Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 28 December 2025

R.W.D. Cameron
Prometheus Systems

26 June 2025

Best Practices for Secure ICAP Deployment with Credential-Based TLS
draft-cameron-secure-icap-bcp-00

Abstract

This document defines a Best Current Practice for deploying the Internet Content Adaptation Protocol (ICAP) in a secure manner. It introduces a mechanism for symmetric encryption using TLS, with client authentication based on user credentials. The aim is to provide confidentiality and integrity of ICAP traffic in modern zero-trust and content-sensitive environments.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 28 December 2025.

Table of Contents

1. Introduction

ICAP [RFC3507] provides offloading of content adaptation to dedicated servers but lacks native support for transport-layer security and authentication. This document offers operational guidance for deploying ICAP securely using TLS with short-lived credential-based sessions to enable multi-tenant ICAP services.

2. Scope

This BCP is intended for ICAP implementers and administrators deploying ICAP within security-sensitive environments where plaintext transport and persistent sessions are undesirable. This is particularly applicable to public ICAP services serving multiple customers.

3. Authentication and Key Derivation

Clients MUST authenticate using a username/password mechanism encoded via HTTP Basic Authorization header. This credential pair MUST be used to derive a 256-bit session key using PBKDF2 with SHA-256.

The derived session key serves as an additional authentication layer beyond TLS transport security, enabling multi-tenant session isolation and per-customer access control within the ICAP service.

4. TLS Enforcement

ICAP servers MUST accept connections only over TLS 1.2 or newer. The recommended port for secure ICAP ("ICAPS") is 11344. Non-encrypted connections SHOULD be rejected by default.

5. Session Expiry

Session keys and associated TLS sessions MUST expire no later than one (1) hour after creation. Clients SHOULD re-authenticate upon session renewal.

6. ICAP Header Constraints

Implementations MUST include the Authorization header during the initial handshake. TLS renegotiation SHOULD NOT be used. Session rekeying MUST be triggered by explicit re-authentication.

7. Operational Guidance

* Session caching MUST be scoped per client based on the derived session key and authentication credentials.

* Salt and iteration count for PBKDF2 SHOULD be dynamically generated by the server and delivered via TLS extension parameters.

* Clients SHOULD validate server certificates using pinned keys or a trusted CA.

* To prevent downgrade attacks, the Options response MAY advertise "Require-TLS: yes".

8. Security Considerations

* Weak password policies MUST be avoided.

* TLS session resumption SHOULD be disabled.

* Authorization headers MUST NOT be logged or echoed in diagnostics.

9. IANA Considerations

No new IANA actions are required by this document.

10. References

10.1. Normative References

[RFC3507] Elson, J. and A. Cerpa, "Internet Content Adaptation Protocol (ICAP)", RFC 3507, DOI 10.17487/RFC3507, April 2003, <https://www.rfc-editor.org/info/rfc3507>.

Appendix A. Example: ICAP over Secure TLS

```
C: CONNECT icap.example.com:11344
S: <TLS Handshake>
C: Authorization: Basic <base64-encoded credentials>
...
<ICAP request begins within TLS session>
```

Appendix B. Sample Implementation (Pseudo-code)

Client Implementation

```
function connectToSecureICAPServer(username, password):
    tlsSocket = TLS.connect("icap.example.com", 11344,
                            verifyCertificates=True)

    credentials = base64Encode(username + ":" + password)
    authHeader = "Authorization: Basic " + credentials

    // Prepare ICAP OPTIONS request
    request = ICAPRequest()
    request.method = "OPTIONS"
    request.uri = "icap://icap.example.com/reqmod"
    request.headers["Authorization"] = authHeader

    tlsSocket.send(request.serialize())

    response = tlsSocket.receive()
    if response.statusCode == 200:
        print("Secure session established.")
        startSessionTimeout(1 hour)
    else if response.statusCode == 407:
        print("Authentication failed.")
        tlsSocket.close()
```

Server Implementation

```
   function handleSecureICAPConnection(tlsSocket):
       request = parseICAPRequest(tlsSocket.receive())

       if not isValidTLS(tlsSocket):
           tlsSocket.send(ICAPErrorResponse(403))
           return

       if "Authorization" not in request.headers:
           tlsSocket.send(ICAPAuthChallenge(407))
           return

       credentials = parseBasicAuth(request.headers["Authorization"])
       if not validateUser(credentials.username, credentials.password):
           tlsSocket.send(ICAPAuthChallenge(407))
           return

       sessionKey = deriveKeyPBKDF2(
           password=credentials.password,
           salt=generateServerSalt(),
           iterations=100000,
           keyLength=256
       )

       // Mark session valid for 1 hour based on session key
       createSecureSession(sessionKey, ttl=1 hour)

       tlsSocket.send(ICAPSuccessResponse(200))
```

Author’s Address

Ross W.D. Cameron
Prometheus Systems
Email: rwd.cameron@prometheus-systems.co.za