Workgroup: Network Working Group
Internet-Draft:
draft-schinazi-masque-obfuscation-04
Published: 12 March 2021
Intended Status: Experimental
Expires: 13 September 2021
Authors: D. Schinazi
Google LLC

MASQUE Obfuscation

Abstract

This document describes MASQUE Obfuscation. MASQUE Obfuscation is a mechanism that allows co-locating and obfuscating networking applications behind an HTTPS web server. The currently prevalent use-case is to allow running a proxy or VPN server that is indistinguishable from an HTTPS server to any unauthenticated observer. We do not expect major providers and CDNs to deploy this behind their main TLS certificate, as they are not willing to take the risk of getting blocked, as shown when domain fronting was blocked. An expected use would be for individuals to enable this behind their personal websites via easy to configure open-source software.

This document is a straw-man proposal. It does not contain enough details to implement the protocol, and is currently intended to spark discussions on the approach it is taking. Discussion of this work is encouraged to happen on the MASQUE IETF mailing list masque@ietf.org or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/masque-drafts.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes MASQUE Obfuscation. MASQUE Obfuscation is a mechanism that allows co-locating and obfuscating networking applications behind an HTTPS web server. The currently prevalent use-case is to allow running a proxy or VPN server that is indistinguishable from an HTTPS server to any unauthenticated

observer. We do not expect major providers and CDNs to deploy this behind their main TLS certificate, as they are not willing to take the risk of getting blocked, as shown when domain fronting was blocked. An expected use would be for individuals to enable this behind their personal websites via easy to configure open-source software.

This document is a straw-man proposal. It does not contain enough details to implement the protocol, and is currently intended to spark discussions on the approach it is taking. Discussion of this work is encouraged to happen on the MASQUE IETF mailing list masque@ietf.org or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/masque-drafts.

MASQUE Obfuscation is built upon the MASQUE protocol [MASQUE]. MASQUE Obfuscation leverages the efficient head-of-line blocking prevention features of the QUIC transport protocol [QUIC] when MASQUE Obfuscation is used in an HTTP/3 [HTTP3] server. MASQUE Obfuscation can also run in an HTTP/2 server [HTTP2] but at a performance cost.

1.1. Conventions and Definitions

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.

2. Usage Scenarios

There are currently multiple usage scenarios that can benefit from MASQUE Obfuscation.

2.1. Protection from Network Providers

Some users may wish to obfuscate the destination of their network traffic from their network provider. This prevents network providers from using data harvested from this network traffic in ways the user did not intend.

2.2. Protection from Web Servers

There are many clients who would rather not establish a direct connection to web servers, for example to avoid location tracking. The clients can do that by running their traffic through a MASQUE Obfuscation server. The web server will only see the IP address of the MASQUE Obfuscation server, not that of the client.

2.3. Onion Routing

Routing traffic through a MASQUE Obfuscation server only provides partial protection against tracking, because the MASQUE Obfuscation server knows the address of the client. Onion routing as it exists today mitigates this issue for TCP/TLS. A MASQUE Obfuscation server could allow onion routing over QUIC.

In this scenario, the client establishes a connection to the MASQUE Obfuscation server, then through that to another MASQUE Obfuscation server, etc. This creates a tree of MASQUE servers rooted at the client. QUIC connections are mapped to a specific branch of the tree. The first MASQUE Obfuscation server knows the actual address of the client, but the other MASQUE Obfuscation servers only know the address of the previous server. To assure reasonable privacy, the path should include at least 3 MASQUE Obfuscation servers.

3. Requirements

This section describes the goals and requirements chosen for MASQUE Obfuscation.

3.1. Invisibility of Usage

An authenticated client using MASQUE Obfuscation appears to observers as a regular HTTPS client. Observers only see that HTTP/3 or HTTP/2 is being used over an encrypted channel. No part of the exchanges between client and server may stick out. Note that traffic analysis is discussed in Section 8.1.

3.2. Invisibility of the Server

To anyone without private keys, the server is indistinguishable from a regular web server. It is impossible to send an unauthenticated probe that the server would reply to differently than if it were a normal web server.

3.3. Fallback to HTTP/2 over TLS over TCP

When QUIC is blocked, MASQUE Obfuscation can run over TCP and still satisfy previous requirements. Note that in this scenario performance may be negatively impacted.

4. Overview of the Mechanism

The server runs an HTTPS server on port 443, and has a valid TLS certificate for its domain. The client has a public/private key pair, and the server maintains a list of authorized MASQUE Obfuscation clients, and their public key. (Alternatively, clients can also be authenticated using a shared secret.) The client starts

by establishing a regular HTTPS connection to the server (HTTP/3 over QUIC or HTTP/2 over TLS 1.3 [TLS13] over TCP), and validates the server's TLS certificate as it normally would for HTTPS. If validation fails, the connection is aborted. At this point the client can send regular unauthenticated HTTP requests to the server. When it wishes to start MASQUE Obfuscation, the client uses HTTP Transport Authentication [TRANSPORT-AUTH] to prove its possession of its associated key. The client sends the Transport-Authentication header alongside its MASQUE Negotiation request.

When the server receives the MASQUE Negotiation request, it authenticates the client and if that fails responds with code "404 Not Found", making sure its response is the same as what it would return for any unexpected POST request. If authentication succeeds, the server sends its list of supported MASQUE applications and the client can start using them.

5. Connection Resumption

Clients MUST NOT attempt to "resume" MASQUE Obfuscation state similarly to how TLS sessions can be resumed. Every new QUIC or TLS connection requires fully authenticating the client and server. QUIC 0-RTT and TLS early data MUST NOT be used with MASQUE Obfuscation as they are not forward secure.

6. Path MTU Discovery

In the main deployment of this mechanism, QUIC will be used between client and server, and that will most likely be the smallest MTU link in the path due to QUIC header and authentication tag overhead. The client is responsible for not sending overly large UDP packets and notifying the server of the low MTU. Therefore PMTUD is currently seen as out of scope of this document.

7. Operation over HTTP/2

We will need to define the details of how to run MASQUE over HTTP/2. When running over HTTP/2, MASQUE uses the Extended CONNECT method to negotiate the use of datagrams over an HTTP/2 stream [HTTP2-TRANSPORT].

MASQUE Obfuscation implementations SHOULD discover that HTTP/3 is available (as opposed to only HTTP/2) using the same mechanism as regular HTTP traffic. This current standardized mechanism for this is HTTP Alternative Services [ALT-SVC], but future mechanisms such as [HTTPSSVC] can be used if they become widespread.

MASQUE Obfuscation implementations using HTTP/3 MUST support the fallback to HTTP/2 to avoid incentivizing censors to block HTTP/3 or QUIC.

When the client wishes to use the "UDP Proxying" MASQUE application over HTTP/2, the client opens a new stream with a CONNECT request to the "masque-udp-proxy" protocol and then sends datagrams encapsulated inside the stream with a two-byte length prefix in network byte order. The target IP and port are sent as part of the URL query. Resetting that stream instructs the server to release any associated resources.

When the client wishes to use the "IP Proxying" MASQUE application over HTTP/2, the client opens a new stream with a CONNECT request to the "masque-ip-proxy" protocol and then sends IP datagrams with a two byte length prefix. The server can inspect the IP datagram to look for the destination address in the IP header.

8. Security Considerations

Here be dragons. TODO: slay the dragons.

8.1. Traffic Analysis

While MASQUE Obfuscation ensures that proxied traffic appears similar to regular HTTP traffic, it doesn't inherently defeat traffic analysis. However, the fact that MASQUE leverages QUIC allows it to segment STREAM frames over multiple packets and add PADDING frames to change the observable characteristics of its encrypted traffic. The exact details of how to change traffic patterns to defeat traffic analysis is considered an open research question and is out of scope for this document.

When multiple MASQUE Obfuscation servers are available, a client can leverage QUIC connection migration to seamlessly transition its end-to-end QUIC connections by treating separate MASQUE Obfuscation servers as different paths. This could afford an additional level of obfuscation in hopes of rendering traffic analysis less effective.

8.2. Untrusted Servers

As with any proxy or VPN technology, MASQUE Obfuscation hides some of the client's private information (such as who they are communicating with) from their network provider by transferring that information to the MASQUE server. It is paramount that clients only use MASQUE Obfuscation servers that they trust, as a malicious actor could easily setup a MASQUE Obfuscation server and advertise it as a privacy solution in hopes of attracting users to send it their traffic.

9. IANA Considerations

We will need to register the "masque-udp-proxy" and "masque-ip-proxy" extended HTTP CONNECT protocols.

10. References

10.1. Normative References

[ALT-SVC] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://www.rfc-editor.org/rfc/rfc7838>.

[HTTP2] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/rfc/rfc7540>.

[HTTP2-TRANSPORT] Kinnear, E. and T. Pauly, "Using HTTP/2 as a Transport for Arbitrary Bytestreams", Work in Progress, Internet-Draft, draft-kinnear-httpbis-http2-transport-02, 4 November 2019, <https://tools.ietf.org/html/draft-kinnear-httpbis-http2-transport-02>.

[HTTP3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <https://tools.ietf.org/html/draft-ietf-quic-http-34>.

[MASQUE] Schinazi, D., "The MASQUE Protocol", Work in Progress, Internet-Draft, draft-schinazi-masque-protocol-02, 9 September 2020, <https://tools.ietf.org/html/draft-schinazi-masque-protocol-02>.

[QUIC] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-34, 14 January 2021, <https://tools.ietf.org/html/draft-ietf-quic-transport-34>.

[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>.

[TLS13] 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>.

[TRANSPORT-AUTH] Schinazi, D., "HTTP Transport Authentication", Work in Progress, Internet-Draft, draft-schinazi-httpbis-

transport-auth-05, 12 March 2021, <https://tools.ietf.org/html/draft-schinazi-httpbis-transport-auth-05>.

10.2. Informative References

[HTTPSSVC] Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPSSVC)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-httpssvc-03, 11 June 2020, <https://tools.ietf.org/html/draft-ietf-dnsop-svcb-httpssvc-03>.

[I-D.ietf-httpbis-http2-secondary-certs]
Bishop, M., Sullivan, N., and M. Thomson, "Secondary Certificate Authentication in HTTP/2", Work in Progress, Internet-Draft, draft-ietf-httpbis-http2-secondary-certs-06, 14 May 2020, <https://tools.ietf.org/html/draft-ietf-httpbis-http2-secondary-certs-06>.

[I-D.pardue-httpbis-http-network-tunnelling]
Pardue, L., "HTTP-initiated Network Tunnelling (HiNT)", Work in Progress, Internet-Draft, draft-pardue-httpbis-http-network-tunnelling-01, 18 October 2018, <https://tools.ietf.org/html/draft-pardue-httpbis-http-network-tunnelling-01>.

[I-D.schwartz-httpbis-helium]
Schwartz, B., "Hybrid Encapsulation Layer for IP and UDP Messages (HELIUM)", Work in Progress, Internet-Draft, draft-schwartz-httpbis-helium-00, 25 June 2018, <https://tools.ietf.org/html/draft-schwartz-httpbis-helium-00>.

[I-D.sullivan-tls-post-handshake-auth]
Sullivan, N., Thomson, M., and M. Bishop, "Post-Handshake Authentication in TLS", Work in Progress, Internet-Draft, draft-sullivan-tls-post-handshake-auth-00, 5 August 2016, <https://tools.ietf.org/html/draft-sullivan-tls-post-handshake-auth-00>.

[RFC8441] McManus, P., "Bootstrapping WebSockets with HTTP/2", RFC 8441, DOI 10.17487/RFC8441, September 2018, <https://www.rfc-editor.org/rfc/rfc8441>.

[RFC8471] Popov, A., Ed., Nystroem, M., Balfanz, D., and J. Hodges, "The Token Binding Protocol Version 1.0", RFC 8471, DOI 10.17487/RFC8471, October 2018, <https://www.rfc-editor.org/rfc/rfc8471>.

Acknowledgments

This proposal was inspired directly or indirectly by prior work from many people. In particular, this work is related to [I-D.schwartz-httpbis-helium] and [I-D.pardue-httpbis-http-network-tunnelling]. The mechanism used to run the MASQUE protocol over HTTP/2 streams was inspired by [RFC8441]. Brendan Moran is to thank for the idea of leveraging connection migration across MASQUE servers. The author would also like to thank Nick Harper, Christian Huitema, Marcus Ihlar, Eric Kinnear, Mirja Kuehlewind, Lucas Pardue, Tommy Pauly, Zaheduzzaman Sarker, Ben Schwartz, and Christopher A. Wood for their input.

The author would like to express immense gratitude to Christophe A., an inspiration and true leader of VPNs.

Design Justifications

Using an exported key as a nonce allows us to prevent replay attacks (since it depends on randomness from both endpoints of the TLS connection) without requiring the server to send an explicit nonce before it has authenticated the client. Adding an explicit nonce mechanism would expose the server as it would need to send these nonces to clients that have not been authenticated yet.

The rationale for a separate MASQUE protocol stream is to allow server-initiated messages. If we were to use HTTP semantics, we would only be able to support the client-initiated request-response model. We could have used WebSocket for this purpose but that would have added wire overhead and dependencies without providing useful features.

There are many other ways to authenticate HTTP, however the authentication used here needs to work in a single client-initiated message to meet the requirement of not exposing the server.

The current proposal would also work with TLS 1.2, but in that case TLS false start and renegotiation must be disabled, and the extended master secret and renegotiation indication TLS extensions must be enabled.

If the server or client want to hide that HTTP/2 is used, the client can set its ALPN to an older version of HTTP and then use the Upgrade header to upgrade to HTTP/2 inside the TLS encryption.

The client authentication used here is similar to how Token Binding [RFC8471] operates, but it has very different goals. MASQUE does not use token binding directly because using token binding requires sending the token_binding TLS extension in the TLS ClientHello, and that would stick out compared to a regular TLS connection.

TLS post-handshake authentication [I-D.sullivan-tls-post-handshake-auth] is not used by this proposal because that requires sending the "post_handshake_auth" extension in the TLS ClientHello, and that would stick out from a regular HTTPS connection.

Client authentication could have benefited from Secondary Certificate Authentication in HTTP/2 [I-D.ietf-httpbis-http2-secondary-certs], however that has two downsides: it requires the server advertising that it supports it in its SETTINGS, and it cannot be sent unprompted by the client, so the server would have to request authentication. Both of these would make the server stick out from regular HTTP/2 servers.

MASQUE proposes a new client authentication method (as opposed to reusing something like HTTP basic authentication) because HTTP authentication methods are conceptually per-request (they need to be repeated on each request) whereas the new method is bound to the underlying connection (be it QUIC or TLS). In particular, this allows sending QUIC DATAGRAM frames without authenticating every frame individually. Additionally, HMAC and asymmetric keying are preferred to sending a password for client authentication since they have a tighter security bound. Going into the design rationale, HMACs (and signatures) need some data to sign, and to avoid replay attacks that should be a fresh nonce provided by the remote peer. Having the server provide an explicit nonce would leak the existence of the server so we use TLS keying material exporters as they provide us with a nonce that contains entropy from the server without requiring explicit communication.

Author's Address

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043,
United States of America

Email: dschinazi.ietf@gmail.com