Network Working Group
Internet-Draft
Intended status: Historic
Expires: 20 August 2026

D. J. Vance
Independent

16 February 2026

SOCKS Protocol Version 4A
draft-vance-socks-v4a-01

Abstract

This document specifies SOCKS 4A, an extension to the SOCKS Version 4 protocol. This extension 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 20 August 2026.

Copyright Notice

Table of Contents

1. Introduction

The original SOCKSv4 protocol 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 SOCKS 4A extension 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

[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>.

[SOCKS] Koblas, D., "SOCKS", 1992 Usenix Security Symposium , 1992.

[SOCKS4] Lee, Y.-D., "SOCKS: A protocol for TCP proxy across firewalls", n.d., <https://www.openssh.org/txt/socks4.protocol>.

[SOCKS4a] Lee, Y.-D., "SOCKS 4A: A Simple Extension to SOCKS 4 Protocol", n.d., <https://www.openssh.org/txt/socks4a.protocol>.

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>.

[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>.

[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>.

[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>.

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

[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 and Implementation Notes

The following behaviors were observed in historical deployments of SOCKS 4A to address specific network constraints and interoperability challenges.

A.1. Proxy Chaining and Relaying

In multi-tiered network environments, a SOCKS server (the "intermediate proxy") may itself be configured to use another SOCKS server (the "upstream proxy") for outbound connectivity. When an intermediate proxy receives a SOCKS 4A request:

* Recursive Resolution: The intermediate proxy may attempt to resolve the DOMAIN locally. If successful, it may then downgrade the request to a standard SOCKSv4 CONNECT/BIND using the resolved IPv4 address when communicating with the upstream proxy.

* Transparent Relaying: If the intermediate proxy lacks DNS access or is configured for "blind" relaying, it passes the SOCKS 4A request— including the 0.0.0.x DSTIP signaling and the DOMAIN field— intact to the upstream proxy. This delegates the resolution responsibility to the edge of the network.

This mechanism was frequently employed in "firewall-behind-firewall" scenarios where only the outermost gateway possessed external name resolution capabilities.

A.2. Client-Side Resolution "Leakage" and Server Robustness

While the SOCKS 4A extension was primarily designed for clients unable to perform local DNS lookups, many "SOCKSified" application libraries (such as those using LD_PRELOAD or global proxy settings) exhibited "leaky" behavior.

* Pre-resolution: A client might resolve a domain name locally but still initiate a SOCKS 4A request using that domain name rather than the resolved IP address.

* Server Interoperability: To ensure maximum compatibility with various client stacks, historical SOCKS 4A server implementations typically did not validate whether the client _needed_ to use 4A. A server would process any request matching the 0.0.0.x DSTIP pattern as a 4A request, regardless of the client’s network location or supposed capabilities.

This permissive approach was essential for maintaining a uniform interface across diverse application environments, though it occasionally resulted in redundant DNS queries if both the client and the server performed the same resolution.

Appendix B. Security Analysis

This section provides an analysis of the security implications introduced by the SOCKS 4A extension. As an extension to SOCKSv4, it inherits the fundamental insecurities of the base protocol while introducing new vectors related to remote name resolution.

B.1. DNS Privacy and information Leakage

SOCKS 4A functions as a countermeasure against DNS leakage at the client-side network layer. In the base SOCKSv4 protocol, the Requirement for the client to provide a literal IPv4 address necessitates a local DNS lookup. This transaction is typically unencrypted and occurs outside the proxy tunnel, exposing the destination hostname to local network observers and the DNS recursive resolver.

By delegating resolution to the SOCKS server, the client encapsulates the intent (the DOMAIN string) within the TCP session established to the SOCKS server. However, this merely shifts the point of leakage; the SOCKS server’ s own DNS queries may still be observable unless the server implements encrypted DNS transport (e.g., DNS over TLS).

B.2. Server-Side Request Forgery

The SOCKS 4A resolution mechanism enables a primitive form of Server-Side Request Forgery. Because the server performs resolution and subsequent connection on behalf of the client, a malicious client may use the SOCKS server to:

* Probe Internal Infrastructure: Access or scan hostnames and IP addresses that are non-routable or firewalled from the public internet but reachable from the SOCKS server’ s internal interface.

* Resolve Split-Horizon DNS: Enumerate internal DNS records that are only visible to the SOCKS server’s configured resolvers.

Implementations SHOULD employ strict egress filtering and Access Control Lists (ACLs) to prevent the SOCKS server from connecting to loopback addresses (127.0.0.0/8), private address space (RFC 1918), or link-local addresses.

B.3. Denial of Service and Resource Exhaustion

The variable-length nature of the SOCKS 4A request introduces two primary vectors for resource exhaustion:

1. Memory Exhaustion: A SOCKS 4A request involves two variable-length NULL-terminated strings (USERID and DOMAIN). An implementation that fails to enforce strict bounds on these fields during the "read-until-NULL" phase is vulnerable to heap exhaustion. Servers MUST enforce a maximum buffer limit (RECOMMENDED 255 octets for DOMAIN) and terminate connections that exceed this limit without a NULL terminator.

2. Resolver Tarpitting: DNS resolution is an asynchronous, I/O-bound operation. A client may initiate numerous concurrent 4A requests targeting non-responsive or slow DNS authoritative servers. This can exhaust the server’s thread pool or file descriptors. Servers MUST implement a per-request resolution timeout.

B.4. Lack of Cryptographic Integrity and Authentication

SOCKS 4A, like its predecessor, provides no facility for session encryption, message integrity, or robust authentication.

* Identity Spoofing: The USERID field is provided by the client without any cryptographic proof of identity. It is trivial to spoof and SHOULD NOT be relied upon for security-critical authorization.

* Active Interception: The entire handshake, including the DOMAIN string, is transmitted in plaintext. An attacker in the path between the client and the SOCKS server can perform a Man-in-the-Middle (MITM) attack, observing the destination domain or modifying the server’s reply to redirect the client.

Implementations requiring confidentiality or integrity MUST wrap the SOCKS 4A transaction in a secure transport layer, such as TLS or an SSH tunnel.

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.

Author's Address

Daniel James Vance
Independent
Email: djvanc@outlook.com