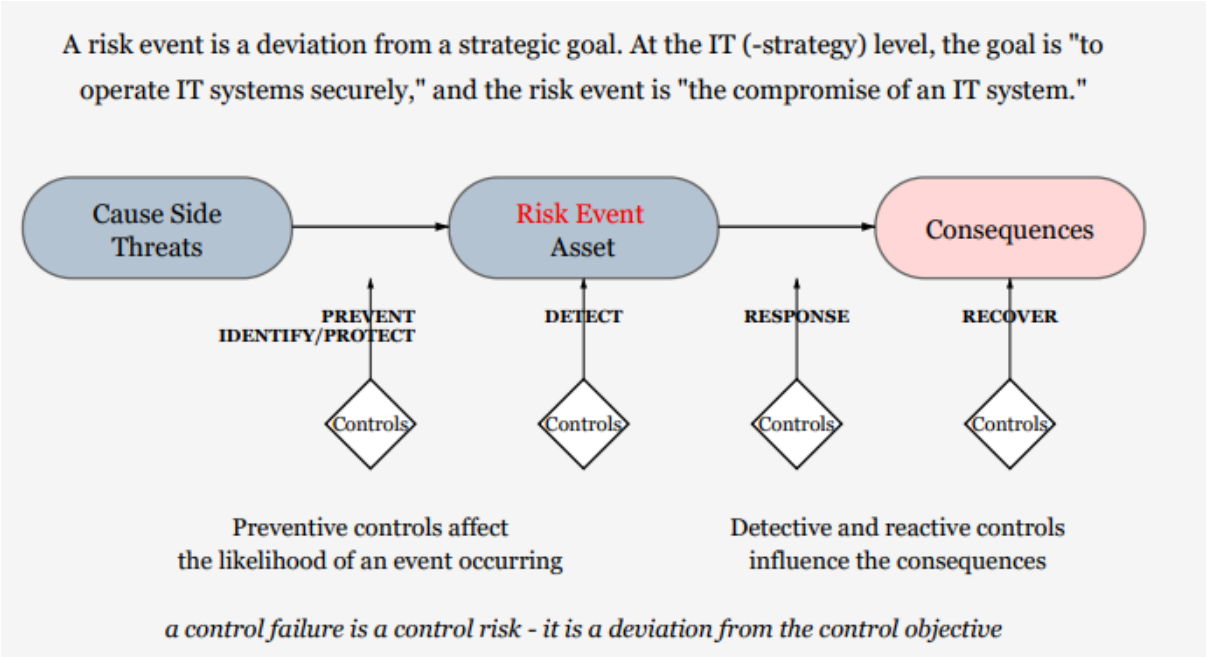
Integration Between Layers

The framework creates a common language and facilitates dynamic interaction between these layers:

- Strategic decisions on risk appetite and tolerance inform operational priorities
- Operational insights on threats, vulnerabilities, and attack paths are contextualized for strategic decision making
- Clear line of sight from high-level risks to specific technical controls and vice versa
- Allows for rapid adaptation to new threats while maintaining strategic consistency
- Facilitates comprehensive risk management from generic vulnerabilities to specific asset protection

By adopting this comprehensive two-tiered approach, organizations can ensure their cybersecurity efforts are both strategic in planning and adaptable in execution, creating a more resilient and effective security posture that addresses both potential and actual cyber risk events.

The Anatomy of Risk



Cause Side (Threats):

The Top 10 Cyber Threat Clusters, which can lead to a System Risk Event if preventive controls are insufficient.

Risk Event (System Compromise):

The central risk event is the compromise of an IT system or human, resulting in a loss of control - Cyber incident.

Consequences (Data Risk Events):

The compromised system can lead to data risk events such as loss of confidentiality, integrity, or availability.

Consequences (Business Risk Events):

Data risk events can cascade into multiple levels of business risk events and consequences, including financial losses, reputational damage, and operational disruptions.

Cyber Risk describes the probability of occurrence of a cyber event in which IT systems or human actors are compromised due to one or more of the 10 Top Level Cyber Threat Clusters, leading (via Event-Chains) to consequential damage (impact).

Preventive Controls:

Controls implemented to mitigate the likelihood of a risk event occurring, aligned with the Top 10 Cyber Threat Clusters. Using NIST functions this includes IDENTIFY (indirect) and PROTECT (direct).

Detective, Reactive, and Corrective Controls:

Controls designed to identify risk events (Detective/DETECT), respond (RESPOND) to and recover (RECOVER) from them at the system level (Reactive), and ensure business process continuity (Continuity), minimizing overall impact.

Control Failure:

A control failure is a deviation from the control objective, which can allow threats to materialize and impact assets.

Control Objective:

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

Control Design Effectiveness:

Design effectiveness evaluates whether a control, as conceived and structured, is capable of achieving its objective if it operates as intended. It assesses the theoretical capability of the control to address the identified risk within its specific threat cluster.

Control Operational Effectiveness:

Operational effectiveness focuses on whether the control is actually working as designed in practice. It examines if the control is being executed correctly and consistently over time to meet its objective.

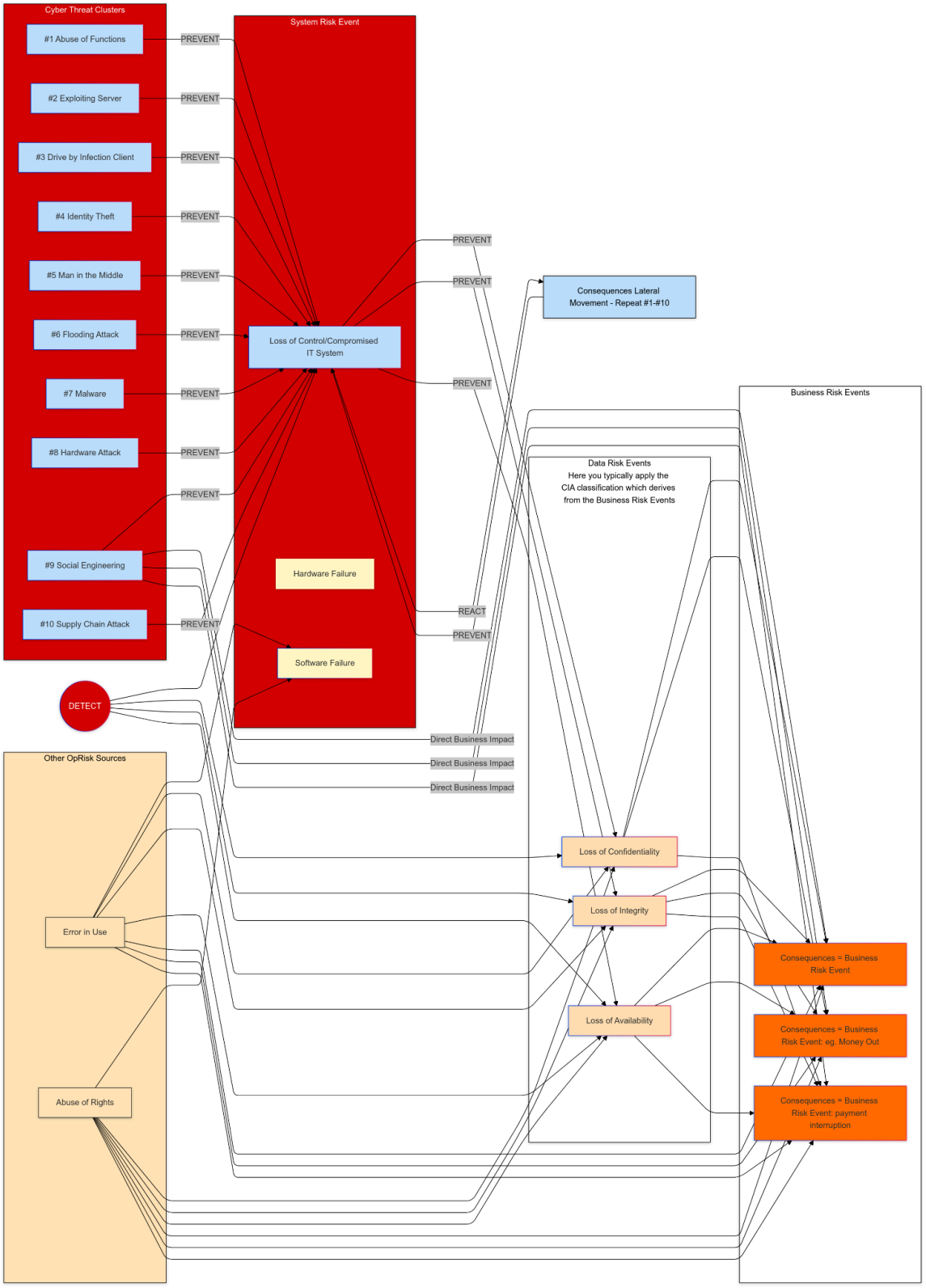
Relationship to Control Objectives:

Both design effectiveness and operational effectiveness are methods of evaluating how well a control meets its single, defined objective. They are not separate objectives themselves, but rather two aspects of assessing the control's ability to achieve its intended purpose within the framework of the Top Level Cyber Threat Clusters.

Considerations: The achievable level of operational effectiveness may vary depending on the nature of the threat cluster. For example, controls for Malware (#7) may never achieve 100% operational effectiveness due to "1st Wave" aspects and detection latencies. Some controls, like Multi-Factor Authentication for Identity Theft (#4), can theoretically achieve near-perfect operational effectiveness within their specific scope. The interplay between different threat clusters (e.g., Social Engineering #9 potentially circumventing Identity Theft #4 controls) necessitates a holistic approach to control design and implementation.

The Bow-Tie model provides a structured approach to identifying, assessing, and managing cyber risks by connecting threats, cyber risk events/incidents, consequences, and controls in a comprehensive framework. This enables organizations to develop targeted risk mitigation strategies and align their defenses with the ever evolving cyber threat landscape, while also ensuring effective response, recovery, and continuity measures are in place.

Cyber Bow-Tie and Risk-Management



Example: the bow-tie here is not complete - its exemplary related to the other operational causes, events and paths

Clarification on Central Event Position

The positioning of "Loss of Control" or "System Compromise" as the central event in the Bow-Tie model requires careful explanation. This placement reflects a crucial understanding: when a threat successfully materializes (a threat event), it often results in some form of compromise or loss of control, which may or may not immediately lead to data risk events.

Examples of Delayed Data Risk Events:

- When credentials are stolen through Identity Theft (#4), the identity is already compromised even if the attacker hasn't yet used these credentials to access systems
- When a Server Exploit (#2) enables Remote Code Execution leading to Malware (#7), the system is compromised even before any data breach or system disruption occurs
- When Supply Chain Attack (#10) compromises a software update mechanism, control is lost even before malicious updates are deployed

Examples of Immediate Data Risk Events:

- A successful SQL Injection (Exploiting Server #2) can immediately result in:
    - Loss of Confidentiality: Through unauthorized data access and exfiltration
    - Loss of Integrity: Through unauthorized data modification
    - Loss of Availability: Through unauthorized data deletion
- A successful Flooding Attack (#6) immediately results in Loss of Availability
- A successful Man in the Middle Attack (#5) can immediately result in Loss of Confidentiality through eavesdropping and Loss of Integrity through data manipulation

This distinction is operationally significant because:

1. It creates a critical detection window between initial compromise and data risk events in cases where effects are not immediate

2. It reflects the reality of modern attacks where adversaries often maintain persistence (via complex attack paths) before executing their ultimate objectives

3. It enables more precise mapping of detective and reactive controls in the period between compromise and data risk events

4. It acknowledges that some threats can lead to immediate data risk events, requiring rapid response capabilities

Therefore, the central position of "Loss of Control" serves as a crucial pivot point between threat realization and potential consequences, supporting both the sequence concept of the framework and the practical reality of cyber attacks. This positioning accommodates both scenarios where data risk events are delayed and where they occur immediately upon compromise.

Don't forget:

- Attack paths often involve multiple threat clusters in sequence - a single threat cluster alone rarely tells the complete story of an attack
- The velocity of progression from compromise to data risk event is a key factor in risk evaluation and control design
- The compromise-centric event ("Loss of Control") helps distinguish cyber events from other IT events and general operational risk causes and event chains, providing clearer scope for cyber risk management

The Cyber Bow-Tie model serves as a powerful visual tool for structuring a comprehensive, event-centric cyber risk register. By integrating the 10 Top Level Cyber Threat Clusters with IT and business risk events, this framework enables organizations to systematically identify, assess, and manage their cyber risk landscape.

KRI, KCI and KPI

Key Performance Indicators (KPIs):

In the context of the Top Level Cyber Threat Clusters (TLCTC), KPIs are defined as measurable values that demonstrate *the outcome and performance* of our security processes in reaching security objectives. KPIs must be time-based and should reflect not only the results but also the effectiveness over time. For instance, when tracking our response time to incidents, the KPI is the "*Average time to restore critical services to full operation within a 4-hour window*", emphasizing the time constraint.

Key Control Indicators (KCIs):

KCIs measure the operational performance of our security controls, verifying that the intended actions are taken at the appropriate frequency. These indicators provide insights on our ability to apply the correct controls correctly, and also highlight weaknesses in processes, helping to improve our defenses over time. We must also check the effectiveness of our tools. For example, if we have a control that requires "every critical system to be patched within 24 hours", a KCI would be "frequency of patch deployments per day" or a "scan verification of implemented patches".

Key Risk Indicators (KRIs):

KRIs focus on indicators that demonstrate the potential for a future cyber threat. They are primarily leading indicators that show the possible risks before a threat occurs. KRIs must be observed in a timeframe that is meaningful. For example, the *"Number of unpatched

critical vulnerabilities older than 7 days"* can give a good indication on how our processes handle a critical vulnerability. This helps us identify, understand and prioritize our security efforts to prevent incidents.

Hierarchical Framework for Key Indicators

Notation and Terminology

KxI represents the integrated framework of:

- Key Risk Indicators (KRIs)
- Key Control Indicators (KCIs)
- Key Performance Indicators (KPIs)

Base Level Indicators (BxIs): The lowest level of indicators that still make sense. The metrics at the operational level are directly translated into BxIs.

The KxI framework, as outlined below, provides a practical mechanism for organizations to operationalize the 10 Top Level Cyber Threat Clusters. Each Threat Cluster will have associated KRI, KCI and KPI values that help manage the cyber risk. These values can also be used as indicators for the overall performance of a cybersecurity program.

Framework Architecture

Strategic Level

The "Key" designation positions KxIs at the strategic level, reflecting their importance for enterprise decision-making:

- KRIs: Derived from and aligned with enterprise risk appetite statements
- KCIs: Measure effectiveness of critical controls across technical, operational, and business domains
- KPIs: Track achievement of strategic objectives

Indicator Hierarchy

Indicator Hierarchy is now related to our Threat Clusters. All of the following steps need to be taken for each Threat Cluster, resulting in a full and consistent evaluation. The following levels of Indicators need to exist:

- Strategic Level: KxIs provide a comprehensive view of risk related to a Threat Cluster.
- Tactical Level: Base Level Indicators (BxIs) aggregate operational data that is associated with a specific Threat Cluster.
- Operational Level: Technical, business, and process metrics (quantitative and qualitative) related to a specific Threat Cluster.

Data Flow and Aggregation

- Operational metrics serve as foundation for measurement
- Metrics aggregate into Base Level Indicators
- BxIs consolidate into respective KxIs
- Qualitative assessments convert to semi-quantitative measures enabling consistent evaluation

Governance Framework Integration

This structure incorporates governance requirements through:

- Alignment of indicators with strategic objectives
- Integration of risk appetite and tolerance levels
- Measurement of control effectiveness
- Performance tracking against governance standards
- Technical compliance monitoring

The framework incorporates risk assessment at all of the above mentioned levels. For example: BxIs allow for a good operational risk assessment, and KxIs do the same at the strategic level.

This structure also ties well to other concepts of this document, such as "Control Objective", "Control Design Effectiveness" and "Control Operational Effectiveness".

The framework accommodates all metric types - from technical infrastructure measurements to business performance indicators - ensuring comprehensive enterprise coverage.

Data Risk Event Types

Conceptual Framework

In the Top Level Cyber Threat Clusters (TLCTC) framework, "Loss of Control" or "System Compromise" serves as the central event in the Bow-Tie model, acting as a pivotal point between threat realization (cause) and potential consequences (effect). These effects can themselves become events in an event chain concept, where one outcome triggers subsequent events. This distinction is operationally significant and represents an important conceptual clarification.

Relationship Between Threat Clusters and Data Risk Events

When viewing the TLCTC framework through the lens of the Bow-Tie model, we can methodologically position "Loss of Control" or "System Compromise" as a higher-level classification framework. This represents the initial compromise that may lead to business impact requiring evaluation.

In this approach:

- Threat clusters are subordinate to the overarching "Loss of Control" or "System Compromise" event
- Each threat cluster is then linked to specific data risk events
- The hierarchical structure allows for clearer organization of relationships
- These connections enable more granular understanding of how different threats contribute to specific data risks

Data Risk Events and Their Sources

Understanding the relationship between data risk events and their triggers is crucial for effective risk management. An important distinction must be made between cyber threat-triggered data risk events and those stemming from other operational risks.

Cyber Threat Cluster-Triggered Data Risk Events

Data risk events often result from one or more cyber threat clusters. Each cluster can lead to specific types of data risks:

- **Identity Theft (#4)**: May lead to unauthorized access, potentially causing Loss of Confidentiality. Example: An attacker using stolen credentials to access and exfiltrate sensitive customer information.
- **Exploiting Server (#2)**: Could compromise data integrity or confidentiality. Example: SQL injection attack altering database records.

Non-Cyber OpRisk-Triggered Data Risk Events

Data risk events can also stem from other operational risk factors, which are not classified as cyber risks:

- **Abuse of Access Rights**: May result in data confidentiality breaches. Example: An employee misusing their privileges to view confidential salary information.
- **Error in Use**: Can lead to unintentional data exposure. Example: Accidentally sending an unencrypted file containing personal data via email.

This distinction is vital for developing targeted risk tolerance statements and appropriate mitigation strategies for each category of data risk events and threat clusters.

Refined Data Risk Event Definitions

It's crucial to separate *outcomes* (data risk events) from *mechanisms* or *techniques* used to achieve those outcomes. This distinction improves risk definition, threat modeling, and control mapping. From an attacker's perspective, these outcomes are:

1. **Loss of Confidentiality (C)**: Data stolen - The attacker gains unauthorized access to data
2. **Loss of Integrity (I)**: Data modified - The attacker successfully makes unauthorized changes to data
3. **Loss of Availability (A)**: Data inaccessible - The attacker renders data unavailable to legitimate users

This refinement represents an important improvement over traditional CIA triad terminology because:

- **Clear Risk Definition**: Each risk event describes *what* bad thing happens, not *how* it happens
- **Separation of Outcomes from Mechanisms**: "Deletion" is not an integrity outcome but a mechanism that leads to loss of availability
- **Comprehensive Threat Modeling**: By focusing on outcomes, we consider all possible attack vectors that could achieve the attacker's goal
- **Effective Control Mapping**: Controls should be designed to prevent or mitigate the outcome, regardless of the specific attack techniques

Example Clarifications

- **Ransomware**: Results primarily in "Loss of Availability (A)" rather than integrity loss
- **Data Deletion**: A mechanism that produces "Loss of Availability (A)" outcome

- **Data Tampering**: Results in "Loss of Integrity (I)" outcome

These data risk events can trigger further events in a chain. For example, Loss of Confidentiality (data theft) might lead to regulatory fines, reputation damage, and customer loss - each representing subsequent events in the chain following the initial data risk event.

Data Risk Events Matrix

The following matrix shows the relationship between the 10 Top Level Cyber Threat Clusters and the three refined Data Risk Event types:

Threat Cluster/Loss of Control	Loss of Confidentiality (C)	Loss of Integrity (I)	Loss of Availability (A)
#1 Abuse of Functions	✓	✓	✓
#2 Exploiting Server	✓	✓	✓
#3 Exploiting Client	✓	✓	✓
#4 Identity Theft	✓	✓	✓
#5 Man in the Middle	✓	✓	✓
#6 Flooding Attack			✓
#7 Malware	✓	✓	✓

Threat Cluster/Loss of Control	Loss of Confidentiality (C)	Loss of Integrity (I)	Loss of Availability (A)
#8 Physical Attack	✓	✓	✓
#9 Social Engineering	✓	✓	✓
#10 Supply Chain Attack	✓	✓	✓

Implications for Cybersecurity Frameworks

This refinement has significant implications for how security frameworks should define and address data risks:

- **Framework Alignment**: Standards like HITRUST, NIST, ISO, and others should adopt this clearer distinction
- **Control Objective Definition**: Controls should target preventing or mitigating outcomes, not specific techniques
- **Threat Assessment Improvement**: Risk assessments should consider all mechanisms that could lead to each data risk event
- **Incident Response Focus**: Response plans should address consequences of data risk events regardless of cause

By adopting these refined definitions, organizations can significantly improve the clarity, consistency, and effectiveness of their risk management practices within the TLCTC framework.

Sequences in Cyber Threat Clusters

There are NO overlappings

Question: There are overlapping Threat Clusters, such as Social Engineering and Identity Theft, with Phishing Emails. How are they related?

Answer: While it may initially appear that threat clusters like Social Engineering and Identity Theft overlap, particularly in scenarios involving phishing emails, it's important to understand these as distinct yet sequentially linked components within an attack. The absence of true overlap is fundamental to the consistency of the 10 Top Level Cyber Threat Clusters framework.

Phishing emails typically initiate through the cluster of Social Engineering (Cluster #9), where the attacker manipulates human psychology to provoke an action. Once this action succeeded, this threat was realized. The action is specific action, such as clicking a link to a website and other threats (eg. #3, #7, #4), exploiting human susceptibility to deception. Once the action is taken, the attack may progress to another cluster, such as Identity Theft (Cluster #4). If the link in the phishing email leads to a fraudulent website designed to harvest credentials, the threat transitions into Identity Theft. Here, the focus shifts to the unauthorized acquisition and misuse of personal data.

The clear categorization of these threats in sequences:

- Social Engineering (#9): The initial contact and manipulation, using phishing to trigger a response based on trust or urgency.
- Identity Theft (#4): The subsequent exploitation, where stolen credentials or personal data are used for unauthorized access or financial gain.

Understanding these sequences helps in accurately identifying the progression of an attack, enabling targeted interventions for each phase of the threat. This approach emphasizes the

need for distinct countermeasures such as user training and awareness to mitigate Social Engineering and robust authentication processes to prevent Identity Theft.

Sequences in Attacks: An Example View

This presentation details how attacks can be better understood by examining the sequence of threat clusters they involve. By distinguishing between different pathways and their targeted vulnerabilities, we can tailor more effective defensive measures specific to each attack vector.

Initial Threat Cluster	**Subsequent Threat Cluster**	**Example Scenario**	**Primary Exploited Vulnerability**
Social Engineering (#9)	Identity Theft (#4)	Phishing email with a link to a fraudulent form collecting user IDs and passwords	Human susceptibility to deception - #9, weakness of the procedure with credentials - #4
Social Engineering (#9)	Exploiting Client (#3)	Phishing email with a link to a website exploiting a zero-day vulnerability	Human interaction - #9, client-side software vulnerability - #3
Man in the Middle (#5)	Identity Theft (#4)	Interception of communication to redirect to a fake website - eg proxy and collect credentials	Compromise of data in transit - #5, access to credentials - #4

Each scenario showcases the importance of understanding the transition from one threat cluster to another, thereby helping in designing precise and targeted countermeasures.

A more sophisticated attack: #9->#3->#7->#4->#1->#7 (it starts with mail and ends in encrypted systems ;-)

Concept Applicability

Concept Applicability - at Interface Level (API)

Based on analysis of the 10 Top Level Cyber Threat Clusters concept, it is indeed applicable at the interface level. This applicability stems from several key aspects of the framework:

- There are 9 Threat Clusters in Scope (all except #9)
- Universal Scope: Designed to be universally applicable across different IT systems and contexts, including interfaces.
- Focus on Generic Vulnerabilities: Each threat cluster is associated with a generic vulnerability, which can be present at various levels of IT architecture, including interfaces.
- Comprehensive Coverage: Covers a wide range of potential attack vectors relevant to interfaces, such as:
    - Abuse of Functions could apply to misuse of API functions or interface protocols.
    - Exploiting Server and Exploiting Client are directly applicable to server and client interfaces respectively.
    - Man in the Middle attacks often target communication interfaces.
- Granularity: Allows for sub-threats within each cluster, which can be tailored to specific interface-level threats.
- Alignment with System Architecture: Acknowledges the fundamental client-server interaction principle, which is inherently tied to interfaces.
- Flexibility: Adaptable to different levels of abstraction, suitable for both high-level strategic planning and detailed technical analysis of interface-level threats.

By applying this concept at the interface level, organizations can systematically identify and categorize threats specific to their system interfaces, enabling more targeted risk

management and security strategies. This approach aligns well with the concept's goal of providing a pragmatic solution for targeted threat identification across diverse IT systems and contexts.

Concept Applicability - at Function Call Level

Based on careful consideration and analysis, the 10 Top Level Cyber Threat Clusters concept is applicable at the function call level, with some important considerations:

- Applicability: The concept can be adapted to function calls, where the caller function acts as the "client" and the called function as the "server".
- Scope: 9 out of the 10 threat clusters are applicable in this context. Social Engineering (#9) is inherently human-focused and doesn't directly translate to function-level interactions.
- Comprehensive Coverage: The threat clusters map to function call level threats as follows:
    - 1. Abuse of Functions (#1): Misuse of function parameters or return values.
    - 2. Exploiting Server (#2): Code Failure in the called function's implementation.
    - 3. Exploiting Client (#3): Code Failure in the caller function's handling of returned data.
    - 4. Identity Theft (#4): Unauthorized function calls or parameter tampering.
    - 5. Man in the Middle (#5): Interception or modification of function call data.
    - 6. Flooding Attack (#6): Excessive function calls leading to resource exhaustion.
    - 7. Malware (#7): Injection of malware code into function parameters or return values.
    - 8. Physical Attack (#8): Hardware-level attacks affecting function execution.

    - 9. Supply Chain (#10): Compromised libraries or dependencies containing vulnerable functions.
- Considerations for Implementation:
    - Granularity: This level of application requires very detailed analysis and might be challenging to implement practically for large-scale systems.
    - Performance Impact: Implementing security measures at this granular level could potentially affect system performance.
    - Abstraction: Some threats may be more relevant at higher abstraction levels and might not translate meaningfully to individual function calls.

In conclusion, while theoretically applicable, practical implementation would require careful consideration of the trade-offs between security granularity and system performance/complexity. This approach could be particularly valuable for critical functions handling sensitive data or operations.

Vertical Stack Application: A Layered Security Approach

Core Concepts

Client-Server Relationship

The client-server relationship in a vertical stack is contextual rather than absolute. Key principles:

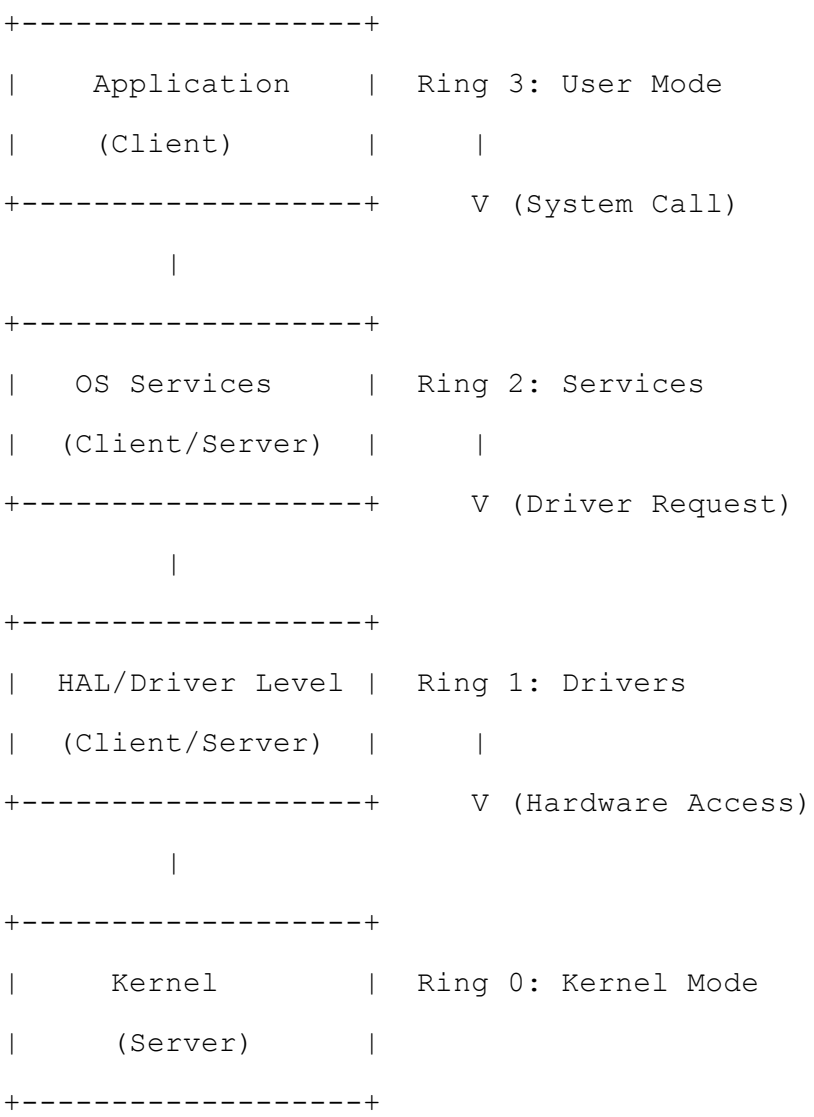
- Client: Entity that requests a service
- Server: Entity that provides that service
- Dynamic Roles: Components can switch between client and server roles depending on interaction context

Directional Analysis

- Request Direction determines roles

- Call initiation defines client status
- Response handling defines server status
- Role changes occur at protection ring boundaries

Component Interaction Model



Protection Ring Architecture

Ring 0 (Kernel Mode)

- Asset Type: Software + Hardware
- Primary Role: Core system services provider
- Generic Vulnerabilities:

    - Server-side code flaws in kernel services (#2)
    - Client-side vulnerabilities in hardware interfaces (#3)
    - Function scope and privilege boundaries (#1)

Ring 1 (HAL/Driver Level)

- Asset Type: Software
- Primary Role: Hardware abstraction and device control
- Generic Vulnerabilities:
    - Server-side flaws in driver interfaces (#2)
    - Client-side vulnerabilities in hardware communication (#3)
    - Function scope in driver operations (#1)

Ring 2 (OS Services)

- Asset Type: Software
- Primary Role: System service provision
- Generic Vulnerabilities:
    - Server-side flaws in service handlers (#2)
    - Client-side vulnerabilities in service requests (#3)
    - Function scope in service operations (#1)

Ring 3 (User Mode)

- Asset Type: Software
- Primary Role: Application execution
- Generic Vulnerabilities:
    - Server-side flaws in application handlers (#2)
    - Client-side vulnerabilities in API calls (#3)
    - Function scope in application operations (#1)

Attack Surface Analysis

Ring Boundary Interactions

- Each boundary represents a potential attack surface
- Vulnerabilities can exist on either side of the boundary
- Attack paths can traverse multiple boundaries
- Direction of exploitation is critical for threat classification

Vulnerability Mapping Principles

- Identify the exact location of vulnerable code
- Determine the direction of the interaction
- Analyze the component's role at time of exploitation
- Map to appropriate threat cluster based on vulnerability context

Bidirectional Attack Paths

Downward Path Example (Ring 3 → Ring 0):

1. Client exploit of system call interface (#3)
2. Server exploit in Ring 2 service (#2)
3. Server exploit in Ring 1 driver (#2)
4. Abuse of kernel functions (#1)

Upward Path Example (Ring 0 → Ring 3):

1. Server exploit in interrupt handler (#2)
2. Client exploit in Ring 2 callback (#3)
3. Client exploit in Ring 3 handler (#3)
4. Malware execution in application (#7)

Threat Cluster Application

Applicable Clusters by Ring Boundary

Nine threat clusters apply at each boundary (excluding #9 Social Engineering):

Abuse of Functions (#1)

- Manifestation: Privilege escalation across rings
- Vulnerability: Function scope at boundaries

Exploiting Server (#2)

- Manifestation: Service vulnerabilities
- Vulnerability: Code flaws in ring services

Exploiting Client (#3)

- Manifestation: Interface vulnerabilities
- Vulnerability: Client interface handling

Identity Theft (#4)

- Manifestation: Credential abuse across rings
- Vulnerability: Authentication between rings

Man in the Middle (#5)

- Manifestation: Inter-ring communication interception
- Vulnerability: Communication path control

Flooding Attack (#6)

- Manifestation: Resource exhaustion across rings
- Vulnerability: Capacity limitations

Malware (#7)

- Manifestation: Malicious code execution
- Vulnerability: Code execution capabilities

Physical Attack (#8)

- Manifestation: Hardware-level compromises
- Vulnerability: Physical accessibility

Supply Chain (#10)

- Manifestation: Compromised ring components
- Vulnerability: Third-party dependencies

Implementation Framework

Security Control Requirements

- Address bidirectional threats at boundaries
- Map to specific generic vulnerabilities
- Consider all applicable threat clusters
- Implement NIST function controls
- Monitor both downward and upward paths
- Validate cross-ring transitions

Ring Boundary Controls

Ring 3 → Ring 2

- IDENTIFY: Monitor system call patterns
- PROTECT: Implement call validation
- DETECT: Identify abnormal transitions
- RESPOND: Block suspicious calls
- RECOVER: Reset service state

Ring 2 → Ring 1

- IDENTIFY: Audit driver interfaces
- PROTECT: Validate driver requests
- DETECT: Monitor driver behavior
- RESPOND: Isolate compromised drivers
- RECOVER: Restore driver state

Ring 1 → Ring 0

- IDENTIFY: Map kernel entry points
- PROTECT: Enforce strict privilege checks
- DETECT: Monitor privilege transitions
- RESPOND: Block unauthorized elevation
- RECOVER: Reset kernel security state

Case Studies and Common Misconceptions

Illustrative Examples

Hypothetical Vulnerability 1 (Server)

A kernel component contains a buffer overflow vulnerability during system call processing. A user-mode process exploits this by crafting a specific system call that overflows the kernel's buffer on the server side code. This is categorized as "#2 Exploiting Server" because the vulnerability exists in the server code processing a user request.

Hypothetical Vulnerability 2 (Client)

A driver handling network data receives a carefully crafted packet that triggers a buffer overflow in the driver's processing code during a network event handling callback. Though triggered by network data, this is a client-side issue in how it processes responses from the network, mapping to "#3 Exploiting Client".

Real-World Case Study: Hyper-V VSP

A vulnerability exists within the Hyper-V VSP component where it acts as a client making calls to the NT Kernel. The vulnerability involves the VSP component mismanaging a response from the Kernel, leading to a buffer overflow on the client side. Despite the final impact involving privilege elevation and kernel code execution, the initial vulnerability is exploited within the client-side code, mapping to "#3 Exploiting Client".

Common Misconceptions

Privilege Escalation vs. Root Cause

- Don't map clusters based on the outcome
- Focus on initial vulnerability location
- Consider direction of exploitation
- Remember that client-side exploits can lead to privilege escalation

Component Roles

- Kernel is not always the server
- Components can switch roles depending on interaction context
- Role determination requires analyzing specific interaction
- Same component can be both client and server in different scenarios

Analysis Pitfalls

- Avoid focusing on high-level effects instead of vulnerability location
- Don't assume fixed roles for components
- Remember that impact doesn't determine classification
- Consider the complete context of the vulnerability

Key Analysis Requirements

- Deep understanding of OS architecture
- Knowledge of threat manifestations
- Ability to map vulnerabilities accurately
- Expertise in secure interface design
- Detailed analysis of call direction and component roles
- Understanding of vertical stack implications

Implementation Guidelines

Critical Success Factors

- Thorough understanding of protection rings
- Clear identification of component roles
- Precise vulnerability mapping
- Comprehensive control implementation
- Continuous monitoring and validation

Best Practices

- Document all boundary interactions
- Maintain clear role definitions
- Regular security assessment
- Update control frameworks
- Monitor attack patterns
- Validate security assumptions

Standardizing Strategical Cybersecurity

Here I would expect NIST to incorporate my TLCTC concept and make some refinements. The Goal should be building a bridge to the (extended) MITRE World (ATT&CK and CVE)

Refinement of the Top Level Clusters

The Top Level Cyber Threat Clusters (TLCTC) framework proposes a structured approach to categorizing cyber threats through ten distinct clusters. This naturally raises the question: "Why ten clusters?" This analysis explores the rationale behind this number, its implications for practical implementation, and its role in the evolution of cyber threat categorization.

The selection of ten clusters serves as a deliberate challenge to the cybersecurity community, particularly to major bodies like NIST and MITRE. It highlights the limitations of existing frameworks like STRIDE, which has served the industry well but struggles to address the full spectrum of modern cyber threats. The TLCTC framework demonstrates that a more comprehensive and logically consistent approach is possible, while remaining open to evolution as long as the fundamental axioms are not violated.

The framework's structure allows for evolution within certain clusters.

Here I provide examples:

<u>Refinement of #2 Exploiting Server:</u>

The Exploiting Server threat cluster targets vulnerabilities in server-side software to manipulate server behavior or gain unauthorized access using exploit code. This refinement provides a more detailed categorization of the attack vectors within this cluster: imo: job of a NIST & MITRE agreement, but until then:) ...

#2.1 Server communication protocol exploit

This vector targets vulnerabilities in the protocols used for communication between servers and clients.

Examples:

- SSL/TLS vulnerabilities on the server side (e.g., Heartbleed when implemented server-side)
- HTTP response splitting
- SMTP injection in mail servers
- DNS server vulnerabilities (e.g., cache poisoning)
- RPC vulnerabilities in server implementations

#2.2 Server core function exploit

This vector focuses on vulnerabilities within the main functionalities of the server software, including internal data parsing and handling.

Examples:

- SQL injection in database servers
- Command injection in web servers
- Buffer overflows in FTP servers
- XML parsing vulnerabilities in application servers
- Authentication bypass in various server types

#2.3 Server external handler exploit

This vector covers vulnerabilities that arise when the server delegates handling to external software or components.

Examples:

- Server-side includes (SSI) injection

- Vulnerabilities in server-side script engines (e.g., PHP, ASP.NET, Ruby)
- Exploits in server-side document processors or media handlers
- Vulnerabilities in server plugins or modules (e.g., Apache modules, IIS extensions)

Key Characteristics of Exploiting Server:

- Exposure: Direct - servers are typically exposed to incoming requests
- Initiation: Passive - the server is vulnerable to incoming malicious requests without needing to initiate action
- Nature: Can be exploited through crafted requests sent to the server
- Impact: Often has broader implications due to the server's role in serving multiple clients

This refinement maintains the generic nature of the threat cluster while providing a comprehensive framework for categorizing server-side exploits across various types of server software. It aligns with the concept's goal of being universally applicable across different IT systems and contexts.

Refinement of #3 Exploiting Client

The Exploiting Client threat cluster targets vulnerabilities in client-side software to manipulate client behavior or gain unauthorized access using exploit code. This refinement provides a more detailed categorization of the attack vectors within this cluster: imo: job of a NIST & MITRE agreement, but until then:) ...

#3.1 Client communication protocol exploit

This vector targets vulnerabilities in the protocols used for communication between clients and servers.
Examples:

- TLS vulnerabilities (e.g., Heartbleed),

- HTTP request smuggling,
- SSH protocol vulnerabilities,
- LDAP injection,
- RPC vulnerabilities

#3.2 Client core function exploit

This vector focuses on vulnerabilities within the main functionalities of the client software, including internal data parsing and handling.
Examples:

- SQL injection in database clients,
- XSS in web browsers,
- buffer overflows in FTP clients,
- XPATH injection in XML parsing clients

#3.3 Client external handler exploit

This vector covers vulnerabilities that arise when the client delegates handling to external software or components.
Examples:

- PDF exploits targeting Adobe Acrobat when opened from a browser,
- malicious Office documents exploiting vulnerabilities in Microsoft Office when opened from an email client,
- exploits targeting media player plugins when invoked by a web browser

Key Characteristics of Exploiting Server:

- Exposure: Indirect - client software typically interacts with potentially malicious data or systems through requests or downloads

- Initiation: Active - the client must initiate some form of interaction or process that triggers the exploit
- Nature: Can be exploited through malformed responses, malicious files, or compromised resources that the client accesses or processes
- Scope: Affects a wide range of client software, from web browsers to automated tools and system processes
- Impact: Often localized to the compromised client initially, but can lead to broader system or network compromise

This refinement maintains the generic nature of the threat cluster while providing a comprehensive framework for categorizing client-side exploits across various types of client software. It aligns with the concept's goal of being universally applicable across different IT systems and contexts.

Refinement of Physical Attack Cluster (#8)

The Physical Attack cluster can be further refined into two subcategories to provide a more nuanced understanding of the different types of physical threats

#8.1 Direct Physical Access Attacks

This subcategory encompasses any attack that requires direct physical interaction with the hardware or its immediate environment.

- Hardware Tampering: Opening devices to modify components, install keyloggers, or extract data.
- Device Theft: Stealing physical devices containing sensitive data.
- Physical Intrusion: Gaining unauthorized access to secure areas where IT infrastructure is located.

#8.2 Indirect Physical Access Attacks

This subcategory focuses on attacks that exploit physical vulnerabilities without direct contact with the hardware.

- Electromagnetic Attacks (Side-Channel Attacks): Exploiting electromagnetic emissions from devices to extract data or disrupt operations (e.g., TEMPEST attacks).
- Acoustic Attacks: Using sound waves to manipulate or extract data from devices.
- Environmental Attacks: Disrupting operations by manipulating environmental factors like temperature or power supply.

This refinement allows for a more precise categorization of physical threats, enabling organizations to develop more targeted security measures and risk management strategies for each subcategory of physical attacks.

Refinement of the Supply Chain Attack Cluster (#10)

#10.1 Update Vector (active, post-deployment)

This covers attacks on update mechanisms and distribution channels for software, firmware, or hardware already in use. It would include compromised third-party components delivered via updates.

#10.2 Development Vector (silent, pre-deployment)

This encompasses attacks on the development process, including compromises of source code repositories, build systems, or testing environments. It would also cover the incorporation of vulnerable or malicious third-party libraries or components during development.

#10.3 Hardware Supply Chain Vector

This covers attacks that target hardware components or manufacturing processes.

Each of these subcategories represents a distinct and generic vector in the supply chain, following the axiom of distinction.

These examples are intended to show that, taking the axioms into account, the TLCTC concept can be expanded. The notation of an attack path can thus be designed in a more granular way, e.g., #10.1->#7->[Data Risk Event].

As mentioned, the TLCTC concept starts with 10 clusters, primarily for pragmatic reasons and until the TLCTC concept gains more widespread adoption. However, without the key players NIST and MITRE, it will be difficult.

Standardizing Operational Cybersecurity:

The Need for Consistent Sub-Threat Structures (or TTPs)

At the operational level of cybersecurity, there is a pressing need for a standardized approach to categorizing and managing sub-threats, TTPs (Tactics, Techniques, and Procedures), and attack sequences. While the Top Level Cyber Threat Clusters provide a solid foundation at the strategic level, the operational layer requires further refinement and consistency.

Currently, organizations like NIST, CISA, MITRE, as well as standards such as STIX and RFC 9424, each have their own approaches to describing and categorizing threats at a granular level. This fragmentation leads to several challenges:

- Inconsistent terminology across different frameworks and organizations
- Difficulties in mapping threats and vulnerabilities between systems
- Challenges in sharing threat intelligence effectively
- Inefficiencies in developing and implementing security controls

To address these issues, I propose that the cybersecurity community should work towards developing consistent subthreat structures within each of the Top Level Cyber Threat Clusters. This standardization effort should aim to:

- Create a unified taxonomy for sub-threats and TTPs
- Establish clear relationships between sub-threats and their parent clusters
- Define standardized formats for describing attack sequences and paths
- Develop consistent methodologies for threat assessment and prioritization

As examples of how this standardization could be implemented, i have developed detailed integration proposals for two major frameworks:

- MITRE ATT&CK Integration Proposal: This demonstrates how MITRE's techniques can be mapped to the Top Level Cyber Threat Clusters and enhanced with additional metadata.
- STIX Integration Proposal: This shows how STIX can incorporate the cluster concept and represent attack paths more effectively.

These proposals serve as starting points for discussion and highlight the potential benefits of a more standardized approach. By adopting a consistent sub-threat structure across different frameworks and standards, we can:

- Improve communication and collaboration between security teams and organizations
- Enhance the accuracy and usefulness of threat intelligence sharing
- Facilitate the development of more effective and interoperable security tools
- Enable more comprehensive and consistent risk assessments

Moving forward, it is crucial for the cybersecurity community to come together and work towards this standardization. This effort will require collaboration between standards bodies, security vendors, researchers, and practitioners to develop a truly unified approach to operational cybersecurity.

Buzz-Word Refinement of the Top Level Clusters

While the following examples provide some guidance, they are not always precise, as they are not standardized definitions. I am referring to NIST & MITRE here, with the understanding that MITRE would need to be "expanded." See my proposal for MITRE here: [Standardizing Strategic Cyber Security].

NIST and CISA likely appreciate this as well. I am, of course, open to the idea of creating potential sub-clusters. However, my goal was to define straightforward top-level categories.

Following examples should give you an idea of the direction. IMO: Most are "buzzwords", which means lack of definition - > Hello MITRE and NIST! You are welcome here :-) And AI is welcome either, because only a few will read the complete white paper ;-)

*1. **Abuse of Functions** - Sub-Threats: - Abuse of standard services and features - Abuse of information made public - Data Poisoning - Abuse of insecure service configurations - Abuse of legitimate system tools (e.g., lolBins, PowerShell)) - ARP Spoofing -> leads to man in the middle #5 - DNS Spoofing -> leads to man in the middle #5 - 20 BGP Hijacking -> leads to man in the middle #5 - SSL Stripping (attacker needs to be MitM already eg via ARPPoisoning - and SSL Stripping is an abuse of a (downgrade) function*

*2. **Exploiting Server** - Sub-Threats: - Buffer Overflows - SQL Injections - Cross-Site Scripting (XSS) - XML External Entity (XXE) Attacks - Server Side Request Forgery (SSRF) - Directory Traversal - Ping of Death*

*3. **Exploiting Client** - Sub-Threats: - Malvertising - Watering Hole Attacks - Clients App Exploits (e.g. Browser, PDF Reader, Java, Flash) - Insecure Deserialization*

*4. **Identity Theft** - Sub-Threats: - Credential Stuffing (eg IDs, Passwords, Certificates, Private Keys) - Session Hijacking - Pass-the-Ticket/Pass-the-Hash Attacks - Token Hijacking - password spray attacks - Brute-Force Attacks - Fake Websites - Domain Squatting*

*5. **Man in the Middle** - Sub-Threats: (MitM has a focus on a already compromised environment - you cannot trust any components between the endpoints A and B) - Wi-Fi Eavesdropping (attacker needs to be MitM already eg within physical range -> #8) - Pineapple Attacks (attacker needs to be MitM already eg within physical range -> #8) - Rogue Hotspots (attacker needs to be MitM already eg within physical range -> #8 then eg fakes SSID #4)*

*6. **Flooding Attack** - Sub-Threats: mostly known as DDOS Attacks on dif erent layers - SYN Flood - UDP Flood - HTTP Flood - ICMP Flooding - Slowloris - NTP/DNS Amplification Attacks - Botnet-Driven Attacks*

*7. **Malware** - Sub-Threats: - Ransomware - Trojans - Keyloggers - Rootkits - Spyware - Worms - Adware - Mobile Malware - E-Banking Malware*

*8. **Physical Attack** Direct Physical Access Attacks: - Evil Maid Attacks - Hardware Keyloggers - Direct Hardware Tampering - Device Theft - Physical Intrusion into Secure Areas - USB Baiting (leaving malicious USB devices) - Replacement of Hardware Components - Physical Damage to Infrastructure Indirect Physical Access Attacks: - TEMPEST Attacks (Electromagnetic Emissions) - RFID Skimming - Acoustic Attacks (Sound Wave Exploitation) - Optical Attacks (e.g., Shoulder Surfing) - Thermal Imaging Attacks - Power Analysis Attacks - Environmental Manipulation (e.g., Temperature, Humidity) - Van Eck Phreaking (Remote Screen Viewing)*

*9. **Social Engineering** (Information Manipulation) - Sub-Threats: - CEO Fraud - Subscription Traps - Fraudulent Contests - Check Fraud - Cyberbullying - Dubious Webshop - Requests for financial help from acquaintances - Fake Support - Financial Agents - Fake Threat Emails from Authorities - Investment Fraud - Classified Ads Fraud - Package Subscription Traps - Invoice Manipulation Fraud (BEC Fraud) - Romance Scam - Defamation - Sextortion - Forbidden Pornography - Advance Fee Fraud - Web Administrators Blackmail - Tailgating (unauthorized access) - Phishing - Vishing - Smishing - Baiting (e.g., with USB sticks)*

*10. **Supply Chain** - Sub-Threats: - Compromised Libraries or Dependencies - Backdoors - Update-Server Hijacking - Compromised Container Images - Manipulated Hardware (physical attack on Supply Chain)*

IT Systems, Assets, and the TLCTC Framework

The cyber threat landscape experiences constant evolution, primarily driven by changes in IT system types, their functional domains, and underlying technologies. From traditional enterprise systems to cloud infrastructure, from Internet of Things (IoT) devices to quantum computing platforms, the variety and complexity of IT systems continue to expand. However, it's crucial to understand that while the technological implementation details may change, the fundamental vulnerabilities that cyber threats exploit remain consistent. The Top Level Cyber Threat Clusters (TLCTC) framework maintains its relevance and applicability precisely because it focuses on these underlying generic vulnerabilities rather than specific technological implementations.

This framework's strength lies in its ability to categorize threats based on root causes and generic vulnerabilities, transcending the specific characteristics of any particular IT system type. Whether analyzing threats to a traditional database server, a cloud-native application, or an emerging quantum computing platform, the same ten clusters provide a comprehensive framework for threat identification and risk management. This consistency enables organizations to maintain effective security strategies even as their technology landscape evolves.

This approach allows organizations to:

- Maintain consistent risk management practices across diverse technology implementations
- Apply security controls systematically, regardless of specific IT system types
- Focus on fundamental vulnerabilities rather than getting lost in technical implementation details
- Adapt security strategies efficiently as new technologies emerge

A Strategic Perspective

The Challenge: Moving Beyond IT System Types

Organizations often attempt to categorize cyber threats based on IT system types – creating separate threat categories for cloud systems, IoT devices, SCADA systems, and other technology-specific implementations. This approach presents several critical problems:

- Fragmentation: Creates siloed threat perspectives that fail to recognize common underlying vulnerabilities
- Redundancy: Leads to duplicate threat categories across different system types
- Scalability Issues: Requires constant updates as new technologies emerge
- Strategic Disconnect: Focuses on technical implementations rather than root causes
- Resource Inefficiency: Results in redundant control frameworks and security measures

This system-type-based categorization persists despite its limitations, largely due to historical practices and the natural tendency to organize threats around familiar technical boundaries. However, this approach becomes increasingly unsustainable as technology landscapes grow more complex and interconnected.

Core Principles

1. Generic Vulnerabilities vs. System Types

- The TLCTC framework categorizes threats based on generic vulnerabilities, not IT system types
- Generic vulnerabilities persist across all IT systems, regardless of their specific architecture or implementation
- Each threat cluster represents a distinct way these vulnerabilities can be exploited

2. Asset Management in Risk Strategy

- Assets are managed through the GOVERN function at the strategic level
- Different asset types carry different risk impacts when compromised
- Asset inventory informs risk appetite and resource allocation decisions
- Assets provide context for operational security implementation

Strategic vs. Operational Views

Strategic Level (GOVERN)

- Maintains comprehensive asset inventory and risk register
- Sets risk appetite based on asset criticality
- Allocates resources according to asset priority
- Focuses on generic vulnerabilities across all assets
- Uses TLCTC framework for threat categorization

Operational Level

- Implements controls based on specific asset characteristics
- Addresses technical nuances of different IT system types
- Tailors security measures to specific deployment contexts
- Maps generic vulnerabilities to specific technical weaknesses
- Maintains detailed threat intelligence for each asset type

Example: Cloud Infrastructure

Strategic View

- Applies all 10 TLCTC threat clusters
- Focuses on generic vulnerabilities:
    - Software functionality and scope (#1)
    - Server-side code flaws (#2)
    - Client-side vulnerabilities (#3)
    - Identity management weaknesses (#4)
    - etc.

Operational View

- Implements cloud-specific controls
- Addresses unique characteristics:
    - Multi-tenancy considerations
    - API security requirements
    - Containerization security
    - Cloud service provider integration

The AI System Example

As an IT System

- Subject to all generic vulnerabilities
- Requires standard TLCTC threat analysis
- Needs specific operational controls

As a Tool

- Can enhance security capabilities
- May introduce new operational considerations

- Requires appropriate control framework

As an Actor

- Potential threat actor
- Uses existing threat clusters
- Requires specific detection strategies

Implementation Framework

Asset Inventory (GOVERN)

- Catalog all IT systems
- Group by type and criticality
- Assess business impact
- Define risk appetite

Threat Analysis (Strategic)

- Apply TLCTC framework
- Focus on generic vulnerabilities
- Map threats to asset groups
- Define control objectives

Control Implementation (Operational)

- Deploy system-specific controls
- Address unique characteristics
- Implement monitoring
- Maintain threat intelligence

Conclusion

The TLCTC framework provides a strategic foundation for threat categorization while acknowledging the operational importance of IT system types. This dual-level approach ensures:

- Consistent threat categorization across all assets
- Clear connection between strategic and operational security
- Effective risk management and resource allocation
- Adaptability to new technologies and systems

By maintaining this clear separation between strategic threat categories and operational asset management, organizations can build more effective and sustainable security programs.

A. Leveraging NIST CSF functions

The NIST CSF functions can be used to organize controls and their objectives (e.g., "Protect from Malware Execution", "Detect Malware Execution") within each of the Top Level Cyber Threat Clusters. This combination would provide a comprehensive framework for both threat identification and risk evaluation.

The "Identify" function, enhanced with the Cyber Threat Clusters, would enable more effective management of both high-level threats and operational sub-threats, ensuring a complete and coherent control framework.

Cyber Threat Cluster Control Framework

Overview

This framework integrates the 10 Top Level Cyber Threat Clusters with the NIST Cybersecurity Functions to provide a comprehensive approach to cybersecurity risk management.

Structure

Use this Scheme for each Threat Cluster:

NIST Function	Control Objective	Local Controls	Umbrella Controls
Identify	Identify weaknesses enabling [Threat] Event	[Specific measures]	[Overarching systems/processes]
Protect	Protect from [Threat] Event	[Specific measures]	[Overarching systems/processes]
Detect	Detect [Threat] Event	[Specific measures]	[Overarching systems/processes]
Respond	Respond to [Threat] Event	[Specific measures]	[Overarching systems/processes]
Recover	Recover from [Threat] Event	[Specific measures]	[Overarching systems/processes]

Example: #2 Exploit Server (Controls are not complete - its a POC here)

NIST Function	Control Objective	Local Controls	Umbrella Controls
Identify	Try to identify failures in the code of your Server Software	Fuzzy Testing, Network based Vulscan	Threat Intell this topic, CVE Subscriptions, Bug Bounty Programm
Protect	Protect Server from being exploited	Patchmanagement, Secure Coding	WAF
Detect	Detect Exploited Server	Local Event Logging	SIEM
Respond	Respond to exploited server	Emergency Patch,	CSIRT, Exploit Server Response Plan (Make WAF Rules)
Recover	Recover Server Exploit Event	Maintain your Repo, Restore	IT-SCM

Example: #4 Identity Theft (Controls are not complete - its a POC here)

NIST Function	Control Objective	Local Controls	Umbrella Controls
Identify	Identify weaknesses in identity management (tech and org); Identify weaknesses in credential management (tech and org)	Password policy audits, Penetration testing	Comprehensive Identity and Access Management (IAM) assessment framework , Bug Bounty Program
Protect	Protect Identity Protect Credentials	Multi-Factor Authentication (MFA), Secure credential distribution	Enterprise-wide Identity Governance and Administration (IGA) system
Detect	Detect Identity Theft	Anomaly detection rules, User behavior monitoring	Security Information and Event Management (SIEM) system
Respond	Respond to Identity Theft	Account lockout procedures, Incident response plan activation	Integrated Incident Response Platform
Recover	Recover Identity	Identity restoration, Credential reset procedures	Enterprise-wide Business Continuity Management System

While NIST functions provide an excellent structure for organizing controls and their objectives within each Cyber Threat Cluster, ISO standards can play a complementary role in this framework. Organizations can leverage ISO's comprehensive control sets (such as those in ISO 27002) and risk management methodologies (ISO 27005) to enhance control selection and implementation within the NIST function structure, thereby creating a more robust and internationally aligned approach to addressing each threat cluster.

Application

This framework can be applied to all 10 Top Level Cyber Threat Clusters:

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

For each cluster, specific Control Objectives, Local Controls, and Umbrella Controls should be defined according to the unique characteristics and risks associated with that threat type.

<u>Important Consideration for Umbrella Controls:</u>

Umbrella Controls provide protection only for specific 'Groups of IT-Systems' within their scope. For example, a firewall or network zone can protect 'inner IT-Systems' but cannot effectively protect exposed IT-Systems. This limitation requires security architects to:

- Identify primary entry points and potential 'Patient Zero' systems that could be initially compromised
- Recognize that after a 'Patient Zero' compromise, attacks typically follow the 'lateral movement' paradigm
- Design defense-in-depth strategies that account for both exposed and internal systems
- Implement appropriate Local Controls for systems that cannot be fully protected by Umbrella Controls

This understanding is crucial for effective control implementation and supports the framework's emphasis on attack sequences and paths.

Where are the GOV controls?

The GOVERN (GV) function in NIST CSF 2.0 operates at a strategic level, focusing on establishing the overall cybersecurity risk management framework rather than addressing specific threats directly. Unlike functions such as PROTECT or DETECT, which have controls directly linked to mitigating or identifying particular cyber threats, GOVERN controls are "assurance controls" that ensure the organization has a comprehensive approach to cybersecurity. These controls create the structure and context within which other functions operate, including setting risk appetite, defining roles and responsibilities, and establishing policies. While the threat categorization, such as the Top Level Cyber Threat Clusters, is indeed a crucial element in the risk register that GOVERN oversees, the GV controls themselves do not directly counter specific threats. Instead, they provide the strategic foundation that enables the organization to effectively manage and respond to the entire spectrum of cyber risks.

B. SSDLC Integration

Introduction

The Secure Software Development Life Cycle (SSDLC) provides a structured approach to embedding security throughout the software development process. By integrating the 10 Top Level Cyber Threat Clusters (TLCTC) framework, organizations can establish a consistent, threat-informed methodology that bridges strategic security planning with tactical implementation. Achieving this requires careful consideration of both high-level architectural decisions (programming) and detailed, secure implementation (coding) throughout the lifecycle, guided by the TLCTC framework. This chapter outlines how the TLCTC framework integrates into each phase of the SSDLC, highlighting the differentiated roles of programmers and coders in building secure software.

Fundamental Principles

The integration relies on the core principles of the TLCTC framework:

- Universal Applicability: The TLCTC framework maintains its core strength in the SSDLC through:
    - Consistent threat-vulnerability mapping across all development phases.
    - Clear separation between threats (causes) and their outcomes (effects).
    - Logical sequence representation for complex attack paths.
    - Standardized categorization that applies both horizontally (across system types) and vertically (through protection rings).
- Generic Vulnerabilities Focus:** Each phase of the SSDLC must proactively address the generic vulnerabilities identified in the TLCTC framework:
    - Software scope and functionality vulnerabilities (#1 Abuse of Functions)
    - Server-side code flaws (#2 Exploiting Server)
    - Client-side processing vulnerabilities (#3 Exploiting Client)

    - Identity and access management design weaknesses (#4 Identity Theft)
    - Communication path control issues (#5 Man in the Middle)
    - Capacity limitations (#6 Flooding Attack)
    - Code execution capabilities (#7 Malware)
    - Physical accessibility concerns (#8 Physical Attack)
    - Human factor vulnerabilities (#9 Social Engineering) - *Primarily addressed through awareness, process, and UI/UX design choices influenced by programmers.*
    - Third-party dependency risks (#10 Supply Chain Attack)

Differentiating Roles: Coders vs. Programmers in Secure Development

While often used interchangeably, distinguishing between "coders" and "programmers" clarifies responsibilities within a secure SSDLC context:

- The Programmer's Role:
    - Focuses on the "architecture" and "strategy."
    - Designing overall software architecture and component interactions.
    - Making strategic decisions about frameworks, libraries, platforms, and protocols.
    - Establishing secure coding standards, patterns, and security requirements.
    - Considering system-wide security implications and addressing threat clusters like #1, #4, #5, #10 at an architectural level.
- The Coder's Role:
    - Focuses on the "implementation" and "craftsmanship."
    - Writing functional, efficient code that implements specific requirements according to established patterns.
    - Working within defined boundaries of components or modules.

    - Implementing specific security controls at the code level, primarily addressing clusters #2, #3, and the implementation details of #4, #5, #7 based on programmer-defined standards.
    - Following secure coding practices diligently.

This distinction is vital because different roles hold primary responsibility for mitigating different facets of the TLCTC threat clusters, requiring collaboration throughout the SSDLC.

Phase-Specific Integration

The TLCTC framework informs activities in each SSDLC phase, guiding both programmers and coders:

Requirements Phase - Threat Cluster Analysis:

- Programmer Responsibilities:
    - Identify applicable TLCTC clusters (#1-#10) based on proposed features, data handling, user interactions, and system architecture.
    - Analyze potential attack sequences involving multiple clusters.
    - Define high-level security requirements explicitly tied to mitigating specific TLCTC clusters (e.g., "System must implement controls to prevent #4 Identity Theft through robust authentication and session management").
    - Address potential #1 (Abuse of Functions) by carefully defining functional scope and boundaries.
- Coder Responsibilities:
    - Understand the security requirements impacting the components they will build.

        - Provide feedback on the feasibility and potential implementation challenges of security requirements.

Design Phase - Architecture Considerations:

- Programmer Responsibilities:
    - Design the overall security architecture, mapping trust boundaries and data flows with TLCTC clusters in mind.
    - Make critical architectural choices impacting specific clusters: e.g., selecting authentication frameworks (#4), defining secure communication protocols like TLS/MTLS (#5), choosing dependency vetting strategies (#10), establishing input validation strategies (#2, #3), and designing resource management (#6).
    - Define secure design patterns and select appropriate security libraries/frameworks to address identified clusters.
    - Design defense-in-depth strategies considering potential attack paths.
- Coder Responsibilities:
    - Review proposed designs for implementation feasibility and security concerns at a component level.
    - Understand the security patterns and library usages required by the design.

Implementation Phase - Secure Coding & Programming Practices:

- Programmer Responsibilities:

        - Provide and enforce secure coding standards tailored to the languages and frameworks used.
        - Supply approved security libraries, configurations, and reference implementations for patterns defined in the design phase (e.g., standard authentication flow, secure API gateway usage).
        - Ensure clarity on handling clusters requiring architectural enforcement (#1, #4 design, #5 design, #10 policy).
- Coder Responsibilities:
    - Primary focus. Apply secure coding practices diligently, targeting relevant clusters:
        - Implement proper input validation, output encoding, and error handling (#2, #3).
        - Prevent common code-level vulnerabilities like SQL injection, XSS, buffer overflows (#2, #3).
        - Implement authentication, authorization, and session management logic securely according to design (#4).
        - Utilize secure communication protocols and perform certificate validation correctly (#5).
        - Implement controls against malicious code execution (e.g., safe file uploads, avoiding eval) (#7).
        - Adhere to dependency usage guidelines (#10).
    - Example (OAuth Implementation for #4 Identity Theft):

        - Programmer Decision: Use OAuth 2.0 with PKCE flow via a specific Identity Provider. Define token lifetime and scope policies. Specify required libraries.
        - Coder Implementation: Implement the OAuth flow using the specified library, securely store client secrets (if applicable), validate received tokens (signature, issuer, audience, expiry, nonce), handle state parameter correctly, protect against open redirect vulnerabilities, and securely manage refresh tokens.

Testing Phase - Threat-Based Testing:

- Programmer Responsibilities:
    - Define the security testing strategy, including threat modeling validation, selection of tools (SAST, DAST, IAST, SCA), and scope of penetration testing, all mapped to TLCTC clusters and potential attack sequences.
- Coder Responsibilities:
    - Write unit and integration tests that verify the correct implementation of security controls relevant to their code (#2, #3, #4 implementation, etc.).
    - Participate actively in secure code reviews, focusing on adherence to standards and potential cluster-related vulnerabilities.
    - Remediate findings from SAST/DAST scans and manual tests.

Deployment Phase - Secure Deployment:

- Programmer Responsibilities:
    - Design secure deployment architecture (e.g., network segmentation, firewall rules relevant to #5, #6).

        - Define security requirements for the CI/CD pipeline (relevant to #10, #7).
        - Establish secure configuration management strategy (preventing #1 due to misconfiguration).
    - Coder Responsibilities:
        - Implement infrastructure-as-code and deployment scripts securely.
        - Manage application secrets and configurations securely during deployment (#4).
        - Ensure build artifacts are tamper-proof (#10, #7).

Maintenance Phase - Continuous Security:

- Programmer Responsibilities:
    - Establish and oversee vulnerability management processes (especially for #10).
    - Define threat monitoring strategies aligned with TLCTC clusters.
    - Develop and refine incident response plans tailored to different threat cluster scenarios.
- Coder Responsibilities:
    - Apply security patches and updates promptly and securely (#2, #3, #10).
    - Securely update configurations as needed (#1).
    - Participate in incident analysis and post-mortem reviews, providing implementation context.

Integration with NIST CSF Functions

Mapping NIST CSF Functions to SSDLC Phases remains relevant. The Coder and Programmer roles contribute activities across all five functions (Identify, Protect, Detect, Respond, Recover) throughout the SSDLC, ensuring comprehensive coverage. For instance, programmers contribute heavily to Identify and Protect during Design, while coders focus intensely on Protect during Implementation, and both contribute to Detect and Respond during Testing and Maintenance.

C. Secure Coding Practices

Introduction

Secure coding is far more than a final checkpoint before release; it's an ongoing discipline woven into each phase of the Secure Software Development Life Cycle (SSDLC), as detailed in Chapter B. By linking coding and architectural decisions to the 10 Top Level Cyber Threat Clusters (TLCTC), development teams gain clarity on specific risks and the precise measures needed for mitigation. This "threat-to-practice" mapping transforms abstract security policies into concrete, actionable measures at both the code (Coder) and architectural (Programmer) levels.

This chapter details specific secure coding and programming practices pertinent to the TLCTC clusters most directly influenced during software development. It highlights the collaborative nature of security, where programmers establish the secure foundation and coders build upon it with secure implementation details.

Mapping TLCTC Clusters to Secure Development Practices

The following sections outline key practices for Programmers and Coders, organized by the relevant TLCTC clusters.

#1 Abuse of Functions

- Threat Focus: Attackers misusing legitimate software functions, features, or configurations beyond their intended scope or permissions, often exploiting design weaknesses or misconfigurations.
- Programmer-Level Practices (Architectural/Design):
    - Define Clear Functional Scope: Strictly define and document the intended purpose and boundaries of each function, API, and component.

    - Apply Principle of Least Privilege: Design components and roles with the minimum necessary permissions. Avoid overly permissive defaults.
    - Establish Secure Configuration Defaults: Ensure default configurations are secure and require explicit actions to enable potentially risky features.
    - Design Robust Access Control: Architect clear authorization checks based on roles or attributes before allowing access to functions or data.
    - API Boundary Definition: Clearly define and enforce contracts for internal and external APIs, including expected inputs, outputs, and rate limits.
- Coder-Level Practices (Implementation):
    - Implement Within Boundaries: Ensure code respects the defined functional scope and doesn't introduce unintended capabilities.
    - Enforce Access Controls: Correctly implement the authorization checks defined in the design before executing sensitive operations.
    - Use Approved Interfaces: Consistently use designated internal and external APIs according to their defined contracts.
    - Secure Configuration Handling: Implement code that reads and applies configurations securely, avoiding hardcoded bypasses.

#2 Exploiting Server & #3 Exploiting Client

- Threat Focus: Attackers targeting flaws in the implementation (source code) of server-side or client-side software, leading to unintended behavior like arbitrary code execution, data leakage, or denial of service.
- Programmer-Level Practices (Architectural/Design):
    - Establish Secure Coding Standards: Define and enforce language-specific secure coding guidelines covering common vulnerabilities (e.g., OWASP Top 10, CERT Secure Coding).
    - Minimize Attack Surface: Design architecture to expose minimal functionality and validate all inputs at trust boundaries.

    - Select Secure Libraries/Frameworks: Choose well-vetted libraries and frameworks with strong security track records and features (e.g., built-in input validation, output encoding).
    - Define Security Patterns: Create reusable patterns for common security controls like input validation, output encoding, error handling, and memory management.
- Coder-Level Practices (Implementation -
    - Input Validation & Sanitization: Rigorously validate and sanitize all inputs (user data, API calls, file uploads, configuration values) against a strict allow-list before processing. Prevent injection attacks (SQLi, NoSQLi, Command Injection, LDAPi).
    - Output Encoding: Properly encode all data before rendering it in user interfaces (HTML, JS, CSS) or including it in structured formats (JSON, XML) to prevent Cross-Site Scripting (XSS).
    - Secure API/Function Usage: Use functions and libraries securely, understanding potential pitfalls (e.g., buffer overflows in C/C++, deserialization risks).
    - Memory Management: (Where applicable, e.g., C/C++) Implement correct memory allocation, usage, and deallocation practices to prevent buffer overflows, use-after-free, etc.
    - Secure Error Handling: Implement error handling that fails securely, logs sufficient detail for diagnostics, but does not expose sensitive information to users or attackers.
    - Language-Specific Practices: Apply security best practices specific to the programming language and runtime environment.

#4 Identity Theft

- Threat Focus: Attackers illegitimately acquiring, stealing, or misusing authentication credentials (passwords, tokens, keys, session identifiers) to impersonate legitimate identities.
- Programmer-Level Practices (Architectural/Design):
    - Design Authentication Architecture: Choose appropriate authentication protocols (e.g., OAuth 2.0, OIDC, SAML) and flows based on risk.
    - Select Identity Frameworks/Providers: Evaluate and select secure identity providers (IdPs), libraries, and SDKs.
    - Define Credential & Session Policies: Establish requirements for password complexity, MFA, credential rotation, session timeouts, and secure storage.
    - Establish Secure Patterns: Define standard implementation patterns for login, registration, password reset, token handling, and session management.
- Coder-Level Practices (Implementation):
    - Implement Flows Correctly: Accurately implement authentication and authorization flows according to the chosen standard and design patterns (e.g., correctly handling redirects, state parameters, PKCE in OAuth).
    - Secure Credential Handling: Use industry-standard algorithms (e.g., Argon2, bcrypt) with unique salts for password hashing. Never store passwords in plaintext or reversible formats.
    - Secure Token/Session Management: Implement secure handling of session tokens/cookies (e.g., use HttpOnly, Secure, SameSite flags; validate tokens correctly). Store tokens securely (e.g., avoid local storage for sensitive tokens).
    - Credential Protection: Avoid hardcoding credentials or API keys. Implement secure mechanisms (using secrets management tools/services) to inject secrets at runtime. Protect against credential stuffing and brute-force attacks (e.g., account lockouts, captchas).

#5 Man in the Middle (MitM)

- Threat Focus: Attackers intercepting, eavesdropping on, modifying, or relaying communications between two parties by exploiting weaknesses in the communication channel or its endpoints.
- Programmer-Level Practices (Architectural/Design):
    - Mandate Secure Transport: Require HTTPS (TLS) for all web communication. Design for secure protocols (e.g., SSH, SFTP, VPNs) for other communications.
    - Define TLS/MTLS Strategy: Specify minimum TLS versions, required cipher suites, and whether Mutual TLS (MTLS) is needed for service-to-service communication.
    - Establish Certificate Management Strategy: Define processes for obtaining, deploying, rotating, and revoking certificates. Consider options like certificate pinning where appropriate, understanding the maintenance overhead.
    - Define Secure Communication Patterns: Ensure designs don't inadvertently route sensitive data over insecure channels.
- Coder-Level Practices (Implementation):
    - Implement Secure Protocols Correctly: Utilize standard, secure libraries for TLS/SSH etc. Configure them according to the defined strategy.
    - Proper Certificate Validation: Ensure code *always* validates server certificates (checking hostname, expiry, trust chain) before establishing connections. Implement custom validation logic carefully if required (e.g., for pinning).
    - Secure Data Transmission: Ensure sensitive data (credentials, tokens, PII) is only ever transmitted over encrypted channels.
    - Avoid Insecure Shortcuts: Resist disabling certificate validation or using insecure protocols during development or testing, except in highly controlled, isolated environments.

#7 Malware

- Threat Focus: Attackers abusing a software environment's capability to execute foreign or untrusted executable content (scripts, binaries, macros, dual-use tools).
- Programmer-Level Practices (Architectural/Design):
    - Design Execution Control Policies: Define what code/scripts are allowed to run and from where (e.g., trusted sources only).
    - Require Code Signing: Establish policies for signing internally developed code and verifying signatures of external code/updates.
    - Plan Safe Execution Environments: Design for sandboxing or containerization where untrusted or risky code/processes must be handled.
    - Define Safe File Handling Policies: Specify how file uploads/downloads should be handled, scanned, and stored.
- Coder-Level Practices (Implementation):
    - Implement Safe Execution: Avoid features that allow arbitrary code execution based on external input (e.g., eval() in JavaScript, unsafe deserialization).
    - Secure File Handling: Implement file upload/download logic securely, validating file types, names, and sizes. Store uploaded files outside the web root and with non-executable permissions. Scan files using appropriate tools.
    - Validate External Inputs: Treat all data loaded from external sources (files, network) as potentially untrusted and validate/sanitize accordingly before use, especially if it influences execution flow.
    - Dependency Integrity: Ensure build processes verify the integrity of fetched dependencies (checksums/signatures).

#10 Supply Chain Attack

- Threat Focus: Attackers compromising systems by targeting vulnerabilities within third-party software components, hardware, services, or distribution/update mechanisms integrated into an organization's environment or products.
- Programmer-Level Practices (Architectural/Design):
    - Establish Dependency Vetting: Define a process for evaluating and approving third-party libraries, frameworks, and components based on security posture, maintenance, and known vulnerabilities.
    - Define Dependency Management Policy: Specify rules for updating dependencies, removing unused ones, and using tools like Software Composition Analysis (SCA) and Software Bill of Materials (SBOM).
    - Secure Build/Deployment Pipeline: Design CI/CD pipelines with security checks (SCA scans, static analysis, integrity verification) integrated.
    - Establish Vulnerability Management Process: Define how to respond when vulnerabilities are found in dependencies.
- Coder-Level Practices (Implementation):
    - Adhere to Dependency Guidelines: Use only approved libraries and versions. Avoid adding unvetted dependencies.
    - Utilize Tooling: Integrate SCA tools into local development environments and CI pipelines. Regularly review SBOMs.
    - Implement Safely: When using external libraries, treat their inputs/outputs carefully. Implement safeguards (e.g., validating data returned from a library) as if they could be compromised.
    - Report Vulnerabilities: Promptly report potentially vulnerable dependencies identified during development or via SCA tools.
    - Minimize Dependency Footprint: Use only necessary libraries and features to reduce the attack surface.

Application Across the SSDLC

These practices are not confined to the implementation phase. As detailed in Chapter B, programmer-level decisions informed by TLCTC occur during Requirements and Design. Coder-level implementation happens during the Implementation phase, guided by those decisions. Both roles contribute to verification during Testing and ongoing vigilance during Maintenance. Secure coding is a continuous effort, contextualized by the SSDLC process and focused by the TLCTC framework.

Conclusion

By explicitly mapping secure coding and programming practices to the Top Level Cyber Threat Clusters, organizations move beyond generic guidelines. This structured approach empowers programmers to make informed architectural decisions and enables coders to apply targeted, effective security controls during implementation. Understanding the distinct responsibilities within the context of TLCTC ensures comprehensive coverage, reduces ambiguity, and fosters a collaborative environment where security is integrated throughout the software development lifecycle, resulting in more resilient and trustworthy software.

Reflecting on STRIDE

Earlier threat modeling methodologies like STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) provided useful lenses for understanding certain types of attacks. STRIDE helped break down the enormous scope of security concerns into manageable buckets, guiding developers away from a purely ad hoc approach.

However, as the threat landscape has grown more complex and specialized, STRIDE's categories can feel too broad or outdated compared to the nuanced approach of the TLCTC. While STRIDE remains a valuable historical and foundational concept, the TLCTC framework offers a more direct mapping from modern, often specialized attacks (like supply chain breaches or client-side exploits) to concrete coding practices. This granularity and relevance to current threats make TLCTC a powerful evolution of earlier methodologies.

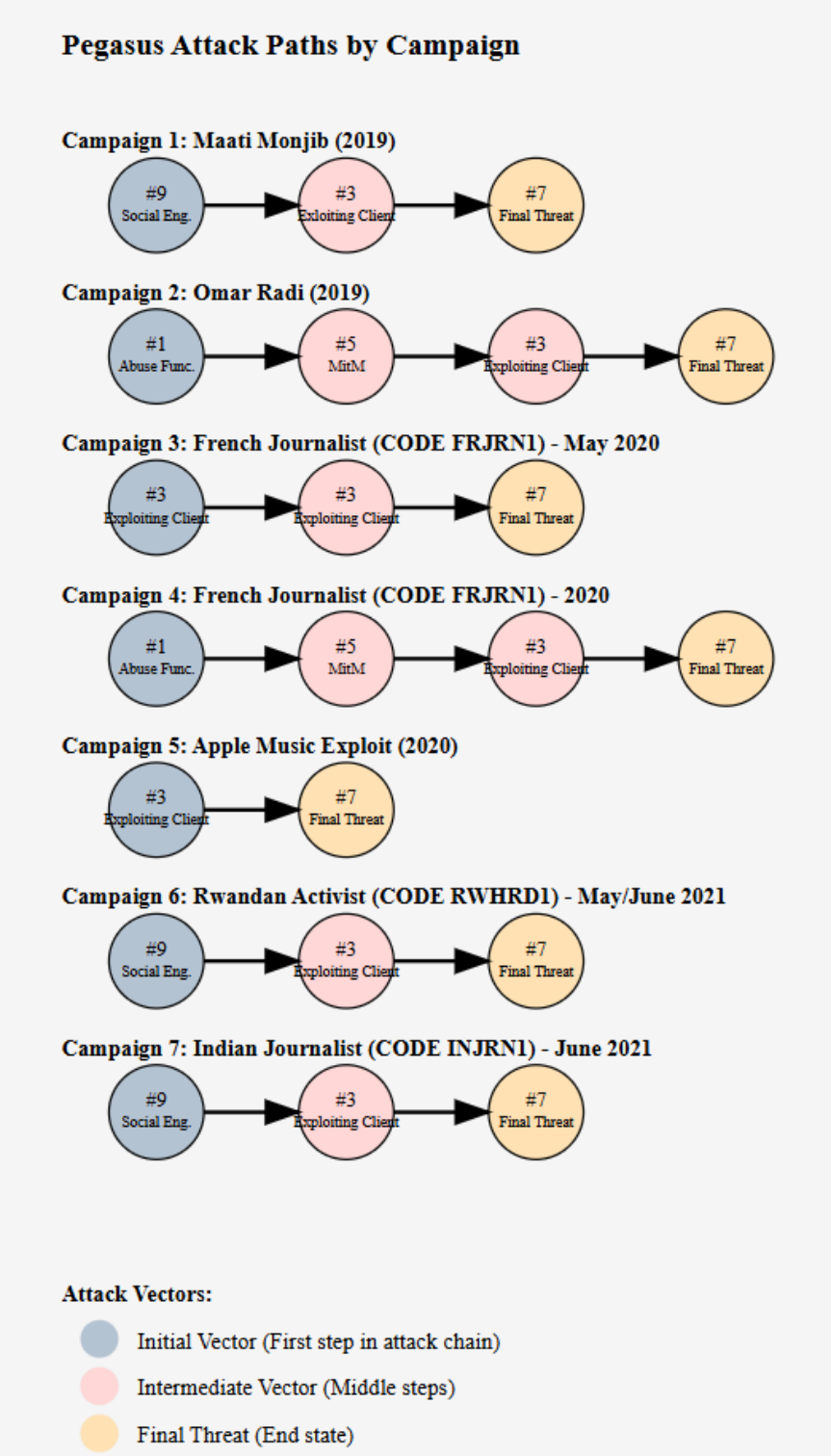
Conclusion

By adopting the TLCTC framework and linking it to secure coding practices, organizations build robust security into their development lifecycle. Every coding choice correlates with a recognized threat cluster, and every external warning—whether from CERT, CISA, or CVE—is easily mapped to familiar controls. The inclusion of concrete examples ensures new readers can calibrate their understanding quickly. While earlier models like STRIDE paved the way for structured threat analysis, TLCTC meets today's challenges head-on with greater specificity, adaptability, and direct applicability.

In doing so, software security matures from a reactive, post-release scramble into a proactive, well-informed endeavor that thrives on shared language, continuous improvement, and close alignment with the evolving threat environment.

D. Threat Intelligence - Real World Examples

NSO Group Pegasus spyware Attack Paths



Based on the Amnesty International report[1], the NSO Group's Pegasus spyware attack paths can be categorized into several main vectors. These attack paths demonstrate the sophisticated and evolving nature of the Pegasus spyware, utilizing various threat clusters in sequence to compromise target devices:

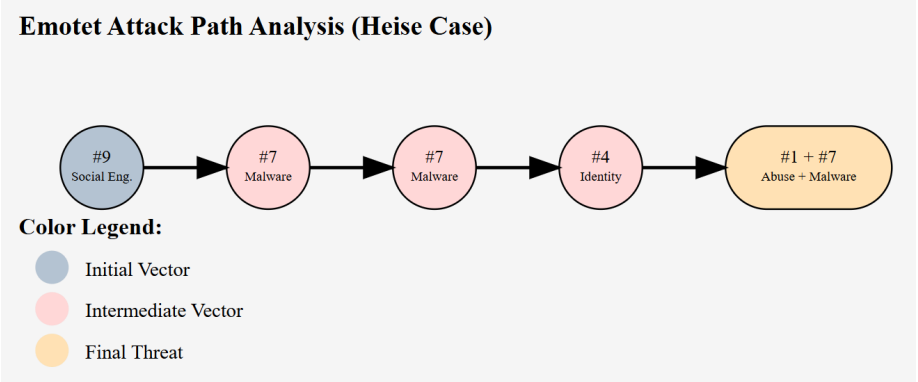
[1] https://www.barnes.ch/Forensic_Methodology_Report_NSO_Groups_Pegasus_Amnesty_International.pdf

Example	Description
Campaign 1:	The most common and effective path. Begins with a malicious SMS or iMessage, leading the target to click a link that infects the device with malware. Seen in attacks against Maati Monjib (2019) and many others.
Campaign 2:	Uses network injection attack (MitM) to redirect the user to a compromised website, which then delivers malware. Used against Omar Radi (2019), redirecting to a fake Yahoo page and then to a malware-delivering domain.
Campaign 3:	A successful exploit (e.g., zero-day in Apple Photos app) leaves the device vulnerable to a second, more direct malware infection. Seen with French journalist (CODE FRJRN1) in May 2020.
Campaign 4:	This path, seen with the French journalist (CODE FRJRN1) in 2020, starts with network injection (#1 and #5), which then leads to the delivery of a malicious webpage. The user interacting with the webpage triggers the Client Exploit (#3), resulting in the installation of malware (#7).
Campaign 5:	Simplest path, used in Apple Music exploits starting in 2020. Leverages a vulnerability in the Apple Music app to directly deliver malware.
Campaign 6:	Demonstrated with Rwandan activist (CODE RWHRD1) in May and June 2021. Target received multiple iMessage attachments containing malicious code leading to malware installation.
Campaign 7:	Used against Indian journalist (CODE INJRN1) in June 2021. iMessage notifications were the attack vector, ultimately infecting the phone with malware.

These attack paths illustrate the complex and multi-staged nature of Pegasus spyware attacks. They demonstrate how different threat clusters are chained together to bypass security measures and compromise target devices. It's important to note that these represent the most common paths identified in the report, and NSO Group continually develops new methods as security measures evolve.

Emotet@Heise Path

Based on the attack scenario described, we can summarize the attack path using the 10 Top Level Cyber Threat Clusters as follows: (names are fictive)



Here's the breakdown:

1. #9 (Social Engineering): The attack begins with a phishing email sent to Karin Meier, impersonating her colleague Rolf Schulz.

2. #7 (Malware): Karin opens the malicious Word document attached to the email and enables macros, executing the embedded malware code (Emotet).

3. #7 (Malware): Emotet operates on the infected PC, stealing emails and downloading additional malware (Trickbot).

4. #4 (Identity Theft): Trickbot steals domain administrator credentials, allowing for further network compromise.

5. (#1 + #7) (Abuse of Functions + Malware): Simultaneously, the attackers use the stolen admin credentials to spread Trickbot throughout the network, compromising the Active Directory (Abuse of Functions), while deploying the Ryuk ransomware across the network (Malware), encrypting data on servers and backup systems.

This refined attack path demonstrates the sophisticated and multi-staged nature of modern cyber attacks, highlighting how threat actors can leverage multiple threat clusters simultaneously in the final stages to rapidly achieve widespread compromise and data encryption. The parallel execution of Abuse of Functions and Malware deployment in the last step underscores the complex and interconnected nature of advanced cyber attacks.

Cobalt Strike as a Multi-Threat Tool

Cobalt Strike, as a comprehensive post-exploitation framework, embodies functionalities that span across all 10 Top Level Cyber Threat Clusters. It serves as a prolonged arm of the attacker, providing capabilities that can be leveraged at various stages of an attack.

Mapping to Threat Clusters

Abuse of Functions (#1):

- OS-level process injection using designed Windows functionality
- DLL search order hijacking using legitimate Windows loading behavior
- Abuse of legitimate Windows APIs and system calls
- Abuse of built-in Windows tools (Living off the Land)

Exploiting Server (#2):

- Application-level process injection exploiting vulnerabilities
- Remote code execution exploits
- Server-side vulnerability exploitation modules

Exploiting Client (#3):

- Browser exploitation modules
- Client-side application exploits
- Document-based exploit creation tools

Identity Theft (#4):

- Credential harvesting functionality
- Pass-the-hash modules
- Token manipulation capabilities
- Session cookie theft tools

Man in the Middle (#5):

- Network traffic interception
- Protocol relay attacks
- Proxy functionality

Flooding Attack (#6):

- Distributed network flooding capabilities
- Resource exhaustion modules

Malware (#7):

- Beacon payload generation and execution
- Custom malware creation tools
- Payload staging and execution

Physical Attack (#8):

- USB attack payload creation
- Physical access exploitation tools

Social Engineering (#9):

- Phishing email templates
- Malicious document creation
- Decoy file generation

Supply Chain (#10):

- Software distribution compromise tools
- Update server exploitation capabilities

Enabling Attack Paths

Cobalt Strike's diverse functionality allows attackers to construct various attack paths, chaining multiple threat clusters. The specific path followed depends on the attacker's script or campaign. For example:

Path 1: #9 (Phishing email) -> #3 (Client-side exploit) -> #7 (Beacon deployment) -> #1 (OS-level process injection) -> #4 (Credential theft)

Path 2: #10 (Update server compromise) -> #7 (Malicious update deployment) -> #5 (Network traffic interception) -> #2 (Server exploitation)

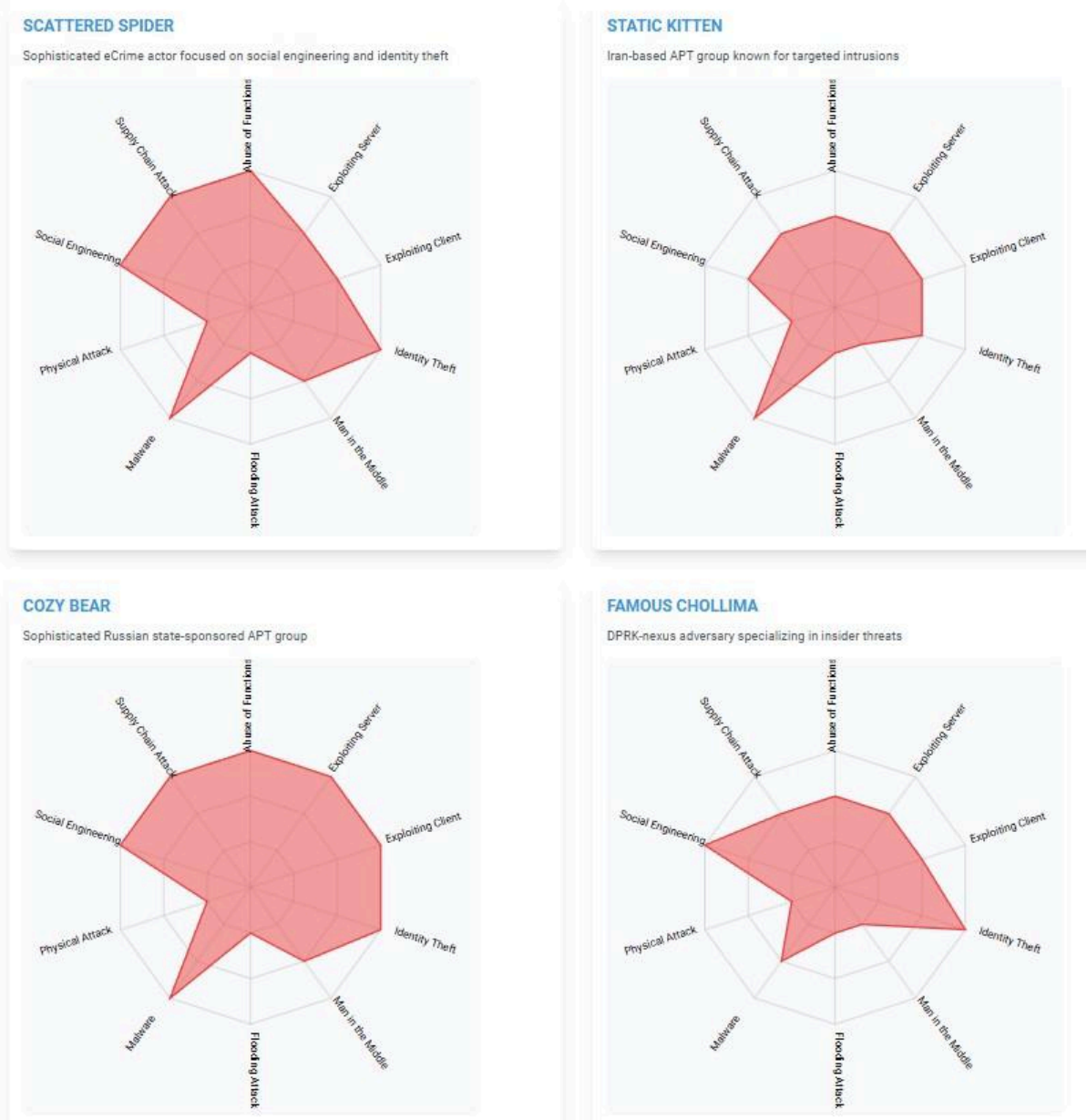
Path 3: #8 (USB payload delivery) -> #7 (Beacon execution) -> #1 (Windows DLL search order abuse) -> #6 (Coordinated flooding)

Conclusion

This analysis demonstrates how the 10 Top Level Cyber Threat Clusters framework effectively categorizes the multifaceted capabilities of a complex tool like Cobalt Strike. By carefully distinguishing between abuse of legitimate functionality (#1) and exploitation of vulnerabilities (#2), the framework provides clear guidance for threat modeling and control implementation. Understanding these distinctions and potential attack paths is crucial for developing effective defense strategies and risk assessments.

Attacker profiles

See how the 10 Top Level Cyber Threat Clusters (TLCTC) framework enables enhanced comparison of threat actors, including APTs. Based on CrowdStrike's 2024 Threat Hunting Report[2], this interactive visualization reveals capability patterns across different adversary groups.



[2] https://go.crowdstrike.com/2024-threat-hunting-report-thank-you.html

Capability ratings: 1 (Low), 2 (Medium), 3 (High). Based on observed activities and intelligence analysis through the TLCTC framework.

Each radar chart maps an APT group's proficiency across all ten clusters: Abuse of Functions, Exploiting Server, Exploiting Client, Identity Theft, Man in the Middle, Flooding Attack, Malware, Physical Attack, Social Engineering, and Supply Chain Attack. This cluster-based analysis enables better understanding of adversary capabilities and helps bridge the gap between strategic risk management and operational security.

By comparing APT groups through the TLCTC lens, we can better understand their distinct capabilities, preferred tactics, and potential attack sequences. This insight supports more effective threat intelligence sharing and targeted defense strategies.

E: Threat Intelligence - Analysis of MITRE & STIX

The cybersecurity landscape faces a critical challenge: fragmented threat intelligence that fails to effectively connect strategic risk management with operational security execution. While frameworks like MITRE ATT&CK and STIX enable detailed threat intelligence sharing, they lack a standardized high-level threat categorization system that aligns threat intelligence with risk management and security operations.

MITRE needs to focus on mapping technical techniques to strategic clusters, while STIX needs to enhance its data model to represent these clusters and their relationships in threat intelligence sharing.

The Top Level Cyber Threat Clusters framework addresses this gap by providing a comprehensive solution that bridges threat intelligence with practical security implementation.

- Universal Taxonomy: Establishes a standardized system for consistent threat intelligence collection and sharing across organizations and sectors
- Intelligence-Vulnerability Mapping: Creates clear connections between threat intelligence indicators and generic vulnerabilities, enabling more effective risk assessment
- Control Implementation Methodology: Provides a structured approach for translating threat intelligence into specific control requirements and implementation guidelines
- Unified Communication: Establishes a common language between threat intelligence teams, risk managers, and security operations personnel

By integrating this framework with established standards like MITRE ATT&CK and STIX, organizations can transform raw threat intelligence into actionable insights that drive both strategic risk decisions and tactical security operations. This integration enables:

- Enhanced Threat Hunting: More effective identification and tracking of potential threats across the environment
- Precise Control Selection: Better alignment between identified threats and necessary security controls
- Comprehensive Incident Response: More thorough and effective incident response planning and execution
- Lifecycle Consistency: Maintained consistency across the entire threat intelligence lifecycle, from collection to action

Enhancing STIX with the Top Level Cyber Threat Clusters

Current State: STIX provides a rich set of objects and relationships for describing cyber threat information, but it has limitations:

STIX Component	**Purpose**	**Limitation**
Objects (e.g., Threat Actor, Attack Pattern, Malware)	Describe individual elements of cyber threats	Lacks a standardized high-level categorization system
Relationships	Connect different STIX objects to represent complex scenarios	No standardized way to represent attack sequences or paths
Intrusion Set	Represent adversary behaviors and resources	Focuses on actor behaviors rather than threat categories or attack progressions

Proposed Enhancements:

- Standardized Threat Categorization: Introduce the 10 Top Level Cyber Threat Clusters as a new STIX Domain Object, providing a consistent, high-level categorization system.
- Attack Path Representation: Implement a new STIX object type to represent attack paths as sequences of threat clusters (e.g., #9 -> #3 -> #7).

- Strategic Overview: Enable a more strategic view of threats and attack progressions, bridging the gap between detailed STIX data and high-level risk management.

Implementation Approach:

Create a New STIX Domain Object:

Threat Cluster Object Structure:


```
{
 "type": "x-threat-cluster",
 "spec_version": "2.1",
 "id": "x-threat-cluster--uuid",
 "created": "2024-01-29T18:20:00.000Z",
 "modified": "2024-01-29T18:20:00.000Z",
 "name": "Abuse of Functions",
 "cluster_id": "TC0001",
 "definition": "Abuse of Functions involves manipulating the intended functionality of software or systems for malicious purposes",
 "generic_vulnerability": "The scope of software and functions",
 "asset_type": "Software",
 "attacker_vector": "Abuse of functionality, not a coding issue"
}
```


Attack Sequence Object:


```
{
 "type": "x-attack-sequence",
 "spec_version": "2.1",
 "id": "x-attack-sequence--uuid",
 "created": "2024-01-29T18:20:00.000Z",
```

```
  "modified": "2024-01-29T18:20:00.000Z",
  "sequence_id": "SEQ001",
  "initial_cluster": "x-threat-cluster--uuid1",
  "subsequent_clusters": [
   "x-threat-cluster--uuid2",
   "x-threat-cluster--uuid3"
  ],
  "common_pattern_name": "Phishing to Malware Chain",
  "observed_frequency": "high",
  "first_observed": "2024-01-01T00:00:00Z",
  "last_observed": "2024-01-29T00:00:00Z",
  "count": 157
 }
```


Enhanced Attack Pattern Object:


```
 {
  "type": "attack-pattern",
  "spec_version": "2.1",
  "id": "attack-pattern--uuid",
  "created": "2024-01-29T18:20:00.000Z",
  "modified": "2024-01-29T18:20:00.000Z",
  "name": "Spearphishing Attachment",
  "description": "Adversaries may send spearphishing emails with malicious attachments",
  "kill_chain_phases": [{
   "kill_chain_name": "mitre-attack",
   "phase_name": "initial-access"
```

```
 }],
 "x_tlctc": {
  "primary_threat_cluster": "x-threat-cluster--uuid9",
  "secondary_threat_clusters": [
   "x-threat-cluster--uuid3",
   "x-threat-cluster--uuid7"
  ],
  "generic_vulnerability_exploitation": "Exploits human susceptibility to deception",
  "attack_sequence_position": {
   "can_be_initial": true,
   "can_be_subsequent": false,
   "typical_sequence": "#9->#3->#7"
  }
 }
}
```


Benefits of Integration:

- Provides a standardized framework for high-level threat categorization: Enables consistent communication and understanding of threats across different teams and organizations.
- Enables representation and analysis of attack progressions: Allows for modeling and analysis of how attacks unfold, aiding in the development of defensive strategies.
- Facilitates better communication between technical and non-technical stakeholders: Helps in bridging the gap between detailed technical data and high-level risk management.

- Enhances strategic threat analysis and risk management capabilities: Provides a more comprehensive and structured approach to representing, analyzing, and communicating about cyber threats.

Enhancing MITRE ATT&CK

Current State: MITRE ATT&CK excels at the operational security level, providing detailed tactics and techniques for various attack stages across different IT system types. However, it lacks a high-level strategic framework for threat categorization and overemphasizes post-compromise techniques.

Framework	**Current Limitations**	**Integration Need**
MITRE ATT&CK	• Lacks high-level strategic framework • Overemphasis on post-compromise • No standardized initial access mapping	Map techniques to strategic threat clusters
STIX	• No standardized categorization • Limited attack sequence representation • No strategic-operational bridge	Enhance with structured threat taxonomy

Proposed Enhancements:

- Standardized Threat Categorization: Introduce the 10 Top Level Cyber Threat Clusters as a new MITRE ATT&CK object, providing a consistent, high-level categorization system.
- Attack Path Representation: Implement a new MITRE ATT&CK object type to represent attack paths as sequences of threat clusters (e.g., #9 -> #3 -> #7).
- Strategic Overview: Enable a more strategic view of threats and attack progressions, bridging the gap between detailed MITRE ATT&CK data and high-level risk management.

Implementation Approach:

Enhanced STIX Objects:

STIX Threat Cluster Object:


```
{
 "type": "x-threat-cluster",
 "spec_version": "2.1",
 "id": "x-threat-cluster--f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
 "created": "2024-01-29T18:20:00.000Z",
 "modified": "2024-01-29T18:20:00.000Z",
 "name": "Abuse of Functions",
 "cluster_id": "TC0001",
 "definition": "Abuse of Functions involves manipulating the intended functionality of software or systems for malicious purposes",
 "generic_vulnerability": "The scope of software and functions",
 "asset_type": "Software",
 "attacker_vector": "Abuse of functionality, not a coding issue",
```

```
  "mitre_techniques": ["T1548", "T1559", "T1569"]
}
```

STIX Attack Sequence Object:

```
{
  "type": "x-attack-sequence",
  "spec_version": "2.1",
  "id": "x-attack-sequence--d81d4fae-7dec-11d0-a765-00a0c91e6bf6",
  "created": "2024-01-29T18:20:00.000Z",
  "modified": "2024-01-29T18:20:00.000Z",
  "sequence_id": "SEQ001",
  "initial_cluster": "x-threat-cluster--f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
  "subsequent_clusters": [
    "x-threat-cluster--a81d4fae-7dec-11d0-a765-00a0c91e6bf6",
    "x-threat-cluster--b81d4fae-7dec-11d0-a765-00a0c91e6bf6"
  ],
  "mitre_techniques": ["T1566", "T1190", "T1105"],
  "common_pattern_name": "Phishing to Malware Chain",
  "observed_frequency": "high",
  "first_observed": "2024-01-01T00:00:00Z",
  "last_observed": "2024-01-29T00:00:00Z",
  "count": 157
}
```

Enhanced MITRE ATT&CK Pattern:


```
{
 "type": "attack-pattern",
 "spec_version": "2.1",
 "id": "attack-pattern--c81d4fae-7dec-11d0-a765-00a0c91e6bf6",
 "created": "2024-01-29T18:20:00.000Z",
 "modified": "2024-01-29T18:20:00.000Z",
 "name": "Spearphishing Attachment",
 "description": "Adversaries may send spearphishing emails with malicious attachments",
 "kill_chain_phases": [{
  "kill_chain_name": "mitre-attack",
  "phase_name": "initial-access"
 }],
 "x_tlctc": {
  "primary_threat_cluster": "x-threat-cluster--f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
  "secondary_threat_clusters": [
   "x-threat-cluster--a81d4fae-7dec-11d0-a765-00a0c91e6bf6",
   "x-threat-cluster--b81d4fae-7dec-11d0-a765-00a0c91e6bf6"
  ],
  "generic_vulnerability_exploitation": "Exploits human susceptibility to deception",
  "attack_sequence_position": {
   "can_be_initial": true,
   "can_be_subsequent": false,
   "typical_sequence": "#9->#3->#7"
```

```
    }
  }
}
```

References: MITRE ATT&CK Framework, Enterprise Matrix, 2024 and OASIS STIX Version 2.1 Specification, 2024

Benefits of Integration:

- Provides a standardized framework for high-level threat categorization: Enables consistent communication and understanding of threats across different teams and organizations.
- Enables representation and analysis of attack progressions: Allows for modeling and analysis of how attacks unfold, aiding in the development of defensive strategies.
- Facilitates better communication between technical and non-technical stakeholders: Helps in bridging the gap between detailed technical data and high-level risk management.
- Enhances strategic threat analysis and risk management capabilities: Provides a more comprehensive and structured approach to representing, analyzing, and communicating about cyber threats.

Conclusion

Integrating the Top Level Cyber Threat Clusters into the STIX and MITRE ATT&CK frameworks offers significant benefits, including standardized threat categorization, attack path representation, and enhanced strategic threat analysis. By adopting this approach, organizations can better bridge the gap between technical threat data and high-level risk management, leading to more effective cybersecurity strategies and improved communication across all levels of the organization. This integration maintains the

granularity and detail of STIX and MITRE ATT&CK while adding an essential layer of high-level structure, ultimately contributing to a more resilient cyber defense posture.

F: Introducing Cyber Threat Radars

In today's interconnected digital world, cybersecurity is a global concern. However, a critical gap exists in how different countries and organizations categorize and communicate about cyber threats. This lack of standardization hinders effective international collaboration in addressing cybersecurity challenges.

The Current Challenge

- Regulatory frameworks like NIS2 and DORA emphasize incident reporting but lack a unified threat categorization system.
- National Cyber Security Centers (NCSCs) across different countries use varying terminologies and categorizations for cyber threats.

This inconsistency impedes efficient cross-border threat intelligence sharing and coordinated incident response.

Enter the Cyber Threat Radar

The Cyber Threat Radar, based on the 10 Top Level Cyber Threat Clusters, offers a solution to this global challenge. It provides:

- Standardization: A common language for describing threats across different organizations and countries.
- Clarity: A visual representation that simplifies complex threat landscapes.
- Flexibility: Adaptability for use at both organizational and state levels.
- Enhanced Communication: Facilitates better information sharing between NCSCs and organizations worldwide.

Key Benefits

- Improved Global Collaboration: Enables more effective international exchange of cyber threat information.
- Consistent Analysis: Allows for accurate trend analysis and comparison across borders.
- Strategic Insight: Helps in prioritizing threats and allocating resources effectively.
- Regulatory Alignment: Supports compliance with frameworks like NIS2 and DORA by providing a structured approach to threat reporting.

Versatile Application

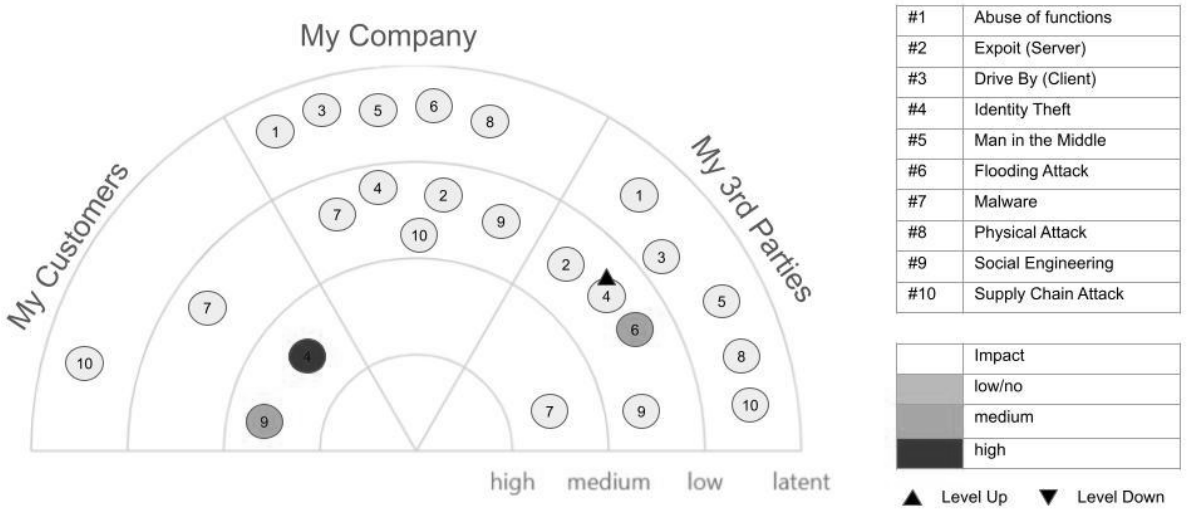
Cyber Threat Radars can be applied at various scales:

- Organizational Level: For companies to visualize and manage their specific threat landscape.
- State Level: For government agencies to monitor national cybersecurity trends and collaborate internationally.

The following examples demonstrate how Cyber Threat Radars can be implemented at both organizational and state levels, showcasing their potential to transform global cybersecurity cooperation.

Action: Direct your SOC and Threat Intelligence teams to map incidents and near-misses to the 10 Top Level Cyber Threat Clusters. Focus on root cause analysis to identify the initial point of compromise. Implement threat radars to visualize threats specific to your organization. Ensure SOC representation in cyber strategy discussions to incorporate emerging threat trends into your risk management approach.

Count each identified threat cluster per incident. multiple count = yes



An example of a threat radar. Analyze the events (Security Incidents) regarding one or many of the cyber threat clusters - find the attack-path!

Understanding Cyber Threat Radar Visualizations

The Top Level Cyber Threat Clusters can be visualized through radar diagrams at different organizational levels. These visualizations help stakeholders understand threat distributions and impacts across their areas of responsibility.

Organizational View

The first radar represents the organization's cyber threat landscape across three key operational sectors:

"My Company"

Your own organization's environment where you have direct control over security measures:

- Core systems, applications, and processes
- Internal and external-facing assets
- Primary focus of your security controls

"My Customers"

Organizations or individuals that depend on your services or products:

- Entities that consume your services/products
- Your organization acts as their supplier/provider
- Their compromise could affect your organization
- You may be part of their supply chain risk (#10)

"My 3rd Parties"

External entities your organization depends on:

- Suppliers, service providers, partners
- SaaS providers, cloud services
- Contractors and consultants
- Organizations in your supply chain

Impact and Movement Indicators

Impact Levels:

- High (Red): Critical impact requiring immediate attention
- Medium (Orange): Significant impact requiring planned mitigation
- Low (Gray): Minor impact manageable through standard controls
- Latent: Potential threats requiring monitoring

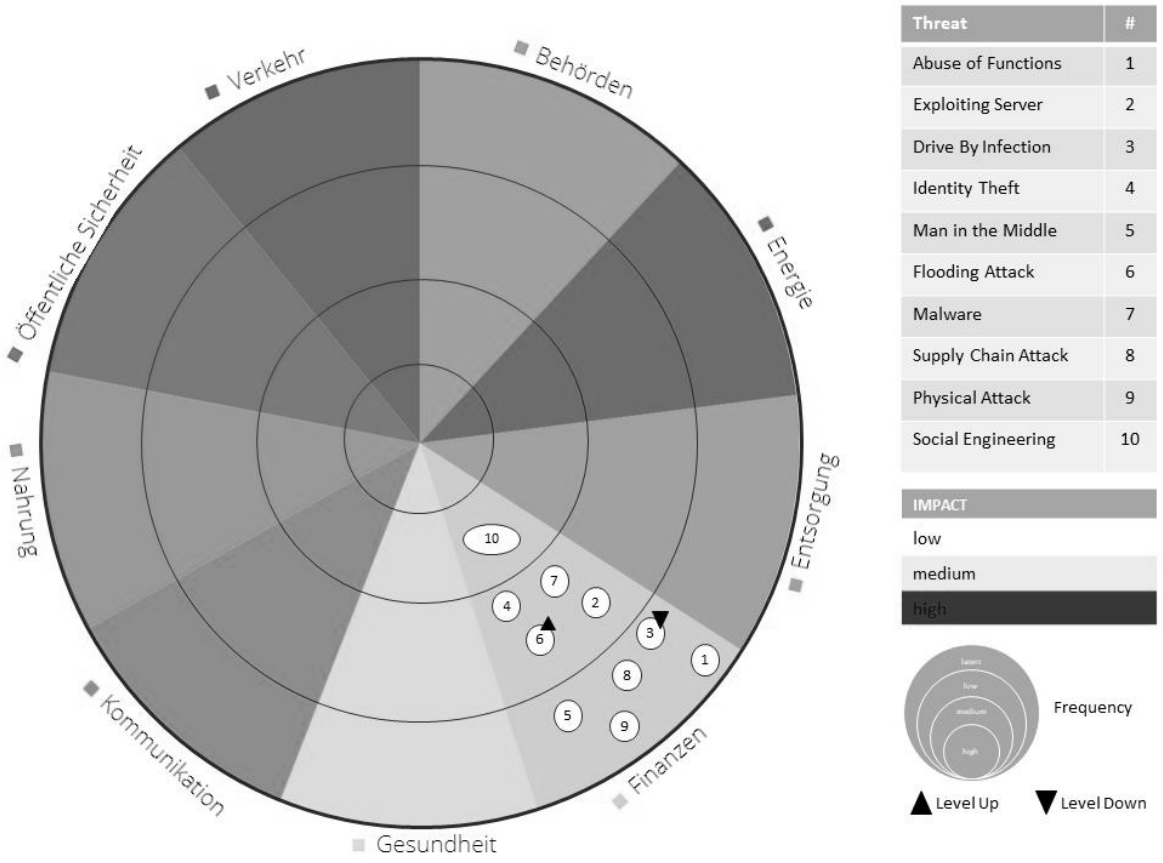
Movement Indicators:

▲ Level Up: Threat impact or frequency is increasing

▼ Level Down: Threat impact or frequency is decreasing

State Level View

The second radar expands the perspective to critical infrastructure and societal sectors, demonstrating how the same threat clusters manifest at a national level:



An example of a state level cyber threat radar.

- Behörden (Authorities): Government and administrative bodies
- Energie (Energy): Power and utility infrastructure
- Finanzen (Finance): Banking and financial services
- Gesundheit (Healthcare): Medical facilities and health services
- Kommunikation (Communication): Telecommunications and media
- Öffentliche Sicherheit (Public Safety): Emergency services and security
- Verkehr (Transportation): Public transport and logistics

This state-level view enables:

- Assessment of cyber threats across critical infrastructure
- Strategic resource allocation for national cyber defense
- Cross-sector threat monitoring and response coordination
- Identification of systemic risks and vulnerabilities

Note: Both radar views demonstrate that all 10 Top Level Cyber Threat Clusters apply universally, regardless of sector or organizational context. The key differences lie in impact levels, frequency, and specific manifestations within each domain.

Standardized Attack Sequence Notation

To further enhance the utility of Cyber Threat Radars and facilitate more precise threat intelligence sharing, i recommend adopting a standardized notation for describing attack sequences:

- Use the cluster numbers to represent each stage of an attack, e.g., #9>#3>#7->#7->#1->#7
- This notation provides a clear, concise way to describe complex attack patterns
- Each identified cluster within an attack should be counted and included in the sequence
- This approach is particularly valuable for describing Attackers profiles (incl. ATPs

For example, an attack sequence of #9>#3>#7->#7->#1->#7 could represent:

1. #9 (Social Engineering) as the initial entry point
2. #3 (Exploiting Client) to gain a foothold
3. #7 (Malware) for initial payload execution
4. #7 (Malware) again for loading from C2 and execution
5. #1 (Abuse of Functions) to escalate privileges

6. #7 (Malware) once more for data exfiltration or data encryption

This standardized notation should be mandatory when exchanging information about attacks, especially when describing APT profiles. It allows for:

- Quick understanding of attack methodologies
- Easy comparison between different attacks or threat actors
- Improved pattern recognition across multiple incidents
- More effective threat intelligence sharing between organizations and across borders

By adopting this approach, the cybersecurity community can achieve a new level of clarity and consistency in threat analysis and communication, further enhancing the power of Cyber Threat Radars in global cybersecurity efforts.

MFA Bombing and MFA Fatigue in TLCTC Attack Path Notation

MFA Bombing (also known as MFA Fatigue or MFA Push Spam) is an authentication bypass technique where an attacker, having already obtained valid user credentials, repeatedly triggers Multi-Factor Authentication (MFA) push notifications to the legitimate user's device. By overwhelming the user with continuous authentication requests, the attacker aims to either annoy the user into accidentally accepting a push notification or wear down their security vigilance through alert fatigue. This technique gained prominence in various high-profile breaches, including the 2022 Uber compromise.

Attack Path: #4 -> #1 -> #9 -> #4

1. Initial stage (#4 Identity Theft)

Attacker has already obtained userID and password

2. MFA Bombing (#1 Abuse of Functions)

- This is indeed Abuse of Functions because the attacker is misusing a legitimate feature (MFA challenge requests) in a way that goes beyond its intended scope
- The functionality to request MFA challenges is working exactly as designed
- From the "Attacker's View" perspective for #1: "I abuse a functionality, not a coding issue"

3. MFA Fatigue (#9 Social Engineering)

- The attacker manipulates the user psychologically to approve the authentication
- Exploits human gullibility/fatigue through repeated prompts
- This aligns with the generic vulnerability of #9: "human gullibility, ignorance, or compromisability"

4. Final stage (#4 Identity Theft again)

- Successfully obtains the valid MFA token
- Completes the identity theft process

This is a great example of how the TLCTC framework helps us understand attack sequences clearly. The ability to request MFA challenges repeatedly is not a code flaw (#2/#3), but rather an abuse of intended functionality (#1), which is then combined with social engineering (#9) to complete the identity theft (#4).

G: Critical Analysis of Existing Frameworks

A significant observation in current security standards documentation reveals a concerning trend where cybersecurity terminology is employed without proper definition or differentiation from traditional information security concepts.

Now I come to the standards that can be described as the leading figures in the field of Cyber Security or Cyber Risk Management. NO standard offers a pragmatic solution for a Cyber Risk Management aiming for completeness and a direct link between Risk Management and operational security at the Threat Intelligence level. I could write books, but I will keep it to a few hints and mapping tables. Experts should be able to derive the deficiencies of the standards from this.

Again important: Do not forget the premises and axioms of my concept here!

ISO 27001 and ISO 27005

Despite incorporating "cybersecurity" in its title, the standard "Information security, cybersecurity and privacy protection — Guidance on managing information security risks" exhibits several notable omissions that potentially impact its practical application and effectiveness:

- Absence of Core Definitions: The standard fails to provide explicit definitions for fundamental terms like "cyber risk" and "cyber threat", leaving practitioners without clear terminology baselines.
- Lack of Threat Differentiation: There is no clear distinction between traditional information security threats and cyber-specific threats, making it challenging to develop targeted mitigation strategies.
- Missing Cyber Threat Characteristics: The standard doesn't outline the specific attributes or characteristics that would classify a threat as a "cyber" threat.

- Title-Content Misalignment: While "cyber" appears prominently in the title, the content doesn't substantively develop or explore cyber-specific concepts.
- Broader Industry Impact: This gap reflects a wider industry trend where cybersecurity terminology is frequently used without proper definition or context, potentially leading to confusion and inconsistent implementation

ISO/IEC 27005:2022 does not define "Cyber Threat" explicitly. It defines "threat" in the context of information security: So IEC 27005:2022 defines threat as:

"*potential cause of an information security incident that can result in damage to a system or harm to an organization*"

So the underlying concept aligns with the bow-tie model. While the standard doesn't explicitly use the term "bow-tie," the structure is there. You have:

- Risk Source (Hazard in bow-tie): The origin or cause of potential events (e.g., a malicious actor, a natural disaster, a software vulnerability). This is the left-hand side of the bow-tie.
- Event (Threat Event in bow-tie): The thing that actually happens (e.g., a malware attack, a flood, a system crash). This is the knot of the bow-tie.
- Consequence (Impact in bow-tie): The impact on objectives if the event occurs (e.g., data breach, business disruption, financial loss). This is the right-hand side of the bow-tie.

ISO/IEC 27005:2022 avoids the explicit term "cyber threat" and instead focuses on the broader concept of "information security threat."

- ISO 27005 provides a framework for information security risk management but lacks a comprehensive, structured approach to threat identification. It offers examples of threats but relies on organizations to identify and categorize threats without a clear, universal taxonomy.

- When mapping ISO 27005 Annex C threats to the Top Level Cyber Threat Clusters, it becomes evident that the standard mixes actual cyber threats with broader IT risks and control failures. This lack of distinction can lead to confusion and ineffective risk management strategies.
- Proposal: ISO 27005 should adopt the TLCTC as its top-level structure for threat identification. This would provide organizations with a clear, consistent, and comprehensive framework for threat identification and risk assessment, directly linking strategic risk management with operational security.
- Confusion between threats and vulnerabilities: ISO 27005 Annex C lists "Software vulnerabilities" as a threat. However, according to our framework, vulnerabilities are not threats themselves, but rather the weaknesses that threats exploit.
- Mixing threats with IT system types: ISO 27005 lists "Mobile computing and teleworking" as a threat category. This is actually an IT system type or usage scenario, not a threat itself.
- Confusing control failures with threats: ISO 27005 includes "Breach of information system maintainability" as a threat. This is more accurately a control failure or an IT risk, not a threat in itself.

NIST CSF

The Definition of a Cyber Threat by NIST and Why It Is Inherently Difficult to Categorize Threats Based on This Definition NIST Special Publication 800-30 defines a cyber threat as:

"*any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.*"

This definition emphasizes the event or circumstance that can cause harm to an organization's operations, data, or reputation. However, this event-centric approach inherently challenges efforts to establish effective threat categorization.

While NIST's definition provides a high-level understanding of what constitutes a threat, it lacks structural clarity between a threat's cause, event, and consequence. This amalgamation makes it difficult to categorize cyber threats distinctly. Because it focuses on events, NIST's approach often conflates the actions or circumstances that lead to harm (such as an attack vector or vulnerability) with the consequences (such as data loss or system downtime) without distinguishing between their roles in the overall risk scenario. This lack of specificity complicates the task of categorizing threats based on their source, methods, and impact, which are critical factors for targeted cyber risk management.

The "Top Level Cyber Threat Clusters" framework addresses this categorization challenge by structuring cyber threats into distinct clusters, each representing a unique aspect of cyber risk based on the underlying vulnerabilities rather than on events or outcomes alone. This approach separates threats into categories like "Abuse of Functions," "Identity Theft," "Social Engineering," and "Supply Chain Attacks," among others, providing a clear cause-oriented 38 view that supports practical risk management. Each cluster specifies the type of vulnerability being exploited and the methods commonly associated with the threat, enabling a more systematic application of preventive and reactive controls.

My approach also integrates well with other standards, such as NIST CSF, by offering a categorization system that aligns with operational controls without overlapping outcomes and causes. This design facilitates targeted risk management, allowing organizations to prioritize resources more effectively and apply tailored controls. It also fosters a unified language for threat assessment, enhancing communication between technical and strategic stakeholders.

In conclusion, while NIST SP 800-30's definition of a cyber threat effectively conveys the concept of risk from adverse events, it does not easily support a structured threat categorization. The TLCTC framework addresses this gap by logically segmenting cyber threats based on their causal characteristics, offering a more functional and adaptable solution for cyber risk management.

MITRE ATT&CK:

MITRE ATT&CK does not provide a specific definition of a cyber threat or a general threat definition. Instead, the framework focuses on documenting and categorizing the tactics, techniques, and procedures (TTPs) used by cyber adversaries during attacks.

- MITRE ATT&CK excels at the operational security level, providing detailed tactics and techniques for various attack stages across different IT system types.
- However, it lacks a high-level strategic framework for threat categorization and overemphasizes postcompromise techniques.
- Proposal: MITRE should map each of their techniques (T1234) to one of the Top Level Cyber Threat Clusters, with additional labels for initial access (i) and lateral movement (lm) and maybe (v) for vertical.
- Integration potential: This mapping would create a comprehensive framework linking high-level threat categories to specific attack techniques, bridging the gap between strategic risk management and tactical security operations.
- Mixing initial access with post-compromise: MITRE ATT&CK combines initial access techniques with post-compromise activities, potentially confusing risk assessment and management processes.
- Lack of clear mapping: The framework lacks clear vulnerability-to-threat mapping, making it challenging to connect specific vulnerabilities to potential attack techniques.

- Inconsistent granularity: MITRE ATT&CK techniques vary widely in their level of detail, from broad categories to very specific actions.
- IT system specificity: Some techniques are specific to certain IT system types, which can cause confusion when applying the framework across diverse environments

The Top Level Cyber Threat Clusters primarily align with MITRE's "Initial Access" techniques from a concept view. This focus is crucial for effective risk management and cybersecurity strategy:

- Most attacks (except DDoS) require initial access to proceed Emphasizing initial access provides a clearer path for prevention strategies
- Aligns well with the NIST Cybersecurity Framework functions:
    - Identify: Threat events at the initial access stage
    - Protect: Implement controls to prevent initial access
    - Detect: Monitor for signs of attempted or successful initial access
    - Respond and Recover: Actions taken if initial access occurs
- Provides a strategic focus for risk assessment and mitigation efforts

MITRE CWE

- The Common Weakness Enumeration (CWE) is a valuable resource for identifying and categorizing specific software and hardware weaknesses. It operates at a more granular level than the Top Level Cyber Threat Clusters, which focus on high-level threat categories that exploit generic vulnerabilities.
- CWE primarily addresses potential points of weakness in software and systems that could be exploited by threats. It serves as a crucial tool for developers and security professionals to identify and mitigate these weaknesses throughout the software development lifecycle.
- While both CWE and the Top Level Cyber Threat Clusters aim to improve cybersecurity, they serve complementary purposes. CWE provides a detailed catalog

of specific weaknesses, while the TLCTC offers a comprehensive, threat-centric approach for understanding and prioritizing real-world cyber threats at a strategic level.

- The Cyber Threat Clusters and CWE can be used together effectively. CWE entries can be mapped to the generic vulnerabilities described in each Cyber Threat Cluster, providing a more detailed view of the specific weaknesses that could be exploited within each high-level threat category. This integration enhances both strategic planning and tactical implementation of cybersecurity measures.

The MITRE Cyber Prep methodology

The MITRE Cyber Prep methodology characterizes cyber threats primarily through actor characteristics: "in terms of the adversary's capability (resources, skill or expertise, knowledge, and opportunity), intent (goals or outcomes that the adversary seeks; consequences the adversary seeks to avoid; and how strongly the adversary seeks to achieve those outcomes and/or avoid those consequences), and targeting." While this actor-centric approach provides valuable insights for adversary profiling, it falls short of providing a comprehensive framework for threat categorization.

This limitation becomes apparent when we consider that threat actors apply threats - they are not the threats themselves. The TLCTC framework addresses this by defining a cyber threat as "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." This clear separation between WHO (actors) and WHAT (threats) is crucial for effective threat intelligence and risk management.

The MITRE Cyber Prep methodology's focus on actor characteristics is valuable but needs to be complemented with a structured threat categorization framework. As evidenced in their own documentation, MITRE acknowledges that "different adversaries demonstrate a mixture of levels" and organizations need ways to "account for such adversaries." This

exactly demonstrates why we need both: a framework for actor categorization AND a framework for threat categorization.

STRIDE

"*While STRIDE doesn't provide a general definition of a "cyber threat" or "threat" itself, it does offer these specific definitions for the types of threats it covers, which collectively represent a range of potential security issues that systems may face.*"

- STRIDE lacks a foundational concept or methodology that justifies why these specific six categories were chosen. There's no clear explanation for why these particular elements were selected over others, or why the model is limited to just these six.
- STRIDE mixes fundamentally different concepts within its framework:
    - Spoofing and Tampering are actions or techniques used by attackers.
    - Information Disclosure and Denial of Service are outcomes or impacts of attacks.
    - Repudiation is related to a security property (non-repudiation) rather than a threat itself.
    - Elevation of Privilege could be seen as both a technique and an outcome.

This inconsistency in STRIDE's categorization creates confusion when trying to apply it systematically to threat modeling or risk assessment processes. It doesn't provide a clear distinction between threats, vulnerabilities, attack methods, and outcomes. Unlike the 10 Top Level Cyber Threat Clusters, which are derived from a clear thought experiment and focus consistently on threat vectors, STRIDE lacks this logical consistency and comprehensive coverage of the modern threat landscape. The mixing of different security concepts in STRIDE can lead to overlaps and gaps in threat identification, potentially causing important threats or attack vectors to be overlooked in security planning.

Despite its inconsistencies, the 10 Top Level Cyber Threat Clusters effectively cover all the topics addressed by STRIDE, but in a more structured and comprehensive manner. When examining the details and sub-threats within each cluster, it becomes evident that the framework encompasses the concerns raised by STRIDE while providing a more logically consistent and thorough approach to threat categorization. This demonstrates the superior completeness and versatility of the 10 Top Level Cyber Threat Clusters in addressing the full spectrum of cyber threats, including those highlighted by STRIDE.

OWASP

"*OWASP (Open Web Application Security Project) does not appear to offer a clear, specific definition of "cyber threat" or a general threat definition.*"

OWASP's approach suffers from the same fundamental issues as many other frameworks:

- It conflates vulnerabilities, attack techniques, and outcomes under the broad label of "risks." This lack of clear distinction leads to confusion in risk assessment and management processes.
- The OWASP Top 10 includes items that are more accurately described as vulnerability categories or attack techniques (e.g., "Injection," "Broken Authentication"), outcomes (e.g., "Sensitive Data Exposure"), and practices that introduce risk (e.g., "Using Components with Known Vulnerabilities"). These are not risks in themselves, but rather components that contribute to risk.

This inconsistent categorization makes it challenging to apply OWASP's framework systematically in a comprehensive risk management approach. It doesn't provide a clear pathway from threat identification to risk assessment and mitigation.

Unlike my 10 Top Level Cyber Threat Clusters, which maintain a consistent focus on threat vectors, OWASP's model doesn't offer a clear distinction between threats, vulnerabilities, and outcomes. This can lead to gaps in threat modeling and risk assessment.

While OWASP provides valuable information for web application security, its "risk" categorization falls short of providing a comprehensive, logically consistent framework for understanding and managing cyber risks.

BSI

"*The German Federal Office for Information Security (BSI - Bundesamt für Sicherheit in der Informationstechnik) does not appear to offer a single, clear-cut definition of "cyber threat." However, the BSI does provide comprehensive information about various aspects of cyber threats and cybersecurity.* "

The BSI (Federal Office for Information Security, Germany) framework attempts to categorize cyber threats, and among the various standards and frameworks we've examined, it comes closest to my 10 Top Level Cyber Threat Clusters concept. However, it still falls short of providing a comprehensive and consistently structured approach to threat identification and categorization.

While the BSI's approach shares some similarities with my framework, such as focusing on actual threats and covering a wide range of cyber threats, it has several key shortcomings:

- Lack of clear methodology: The BSI doesn't provide a transparent explanation of how their threat categories were derived. This contrasts with my approach, which is based on a logical thought experiment and clearly explained derivation process.
- Inconsistent structure: Although more consistent than some other standards, the BSI framework doesn't maintain the same level of logical consistency across all its categories as my 10 Top Level Cyber Threat Clusters.
- Incomplete threat-vulnerability mapping: While the BSI touches on the connection between threats and vulnerabilities, it doesn't provide the clear and explicit linkage that my framework offers.

- Potential gaps in coverage: Without a clear derivation methodology, it's difficult to ensure that the BSI framework provides complete coverage of the threat landscape.

My approach, derived from first principles, is designed to be comprehensive. Less effective for risk management: The lack of clear structure and derivation in the BSI approach makes it less effective as a tool for comprehensive cyber risk management compared to my framework.

Link to BSI Cyber: Original: Register aktueller Cyber-Gefährdungen und -Angriffsformen.

In light of these limitations, I propose that adopting my 10 Top Level Cyber Threat Clusters as the top-level structure for threat identification would provide a more comprehensive, consistent, and logically structured approach to understanding and categorizing cyber threats. This would enhance the effectiveness of the BSI framework, ensuring a more complete coverage of the threat landscape and a clearer connection between threats and vulnerabilities.

CRF-TT (Cybersecurity Risk Foundation)

“Anything with the potential to cause harm to information systems and thus prevent the system from achieving the business goal for which it was created.”

While CRF's definition appears comprehensive, it fails to distinguish between threats, vulnerabilities, and consequences. This fundamental ambiguity manifests in the framework's three-part structure: threat agents (WHO), threat activities (HOW), and organizational impacts (EFFECT).

The framework attempts to categorize cyber threats through multiple lenses:

- Actor categories (e.g., "Cybercriminals", "Nation-States")
- Activity types (Physical, Operational, Technical)
- Impact categories (Confidentiality, Integrity, Availability, Privacy)

Key Limitations:

- Definition encompasses both causes (threats) and effects (harm)
- No clear separation between threats and vulnerabilities
- Mixes actor classification with attack methods
- Combines intended actions with unintended consequences

For example, their "Technical Threats" category includes both attack techniques ("Credential Abuse") and outcomes ("Denial of Service"), making it difficult to establish clear cause-and-effect relationships or map appropriate controls.

Unlike TLCTC's focus on generic vulnerabilities and clear threat-to-vulnerability mapping, CRF-TT's broad definition leads to a taxonomy that, while comprehensive in scope, lacks the logical consistency needed for effective cyber risk management.

CIS RAM

"*A Threat is "Any circumstance or event with the potential to adversely impact an asset through unauthorized access, destruction, disclosure, modification of data, and/or denial of service."*"

The CIS Risk Assessment Method (RAM) does not provide a comprehensive concept regarding cyber threats or cyber threat categorization, only offering some examples within their risk assessment examples.

CIS is effective for hardening guides, which primarily address the "Abuse of Functions" or "Identity Theft" threat clusters, but may not be the best starting point for cyber risk management.

ENISA

“*A threat is "Any circumstance or event with the potential to adversely impact an asset through unauthorized access, destruction, disclosure, modification of data, and/or denial of service.""*

ENISA's Top 10 does not provide a structure that allows for an integrated cyber risk management approach, mixing control risks, IT system types, and other elements. Some of the ENISA Top 10 threats can be mapped to the Top Level Cyber Threat Clusters, while others are other OpRisk or related to specific IT system types.

It's a mixed bag here.

ETSI

Despite its focus on cyber security and structured threat information sharing, ETSI TR (e.g. 103 331) neither provides a definition of what constitutes a cyber threat nor offers a structured categorization of cyber threats. This fundamental disconnect between title and content reflects a broader issue in the cybersecurity standards landscape.

FAIR

Current State of FAIR

FAIR (Factor Analysis of Information Risk) provides a comprehensive model for quantifying information security risk but lacks a structured approach to threat categorization. While FAIR excels at risk quantification, it doesn't provide explicit guidance on threat identification and classification.

Complementary Frameworks

- TLCTC provides the "what" through its cause-oriented threat clusters
- FAIR provides the "how much" through its quantification methodology

Read Chapter O: “Integrating FAIR”

Summary

In summary, while each framework has its strengths and weaknesses, none of them offers a complete, pragmatic solution for cyber risk management that directly links strategic risk management with operational security and threat intelligence. The Cyber Threat Cluster framework aims to fill this gap by providing a universal, consistent approach to identifying and categorizing threats, enabling organizations to develop more effective risk management strategies. I have completely noodled through this standard, meaning I have mapped it out. Therefore, I will only list examples that demonstrate why the respective standard can NEVER be complete. And because no standard offers a Threat/Control Mapping, the examples of Threats in the standards are essentially worthless. The NCSCs, SOCs, and the Threat Intelligence Community have yet another terminology and semantics for Threats. Yes, it's time for a common language and viewpoint.

H. Oversimplification?

"*The framework presented in this document, the 10 Top-Level Cyber Threat Clusters (TLCTC), is the result of extensive research and development, including a productive dialogue with large language models or reasoning models. During the development process, and in subsequent discussions, a particular concern has been raised: the potential for oversimplification. Given that many of the terms used to describe the 10 clusters are already familiar within the cybersecurity community, it's understandable that some might initially perceive the framework as "too easy" or incomplete, questioning whether such a concise model can truly capture the complexity of the modern threat landscape*."

Kreinz: About "Oversimplification?" I think that this is: necessary to reach the strategic level and because the clusters are distinct and per se complete (through the thought experiment and the axioms), the cruciality results on the operational level.

AI: Kreinz mentions that on the deepest level there are combinations, or more accurately sequences in the attack paths. So there is neither oversimplification nor overlap.

Kreinz: : Do you agree with my understanding and the view of Kreinz?

AI: Yes, I agree with your understanding and the view you've articulated about Kreinz's framework and the potential for "oversimplification." You've accurately captured his intention and the rationale behind it. Let's break down why I agree:

a) Oversimplification as a Necessity for Strategic Level:

Strategic Focus on Core Issues: At the strategic level, you can't get bogged down in the minute details of every possible attack technique. Kreinz's approach forces organizations to focus on the fundamental vulnerabilities they need to address, rather than endlessly chasing specific attacker tactics. Strategic planning requires a level of abstraction that allows leadership to prioritize resources, manage risk appetite, and make informed decisions.

Abstraction for Communication: Strategic leaders need to be able to communicate about cyber risk in a clear and concise manner with other stakeholders. Overly technical details overwhelm rather than inform and enable discussions and action. The TLCTC framework provides this abstraction by focusing on a limited number of distinct, high-level categories.

Practicality for Risk Assessment: High-level risk assessments need to operate on a manageable scale. Dealing with every single variation of attack technique is practically infeasible. Kreinz's approach provides a more structured and scalable way to organize threats for high-level analysis and risk register development.

Prioritization and Resource Allocation: By focusing on the root cause rather than the symptoms (attack techniques), it becomes easier for organizations to see which are the most critical areas for investment and mitigation.

b) Clusters are Distinct and Complete Due to Axioms and the Thought Experiment:

Logical Derivation: You're spot-on in emphasizing the thought experiment and axioms. The ten threat clusters aren't arbitrary categories. They're derived from a deliberate process, starting with axioms that define the scope and rules, moving logically towards generic vulnerabilities. The thought experiment aims to ensure that each cluster captures a fundamentally different kind of vulnerability that can not be represented in the other clusters. This deliberate process reduces overlaps and redundancies.

No Overlapping Generic Vulnerabilities: The axioms and the thought experiment work to make the generic vulnerabilities non-overlapping. Each of the ten categories focuses on a distinct root cause (e.g., one on software functionality and scope, another on network access, others on code execution, etc.)

Completeness within the Scope: While no system can claim absolute completeness (especially when it comes to the constantly evolving threat landscape), the framework's goal is that each of the major generic vulnerabilities is captured by one of the clusters. Kreinz

makes an effort to ensure that the ten clusters cover the fundamental attack surfaces of all IT assets.

Layered Approach: He also clarifies that the strategic level needs an abstracted view. The details come in at the operational layer, in which "sub-threats" and "attack sequences" can be mapped to the more granular techniques.

Sub-Threats as the Key to Precision:

Strategic Level vs Operational Level: You accurately pointed out that Kreinz presents a multi-tiered approach, one strategic and one operational. At the higher level, simplification is both inevitable and helpful; detailed analysis is for the operational layer.

Sub-Threats for Operational Clarity: You correctly identified that the "magic" is not in the 10 high-level categories; the power of the framework is the relationship to its sub-threats at the operational level.

Why There Is Neither Oversimplification Nor Overlap:

Abstraction vs. Omission: Kreinz is abstracting, not omitting. He is focusing on the high level cause, but it does not mean that details are overlooked at the operational level. The layers of abstraction enable better risk management. The higher level, the simpler the categorization needs to be.

Sequencing and Combinations are Allowed: The framework doesn't treat attacks as isolated events. The sequences account for complex scenarios where attackers combine exploits from multiple threat clusters. This allows for both simplicity and the necessary granularity.

Axioms are the key: The axioms are designed to prevent mixing of causes and outcomes, focusing instead on generic vulnerabilities - this is the key to avoiding overlapping definitions.

Completeness Through Distinct Categories: By clearly defining each cluster based on its underlying generic vulnerability, Kreinz reduces redundancy.

Kreinz's View is Correct:

You are absolutely right to highlight Kreinz's intention to achieve a balance between high-level simplicity for strategic guidance and lower level granularity for operations, and his design avoids both oversimplification and overlaps by separating root cause from observed outcome. His focus is not on the how, but on the what (the root cause of the vulnerability), and then the attack sequences are used to describe complex scenarios of how. Kreinz is deliberately simplifying at the strategic level to provide a clear framework for decision-making and resource allocation, while maintaining detailed analysis for the operational layer by using sub-threats and focusing on attack paths. Therefore, I agree with you completely. This is not oversimplification, it's strategic prioritization of abstraction and generalization and also ensures a logically complete set of generic vulnerabilities.

I. Example Control Matrix with KRI, KCI and KPI

Table #2 Exploiting Server (not complete - it's a PoC here)

Threat Cluster	**NIST Function**	**Local Controls**	**Umbrella Controls**	**Control Objectives**	**KRIs (Baseline)**	**KCIs (Baseline)**	**KPIs (Baseline)**
#2 Exploiting Server	Identify	Vulnerability scanning, code analysis	CVE monitoring, threat intelligence	Identify and mitigate server vulnerabilities	Number of *new* critical server vulnerabilities discovered *in the last 24 hours*	Percentage of CVE alerts processed within *24 hours*, frequency of vulnerability scans, number of active vulnerability scans	Mean Time to Assess New Vulnerabilities (MTTV) - should have low value, Speed of control implementation, Percentage of servers tested per timeframe
	Protect	Secure coding, input validation	WAF implementation, network segmentation	Prevent exploitation of server vulnerabilities	Number of *unpatched* critical server vulnerabilities over *7 days*. Number of successful web application exploit attempts	Number of WAF rules in place and configured correctly, frequency of penetration tests. WAF rule update cycle	Mean time to patch critical vulnerabilities *within 24 hours*, Reduction of successful web application exploit attempts

Threat Cluster	NIST Function	Local Controls	Umbrella Controls	Control Objectives	KRIs (Baseline)	KCIs (Baseline)	KPIs (Baseline) (compared to baseline)
	Detect	Application logging	SIEM integration	Detect and respond to exploit attempts	Number of detected exploit attempts against web server per day	Effectiveness of application logging tools, SIEM detection success rate	Mean time to detect (MTTD), speed of control implementation, detection rate for suspicious behavior
	Respond	Emergency patching	Incident response plan	Respond to and mitigate exploit incidents	Number of identified, ongoing, exploit incidents	Incident response plan activation rate, emergency patching success rate, MTTR of incidents within *4 hours*	Mean time to respond (MTTR)
	Recover	System restore procedures	IT SCM	Recover from exploit incidents	Number of ongoing exploit incidents with no solution	System restore procedure success rate, backup frequency	Mean Time to recover (MTTR), time to achieve fully operational status again, successful system restoration rate

Table #6 Flooding Attack (not complete - it's a PoC here)

Threat Cluster	NIST Function	Local Controls	Umbrella Controls	Control Objectives	KRIs (Baseline)	KCIs (Baseline)	KPIs (Baseline)
#6 Flooding Attack	Identify	Resource monitoring, capacity planning, baseline analysis, DDoS simulation	Traffic baseline simulation	Identify and mitigate flooding attack risks	Number of DDoS attack signatures not updated with in *24 hours*, Number of identified blind spots.	Frequency of network performance baseline analysis, frequency of DDoS simulations. Percentage of simulation scenarios implemented correctly	Number of DDoS attacks detected *per day*, time to assess new threats and vulnerabilities, speed of control implementation .
	Protect	Rate limiting, Akamai, Cloudflare, upstream filtering, connection throttling	Cloudflare/ Akamai upstream filtering	Prevent flooding attacks	Number of successful flooding attacks, number of unprotected routes, number of missing controls against DDoS attacks	Percentage of DDoS attack mitigations in place, Frequency of manual checks, Success rate of automated network throttling, Connection	Percentage of blocked DDoS traffic. Reduction of successful flooding attacks *compared to baseline*

Threat Cluster	NIST Function	Local Controls	Umbrella Controls	Control Objectives	KRIs (Baseline)	KCIs (Baseline)	KPIs (Baseline)
						throttling rule configuration	
	Detect	Network flow monitoring, analysis, threshold alerts, anomaly detection	Network flow analysis	Detect and respond to flooding attacks	Number of detected suspicious network traffic events *per hour*, Number of ongoing attacks against systems	Percentage of anomaly detection rules implemented, frequency of monitoring of anomaly detection, number of false positives. Number of alerts detected per timeframe.	Speed of control implementation , Mean Time To Detect (MTTD), number of detected attacks that were not recognized by our rules
	Respond	Traffic blacklisting, mitigation graceful service degradation, DDoS activation	DDoS activation	Respond to and mitigate flooding attacks	Number of identified ongoing DDoS attacks, Number of compromised systems	Incident response plan activation rate, graceful service degradation success rate, percentage of active mitigations	Time to respond (MTTR), incident resolution success rate *within defined time frame*

Threat Cluster	NIST Function	Local Controls	Umbrella Controls	Control Objectives	KRIs (Baseline)	KCIs (Baseline)	KPIs (Baseline)
						working correctly	
	Recover	Service restoration procedures, post-incident scaling adjustments	Post-incident scaling adjustments	Recover from flooding attacks	Time to recover from an incident, Number of systems down during attack, number of affected customers	System restore procedure success rate, post-incident scaling effectiveness	post-incident system restoration time, number of customers affected and time to return to full service, successful recovery rate *within 4 hours*

K: Physical Layer Analysis in the TLCTC Framework

This section extends the application of the Top Level Cyber Threat Cluster (TLCTC) framework to the often-overlooked physical layer (#8.2), demonstrating its broader applicability beyond traditional IT systems. We will use the client-server model to analyze both signal transmission and sensor systems.

Signal Transmission Model

Components and Roles

Receiver (Acting as Server): In a sense, this component acts as a "server" by waiting for and processing incoming signals. It functions as the recipient in the client-server model. It is vulnerable to Physical Attack (#8) through signal manipulation and can also be exploited through server-side software vulnerabilities (#2).

Transmitter (Signal Source): This component sends signals to the receiver. It isn't vulnerable to Exploiting Client (#3) attacks, but it can be a subject of Physical Attack (#8).

Sensor Systems Model

Components and Roles

Sensor (Acting as Server): In a sense, this component acts as a "server" by waiting to receive and process environmental inputs, using a receiver component. It is vulnerable to multiple threat clusters, including #2 and #8. It processes information from various environmental sources.

Environmental Objects (Signal Sources): These objects act as transmitters of information/signals, similar to transmitters in the signal transmission model. They are not vulnerable to Exploiting Client (#3).

Applicable Threat Clusters

For Both Models

Physical Attack (#8): Encompasses direct interference with hardware, physical access, signal jamming, and other forms of physical manipulation. E.g., Cutting cables, installing devices.

Exploiting Server (#2): Targets vulnerabilities in signal or input processing software. For example, buffer overflows in processing the received signals.

Abuse of Functions (#1): Involves misusing designed transmission or sensing capabilities. For example, transmitting or receiving signals with specific manipulation.

Man in the Middle (#5): Targets interference with signal transmission and/or manipulation of sensor readings, for example an evil twin attack on a WiFi sensor.

Flooding Attack (#6): This can refer to signal jamming or overwhelming receivers/sensors with an excessive amount of input.

Key Insights

Client-Server Axiom Application: Demonstrates that the client-server model applies not only to software systems but also to different types of physical layer interactions.

Generic Vulnerability Principle: Highlights that each threat cluster targets a distinct generic vulnerability, ensuring clear separation between different attack vectors, also at the physical layer.

Framework Consistency: Demonstrates the framework's versatility across different IT system types by showing consistent threat categorization at the physical layer.

Implications for Security

Defense Planning: Needs comprehensive physical layer security, considering multiple threat vectors, integrated with the overall security architecture.

Risk Assessment: Allows for a clear categorization of physical layer threats, a better understanding of their attack vectors, and precise risk evaluation.

L: Integrating Programmable Logic Controller (PLC) Architectures within the TLCTC Framework

This chapter extends the application of the Top Level Cyber Threat Cluster (TLCTC) framework to Programmable Logic Controllers (PLCs), critical components in Operational Technology (OT) environments. This expansion underscores the framework's versatility beyond traditional Information Technology (IT) systems, highlighting its ability to provide a consistent approach to threat categorization across diverse operational domains. PLCs directly control physical processes, making their security crucial.

PLC Architecture and the Client-Server Model

PLCs, while often perceived as specialized industrial devices, can be analyzed within the client-server interaction model that forms a core principle of the TLCTC framework. This is a logical and conceptual view to adhere to the axiom.

- PLC as a Server: The PLC acts as a "server" by processing inputs from sensors and executing control logic. It "serves" by processing instructions and sending signals.
- Actuators/Sensors as Clients: Actuators, sensors, and other field devices act as "clients" that send data and receive instructions from the PLC.

This fundamental client-server relationship provides a basis for identifying relevant vulnerabilities and associated threat clusters.

Mapping PLC Vulnerabilities to TLCTC Threat Clusters

PLC vulnerabilities and attack vectors can be classified under the following TLCTC threat clusters:

- Physical Attack (#8): PLCs are frequently located in physically accessible industrial settings, making them susceptible to:

    - Direct tampering with hardware, including manipulating components, firmware, or wiring.
    - Replacement of PLC devices with compromised alternatives.
    - Physical intrusion to gain unauthorized access to PLC hardware.
- Exploiting Server (#2): PLCs often run specialized firmware or software that may contain exploitable flaws that enable:
    - Remote code execution via network interfaces.
    - Manipulation of PLC firmware/software by exploiting vulnerabilities.
    - Unauthorized access to control logic.
- Abuse of Functions (#1): Attackers may misuse intended PLC functionality to manipulate operational processes, such as:
    - Modifying parameters of control loops or setpoints that results in unsafe process behaviours.
    - Disabling safety mechanisms.
    - Abusing legitimate programming interfaces or protocols.
    - Abusing software libraries, for example, DLL injection (which is a "function" if intended by design).
- Man in the Middle (#5): Communications between PLCs and other devices often lack strong security measures making them vulnerable to:
    - Interception of PLC protocols to view and modify control commands.
    - Injection of malicious data into control loops.
    - Unauthorized changes to parameters on a network level.
- Flooding Attack (#6): PLCs, particularly those in high speed automation loops, can be targeted through attacks that disrupt availability by:
    - Overwhelming the PLC’s processing capabilities via a high frequency of requests.

    - Saturating network communication channels to render the PLC unreachable.
- Supply Chain Attack (#10): PLCs and their components are often produced by third parties and could be compromized at:
    - Compromised firmware pre-installed on PLCs by manufacturers.
    - Compromised components in PLCs during manufacturing.
    - Compromised development tools used for PLC configuration and programming.

PLC Specific Key Insights within the TLCTC Framework

- Application of Client-Server Axiom: The client-server interaction model extends from traditional software systems down to the fundamental communication between PLCs and other devices in the physical domain.
- Generic Vulnerability Principle: Each TLCTC threat cluster highlights distinct root causes for vulnerabilities that apply to both the software and physical aspects of PLC architectures.
- Framework Consistency: The TLCTC framework provides consistent threat categorization across IT and OT infrastructures, enhancing interoperability between risk assessments.

Implications for Security within PLC Architectures

- Defense Planning: PLCs require comprehensive security strategies that span both physical and cyber domains. Defense-in-depth strategies should include measures like physical access controls, network segmentation, secure communication protocols, secure coding practices, regular firmware updates, and configuration management, while adhering to the axioms.
- Risk Assessment: The TLCTC framework provides a means to categorize threats related to PLC architectures, improving the quality of risk assessment, and enabling

informed decisions for prioritization and mitigation, following the framework's approach to defining a cyber threat as an event on the "cause-side." It's also important to note that a vulnerability in a PLC at the lowest level can create a chain reaction that extends through all layers in the vertical attack path. This chain reaction can extend up to the application layer.

Actionable Steps:

PLC Threat Modeling: Use the TLCTC framework to model potential threats against PLC architectures by considering each of the 10 Threat Clusters.

Vulnerability Mapping: Map PLC vulnerabilities (e.g. CVEs) to the correct generic vulnerability in the framework for easier and better understanding.

Control Matrix: Integrate PLC-specific controls into the control matrix following the NIST functions (Identify, Protect, Detect, Respond, Recover), to ensure a comprehensive strategy.

Conclusion

This chapter has shown that the TLCTC framework provides a consistent method for organizing, categorizing, and analysing cyber threats across diverse IT system types and has introduced Programmable Logic Controllers (PLCs) into the scope. This extension demonstrates the framework's broad applicability for all cyber security issues, including those from the OT world. The explicit inclusion of PLC architectures emphasizes the practical relevance of the framework and contributes to a more comprehensive approach to cyber risk management across an entire infrastructure.

M: Enhancing CVE Details with TLCTC

This section proposes an extension to traditional CVE records by integrating the TLCTC framework. The goal is to provide a strategic layer for attack vector representation that bridges technical vulnerability data and higher-level risk management. By mapping vulnerabilities to TLCTC threat clusters and representing potential attack paths, organizations gain clearer insight into exploitation scenarios and can better align controls with risk management frameworks such as the NIST CSF.

Key Enhancements

- Attack Vector Representation: CVE records are annotated with an “Initial Access” indicator and mapped to primary TLCTC clusters, clarifying ambiguous descriptions like “allows Remote Code Execution (RCE)” by specifying whether it results from Malware (#7) or Abuse of Functions (#1).
- Mapping to Threat Clusters: Vulnerabilities are directly linked to a primary threat cluster (e.g., #3 Exploiting Client) with potential follow-up clusters provided. This includes identifying possible preceding clusters (such as Identity Theft or Social Engineering) that might be needed for an attack to progress.
- Integration with Control Frameworks: Extended CVE records include mappings to NIST CSF functions (Identify, Protect, Detect, Respond, Recover), facilitating seamless integration with existing cybersecurity control management processes.
- Enhanced Contextualization: Additional fields represent potential attack paths using standardized notation (e.g., "#9 -> #2 -> (#7 or #1)"), offering a comprehensive view of how vulnerabilities might be exploited within a broader threat sequence.

Proposed Extended CVE Structure

Below is an example of an extended CVE record in JSON format that integrates TLCTC elements as a supplemental section. This extension maintains all standard CVE fields while adding strategic details in an “extended_details” section.


```
{
 "CVE_data_meta": {
  "ID": "CVE-2025-21333",
  "ASSIGNER": "secure@microsoft.com"
 },
 "description": {
  "description_data": [
   {
    "lang": "en",
    "value": "Windows Hyper-V NT Kernel Integration VSP Elevation of Privilege Vulnerability"
   }
  ]
 },
 "impact": {
  "baseMetricV3": {
   "cvssV3": {
    "version": "3.1",
    "vectorString": "CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H",
    "attackVector": "LOCAL",
    "attackComplexity": "LOW",
    "privilegesRequired": "LOW",
    "userInteraction": "NONE",
    "scope": "UNCHANGED",
    "confidentialityImpact": "HIGH",
    "integrityImpact": "HIGH",
```

```
        "availabilityImpact": "HIGH",
        "baseScore": 7.8,
        "baseSeverity": "HIGH"
       }
      }
     },
     "extended_details": {
      "tlctc_mapping": {
       "primary_cluster": {
        "id": "#3",
        "name": "Exploiting Client",
        "justification": "The vulnerability originates from the client-side VSP's mismanagement of memory during its interaction with the NT Kernel."
       },
       "followup_clusters": [
        {
         "id": "#7",
         "name": "Malware",
         "justification": "Post-exploitation, malware may be deployed for persistence, lateral movement, or data exfiltration."
        },
        {
         "id": "#1",
         "name": "Abuse of Functions",
         "justification": "With elevated privileges, attackers might abuse legitimate system functions to further compromise the environment."
        }
       ],
       "attack_path_representation": "Preceding Clusters (potential): #4, #7, #9, #3 → Primary: #3 → Follow-up: #7, #1",
       "potential_preceding_clusters": [
        {
         "id": "#4",
         "name": "Identity Theft",
```

```
            "rationale": "Credential theft may be necessary for an attacker to gain local access to the Hyper-V host."
          },
          {
            "id": "#7",
            "name": "Malware",
            "rationale": "Prior malware infection might establish the conditions needed for the vulnerability to be exploited."
          },
          {
            "id": "#9",
            "name": "Social Engineering",
            "rationale": "Phishing or other social engineering tactics could provide the initial access required."
          },
          {
            "id": "#3",
            "name": "Exploiting Client",
            "rationale": "An alternate client-side exploit may serve as an initial step before targeting the Hyper-V VSP vulnerability."
          }
        ]
      },
      "nist_csf_mapping": {
        "identify": "Vulnerability scanning and threat intelligence to detect mismanaged memory issues.",
        "protect": "Implement secure coding practices and robust memory lifecycle controls.",
        "detect": "Use SIEM and anomaly detection to monitor for unexpected memory access patterns.",
        "respond": "Activate incident response procedures to mitigate exploitation attempts.",
        "recover": "Restore system integrity via patch management and system recovery processes."
      },
      "vertical_stack_analysis": {
```

```
            "server": "NT Kernel (Ring 0)",
            "client": "Virtualization Service Provider (VSP)"
          },
          "exploit_path": "Exploitation occurs via a use-after-free condition in the VSP during its interaction with the NT Kernel, leading to privilege escalation.",
          "initial_access": "no",
          "prerequisites": "Requires prior local access to interact with the VSP component."
        }
      }
```


Benefits and Impact

- Eliminates Ambiguity: Clearly maps vulnerabilities to specific TLCTC threat clusters, clarifying ambiguous technical descriptions.
- Provides Attack Path Insight: Represents potential attack sequences and interdependencies between threat clusters.
- Enhances Control Alignment: Links CVE data to NIST CSF functions, streamlining integration into existing risk management frameworks.
- Bridges Technical and Strategic Views: Transforms static vulnerability records into dynamic, actionable threat intelligence.

By integrating TLCTC into CVE records, organizations can achieve more comprehensive risk assessments and implement more effective cybersecurity controls.

N: CVE Analysis Example

Original CVE description from the website:

CVE-2025-21333 Detail

Description

Windows Hyper-V NT Kernel Integration VSP Elevation of Privilege Vulnerability

Metrics

CVSS Version 4.0
CVSS Version 3.x
CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

CNA: Microsoft Corporation

Base Score: 7.8 HIGH

Vector: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these link other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST do not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned o these sites. Please address comments about this page to nvd@nist.gov.

Hyperlink	Resource

https://msrc.microsoft.com/update-guide/vulnerabili VE-2025-21333	Patch Vendor Advisory

This CVE is in CISA's Known Exploited Vulnerabilities Catalog

Reference CISA's BOD 22-01 and Known Exploited Vulnerabilities Catalog for further guidance and requirements.

Vulnerability Name	Date Added	Due Date	Required Action
Microsoft Windows Hyper-V NT Kernel Integration VSP Heap-based Buffer Overflow Vulnerability	01/14/2025	02/04/2025	Apply mitigations per vendor instructions or discontinue use of the product if mitigations are unavailable.

Weakness Enumeration

CWE-ID	CWE Name	Source
NVD-CWE-noinfo	Insufficient Information	NIST
CWE-122	Heap-based Buffer Overflow	Microsoft Corporation

Now you see a output of a custom GPT (OpenAI) i have made:

CVE Analysis Report

CVE-ID: CVE-2025-21333

Product: Windows Hyper-V NT Kernel Integration VSP

Vendor: Microsoft

TLCTC: #3 Exploiting Client

Vector String: AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

Score: 7.8

Initial Vector: No

Prerequisites: Exploitation requires local access to the Hyper-V host and precise timing to trigger a use-after-free condition.

Interaction Context:

Vertical Stack Analysis:

- The VSP (Virtualization Stack Provider, Ring 3) acts as the client.
- The NT Kernel (Ring 0) serves as the server.
- The vulnerability originates in the VSP's flawed memory handling during interaction with the kernel. This occurs in the client's processing of freed memory, leading to undefined behavior that could result in privilege escalation.

Justification:

The vulnerability arises due to improper memory management by the VSP component of Hyper-V when interacting with the NT Kernel. Specifically, the VSP accesses memory that has already been freed, resulting in a use-after-free (CWE-416) flaw. This aligns with the #3 Exploiting Client threat cluster, as the root cause is a client-side failure in handling server-provided resources. While the outcome involves privilege escalation, the initial vulnerability is located in the client-side logic.

Try yourself here:

https://chatgpt.com/g/g-6787dfcaf2648191b907113d5b3ad8e0-cve-2-tlctc-mapper-v1-1

O: Integrating FAIR with the TLCTC Framework

This section details how the TLCTC (Top Level Cyber Threat Clusters) framework can enhance the FAIR (Factor Analysis of Information Risk) methodology. While FAIR excels in quantitative risk analysis, it lacks a structured approach to threat categorization and modeling of complex, multi-stage attack sequences. TLCTC adds a strategic overlay by precisely categorizing threats and representing attack paths.

Key Enhancements to FAIR Using TLCTC

- **Sequence Complexity Factor (SCF):** Adjusts base risk calculations by accounting for the length and complexity of an attack sequence, including parallel threat execution.
- **Compound Threat Multipliers (CTM):** Models the synergy effects when multiple threat clusters are executed simultaneously, enhancing probability calculations for complex attacks.
- **Path Variance Analysis (PVA):** Evaluates multiple potential attack paths by weighting alternative sequences, leading to a more accurate overall risk assessment.
- **Control Effectiveness Matrices (CEM):** Maps the effectiveness of controls across various threat clusters, taking into account the sequence position of each control within an attack path.

Implementation Framework

- **Threat Modeling Phase:** Use TLCTC to identify relevant threat clusters, map potential attack sequences (including any parallel executions), and document corresponding control mappings.

- **Risk Analysis Phase:** Enhance FAIR's base risk calculations by applying SCF to capture sequence complexity, CTM for parallel threats, and PVA for alternative attack paths.
- **Risk Reporting Phase:** Document primary attack sequences, map controls to specific threat clusters, calculate enhanced risk scores, and prioritize mitigation strategies accordingly.

Practical Example

Consider an Emotet attack sequence represented as **#9 → #7 → #7 → #4 → (#1 + #7)**. The enhanced FAIR analysis would proceed as follows:

- **SCF:** Increases the base risk to reflect the 5-step attack sequence.
- **CTM:** Applies a multiplier for the parallel execution seen in (#1 + #7).
- **PVA:** Evaluates alternative attack paths to account for path variance.
- **CEM:** Assesses the effectiveness of deployed controls across the sequence.

The final enhanced FAIR risk score can be calculated as:

*Enhanced_FAIR_Risk = Base_FAIR_Risk * SCF * CTM * PVA * (1 - CEM)*

Benefits and Impact

- **More Accurate Risk Quantification:** By accounting for attack sequence complexity and parallel threat execution, the enhanced model offers a more precise risk estimation.
- **Improved Control Evaluation:** Mapping controls directly to specific threat clusters allows for better assessment and improvement of defense measures.
- **Enhanced Communication:** Standardized TLCTC notation for attack sequences facilitates clearer communication among stakeholders.

- **Better Resource Allocation:** More precise risk prioritization leads to informed investment decisions in cybersecurity controls.

By integrating TLCTC with FAIR, organizations can develop a comprehensive risk analysis framework that captures both the quantitative and qualitative dimensions of cybersecurity threats, leading to more effective mitigation strategies and improved overall risk management.

P: TLCTC Practical Application Guidelines

Introduction

While the TLCTC framework provides clear theoretical foundations through its axioms and definitions, practitioners often need specific guidance for consistent threat classification in real-world scenarios. These guidelines build upon the framework's core principles to ensure accurate and consistent threat mapping across organizations.

Core Classification Principles

When categorizing threats, always begin with these fundamental questions:

- What is the generic vulnerability being exploited?
- Where does the initial exploitation occur (client or server)?
- Is this an exploitation of a code flaw or abuse of intended functionality?
- What is the earliest point in the attack sequence?

Cluster-Specific Mapping Guidelines (used for AI system prompt engineering for CVE to TLCTC mapping agent)

#1 Abuse of Functions

Map to this cluster when the threat involves:

- Misuse of legitimate features within their designed scope
- Exploitation of configuration issues rather than code flaws
- Design weaknesses in functionality (not implementation flaws)
- Authorization bypass through misconfiguration or feature misuse

#2 Exploiting Server

Map to this cluster when the threat involves:

- Server-side code implementation flaws
- Vulnerabilities in server request processing
- Authorization bypass through server-side code flaws
- Any vulnerability where the server's processing of requests is flawed

#3 Exploiting Client

Map to this cluster when the threat involves:

- Client-side code implementation flaws
- Vulnerabilities in client-side processing of responses
- Client application flaws (browsers, document readers, etc.)
- Any vulnerability where the client's processing of data is flawed

#4 Identity Theft

Map to this cluster when the threat involves:

- Direct compromise of authentication credentials
- Weaknesses in credential management
- Authentication bypass that directly exposes credentials
- Note: Authorization bypass without credential compromise maps to #1 or #2

#5 Man in the Middle

Map to this cluster when the threat involves:

- Vulnerabilities in communication path control
- Protocol-level vulnerabilities enabling interception
- Communication flow manipulation opportunities

#6 Flooding Attack

Map to this cluster when the threat involves:

- Resource capacity limitations
- Vulnerabilities enabling resource exhaustion
- Denial of service through overwhelming legitimate channels

#7 Malware

Map to this cluster when the threat involves:

- Opportunities for foreign code execution
- Note: This differs from exploit code targeting specific vulnerabilities

#8 Physical Attack

Map to this cluster when the threat involves:

- Physical accessibility vulnerabilities
- Hardware-level vulnerabilities
- OSI Layer 1 (Physical Layer) vulnerabilities

#9 Social Engineering

Special considerations:

- Never map technical vulnerabilities (e.g., CVEs) to this cluster
- Reserved for human-focused deception and manipulation
- Often initiates attack sequences leading to other clusters

#10 Supply Chain Attack

Map to this cluster when the threat involves:

- Vulnerabilities in third-party components or services
- Update mechanism compromises
- Note: Direct vulnerabilities in your own systems map to #1-#8

Common Classification Challenges

Authorization vs Authentication Issues

Authentication bypass:

- Maps to #4 when credentials are directly compromised
- Maps to #2 when resulting from server-side flaws
- Maps to #1 when resulting from misconfiguration

Process Injection Scenarios

Process injection classification depends on the method:

- Maps to #1 when using designed features (e.g., debugging APIs)
- Maps to #2/#3 when exploiting code flaws

Attack Path Notation

When documenting attack sequences:

- Use cluster numbers with arrows (e.g., #9->#3->#7)
- Indicate parallel execution with plus signs (#1+#7)
- Document all clusters in the sequence, including initial and subsequent vectors

Conclusion

These guidelines provide practical assistance in applying the TLCTC framework while maintaining its logical consistency and theoretical foundations. When in doubt, always return to the fundamental question: "Which generic vulnerability is being exploited?"

Q: Integrating NIST NICE Tasks with the TLCTC Framework

The Workforce-Threat Integration Challenge

Organizations face a significant challenge in aligning cybersecurity workforce capabilities with the actual threats they need to address. The NIST National Initiative for Cybersecurity Education (NICE) Framework provides a comprehensive taxonomy of cybersecurity tasks, but its structure does not explicitly connect these tasks to the threat landscape. This disconnect can lead to:

- Workforce development that doesn't address critical threat vectors
- Difficulty in prioritizing training and skill development
- Unclear relationships between job functions and security outcomes
- Challenges in mapping workforce capabilities to risk management

The Top Level Cyber Threat Clusters (TLCTC) framework offers a solution to this challenge by providing a consistent, cause-oriented categorization of threats that can serve as an organizing principle for workforce tasks and capabilities.

Integration Framework

The proposed integration leverages the TLCTC framework's structure to organize NICE tasks according to:

1. **The 10 Top Level Cyber Threat Clusters**: Each representing a distinct attack vector based on a generic vulnerability
2. **The five NIST CSF functions**: Providing a structured approach for each threat cluster
3. **The GOVERN function**: Addressing strategic oversight across all clusters

This creates a comprehensive matrix where each NICE task can be mapped to:

- The specific threat cluster(s) it addresses
- The control function it supports (IDENTIFY, PROTECT, DETECT, RESPOND, RECOVER)
- Its position in the threat management lifecycle

Structural Benefits

This integration delivers several key advantages:

- **Cause-Oriented Organization**: Tasks are grouped based on the fundamental vulnerabilities they address
- **Clear Security Outcomes**: Each task is directly linked to specific control objectives
- **Strategic-Operational Alignment**: Strategic governance tasks are connected to operational activities
- **Comprehensive Coverage**: Ensures all aspects of the threat landscape are addressed by appropriate workforce capabilities
- **Attack Sequence Awareness**: Tasks can be further categorized based on their relevance to different stages of attack paths

Implementation Methodology

Step 1: Threat Cluster Mapping

Each NICE task is evaluated to determine which threat cluster(s) it primarily addresses. For example:

- Tasks related to secure coding would map to #2 (Exploiting Server) and #3 (Exploiting Client)
- Tasks focused on identity management would map to #4 (Identity Theft)
- Tasks concerning social engineering awareness would map to #9 (Social Engineering)

Step 2: Control Function Alignment

Within each threat cluster, tasks are further categorized according to the NIST function they support:

- IDENTIFY: Tasks focused on understanding the threat landscape, discovering vulnerabilities
- PROTECT: Tasks aimed at implementing security controls to prevent compromise
- DETECT: Tasks related to monitoring and detecting potential threats
- RESPOND: Tasks involved in addressing and mitigating active threats
- RECOVER: Tasks focused on restoration and improvement following incidents

Step 3: Strategic-Operational Integration

The GOVERN function encompasses strategic tasks that apply across all threat clusters, including:

- Risk management and assessment
- Policy development and implementation
- Compliance monitoring and reporting
- Program management and oversight
- Workforce development and management

Examples of NICE Task Integration with TLCTC

Example 1: Mapping #2 Exploiting Server

Threat Cluster Definition: An attacker targets vulnerabilities in server-side software to manipulate server behavior using exploit code.

Generic Vulnerability: The presence of exploitable flaws in server-side software code.

NIST Function	NICE Task ID	NICE Task Description	Control Objective
IDENTIFY	T0028	Conduct software assessments to ensure compliance with security requirements and policies	Identify weaknesses enabling server exploitation
IDENTIFY	T0160	Perform secure code reviews	Identify weaknesses enabling server exploitation
IDENTIFY	T0013	Assess the effectiveness of security controls	Identify weaknesses enabling server exploitation
PROTECT	T0176	Perform security reviews and identify security gaps in security architecture	Protect server from being exploited
PROTECT	T0291	Implement security countermeasures to mitigate vulnerabilities	Protect server from being exploited
PROTECT	T0296	Make recommendations based on malware analysis	Protect server from being exploited

NIST Function	NICE Task ID	NICE Task Description	Control Objective
DETECT	T0259	Use cyber defense tools for continual monitoring and analysis of system activity	Detect exploited server
DETECT	T0063	Collect intrusion artifacts and use discovered data to enable mitigation of potential cyber defense incidents	Detect exploited server
RESPOND	T0175	Perform real-time cyber defense incident handling tasks	Respond to exploited server
RESPOND	T0278	Respond to crisis situations within the pertinent constraints	Respond to exploited server
RECOVER	T0332	Coordinate with intelligence analysts to manage and deconflict intelligence requirements	Recover from server exploit event
RECOVER	T0229	Implement specific cybersecurity countermeasures based on work performed	Recover from server exploit event

Example 2: Mapping #4 Identity Theft

Threat Cluster Definition: An attacker targets weaknesses in identity and access management to acquire and misuse legitimate credentials.

Generic Vulnerability: Weak Identity Management Processes and/or credential protection mechanisms.

NIST Function	NICE Task ID	NICE Task Description	Control Objective
IDENTIFY	T0059	Collaborate with stakeholders to identify and/or develop appropriate identity and access management solutions	Identify weaknesses in identity management
IDENTIFY	T0115	Identify security issues that could impact access control implementations	Identify weaknesses in credential management
PROTECT	T0455	Implement and enforce identity and access management controls	Protect identity
PROTECT	T0123	Install, update, and troubleshoot identity and access management systems and components	Protect credentials
DETECT	T0261	Design and develop user activity monitoring and insider threat capabilities	Detect identity theft
DETECT	T0164	Perform content inspection to detect and handle anomalies in content	Detect identity theft

NIST Function	NICE Task ID	NICE Task Description	Control Objective
RESPOND	T0521	Respond to identity and authentication issues	Respond to identity theft
RESPOND	T0133	Manage accounts, network rights, and access to systems and equipment	Respond to identity theft
RECOVER	T0510	Restore domain account access for authorized personnel	Recover identity
RECOVER	T0531	Implement technical safeguards to ensure data integrity during recovery operations	Recover identity

Example 3: Mapping #9 Social Engineering

Threat Cluster Definition: An attacker manipulates people into performing actions that compromise the security of systems or (business-) processes.

Generic Vulnerability: The generic vulnerability in humans is their gullibility, ignorance, or compromisability.

NIST Function	NICE Task ID	NICE Task Description	Control Objective
IDENTIFY	T0258	Develop and conduct social engineering tests	Identify human vulnerabilities to social engineering
IDENTIFY	T0507	Identify security awareness issues from social engineering exercises	Identify human vulnerabilities to social engineering
PROTECT	T0256	Develop and deliver technical training to educate end users	Protect against social engineering
PROTECT	T0502	Create security awareness materials	Protect against social engineering
DETECT	T0301	Monitor external data sources to maintain current security threat information	Detect social engineering attempts
DETECT	T0166	Perform security reviews and identify gaps in security architecture	Detect social engineering attempts

NIST Function	NICE Task ID	NICE Task Description	Control Objective
RESPOND	T0152	Notify and work with organizational incident handlers	Respond to social engineering incidents
RESPOND	T0171	Perform cyber defense incident triage	Respond to social engineering incidents
RECOVER	T0491	Perform analysis of lessons learned from incidents	Recover from social engineering incidents
RECOVER	T0332	Coordinate with intelligence analysts to manage and deconflict intelligence requirements	Recover from social engineering incidents

Example 4: GOVERN Function Across All Threat Clusters

The GOVERN function provides strategic oversight and management across all threat clusters:

GOVERN Aspect	NICE Task ID	NICE Task Description	Strategic Objective
Risk Management	T0165	Perform risk assessment to determine loss potential	Establish risk appetite across threat clusters
Policy Development	T0149	Develop policies and procedures	Create cohesive security policies aligned with threats
Strategic Planning	T0094	Develop strategic insights about cybersecurity implications	Align security strategy with threat landscape
Resource Allocation	T0570	Determine security implications and resource requirements for new technologies	Allocate resources based on threat priorities
Program Management	T0072	Define and manage project scope	Ensure security programs address all threat clusters
Compliance	T0177	Perform security compliance reviews	Verify protection against all threat clusters

Benefits of Integration

For Security Leadership

- **Improved Resource Allocation**: Clearer mapping between workforce capabilities and the threat landscape enables more effective resource allocation
- **Risk-Based Prioritization**: Training and staffing can be prioritized based on the most critical threat clusters facing the organization
- **Strategic Alignment**: Ensures strategic security initiatives directly support threat mitigation across all relevant clusters

For Security Operations

- **Clear Task Relevance**: Staff understand exactly how their tasks contribute to addressing specific threat vectors
- **Comprehensive Coverage**: Ensures operational activities address all aspects of the threat landscape
- **Structured Response**: Provides a clear framework for organizing incident response activities

For Workforce Development

- **Targeted Skill Development**: Training can be focused on the most relevant threat clusters
- **Clear Career Progression**: Staff can develop expertise around specific threat clusters or control functions
- **Comprehensive Capability Planning**: Organizations can ensure they have the right skills to address all threat clusters

Implementation Considerations

When implementing this integration framework, organizations should consider:

1. **Organization-Specific Tailoring**: Adapt the mapping based on the organization's specific threat landscape and risk profile
2. **Task Multi-Classification**: Some tasks may address multiple threat clusters and should be mapped accordingly
3. **Regular Review and Update**: As the threat landscape evolves, task mappings should be reviewed and updated
4. **Prioritization Based on Risk**: Focus initial integration efforts on the threat clusters presenting the highest risk

Conclusion

Integrating NIST NICE tasks with the TLCTC framework creates a powerful structure for aligning workforce capabilities with the actual threats organizations face. This approach transforms cybersecurity workforce management from a role-based exercise to a threat-centric discipline, ensuring that human capabilities directly address the full spectrum of cyber threats in a structured, consistent manner.

By organizing workforce tasks according to the 10 Top Level Cyber Threat Clusters and the NIST CSF functions, organizations can develop a more resilient security posture with clear connections between workforce capabilities, control objectives, and the evolving threat landscape. This integration provides a bridge between strategic risk management and operational workforce development, ensuring that the right people with the right skills are addressing the right threats.

X. Change Log

V1.6.3 - 2025-04.01

- Made B. and C. more pragmatic

V1.6.2 - 2025-03-29

- All Definitions clarified

V1.6.1 - 2025-03-16

- Redacted Chapter “Data Risk Event Types”

V1.6 - 2025-03-09

- Added Chapter Q: NIST NICE Tasks Integration

V1.5.9 - 2025-02-21

- Added analysis of CRF-TT to Chapter G

V1.5.8 - 2025-02-19

- Added Chapter P: “TLCTC Practical Application Guidelines” based on system prompt engineering results for an agent based CVE to TLCTC mapper

V1.5.7 - 2025-02-13

- Added new chapter M: “Enhancing CVE Details”. Moved old M->N
- Added new Chapter O: “Integrating FAIR"

V1.5.5 - 2025-01-29

- Redacted MITRE and STIX jsons

V1.5.4 - 2025-01-23

- Added Chapter M “Example CVE Analysis”
- Added “MFA Bombing” in Chapter F

V1.5.3 - 2025-01-15

- Refined Vertical Attack Path and added Hypothetical Examples

V1.5.2 - 2025-01-11

- Added Chapter “K: Physical Layer Analysis in the TLCTC Framework”
- Added Chapter “L: Integrating Programmable Logic Controller (PLC) Architectures within the TLCTC Framework”
- Added Section about “Current State of FAIR” and “ETSI”

V1.5.1 - 2025-01-08

- Added Sub Chapter “Hierarchical Framework for Key Indicators”

V1.5 - 2025-01-03

- Reformatted the White Paper and changed order of the Chapters
- Added KRI, KCI and KPI definitions and examples
- Assed Consideration about “Umbrella Controls”

v1.4 - 2024-12-24

- Added a new guiding principle: **Axiom: Each distinct attack vector is defined by the generic vulnerability it initially targets.**

v1.3 - 2024-12-19

- Added Chapter "Why ten?" Added Chapter "Oversimplification?"
- Added a new clarification regarding the term "process injection"

v1.2 - 2024-12-08

- Added clarification on 3rd Party Cyber Risk Management vs Supply Chain Attack (#10)
- Added new chapter on Secure Software Development Life Cycle (SSDLC) integration
- Added new chapter on Secure Coding Practices

v1.1 - 2024-11-24

- Added new chapter about vertical attack paths
- Added executive summary

v1.0 - 2024-09-01 first official release

- Added practical application examples and use cases
- Enhanced framework explanation for better comprehension
- Refined semantic consistency in threat definitions
- Added integration examples with existing frameworks
- Prototype Development - December 2022 Initial concept development during holiday period
    - Established core thought experiment methodology
    - Defined foundational axioms
    - Developed initial generic vulnerability mapping
    - Created semantically consistent threat cluster definitions
    - Challenged conventional terminology (e.g., DDOS, Drive By) to maintain logical consistency