Network Working Group
Internet-Draft
Intended status: Historic
Expires: 5 November 2026

D. J. Vance
Independent

4 May 2026

# SOCKS Protocol Version 4A
## draft-vance-socks-v4a-02

## Abstract

This document specifies SOCKS Protocol Version 4A, an independent protocol originated from the SOCKS Protocol Version 4. This protocol allows SOCKS clients to delegate domain name resolution to the SOCKS server. This is particularly useful in environments where the client host cannot resolve the destination host's domain name due to restrictive network policies or lack of DNS access.

## Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/4socks/socks4.

## 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 5 November 2026.

## Copyright Notice

Table of Contents

## 1. Introduction

The original SOCKSv4 protocol ([I-D.vance-socks-v4]) requires the client to provide the destination host's IPv4 address. However, in many firewall configurations, the client resides on a network without direct DNS access to the outside world. SOCKS 4A addresses this by allowing the client to provide a domain name string instead of a resolved IP address.

## 2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification uses the following terms:

* Client (Application Client): The program requesting a connection to an application server through the SOCKS server.

* SOCKS Server: The host, typically at a firewall, that intermediates the connection between the Client and the Application Server.

* Application Server: The host to which the Client ultimately wishes to connect (e.g., a Telnet daemon, an HTTP server).

* TCP Session: A connection established using the Transmission Control Protocol (TCP). SOCKSv4 only supports TCP sessions.

* DSTIP (Destination IP): The IP address of the Application Server, as specified in the SOCKS request.

* DSTPORT (Destination Port): The port number of the Application Server, as specified in the SOCKS request.

* USERID: A variable-length, NULL-terminated string identifying the client's user on the local system.

* NULL: A byte of all zero bits, used to terminate the USERID field.

## 3. Protocol Mechanism

The only behaviors that SOCKS 4A is different from the original SOCKSv4 is triggered by a specific, non-routable pattern in the DSTIP field of a standard SOCKSv4 request.

### 3.1. Request Format

To initiate a SOCKS 4A request (either CONNECT or BIND), the client sends a packet with the following structure:

| Field | Description | Size (bytes) | Value/Notes |
|---|---|---|---|
| VN | Version Number | 1 | 0x04 |
| CD | Command Code | 1 | 0x01 (CONNECT) or 0x02 (BIND) |
| DSTPORT | Destination Port | 2 | Network Byte Order |
| DSTIP | Destination IP | 4 | 0x00, 0x00, 0x00, x (x != 0) |
| USERID | User Identifier | variable | Variable length, NULL terminated |
| DOMAIN | Destination Domain | variable | Variable length, NULL terminated |

Table 1: SOCKS 4A Request Structure

#### 3.1.1. DSTIP Encoding and Signaling

To signal a SOCKS 4A extension request, the client MUST set the first three octets of the DSTIP field to 0x00 and the final octet to a non-zero value in network byte order (i.e., representing an IPv4 address in the range 0.0.0.1 through 0.0.0.255).

This specific address range, part of the 0.0.0.0/8 block, is reserved by IANA for "this host on this network" [RFC1122] and is not a routable destination. This ensures that the 4A signal is syntactically distinct from standard SOCKSv4 requests. A SOCKS server receiving such a DSTIP MUST ignore its numerical value and proceed to extract the destination address from the DOMAIN field as defined in Section 3.1.2.

#### 3.1.2. Destination Domain Name Field

The DOMAIN field contains the fully qualified domain name (FQDN) of the application server. To ensure protocol stability and prevent common parsing errors, the following rules MUST be observed:

* Positioning: The DOMAIN field MUST begin immediately after the NULL (0x00) terminator of the USERID field.

* Encoding: The domain name SHOULD be encoded in US-ASCII. While some implementations support Internationalized Domain Names (IDNs), clients SHOULD use the Punycode-encoded A-label format [RFC5891] to ensure maximum compatibility.

* Termination: The field MUST be terminated by a single NULL (0x00) octet.

* Length Constraints: The DOMAIN string (excluding the terminator) SHOULD NOT exceed *255 octets*, consistent with the maximum length of a FQDN defined in [RFC1035]. Servers SHOULD enforce a maximum buffer limit for this field to mitigate resource exhaustion attacks.

## 4. Server Processing

Upon receipt of a client request, a SOCKS 4A compliant server MUST process the data according to the following sequential states:

### 4.1. Initial Header Parsing

The server MUST first read the fixed-length 8-octet header. It SHALL evaluate the fields as follows:

* VN: If the version number is not 4, the server SHOULD terminate the connection.

* CD: The server determines the requested operation (CONNECT or BIND).

* DSTPORT: The destination port is extracted for later use in the connection attempt.

* DSTIP: The server inspects the four-octet destination IP address to determine the routing mode (Standard SOCKSv4 or SOCKS 4A).

### 4.2. Routing Mode Selection and Field Extraction

The server MUST apply the following logic based on the DSTIP value:

1. SOCKS 4A Signaling: If the first three octets of DSTIP are zero and the fourth octet is non-zero (0.0.0.x, where x != 0), the server SHALL enter the SOCKS 4A extended resolution mode. The server MUST continue to read the input stream to extract the USERID string, defined as all octets up to and including the

first NULL (0x00) terminator. Immediately following the USERID terminator, the server MUST continue reading to extract the DOMAIN string, defined as all octets up to and including the second NULL (0x00) terminator.

2. Standard SOCKSv4 Handling: If the DSTIP does not match the 0.0.0.x pattern (including the case of 0.0.0.0), the server MUST follow the standard SOCKSv4 procedure, extracting only the USERID field. In this mode, the server MUST NOT attempt to read or interpret any data following the first NULL terminator as a domain name.

## 4.3. Name Resolution and Execution

In SOCKS 4A mode, once the DOMAIN string is extracted:

* Resolution: The server SHALL attempt to resolve the ASCII-encoded domain name to a valid IPv4 address using the server's local DNS resolver or host lookup mechanism.

* Successful Resolution: If the domain resolves to one or more IPv4 addresses, the server SHOULD attempt to establish the requested TCP session (for CONNECT) or bind a socket (for BIND) using the first resolvable and reachable address.

* Resolution Failure: If the domain cannot be resolved, or if the resolver returns an error, the server MUST consider the request failed. It SHALL return a reply packet with CD = 91 and MUST immediately close the connection to the client.

## 4.4. Response Generation

Following the completion (success or failure) of the request processing, the server MUST return an 8-octet reply packet. For SOCKS 4A CONNECT operations, the DSTPORT and DSTIP fields in the reply are typically set to zero and SHOULD be ignored by the client. For BIND operations, these fields MUST contain the specific port and IP address where the SOCKS server is listening for the inbound connection.

| Field | Description | Size (bytes) | Value/Notes |
|---|---|---|---|
| VN | Reply Version | 1 | 0x00 (Null byte) |
| CD | Result Code | 1 | 0x5A (Granted), 0x5B (Rejected/Failed), etc. |
| DSTPORT | Destination Port | 2 | Ignored for CONNECT; provided for BIND |
| DSTIP | Destination IP | 4 | Ignored for CONNECT; provided for BIND |

Table 2: SOCKS 4A Reply Structure

## 5. Security Considerations

See Appendix B.

## 6. IANA Considerations

No IANA actions required.

## 7. References

### 7.1. Normative References

[I-D.vance-socks-v4]
Vance, D. J., "SOCKS Protocol Version 4", Work in Progress, Internet-Draft, draft-vance-socks-v4-09, 4 May 2026, <https://datatracker.ietf.org/doc/html/draft-vance-socks-v4-09>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

### 7.2. Informative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/rfc/rfc1122>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/rfc/rfc1918>.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <https://www.rfc-editor.org/rfc/rfc1928>.

[RFC1929] Leech, M., "Username/Password Authentication for SOCKS V5", RFC 1929, DOI 10.17487/RFC1929, March 1996, <https://www.rfc-editor.org/rfc/rfc1929>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/rfc/rfc2827>.

[RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, DOI 10.17487/RFC3365, August 2002, <https://www.rfc-editor.org/rfc/rfc3365>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/rfc/rfc3552>.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <https://www.rfc-editor.org/rfc/rfc3927>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/rfc/rfc4301>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <https://www.rfc-editor.org/rfc/rfc4732>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/rfc/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/rfc/rfc5891>.

[RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <https://www.rfc-editor.org/rfc/rfc7626>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/rfc/rfc7858>.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/rfc/rfc791>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/rfc/rfc9293>.

# Appendix A. Operational Considerations

The following section outlines the operational behaviors and implementation strategies observed in historical deployments of SOCKS 4A. These notes are provided to ensure interoperability and to document how the protocol handles complex network topologies.

## A.1. Multi-tier Proxying and Chaining

In hierarchical network architectures, an intermediate SOCKS server often acts as a client to an upstream SOCKS server. This configuration, known as proxy chaining, requires specific handling of SOCKS 4A requests to maintain the integrity of the resolution delegation.

### A.1.1. Recursive Resolution and Protocol Downgrade

An intermediate proxy receiving a SOCKS 4A request MAY perform local resolution of the DOMAIN field. If the resolution is successful, the intermediate proxy may elect to "downgrade" the request to a standard SOCKSv4 CONNECT or BIND operation when communicating with the upstream server, using the literal IPv4 address obtained from its local resolver.

While this approach reduces the resolution burden on the upstream server, it requires that the intermediate proxy possesses a DNS environment consistent with the client’s expectations. Implementations should be aware that resolving a domain at an intermediate hop may yield different IP addresses (e.g., in Content Delivery Networks) than resolution at the network edge.

### A.1.2. Transparent Relaying

In environments where the intermediate proxy is situated behind a restrictive firewall without direct DNS access, it SHOULD implement transparent relaying. In this mode, the intermediate proxy preserves the SOCKS 4A signaling (the 0.0.0.x DSTIP pattern) and appends the USERID and DOMAIN fields exactly as received from the client when initiating the upstream connection. This ensures that the resolution intent is signaled end-to-end until it reaches a node capable of performing the lookup.

## A.2. Implementation Robustness and Interoperability

In accordance with the general robustness principle—to be conservative in what you send and liberal in what you accept—SOCKS 4A implementations have historically adopted permissive processing logic to accommodate diverse and sometimes non-compliant client behaviors.

### A.2.1. Handling of Pre-resolved Requests

Certain "SOCKSified" libraries or shim layers may resolve the destination hostname locally via the system's standard resolver before initiating the SOCKS handshake. Despite possessing a valid IP address, these clients may still utilize the SOCKS 4A format, placing the original hostname in the DOMAIN field.

To ensure maximum interoperability, a SOCKS 4A-compliant server MUST NOT attempt to validate whether a 4A request was strictly necessary. Any request that matches the 0.0.0.x pattern MUST be processed according to the SOCKS 4A logic described in Section 4, even if the server suspects the client is capable of direct IPv4 addressing.

### A.2.2. Buffer Management and Premature Termination

Historical implementations have occasionally encountered "leaky" or malformed packets where the NULL terminators for the USERID or DOMAIN fields are missing or followed by extraneous data. A robust implementation SHOULD treat the first NULL octet encountered as the definitive end of the field.

Furthermore, if the stream terminates before the second NULL octet (the DOMAIN terminator) is received, the server MUST treat the request as failed and send a rejection reply (Result Code 91). This prevents the server from hanging in a "half-read" state, which could be exploited as a resource exhaustion vector (see Appendix B.4).

### A.2.3. Character Set Consistency

While this document recommends US-ASCII or Punycode (Section 3.1.2), historical deployments have occasionally seen DOMAIN strings containing raw UTF-8 or local codepage octets. Servers SHOULD treat the DOMAIN string as an opaque sequence of octets when passing it to the local resolver. If the local resolver returns an error due to illegal characters, the server MUST return a failure code to the client rather than attempting to "guess" the intended encoding, which can lead to security vulnerabilities through domain name spoofing.

## Appendix B. Security Analysis

The SOCKS 4A protocol is a lightweight shim designed to facilitate TCP proxying with remote name resolution. It operates primarily at the session layer and lacks the cryptographic primitives found in more modern protocols like TLS [RFC8446]. This appendix provides a detailed analysis of the security implications of the protocol, assuming a threat model where an attacker can observe, intercept, and modify traffic between the client, the SOCKS server, and the DNS

infrastructure.

### B.1. Security Deficiencies of the Base Protocol

As an extension of SOCKSv4, SOCKS 4A inherits significant structural vulnerabilities. The protocol provides no mechanisms for mutual authentication, integrity protection, or confidentiality. Consequently, it is inherently susceptible to active man-in-the-middle (MITM) attacks. An attacker positioned between the client and the SOCKS server can silently alter the DSTPORT or DOMAIN fields, effectively redirecting the application traffic to a malicious destination without either party's knowledge.

The USERID field, while intended for identity assertion, provides no cryptographic proof of origin. In the absence of a strong authentication framework as recommended in [RFC1918], this field must be treated as untrusted and unauthenticated information. Relying on USERID for access control decisions is a violation of the principle of least privilege and is highly discouraged.

### B.2. Remote Name Resolution and Information Leakage

One of the primary motivations for SOCKS 4A is the mitigation of "DNS leakage" on the client's local network. By delegating resolution to the SOCKS server, the client avoids issuing plaintext DNS queries that would otherwise expose the destination hostname to local observers [RFC7626]. However, this delegation does not eliminate the risk but rather relocates it to the SOCKS server's network environment.

Unless the SOCKS server employs encrypted DNS transports such as DNS over TLS [RFC7858] or DNS over HTTPS [RFC8484], the resolution process remains transparent to upstream passive monitors. Furthermore, if the client and SOCKS server communicate over an untrusted wide-area network (WAN) without a secure tunnel (e.g., [RFC4301] or [RFC5246]), the DOMAIN string itself is transmitted in the clear, negating any privacy benefits intended by the use of remote resolution.

### B.3. Server-Side Request Forgery Risks

SOCKS 4A servers act as confused deputies by performing network operations on behalf of potentially anonymous clients. This mechanism introduces a significant risk of Server-Side Request Forgery (SSRF). A malicious client may leverage the SOCKS server to probe or attack internal infrastructure that is otherwise shielded from the public internet.

To mitigate this, implementations MUST adhere to the guidance in [RFC2827] regarding network ingress filtering. Servers should be configured with strict egress Access Control Lists (ACLs) to prevent connections to loopback addresses (127.0.0.0/8), private address space [RFC1918], and link-local addresses [RFC3927]. Failure to implement these controls allows an attacker to use the SOCKS server as a scanning proxy to enumerate internal services or exploit vulnerabilities in non-hardened internal applications.

### B.4. Robustness and Resource Exhaustion

The variable-length nature of the USERID and DOMAIN fields introduces vectors for Denial of Service (DoS) attacks. Unlike protocols with explicit length-prefixing, SOCKS 4A relies on NULL terminators. An implementation that performs unbounded reads while searching for a NULL octet is vulnerable to memory exhaustion attacks.

In accordance with [RFC4732], implementations MUST enforce hard limits on the size of the input buffers used for these fields. For the DOMAIN field, a limit of 255 octets is recommended to align with the maximum length of a Fully Qualified Domain Name (FQDN) as specified in [RFC1035]. Furthermore, servers MUST implement per-session timeouts for the resolution phase to prevent "tarpitting" attacks, where a client initiates a large number of requests that target slow or non-responsive DNS authoritative servers, thereby exhausting the server’s thread pool or file descriptors.

### B.5. Protocol Rollback and Downgrade Attacks

While SOCKS 4A was designed to improve upon SOCKSv4, it remains a subset of the capabilities provided by SOCKSv5 [RFC1928]. SOCKSv5 offers robust authentication methods [RFC1929] and support for UDP. However, because SOCKS 4A does not participate in a formal version negotiation (it merely uses a different version octet), it is susceptible to downgrade attacks. An attacker may modify the version octet of a SOCKSv5 request to force the use of SOCKS 4A, thereby stripping away any authentication or encryption requirements mandated by the higher-version configuration.

### B.6. Interaction with Internationalized Domain Names

The use of the DOMAIN field requires careful handling of Internationalized Domain Names (IDNs). As noted in [RFC5890], the interpretation of non-ASCII characters can lead to ambiguity and "homograph" attacks, where a visually similar but different domain is resolved. For maximum security and interoperability, clients SHOULD convert IDNs to A-label format (Punycode) as defined in [RFC5891] before transmission. Servers SHOULD treat the DOMAIN string as an

opaque sequence of octets to be passed to the resolver, while ensuring that the resulting IP address undergoes the filtering described in Appendix B.3.

### B.7. Final Security Note

SOCKS 4A is an aged protocol and lacks modern security features. It should only be used in environments where the communication channel is otherwise secured by a lower-layer technology (such as IPsec) or where the risk of interception and spoofing is deemed acceptable. For all other use cases, the transition to SOCKSv5 [RFC1928] combined with TLS is strongly recommended to ensure the confidentiality and integrity of the session.

## Appendix C. Example Implementations

The following software projects provide full or partial implementations of the SOCKS4A protocol. The author consulted these implementations as practical references during the drafting of this document. It is explicitly stated that the author and this draft are entirely unaffiliated with these projects.

* Dante: A sophisticated SOCKS server implementation for Unix-based systems. It handles SOCKS4A requests within its broader SOCKS4 module, utilizing the specific "null-domain" IP address format to trigger remote DNS resolution.

* GOST (Go Simple Tunnel): A multi-protocol tunnel manager written in Go. It includes a native SOCKS4A handler that supports both the protocol’s command set and its specific address representation.

* 3proxy: A lightweight, multi-platform proxy server. It implements SOCKS4A as part of its compact SOCKS service, supporting the extension for environments where client-side DNS resolution is restricted.

## Appendix D. Relationship with SOCKS 4

See Appendix A of [I-D.vance-socks-v4] for the reason why SOCKS 4A was not an extension of SOCKS Version 4.

## Original Author

Ying-Da Lee
Principal Member Technical Staff
NEC Systems Laboratory, CSTC
ylee@syl.dl.nec.com

David Koblas
Netskope

We sincerely apologize that, due to the document's long history and the passage of time, many early contributors may not have been formally included in this list. We extend our deepest gratitude to all who have contributed to this work. If you believe your name should be added to the acknowledgments, please contact the draft maintainers.

## Contributors

George G. Michaelson
Asia-Pacific Network Information Centre
6 Cordelia St
South Brisbane QLD 4101
Australia
Email: ggm@algebras.org

## Author's Address

Daniel James Vance
Independent
Email: djvanc@outlook.com