Defending Against Mythos-Class Attacks

Applying Structural Design Principles to Build Trustworthy Secure Systems

Dr. Ron Ross

RONROSSECURE, LLC

1. Introduction

The United States and its allies face persistent, evolving, and increasingly sophisticated threats from determined adversaries using advanced AI tools and techniques to conduct cyberattacks against mission critical systems. A next generation AI model that has raised concerns among senior government officials and cybersecurity experts is Claude Mythos, a large language model (LLM) developed by Anthropic to find software vulnerabilities. Specifically, the concerns involve the potential for adversaries to use this AI capability to find and exploit vulnerabilities that exist in the installed code base of systems in critical infrastructures around the world.

The phrase "Mythos-class attack" does not refer to a specific hacker group or adversary, type of malicious code, or set of tactics, techniques and procedures (TTP). It does, however, describe the next generation of AI-driven cyber threats that can be triggered by the capabilities of the Mythos model or LLMs with equivalent capability. When government agencies and cybersecurity experts warn about a Mythos-class attack, they are referring to a fundamental shift in how vulnerabilities are found and weaponized. It represents a transition from human-centric cyber warfare to an autonomous, machine-speed cyber warfare.

The anatomy of a Mythos-class attack is defined by three distinct technical characteristics:

- Autonomous zero-day exploitation
- Multi-stage autonomous attack chains
- Rapid, low-cost exploitation

Traditional, script-based cyberattacks required the discovery of an unmitigated vulnerability (e.g., an unpatched software flaw) and human-developed malicious code to exploit the vulnerability – a resource intensive process. In contrast, Mythos-class models possess the capability, through agentic reasoning, to independently scan code, locate critical vulnerabilities, and rapidly develop fully functional code to carry out the exploit. Traditional cyberattacks also lack flexibility – that is, if they encounter an unexpected security measure or control, the attack fails. Mythos-class attacks are flexible and adaptable. If the AI model encounters a security control or defensive boundary, it can autonomously look for additional exploitable flaws including chaining together multiple flaws to uncover new vulnerabilities. These AI-driven attacks can be carried out rapidly (significantly reducing the time-to-exploit factor) and at a relatively low cost compared to traditional cyberattacks.

In this new era of super-charged AI-driven attack tools, there is an emerging fundamental truth: the root cause of most successful cyberattacks is not a failure of security controls, but a failure of system design [1]. This article examines why the application of the security design principles in NIST SP 800-160 [2] is critical to building trustworthy secure systems that provide "mission resilience" against determined

adversaries employing next generation AI-driven attack tools. It focuses on the foundational design principle of domain separation (i.e., isolation) and the complementary principles that provide layered, defense-in-depth protection and resilience for mission critical systems.

2. The Anatomy of a Modern Cyberattack

Modern cyberattacks have the following characteristics:

- Agent-based architecture with encrypted, polymorphic C2 channels that blend into legitimate traffic
- Modular payload system enabling custom capabilities per target environment
- Built-in lateral movement capabilities including pass-the-hash, pass-the-ticket, Kerberoasting, and SMB/WMI/DCOM traversal
- Privilege escalation modules exploiting OS and application vulnerabilities or abusing legitimate administration interfaces
- Persistence mechanisms spanning registry keys, scheduled tasks, service installation, DLL hijacking, and boot sector implants
- Anti-forensics and detection evasion including process injection, PPID spoofing, timestomping, and living-off-the-land (LOTL) techniques
- OPSEC-aware design: minimal footprint, configurable sleep jitter, encrypted artifact storage

Cyberattacks also follow a consistent lifecycle that directly exploits the absence of structural security properties:

- **Phase 1** — Initial Access: Phishing, supply chain compromise, exploitation of internet-facing services, or use of stolen credentials.
- **Phase 2** — Execution and Establishment: Deployment of an implant (agent/beacon) that establishes encrypted C2 communication back to the adversary's infrastructure.
- **Phase 3** — Discovery: Automated enumeration of the local and adjacent network, accounts, services, and trust relationships.
- **Phase 4** — Lateral Movement: Traversal to other systems within the same network segment using harvested credentials and exploited trust relationships.
- **Phase 5** — Privilege Escalation: Elevation to administrative or system-level rights, enabling access to protected resources and the ability to disable security controls.
- **Phase 6** — Persistence and Defense Evasion: Establishment of multiple redundant footholds; manipulation of logs and detection capabilities.
- **Phase 7** — Mission Execution: Data exfiltration, encryption for ransom, destruction of systems, or long-term intelligence collection.

The critical insight is that Phases 3 through 5 are where structural security properties have their greatest defensive effect. A well-designed system makes lateral movement structurally impossible across domain boundaries, not merely detectable after the fact.

3. Domain Separation: The Bedrock of Structural Security

Among the security design principles in NIST Special Publication 800-160, domain separation stands out as foundational. Without proper domain separation (i.e., isolation), no amount of perimeter security, encryption, or monitoring can prevent an adversary from achieving their objectives once initial access is obtained. The adversary simply moves laterally through the flat, undifferentiated architecture that most organizations have built and deployed.

Domain separation is also a foundational systems engineering requirement. Systems are built with implicit trust between components — trust that adversaries systematically exploit. When a system has no meaningful internal boundaries, a compromised system component (e.g., an endpoint) becomes a springboard to every other system component. The question is not whether an adversary will breach the perimeter. The question is what happens after they do. Without domain separation, the answer is always the same: everything is reachable, everything is vulnerable, and nothing can stop the damage.

The principle of domain separation defined in NIST SP 800-160, states that domains with distinctly different protection needs are physically or logically separated. The separation of domains enables enhanced control and, therefore, protection of system function and the flow of data. Control relative to separated domains limits the extent to which an entity or domain is influenced by or is able to influence some other entity or domain, thereby enhancing the protection of a domain.

The principle of domain separation articulates two key protective effects: susceptibility (protecting a domain from influence by external entities) and containment (protecting external entities from erroneous or malicious behavior originating within the domain). Both effects are critical when defenders face adversaries using Mythos-class attack tools.

Historically, domain separation has been applied to enforce privilege boundaries – including separating administrative domains from user domains. However, a modern extension of this principle is needed to enforce mission-driven separation across the totality of a system's architecture, from the hardware substrate to the application layer.

4. Domain Separation Reduces the Impact of Cyberattacks

Domain separation attacks the adversary's kill chain at its most vulnerable point: the phase between initial compromise and mission achievement. An adversary who compromises a single endpoint in a properly domain-separated architecture has compromised exactly that endpoint, and nothing more.

Containment: Limiting Blast Radius

The containment property of domain separation ensures that a compromised domain cannot directly interact with higher-sensitivity or higher-privilege domains. NIST SP 800-160 states that the principle of domain separation requires a domain to be contained within its own protected subsystem so that the elements within the domain are only directly accessible by procedures or functions of the protected subsystem. Containment against a Mythic [3] or Cobalt Strike [4] beacon means that even if the agent achieves code execution with local administrative rights, it cannot reach across to an adjacent financial system, control system, or classified domain. The C2 operator is trapped in the compromised domain.

Susceptibility Reduction: Blocking Lateral Movement

Modern cyberattacks depend on exploiting trust relationships. When workstations in the same network segment can freely initiate connections to each other, pass-the-hash attacks, SMB relay attacks, and DCOM traversal are straightforward. Domain separation, enforced by mechanisms such as firewalls, data diodes, and cross-domain solutions (CDS), eliminates the implicit trust that makes these attacks possible. Other techniques include software-defined networking (SDN), micro-segmentation, zero trust architectures (ZTA), hardware-enforced isolation, and mandatory access controls (MAC). Applied architecturally, these create chokepoints where every interaction between domains must be explicitly authorized — transforming the attack surface from a flat plane to a set of narrow, monitored interfaces.

Privilege Isolation: Defeating Escalation

Administrative domains separated from user domains mean that even a full administrative compromise of a user workstation does not automatically yield administrative access to the administrative domain. The adversary's privilege escalation path is severed at the domain boundary. Domain credentials and administrative functions are not reachable from the compromised domain. This architectural isolation is qualitatively different from password complexity requirements or multi-factor authentication. Those controls can be bypassed by credential harvesting. Domain separation means that even valid harvested credentials from the user domain do not operate in the administrative domain.

Command and Control (C2) Channel Disruption

Modern cyberattack implants require a functioning encrypted C2 channel to receive instructions and exfiltrate data. In a domain-separated architecture with data diodes or CDS protecting sensitive domains, the C2 callback from a compromised element in a restricted domain may not be physically or logically capable of reaching the adversary's infrastructure. The implant is isolated, unable to communicate, effectively neutralizing its capability.

5. Complementary Security Design Principles

Domain separation is powerful but does not operate in isolation. NIST SP 800-160 defines 30 security design principles. Seven of these are particularly complementary to domain separation in the context of defending against Mythos-class attacks. Together they form an interlocking structural defense that reduces an organization's susceptibility to severe or catastrophic damage and helps to ensure mission resilience.

Least Privilege

Design Principle: *Each system element is allocated privileges that are necessary to accomplish its specified functions but no more.*

Security Relevance: Directly limits the scope of damage achievable from within any single domain. A compromised process or account can only access what it legitimately needs, preventing credential-based escalation even within a domain.

Least privilege and domain separation are synergistic: domain separation sets the coarse-grained boundary between security domains; least privilege enforces fine-grained access control within each domain. Together, they ensure that an adversary who compromises one element in one domain has extremely limited reach. NIST SP 800-160 notes explicitly that least privilege can inform the use of

domain separation — the system modules can be designed so that only the encapsulated elements are directly accessed by the functions within the module. Least privilege defeats the common technique of abusing over-privileged service accounts. When service accounts have only the rights they need, credential harvesting yields credentials of limited utility.

Mediated Access

Design Principle: *All access to and operations on system elements are mediated.*

Security Relevance: Creates mandatory chokepoints at every domain boundary and within domains, forcing all interactions through a policy-based access decision that cyberattack tools cannot bypass without detection.

Mediated access is a foundational principle in the design of trustworthy secure systems. It has two components: access to the system element (only authorized entities gain access) and use of the system element (only authorized operations are permitted). Both are relevant to defending against a Mythos-class attack. Mediated access implements the reference monitor concept at every domain boundary. An implant attempting to cross a domain boundary encounters a policy-based decision point that evaluates whether the requesting entity is authorized for that specific access at that specific time in that specific system state. The C2 operator, operating through a compromised element, has neither the credentials nor the role to pass this check. The reference validation mechanism and reference monitor concept provide design assurance that the mediated access control mechanism is tamper-resistant, always invoked, and small enough to verify. This is what distinguishes an architectural control from a perimeter control that can be bypassed.

Structured Decomposition and Composition

Design Principle: *System complexity is managed through the structured decomposition of the system and the structured composition of the constituent elements to deliver the required capability.*

Security Relevance: Provides the architectural discipline that makes domain separation implementable and verifiable, rather than aspirational.

Domain separation requires that domains be defined, bounded, and maintained as distinct architectural entities. Without the discipline of structured decomposition, the boundaries tend to erode over time as connections are added for convenience, exceptions accumulate, and the intended architecture diverges from the implemented architecture. Structured decomposition provides the engineering methodology to prevent this. Modularity, layering, and partially ordered dependencies are the basis for structured decomposition. Applied to domain separation, this means domains are defined as modules with explicit, controlled interfaces, arranged in layers with well-defined dependency relationships, and composed into the overall system architecture in a manner that can be verified. This principle matters because the adversary's most powerful capability is finding unintended paths: for example, a forgotten firewall rule, an undocumented administrative share, a service that uses credentials from a higher-privilege domain. Structured decomposition and composition discipline minimizes such unintended paths by making the architecture explicit, traceable, and verifiable.

Least Functionality

Design Principle: *Each system element has the capability to accomplish its required functions but no more.*

Security Relevance: Reduces the attack surface available to Mythos-class tools within each domain. Disabling unneeded services eliminates entire categories of exploitation.

Mythos-class tools thrive on the richness of the average enterprise environment: SMB file sharing, PowerShell remoting, DCOM, and legacy administrative interfaces all provide execution and traversal vectors. Least functionality applied rigorously within each domain — disabling or blocking services not required for the domain's mission — eliminates these attack surfaces. The principle acknowledges that commercial off-the-shelf (COTS) components present a challenge, as they often contain functionality beyond what is needed. The solution is to configure the system to enable only what is required and prohibit or restrict what is not. In the context of domain separation, this means each domain presents only the services its mission requires.

Least Sharing

Design Principle: *System resources are shared among system elements only when necessary and among as few elements as possible.*

Security Relevance: Eliminates the shared mechanisms and pathways that Mythos-class tools exploit for lateral movement and information access.

Shared infrastructure is the highway of lateral movement. Shared storage, shared naming services, shared authentication infrastructure, and shared administrative channels all represent information paths between domains that Mythos-class tools exploit. NIST SP 800-160 cites Saltzer and Schroeder's foundational security principle: "Every shared mechanism (especially one involving shared variables) represents a potential information path between users and must be designed with great care." [5] Least sharing reinforces the principle of domain separation by ensuring that the shared elements between domains are explicitly designed, minimized, and controlled, not the accidental consequence of convenience-driven architectural choices.

Least Persistence

Design Principle: *System elements and other resources are available, accessible, and able to fulfill their design intent only for the time for which they are needed.*

Security Relevance: Defeats the persistence mechanisms on which Mythos-class tools depend to maintain a foothold across reboots and sessions.

Persistence is a core capability of modern cyberattack tools. Cobalt Strike, Mythic, and other similar frameworks include extensive persistence mechanism libraries spanning registry autorun keys, scheduled tasks, service installation, DLL search order hijacking, and boot-level implants. Least persistence, applied architecturally, means that processes, services, and credentials that are not currently needed, are removed from a running state and not available to be abused. The least persistence principle defines three conditions that must be satisfied for an element to be usable: (1) presence (installed in memory or storage), (2) accessible (invocable), and (3) active (currently executing). Least persistence targets all three: do not install until needed, deactivate when not in use, and mediate access even when present. This systematically attacks the persistence capabilities that Mythos-class tools rely on.

Anomaly Detection

Design Principle: *Any salient anomaly in the system or its environment is detected in a timely manner that enables effective response action.*

Security Relevance: Provides the sensor layer that detects Mythos-class tool activity at domain boundaries and within domains, enabling response before mission execution.

Domain separation changes the value proposition of anomaly detection fundamentally. In a flat architecture, the adversary's lateral movement traffic looks like normal east-west traffic. There is no structural difference between legitimate and malicious communications. In a domain-separated architecture, any attempt to traverse a domain boundary is either explicitly authorized and logged, or structurally impossible. Anomalous traffic at the domain boundary stands out precisely because the boundary controls define what is normal. The principle emphasizes that anomaly detection requires monitoring system behaviors and outcomes to determine deviations from design intent. Domain separation makes design intent concrete and architectural; that is, all traffic between domains must traverse a defined interface. Any traffic that does not conform to the authorized interface specification is, by definition, anomalous. This transforms anomaly detection from a needle-in-a-haystack problem (finding malicious C2 traffic among millions of legitimate connections) to a targeted problem (detecting unauthorized boundary crossing in a small, well-defined interaction set).

Continuous Protection

Design Principle: *The protection provided for a system element must be effective and uninterrupted during the time that the protection is required.*

Security Relevance: Ensures that domain boundaries and access controls remain effective during state transitions, maintenance, updates, and recovery — the moments Mythos-class tools specifically target.

Mythos-class attacks deliberately target transitional states: systems during updates, services during restart, sessions during authentication, backups during transmission. The adversary's toolkit includes timing attacks that exploit brief windows when protection is degraded. Continuous protection, as an architectural requirement, means that domain separation mechanisms must remain effective across all system states and modes. The principle requires that protection capability be uninterrupted across all relevant system states, modes, and transitions. For the principle of domain separation, this means the boundary enforcement mechanisms must operate during boot, during maintenance windows, during backup and recovery operations, and during system updates — the same moments that enterprise systems are most vulnerable to Mythos-class tools.

6. Integrated Defense Against Mythos-Class Attacks

Each of the principles described above addresses a different attack phase or capability. When applied in combination with domain separation, they create a mutually reinforcing structural defense that is qualitatively different from the additive effect of independent security controls.

Raising the Cost of Attack

A fundamental concept in adversary economics is cost imposition: the defender's objective is to raise the cost and complexity of a successful attack until it exceeds the adversary's willingness or capacity to pay. Perimeter-based security imposes cost only at the initial access phase. Once the perimeter is

breached, the adversary's costs collapse. The application of domain separation and complementary security design principles imposes cost at every phase:

- Initial access yields only a foothold in one domain, not access to the mission-critical system or environment.
- Lateral movement requires overcoming explicit domain boundaries, not just traversing a flat network.
- Privilege escalation within a domain is constrained by least privilege; escalation across domains requires crossing a mediated boundary.
- Persistence mechanisms are systematically denied by least persistence and least functionality.
- Discovery is limited by least sharing; the adversary cannot enumerate what is not accessible.
- Every action at a domain boundary is logged and subject to anomaly detection.

This cost imposition is structural, not operational. It does not depend on the specific skills of the defenders monitoring a security operations center. It does not degrade when defenders are tired, understaffed, or distracted. It persists because it is built into the architecture of the system.

The Role of Systems Engineering

Effective defense against Mythos-class attacks requires principled systems engineering, not just traditional cybersecurity controls and operations. The principles described in this paper are engineering principles — they are realized through architectural decisions made during the design and development of systems. NIST SP 800-160 provides the framework for this engineering approach. The Systems Security Engineering (SSE) framework defines three contexts in which security must be addressed:

- Problem Context (establishing protection objectives and requirements)
- Solution Context (defining the protection strategy and architecture), and
- Trustworthiness Context (demonstrating through evidence that the architecture achieves its objectives).

Domain separation and its complementary security design principles inform all three contexts. In the Problem Context, they inform the definition of security objectives and the identification of loss scenarios that must be prevented. In the Solution Context, they provide the architectural design patterns that achieve those objectives. In the Trustworthiness Context, they provide the basis for claims that the architecture is effective — claims that can be verified through analysis, inspection, and testing.

7. Conclusion

Mythos-class tools represent the next generation of adversary cyberattack capability: autonomous zero-day exploitation, multi-stage autonomous attack chains, and rapid, low-cost exploitation. The cyberattack tools are sophisticated, modular, persistent, and specifically designed to exploit the architectural weaknesses of systems that were built without structural security. Against this threat, controls-based security is insufficient. The adversary's tools are explicitly designed to bypass controls.

Domain separation, as defined in NIST SP 800-160, provides the structural foundation for a defense that cannot be bypassed — only overcome through direct assault on each domain's architecture. When combined with least privilege, least functionality, least sharing, least persistence, structured decomposition and composition, mediated access, anomaly detection, and continuous protection, domain separation creates a mutually reinforcing architectural defense that imposes cost at every phase of the cyberattack lifecycle.

The implication for the broader systems engineering community is clear: trustworthy secure systems are not built by deploying more security products. They are built by applying engineering discipline to the structure of the system itself — ensuring that the architecture makes damaging attacks structurally difficult, not merely operationally detectable. The failure mode is not inadequate technology. It is inadequate engineering.

References

[1] Ross, R. (2016). Rethinking Cybersecurity from the Inside Out. NIST Taking Measure Blog. https://www.nist.gov/blogs/taking-measure/rethinking-cybersecurity-inside-out

[2] Ross, R., Winstead, M., & McEvilley, M. (2022). NIST Special Publication 800-160, Volume 1, Revision 1: Engineering Trustworthy Secure Systems. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-160v1r1

[3] Specter, J. (2019–2024). Mythic C2 Framework. https://github.com/its-a-feature/Mythic

[4] Raphael Mudge. (2012–2024). Cobalt Strike Technical Documentation. Fortra, LLC.

[5] Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. Proceedings of the IEEE, 63(9), 1278–1308.