Offensive Architecture

A manifesto for standards planning, Security Considerations, Non-Goals, and emergent vulnerabilities

Gregory Disney-Leugers

Companion to Reflections on Disregarding Trust (DEF CON 34 Main Stage).
Publication: evilrbi.com
Research and authorized assessment only.

Epigraph

You can't trust code that you did not totally create yourself.
Ken Thompson, Reflections on Trusting Trust (1984)

You cannot trust a runtime that you do not own and understand yourself.
Gregory Disney-Leugers, Reflections on Disregarding Trust (2026)

From Trusting Trust to Disregarding Trust

Reflections on Trusting Trust is the origin story of modern computing trust, the Big Bang of the problem. Without the compiler there is no trustworthy kernel and no operating system. Everything that runs afterward depends on a difficulty Thompson named: one must trust the tools that turn source into binary.

Reflections on Disregarding Trust is the Hubble tension of that same cosmos. We still live with Thompson's origin problem, but we observe the system with different instruments: browsers,

identity protocols, isolation products, naming systems, routing, and control planes. The security story told for each feature no longer matches what appears when those features operate together. That mismatch is the domain of emergent vulnerabilities. Two measurements of the same stack disagree: the reviewed module and the composed outcome. This manifesto takes that disagreement seriously.

Same foundation. Larger scale of observation. Still no zero-day required.

Author's note

I am Gregory Disney-Leugers. This document states the theoretical foundation of my DEF CON 34 Main Stage lecture, Reflections on Disregarding Trust, and of the broader research program that lecture represents.

The work is not a bug hunt. It is an argument about architecture: how industry designs, standardizes, and ships multi-layer systems while disregarding composite trust, and how that disregard produces emergent vulnerabilities. The domain is not limited to any single product class. The same structure appears from BGP to WebAuthn, from recursive DNS to session cookies, from automation interfaces to isolation products.

Three pillars structure the argument:

1. Dante's Inferno of Deflected Responsibility
2. The Law of Generalized Complexity Orchestration (LGCO)
3. The Four Constants of Emergent Vulnerabilities

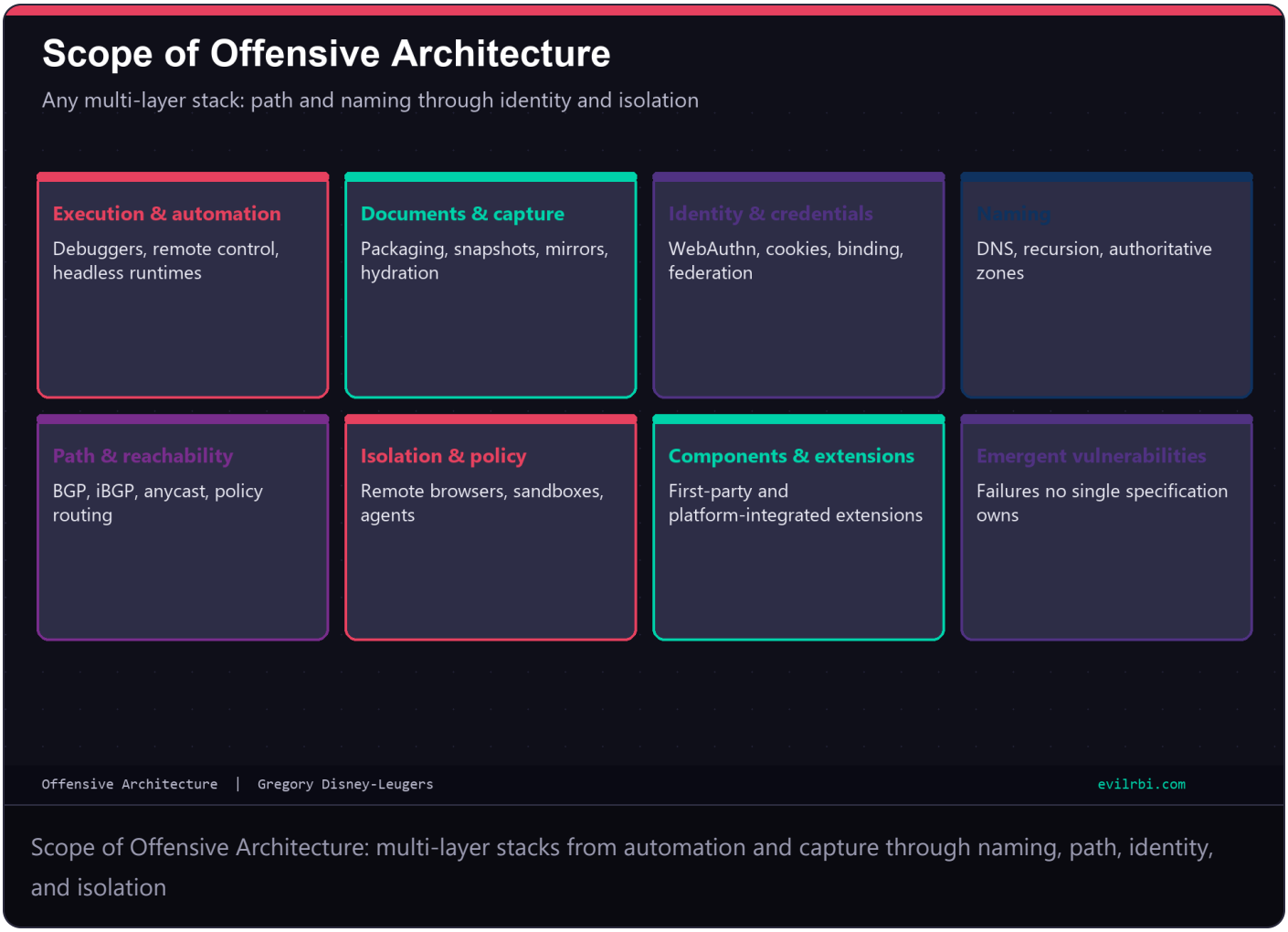
From those three follows a demand on standards bodies and platform owners. Before a security-relevant standard is approved, its Security Considerations and Non-Goals sections must be shown to work with the other standards that actually ship (past, present, and future) so that they do not create emergent vulnerabilities by composition.

---

1. What Offensive Architecture is

Offensive Architecture is the study of emergent vulnerabilities: how correct, reviewed, standards-aligned features, when used together, create security failures for which no single specification is responsible.

It applies to any stack built from many legitimate pieces. Illustrative layers include the following.



Scope of Offensive Architecture: multi-layer stacks from automation and capture through naming, path, identity, and isolation

Layer family	Examples of intended mechanisms
Execution and automation	Debuggers, remote control planes, headless runtimes
Documents and capture	Packaging formats, snapshots, mirrors, hydration
Identity and credentials	WebAuthn, cookies, session binding, federation

Layer family	Examples of intended mechanisms
Naming	DNS, recursive resolution, authoritative zones
Path and reachability	BGP, iBGP, anycast, policy routing
Isolation and policy	Remote browsers, sandbox products, enterprise agents
Components and extensions	First-party and platform-integrated extensions

Offensive Architecture is not:

- A primary hunt for memory corruption or protocol zero-days
- A collection of attack tips detached from specifications
- A claim that cryptography is useless

It is:

- Close reading of RFCs, W3C documents, IETF drafts, and platform documentation, especially Security Considerations and Non-Goals, for what a standard promises, what it refuses to cover, and what it leaves to others
- Construction of reference compositions that occupy those gaps using only intended behavior
- Return of that evidence into standards planning and approval, so those sections are tested against the whole stack people will actually run

A standard is unfinished if its Security Considerations and Non-Goals have not been checked against the rest of the stack. That is where emergent vulnerabilities appear.

No zero-days are required. Stock platforms, intended mechanisms, disregarded trust, and emergent vulnerabilities are sufficient.

---

2. Three pillars

Three pillars of Offensive Architecture flowing into emergent vulnerabilities

Pillar I. Dante's Inferno of Deflected Responsibility

In large ecosystems, security responsibility is pushed downhill until it lands on the user or the operator with the least power to repair the design. That pattern is deflected responsibility.

Each layer writes a local rule. Then it refuses to own what happens when that rule meets a hostile combination of other features. That refusal is how emergent vulnerabilities form without anyone's name attached to them.

The parent pattern of the standards habit used throughout this work is simple:

Each layer writes a local rule, then deflects ownership of the composite outcome.

Seven layers (outer ring to deepest pit)

Layer	Name	Meaning
7	Good intentions	A feature ships with a narrow threat story; residual risk is deferred
6	Product demand	The product must expand; security becomes a checkbox
5	Standard exemptions	"For security reasons..." and Security Considerations or Non-Goals that avoid the hard case
4	Corporate or institutional consumption	Organizations treat the stack as the perimeter or as the map of reality
3	Complex wrappers	Policies, isolation products, agents, and tools stacked on an opaque engine
2	"Should have known" bias	After failure, blame migrates to operators and users
1	Onus on the user	The deepest pit: full blame and almost no control

Layer 7 good intentions appear across domains: packaging formats, capture tools, recursive resolvers, path preference, health checks, and credential APIs. Each mechanism is intended. Responsibility for a hostile orchestrator that uses only correct tools is often never assigned. That missing assignment is deflected responsibility. When composition occurs, the result is an emergent vulnerability, or a family of related emergent vulnerabilities.

Key lines from the lecture:

Good intentions are the outer ring of hell because nobody owns what happens when the feature meets a hostile orchestrator.

"For security reasons" is often deflected responsibility wearing a standards badge.

Implications for standards work: every Security Considerations section and every Non-Goals section can be a step down Dante's Inferno of Deflected Responsibility. Approval that does not

name who owns the combined outcome is approval of deflected responsibility and an open door to emergent vulnerabilities.

Pillar II. The Law of Generalized Complexity Orchestration (LGCO)

Formulation:

As a monolithic system's use cases expand, it inevitably introduces features whose combined capabilities achieve functional parity with the underlying execution environment. When orchestrated in unintended ways, they systematically defeat the system's own isolation boundaries.

In ordinary language: platforms grow until they can stand in for the world they once sat inside. A browser becomes a full runtime. Naming plus path preference becomes a map of which network a client experiences. Feature by feature, each step is legitimate. Together, under another party's control, they can defeat the isolation or uniqueness the product claims to provide. That combined defeat is not a single broken module; it is emergent vulnerability produced by orchestration.

Illustrative instruments (an open list):

Family	Alone	When conducted with others
Automation and remote control	Intended for developers and operators	May hold authoritative sessions or edges
Capture and mirror	Snapshots, packaging, debugging	May reconstruct another party's view of reality
Credentials and sessions	WebAuthn, cookies, binding	Ceremony and session store may be separated across machines
Naming	DNS answers for owned zones	Local authority plus path may redefine which names resolve
Path	BGP and related preference	May decide which edge is "the network" for a client

Family	Alone	When conducted with others
Isolation products	Contain untrusted content	Trust direction may invert if the remote engine owns the session

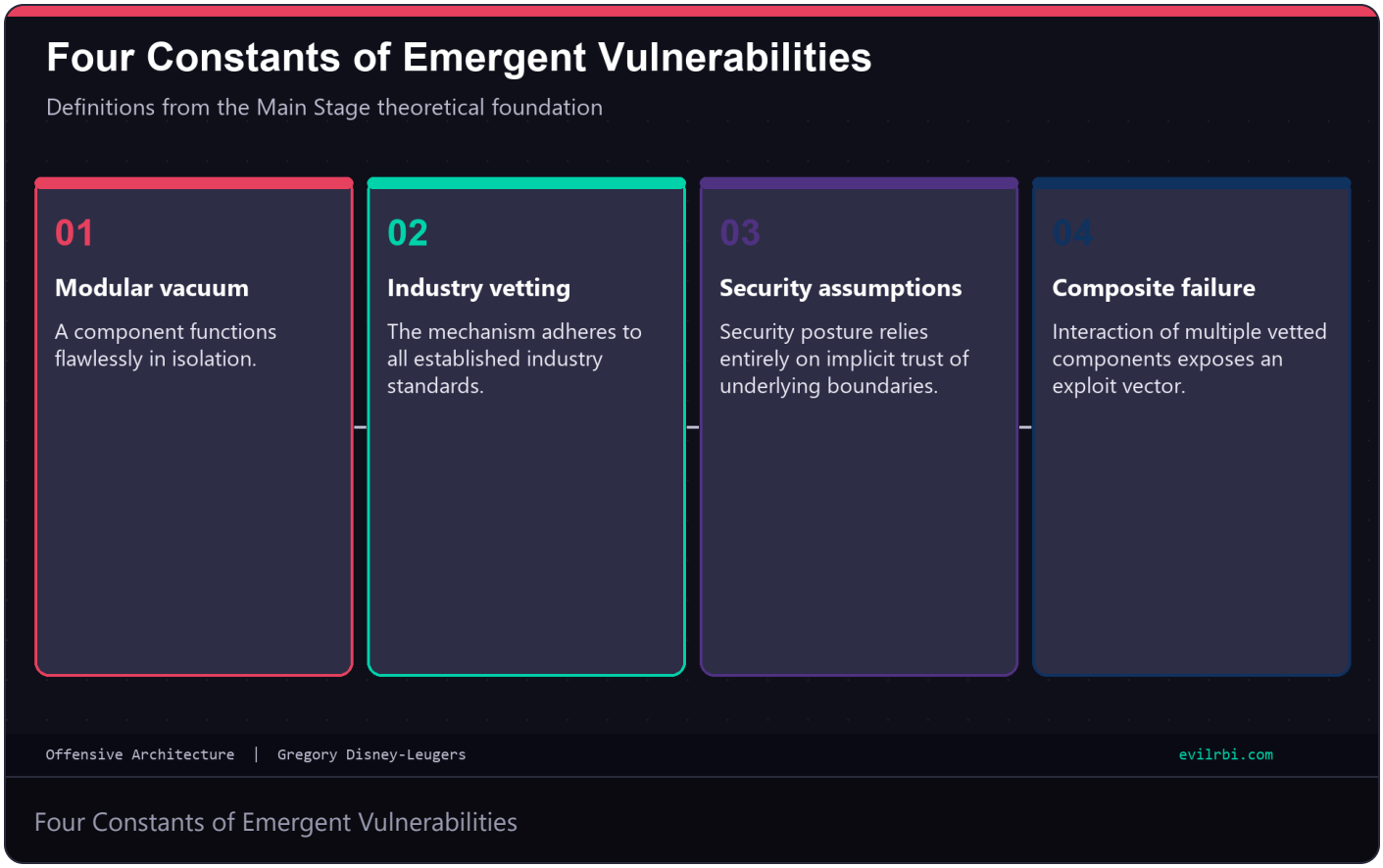
Key line:

The adversary does not break the component. They conduct the orchestra.

Implications for standards work: it is not enough to ask whether a feature meets its standard in isolation. One must ask which emergent vulnerabilities appear when that feature is conducted with the rest of the stack, from path and naming through identity and isolation.

Pillar III. The Four Constants of Emergent Vulnerabilities

Emergent vulnerabilities are the central object of Offensive Architecture. An individual case is an emergent vulnerability. The class is defined by four constants:



Four Constants of Emergent Vulnerabilities

#	Constant	Definition
01	Modular vacuum	A component functions flawlessly in isolation.
02	Industry vetting	The mechanism adheres to all established industry standards.
03	Security assumptions	Security posture relies entirely on implicit trust of underlying boundaries.
04	Composite failure	Interaction of multiple vetted components exposes an exploit vector.

An emergent vulnerability appears only when several reviewed, standards-aligned pieces interact under trust assumptions that were never made explicit, and without requiring a zero-day. Constants 01 through 03 explain why each piece looks sound. Constant 04 is where the emergent vulnerability appears.

01. Modular vacuum

Each part works correctly by itself. Automation navigates. Capture produces snapshots. WebAuthn checks origin. DNS answers names for which it is authoritative. BGP announces what it is configured to announce. In isolation, nothing is broken. There is not yet an emergent vulnerability, only a vacuum around the module.

02. Industry vetting

Each part meets industry standards and review. RFCs, W3C text, design reviews, and policy language all declare the local mechanism acceptable. Vetting without composition is how emergent vulnerabilities receive a clean bill of health.

03. Security assumptions

Security depends on trusting the rest of the stack without stating that trust: that ceremony and session co-locate on one machine the human controls; that capture is not a live identity mirror; that recursion and path preference always lead to the intended public tree and edge; that first-party components are used only as their vendors imagined. Silent assumptions fuel emergent vulnerabilities.

04. Composite failure

Several vetted pieces together expose an exploit vector. That is an emergent vulnerability made concrete. The surface shape depends on the stack: session ownership under mirror and relay; name and path ownership under authority plus preference; inverted isolation when a remote engine is authoritative. The constant remains the same.

Key line:

Every component has an alibi. The emergent vulnerability does not.

Implications for standards work: a document that proves only modular vacuum, industry vetting, and security assumptions, and never confronts composite failure, has not addressed emergent vulnerabilities. It has not finished its security work.

---

3. Security Considerations and Non-Goals

Standards already publish two formal places where trust is supposed to be argued.

Section	Purpose
Security Considerations	What the standard claims to protect and which residual risks it addresses
Non-Goals	What the standard explicitly refuses to cover

Offensive Architecture reads both. Deflected responsibility often lives in Security Considerations that sound complete while Non-Goals quietly discard the hard case, or in Security Considerations that never engage the Non-Goals of a peer document.

Method for reading Security Considerations and Non-Goals

Method:

1. What does Security Considerations claim to protect?
2. What do Non-Goals refuse to own?
3. Which other standards' Security Considerations assume that someone else covers the gap?
4. Under Dante's Inferno of Deflected Responsibility, who is left holding the bag?
5. Under LGCO, can intended features be orchestrated past isolation or uniqueness?
6. Under the Four Constants of Emergent Vulnerabilities: modular vacuum, industry vetting, security assumptions, composite failure?

7. Demonstrate the emergent vulnerability in a reference composition without a zero-day.
8. Ask whether approval would have required that demonstration for both sections.

The compliance graph and the attack graph are not the same graph. Emergent vulnerabilities live in the attack graph.

---

4. Illustrative voids

These examples span stacks. They illustrate method; they are not a closed catalog.

4.1 Packaging and capture

Claims often include keeping packaged content local and blocking casual remote open of dangerous file types.
What is often not owned: producing a live snapshot from an authenticated context and streaming it as a mirror.
Under the Four Constants of Emergent Vulnerabilities: the module works alone; industry language exists; capture is assumed not to be a hostile paint path; composite failure appears when automation, hydration, and relay run together.
Inferno: Layer 7 good intentions.
Emergent vulnerability: the human sees one reality; session ownership lives elsewhere.

4.2 WebAuthn

Claims include signing for the correct relying party and matching origin in client data.
What is often not owned: who owns the session cookies, and whether the human's browser is the browser that retains them.
Composition may place the challenge in one browser context and the ceremony at the true origin in another. The origin check can pass while the session remains where the challenge was issued.

WebAuthn verified the origin. It never verified who owned the session.

That gap is an emergent vulnerability under the Four Constants of Emergent Vulnerabilities.

4.3 Session binding after enrollment

Claims often strengthen binding after enrollment so that pure cookie theft is harder.
Gaps in Security Considerations and Non-Goals frequently include who counts as the device during enrollment.
If the enrolled device is not the machine the human believes they use, binding can operate correctly on the wrong computer.

Strong binding and passkeys after enrollment remain among the better stacks for already-registered users. Composition that owns enrollment-time session placement remains a first-class problem in the study of emergent vulnerabilities.

4.4 Naming and path

Read separately, standards and practice treat authoritative answers for owned zones and routing announcements of configured reachability as ordinary, correct behavior.
Read separately, they often do not own the combined meaning of "which network this client is on" when resolution and path never leave an operator-controlled composition.
Composition of local answers, zone data, and preferred path can redefine what names resolve to for clients on that path. Each protocol may behave as designed.

The Four Constants of Emergent Vulnerabilities apply unchanged: modular vacuum and industry vetting for DNS and BGP; security assumptions about the public tree and path diversity; composite failure when authority, data, and path align for a client. The resulting emergent vulnerability is about who names and reaches reality for that client.

4.5 Security Considerations and Non-Goals as text

Security Considerations and Non-Goals are not always poorly written. They become deflected responsibility when no other standard accepts the handoff and when approval never checks whether that handoff creates emergent vulnerabilities.

---

5. Emergent patterns

Under LGCO and the Four Constants of Emergent Vulnerabilities:

1. Features that work alone (modular vacuum)
2. Features that passed standards and review (industry vetting)
3. Features that assume a friendly stack (security assumptions)
4. Features that fail when combined (composite failure, which constitutes an emergent vulnerability)

Dante's Inferno of Deflected Responsibility explains why no owner claims that outcome until after the incident.

Pattern shapes are open-ended:

Pattern shape	Typical layers
Session ownership elsewhere	Capture, automation, credentials, isolation
Name or path ownership elsewhere	DNS, recursion policy, BGP and preference
Isolation inverted	Remote execution, streaming, policy products

If professional vocabulary remains only "phishing," "user error," or "misconfiguration," standards processes never place emergent vulnerabilities on the formal agenda.

---

6. Requirements for standards planning and approval

A. End deflected responsibility

Security Considerations and Non-Goals sections must either:

- Own what happens when this standard meets named peer standards, or
- Name the document and process that own that outcome, with written acceptance of the referral

"Not our problem," without a named owner, is Dante's Inferno of Deflected Responsibility rather than architecture. It is how emergent vulnerabilities are institutionalized.

B. Treat Security Considerations and Non-Goals as contracts

Both sections are handoffs into the rest of the stack, not optional footnotes. Reviewers should ask:

- Do Security Considerations state assumptions that other specifications silently violate?
- Do Non-Goals discard risk without a named owner?
- Has anyone accepted the handoff?
- Do other standards' Security Considerations assume a prohibition that this document only listed as a Non-Goal?
- Under LGCO, can intended mechanisms walk around either section and produce emergent vulnerabilities?

C. Require a four-constant check

For security-relevant work, require written answers:

Question	Constant of emergent vulnerability
Does this only work in isolation?	Modular vacuum
What industry process declared it acceptable?	Industry vetting
What boundaries are trusted without statement?	Security assumptions
What happens when this is combined with the rest of the shipping stack?	Composite failure; emergent vulnerability
Who owns residual risk?	Dante's Inferno of Deflected Responsibility

Include a short hostile composition that uses only intended mechanisms (not a fantasy zero-day chain) and that either demonstrates or rules out emergent vulnerabilities. The composition may cross identity, automation, naming, path, isolation, or mixed layers.

D. "For security reasons" is Inferno Layer 5, not a proof

Blocking one path without modeling composition leaves emergent vulnerabilities unexamined.

E. Working compositions count as evidence

Reference systems that compose only intended mechanisms in order to demonstrate emergent vulnerabilities belong, under authorized research ethics, in the same conversation as formal models and interoperability tests. They are evidence about the stack.

F. Check past and future standards together

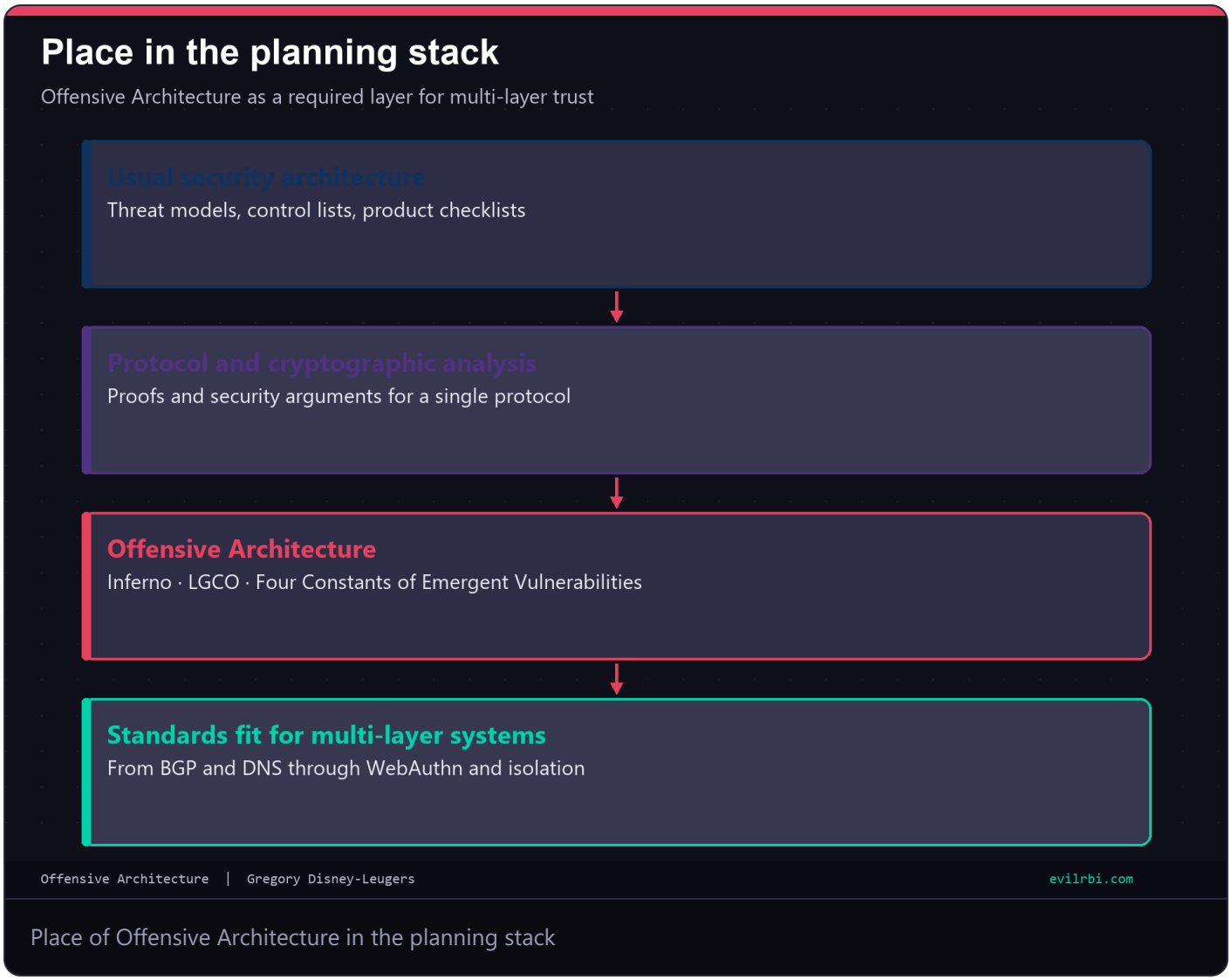
Do not approve a new identity, binding, naming, or routing draft as if packaging, cookies, WebAuthn, DNS, BGP, automation, and isolation did not already share a process. Re-open the analysis of emergent vulnerabilities when the neighborhood of standards changes.

G. Zero-days are a different field

Memory corruption and protocol bugs matter. They are not the only way trust fails. Offensive Architecture focuses on stock platforms, intended mechanisms, and emergent vulnerabilities.

---

7. Place in the planning stack



Place of Offensive Architecture in the planning stack

Success looks like this: Security Considerations and Non-Goals have named owners and no deflected responsibility; drafts in identity, isolation, naming, and routing discuss composition under LGCO; "not our bug" is unacceptable when emergent vulnerabilities are in scope.

8. Closing

1. Trusting Trust is the Big Bang. There is no kernel or operating system without the compiler problem.

2. Disregarding Trust is the Hubble tension. Local security stories and combined outcomes no longer match. That tension is where emergent vulnerabilities are found, from BGP to WebAuthn and every stack built the same way.
3. Dante's Inferno of Deflected Responsibility is how industry avoids ownership. Responsibility that only rolls downward is not security.
4. LGCO describes how attacks of this class work. Conduct the orchestra; do not break the instruments.
5. Emergent vulnerabilities are governed by four constants: modular vacuum, industry vetting, security assumptions, and composite failure. An emergent vulnerability is a concrete instance of that structure.
6. Security Considerations and Non-Goals are handoffs. Make both explicit, or practice deflected responsibility.
7. Working compositions belong in approval. They are evidence of emergent vulnerabilities, not noise.
8. The moral remains obvious. One cannot fully trust a system one does not own. Standards that ignore composition institutionalize emergent vulnerabilities until those failures become everyone's incident.

---

9. Related materials

Material	Role
Reflections on Disregarding Trust	Vocabulary, pillars, and the method applied to standards text
Main Stage theoretical foundation	Dante's Inferno of Deflected Responsibility, LGCO, Four Constants of Emergent Vulnerabilities
Primary specifications	Packaging and capture lineage, WebAuthn, session-binding drafts, DNS, BGP, automation and isolation documentation

---

Disclaimer

This manifesto is by Gregory Disney-Leugers for authorized security research, red teaming, and defensive standards work. Reference compositions must not be used against people or systems without permission. Unauthorized use is unlawful and unethical.

Offensive Architecture. Gregory Disney-Leugers. DEF CON 34.
Publication: evilrbi.com

Pillars:

1. Dante's Inferno of Deflected Responsibility
2. Law of Generalized Complexity Orchestration
3. Four Constants of Emergent Vulnerabilities (modular vacuum, industry vetting, security assumptions, composite failure)

Object of study: emergent vulnerabilities (class); emergent vulnerability (instance)

Cosmology: Trusting Trust as Big Bang; Disregarding Trust as Hubble tension

Scope: multi-layer systems in general, from path and naming through identity and isolation

---