INTERNET-DRAFT
Intended status: Standards Track
Expires: December 26, 2025

R. Bouziane
SecRoot.io

June 26, 2025

OODA-HTTP: Adaptive Security Framework for HTTP Communications
draft-secroot-ooda-http-01

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at https://www.ietf.org/shadow.html

This Internet-Draft will expire on December 26, 2025.

Copyright Notice

Abstract

This document defines OODA-HTTP, a protocol extension and adaptive security framework that applies the Observe-Orient-Decide-Act loop to HTTP/HTTPS communications. OODA-HTTP introduces dynamic behavior at the application layer, enabling real-time detection, decision, and mitigation based on evolving threat contexts. Each HTTP request becomes not just a message, but an observation signal, a decision vector, and a defensive act. This framework introduces headers such as X-OODA-Action to carry contextual scores or actions across clients, servers, and intermediaries. OODA-HTTP transforms passive transport into active self-defense, offering resilience against classic and quantum threats. It is designed as a native defender of the World Wide Web \[Gupta23]\[Martinez22]\[Wang21]\[Kim23].

1. Introduction

Strategic Positioning Note

In warfare, if we do not have the same weapons, we do not even consider engaging. We remain passive.This is the current reality: the HTTP protocol,

even when secured via TLS (HTTPS), remains passive in the face of next-generation threats — especially automated intelligent agents.

OODA-HTTP introduces a paradigm shift. It transforms HTTP from a passive transport protocol into an active defensive actor, capable of observing, orienting, deciding, and acting in context — a necessary evolution in the age of automated attacks.

HTTP and HTTPS communications face increasingly sophisticated threats, including quantum computing-based attacks such as those leveraging Shor's algorithm. OODA-HTTP introduces a real-time adaptive security mechanism based on the OODA loop (Observe-Orient-Decide-Act) to enhance resilience. The framework leverages adaptive security paradigms demonstrated in enterprise networks \[Wang21], cryptographic key rotation in TLS \[Gupta23], and intrusion detection methodologies using OODA principles \[Martinez22].

Traditional HTTPS treats security as a static, pre-negotiated layer, focused on encryption, authentication, and transport integrity. It lacks the ability to adapt to runtime behavior or evolving threat conditions during a session. OODA-HTTP fills this gap by enabling context-aware decision loops and dynamic protocol responses.

This capability can also benefit application-layer components. Client-side applications may interpret OODA-HTTP signals to adjust their behavior in real time. Suspicious sessions can trigger interface restrictions, challenges, or warnings, enabling proactive defense at the user interaction layer.

2. Terminology

The following terms are used throughout this document:

OODA Loop:
A decision-making model consisting of four sequential phases: Observe, Orient, Decide, and Act. Originally developed for military strategy, it is applied here to network communications.

Threat Score:
A numeric or categorical indicator representing the risk level associated with a session, request, or identity. It can be computed locally or distributed, and may evolve over time.

X-OODA-Action:
A custom HTTP header used to transmit OODA-driven decisions or scores between entities (clients, servers, intermediaries).

Post-Quantum Cryptography:
Cryptographic algorithms designed to resist attacks from quantum computers. Mentioned in this draft in relation to resilience against Shor's algorithm.

Shor's Algorithm:
A quantum algorithm \[Shor94] capable of factoring large integers exponentially faster than classical algorithms, posing a threat to widely-used public-key cryptosystems.

Contextual Adaptation:
The ability of an endpoint or middleware to adjust its behavior based on environmental, behavioral, or risk-related context.

Self-Defending Protocol:
A protocol that is capable of taking real-time protective actions based on observed conditions, without requiring external triggers.

3. Architectural Overview

OODA-HTTP introduces a modular security architecture that integrates decision-making capabilities into existing HTTP and HTTPS infrastructures. The architecture is composed of four key components that operate along the OODA loop: telemetry collection (Observe), threat analysis (Orient), decision engines (Decide), and enforcement mechanisms (Act).

The framework can be deployed at various levels: client applications, web servers, reverse proxies, smart edge routers, CDN nodes, or security middleware. It does not replace existing TLS or HTTP protocols but extends them by introducing headers and context-aware evaluation.

A central concept in the architecture is the notion of local or distributed threat scores. These scores guide decisions that may be enforced via the X-OODA-Action header or other configured policies. The system may operate in standalone mode or integrate with machine learning models for adaptive behavior.

The following sections describe how each phase of the OODA loop is implemented within this architecture.

4. OODA-HTTP Phases

OODA-HTTP applies the four phases of the Observe-Orient-Decide-Act (OODA) loop to HTTP communications. Each phase is mapped to technical operations that collectively enable adaptive security in real time.

4.1. Observe

The Observe phase consists of collecting telemetry data from HTTP requests, TLS handshake metadata, timing information, user-agent patterns, and behavioral signals. This data may be extracted by edge devices, proxies, or endpoints and stored locally or streamed to a threat evaluation engine.

4.2. Orient

In the Orient phase, collected data is contextualized and interpreted using rule-based logic or machine learning models. Features may include request frequency, signature anomalies, protocol violations, or correlation with known attack patterns. This phase results in assigning a threat score to the current session or request.

4.3. Decide

Based on the threat score and predefined security policies, the Decide phase selects an appropriate mitigation strategy. The decision may include allowing, delaying, challenging, throttling, or blocking the request. Optionally, the decision can be embedded into the request via the X-OODA-Action header for downstream enforcement.

4.4. Act

The Act phase enforces the decision selected in the previous phase. This may include applying rate limits, returning captchas, rotating TLS keys, logging events, or updating dynamic filters. The act phase may also emit metadata that becomes observable in future requests, enabling feedback loops.

5. Message Formats and Protocol

OODA-HTTP introduces protocol-level extensions for transmitting threat context and decisions using HTTP headers and JSON-based structures.

The primary extension is the X-OODA-Action header, which carries a compact JSON object encoding the outcome of the Orient and Decide phases.

Example format:

X-OODA-Action: {
"score": 72,
"category": "suspicious",
"action": "challenge",
"timestamp": "2025-06-28T15:42:10Z"
}

Fields:

* score (integer): A numeric threat score (e.g., 0100).
* category (string): A textual label for the threat level (e.g., "normal", "suspicious", "malicious").
* action (string): Suggested enforcement action (e.g., "allow", "block", "challenge", "throttle").
* timestamp (string): ISO 8601 UTC timestamp of the decision.

Servers, clients, or intermediaries such as edge devices or security gateways may consume this header and apply local policies accordingly. The header is optional and may be stripped or overwritten depending on policy scope and trust boundaries.

6. Threat Models and Detection

Un paquet peut tout changer.
OODA-HTTP transforms each request into an opportunity for defense, analysis, and adaptation.

OODA-HTTP is designed to detect and mitigate a broad range of threats, spanning classical, behavioral, and quantum-enabled attacks. The threat model includes both protocol-level and application-level vectors, and supports both static and adaptive detection methods.

6.1. Classical Threats

These include traditional attacks such as:

* Denial of Service (DoS) and Distributed DoS (DDoS)
* Slowloris and application-layer flooding
* Bot impersonation and session replay
* Payload manipulation and malformed headers

6.2. Behavioral Threats

These cover attacks identified via patterns of behavior, including:

* Unusual request rates or frequency spikes
* Time-based anomalies (e.g., bursts, silent intervals)
* Identity inconsistency or rotating user agents
* Suspicious navigation or incomplete flows

6.3. Quantum-Adaptive Threats

OODA-HTTP anticipates emerging post-quantum scenarios, such as:

* Precursor attacks preparing for future quantum decryptions
* TLS handshake manipulation to record and break sessions later
* Exploitation of weak randomness or low entropy in key negotiation
* Threats leveraging Shor’ s algorithm \[Shor94] to factor captured

session keys from classical cryptosystems

Adaptive defenses can preemptively respond by applying key rotations \[Gupta23], adjusting handshake profiles, or introducing entropy-hardening techniques.

7. Applications: Case Studies and Practical Validations

OODA-HTTP has been evaluated across simulated and real-world environments to validate its adaptability and effectiveness. The following use cases highlight its potential in operational deployments.

7.1. Burst Attack Detection in CDN Edge Routers

OODA-HTTP deployed on CDN edge nodes enabled the early detection of short-duration, high-intensity traffic bursts aimed at exhausting caching layers. By integrating local telemetry and threat scoring, edge routers adapted QoS and activated throttling rules dynamically.

7.2. Slowloris Mitigation in Reverse Proxies

In reverse proxy deployments, OODA-HTTP successfully identified slow request patterns through timing and header irregularities, triggering ‘X-OODA-Action: block‘ to terminate low-rate attacks before reaching backend applications.

7.3. Bot Detection in Application Gateways

Application-layer integrations combined behavioral scoring and fingerprint rotation detection. OODA-HTTP headers propagated context to frontend gateways, where requests flagged as "malicious" were challenged or dropped.

7.4. Post-Quantum Resilience Simulation

A TLS testbed integrated OODA-HTTP to react to low-entropy handshakes and detect clients using weak key negotiation. Upon scoring these sessions as high-risk, the system enforced real-time key rotation \[Gupta23] and triggered entropy hardening for subsequent sessions.

7.5. Frontend Integration with Reactive Applications

In client-side environments such as single-page applications, OODA-HTTP enables dynamic UI adjustments based on the threat context. For example, an application may read the X-OODA-Action header from server responses and respond accordingly:

* If action = "challenge", trigger a CAPTCHA or 2FA prompt.
* If category = "suspicious", restrict certain UI interactions.
* If score > threshold, log session metadata or notify the user.

This integration supports self-defending application logic, where frontend components become adaptive actors within the OODA loop. Security is no longer passive but embedded into user-facing flows.

OODA-HTTP is the first protocol where the user interface, HTTP transport, and adaptive security speak the same language.

8. Integration with TLS/HTTPS

OODA-HTTP is designed to complement and extend existing TLS and HTTPS infrastructures without breaking compatibility. It operates at the application layer while observing and interacting with the TLS stack through termination proxies, inspection agents, or endpoints.

The protocol can integrate with TLS in the following ways:

* Observing handshake metadata such as cipher suite selection, handshake timing, and session reuse patterns.

* Reacting to entropy metrics or randomness sources used in key negotiation to detect low-entropy or anomalous handshakes.

* Coordinating key rotation triggers based on observed threat thresholds \[Gupta23], especially in environments preparing for post-quantum resilience.

* Tagging TLS sessions with contextual scores or identifiers via out-of-band headers, shared memory, or internal metadata systems.

OODA-HTTP does not alter the TLS protocol itself but enhances its observability and adaptive response capabilities from the HTTP layer.

9. Protocol Engineering Foundations

OODA-HTTP is grounded in classic protocol engineering principles, including modularity, state-awareness, error handling, and feedback loops. It leverages established design techniques to ensure resilience and extensibility across HTTP/TLS infrastructures.

The protocol supports the following foundational characteristics:

* State-driven logic: Each phase of the OODA loop transitions deterministically based on observable input.

* Extensibility: Headers such as X-OODA-Action are optional and can evolve independently of core protocol semantics.

* Backward compatibility: OODA-HTTP does not modify HTTP or TLS wire formats, ensuring that legacy systems can interoperate without disruption.

* Fault tolerance: If any OODA component fails or becomes unreachable, systems default to standard HTTP behavior, maintaining service continuity.

* Measurability: All actions and scores are loggable, audit-friendly, and interpretable by external observers.

These principles align with long-standing guidance in protocol architecture and testing methodologies \[Sar93].

10. Security Considerations

Unlike purely defensive frameworks, OODA-HTTP is both a protocol and a security engine, embedded directly into HTTP interactions.

OODA-HTTP introduces new information flows through headers and decision logic. The following considerations apply:

* Confidentiality: OODA-HTTP headers such as X-OODA-Action may reveal threat assessment details or system behavior. These headers should not be exposed across trust boundaries unless encrypted or filtered.

* Integrity: Decisions derived from threat scores should be protected against tampering. Systems must ensure that OODA headers cannot be forged or overwritten by untrusted intermediaries.

* Fallback safety: If OODA-HTTP components are disabled or unavailable, systems should gracefully default to standard HTTP/TLS behavior to

maintain availability.

* Score manipulation: Care must be taken to avoid creating exploitable feedback loops or incentives for malicious actors to influence threat scores through adversarial behavior. However, attempted score manipulation may itself serve as a signal of systemic vulnerability, revealing weaknesses in session-level security or orchestration flow.

* Human supervision: Automated decisions must be auditable and, where appropriate, overrideable by human operators, especially in sensitive applications.

Implementations should ensure secure defaults, minimize information leakage, and respect separation of concerns between detection, decision, and enforcement layers.

11. IANA Considerations

This document registers a new HTTP header field under the "Permanent Message Header Field Names" registry maintained by IANA.

Header Field Name: X-OODA-Action
Applicable Protocol: HTTP
Status: Provisional
Author/Change Controller: IETF
Specification Document: This document

No new TLS parameters or ALPN identifiers are requested at this time. Future versions may introduce identifiers for deeper protocol-level integration.

12. Vector Engine and Temporal Memory

OODA-HTTP introduces a semantic vector engine that transforms each request into a multi-dimensional observation vector. This engine supports temporal memory, pattern accumulation, and learning-based refinement, enabling the protocol to mature over time.

12.1. Observation Vectors

During the Observe phase, each HTTP interaction is converted into a structured vector containing metadata fields such as:

TLS fingerprint: cipher suite, handshake entropy, session reuse

HTTP metadata: method, headers, frequency, burst timing

Client behavior: agent string variability, navigation flow, interaction latency Environment: IP locality, ASN, device type

These vectors are timestamped and stored in short- or long-term memory modules.

12.2. Orientation Space and Semantic Patterns

The Orient phase processes these vectors using rule-based classification or ML inference models. Similar vectors are grouped and compared across:

Known attack patterns (e.g., slowloris, botnets, smart agents)
Session anomalies (e.g., inconsistent headers, entropy irregularities)
Behavioral deviations from normative baselines

The system learns correlations over time, allowing it to recognize increasingly complex or obfuscated threats.

12.3. Temporal Threat Memory

Unlike static HTTP/HTTPS stacks, OODA-HTTP maintains an evolving threat memory:

Short-term memory detects bursts, replay attempts, and adaptive scanning.

Long-term memory builds threat reputation for agents, sessions, or IP zones.

This memory enables cross-request correlation, historical fingerprinting, and context-based threat anticipation.

12.4. Maturity Through Activity

As OODA-HTTP observes more traffic, it becomes more effective. Each request strengthens its adaptive models, building resilience without explicit retraining. This contrasts with traditional HTTP/HTTPS protocols, which remain blind to behavioral evolution.

OODA-HTTP becomes smarter, faster, and more precise the more it is used—turning activity into capability.

13. References

13.1. Normative References

[Sarikaya19] B. Sarikaya, "Adaptive Security in Wireless Networks: A QoS Perspective," IEEE Communications Surveys & Tutorials, 2019.

[Sar93] B. Sarikaya, "Principles of Protocol Engineering and Protocol Testing," Ellis Horwood, 1993.

13.2. Informative References

[Gupta23] S. Gupta et al., "Automatic Key Rotation for TLS Sessions to Mitigate Quantum Threats," IEEE INFOCOM, 2023.

[Kim23] J. Kim et al., "Real-Time QoS and Security Adaptation in Software-Defined Networking," ACM SIGCOMM, 2023.

[Martinez22] L. Martinez et al., "Applying the OODA Loop to Intrusion Detection: A Case Study," USENIX Security Symposium, 2022.

[Wang21] D. Wang et al., "Implementation and Evaluation of Adaptive Security Frameworks in Enterprise Networks," IEEE Transactions on Network and Service Management, 2021.

[Zhang21] X. Zhang et al., "Threat-aware QoS Management for Secure Wireless Networks," IEEE Transactions on Network and Service Management, 2021.

[Shor94] P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994.

Authors' Addresses

Rachid Bouziane
SecRoot.io
Villa El Majd 171, Tamsna Temara
Rabat, Morocco
Email: contact@secroot.io

Expires: December 26, 2025