Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 9, 2020

D. Schinazi
Google LLC

July 08, 2019

The MASQUE Protocol
draft-schinazi-masque-01

Abstract

This document describes MASQUE (Multiplexed Application Substrate over QUIC Encryption). MASQUE 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 [1] or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/masque-drafts [2].

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 January 9, 2020.

Table of Contents

1. Introduction

This document describes MASQUE (Multiplexed Application Substrate over QUIC Encryption). MASQUE 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 [3] or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/masque-drafts [4].

MASQUE leverages the efficient head-of-line blocking prevention features of the QUIC transport protocol [I-D.ietf-quic-transport] when MASQUE is used in an HTTP/3 [I-D.ietf-quic-http] server. MASQUE can also run in an HTTP/2 server [RFC7540] 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.

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 server. The web server will only see the IP address of the MASQUE server, not that of the client.

2.3. Making a Home Server Available

It is often difficult to connect to a home server. The IP address might change over time. Firewalls in the home router or in the network may block incoming connections. Using a MASQUE server as a rendez-vous point helps resolve these issues.

2.4. Onion Routing

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

In this scenario, the client establishes a connection to the MASQUE server, then through that to another MASQUE 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 server knows the actual address of the client, but the other MASQUE servers only know the address of the previous server. To assure reasonable privacy, the path should include at least 3 MASQUE servers.

3. Requirements

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

3.1. Invisibility of Usage

An authenticated client using MASQUE features 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 7.1.

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 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 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 [RFC8446] 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, the client uses HTTP Transport Authentication (draft-schinazi-httpbis-transport-auth) to prove its possession of its associated key. The client sends the Transport-Authentication header alongside an HTTP CONNECT request for "/.well-known/masque/initial" with the :protocol pseudo-header field set to "masque".

When the server receives this CONNECT request, it authenticates the client and if that fails responds with code "405 Method Not Allowed", making sure its response is the same as what it would return for any unexpected CONNECT request. If authentication succeeds, the server responds with code "101 Switching Protocols", and from then on this HTTP stream is now dedicated to the MASQUE protocol. That protocol provides a reliable bidirectional message exchange mechanism, which is used by the client and server to negotiate what protocol options are supported and enabled by policy, and client VPN configuration such as IP addresses. When using QUIC, this protocol also allows endpoints to negotiate the use of QUIC extensions, such as support for the DATAGRAM extension [I-D.pauly-quic-datagram].

Clients MUST NOT attempt to "resume" MASQUE 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 as they are not forward secure.

5. Mechanisms the Server Can Advertise to Authenticated Clients

Once a server has authenticated the client's MASQUE CONNECT request, it advertises services that the client may use.

5.1. HTTP Proxy

The client can make proxied HTTP requests through the server to other servers. In practice this will mean using the CONNECT method to establish a stream over which to run TLS to a different remote destination. The proxy applies back-pressure to streams in both directions.

5.2. DNS over HTTPS

The client can send DNS queries using DNS over HTTPS (DoH) [RFC8484] to the MASQUE server.

5.3. UDP Proxying

In order to support WebRTC or QUIC to further servers, clients need a way to relay UDP onwards to a remote server. In practice for most widely deployed protocols other than DNS, this involves many datagrams over the same ports. Therefore this mechanism implements that efficiently: clients can use the MASQUE protocol stream to request an UDP association to an IP address and UDP port pair. In QUIC, the server would reply with a DATAGRAM_ID that the client can then use to have UDP datagrams sent to this remote server. Datagrams are then simply transferred between the DATAGRAMs with this ID and the outer server. There will also be a message on the MASQUE protocol stream to request shutdown of a UDP association to save resources when it is no longer needed. When running over TCP, 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 associates resources.

5.4. QUIC Proxying

By leveraging QUIC client connection IDs, a MASQUE server can act as a QUIC proxy while only using one UDP port. The server informs the client of a scheme for client connection IDs (for example, random of a minimum length or vended by the MASQUE server) and then the server can forward those packets to further web servers.

This mechanism can elide the connection IDs on the link between the client and MASQUE server by negotiating a mapping between DATAGRAM_IDs and the tuple (client connection ID, server connection ID, server IP address, server port).

Compared to UDP proxying, this mode has the advantage of only requiring one UDP port to be open on the MASQUE server, and can lower the overhead on the link between client and MASQUE server by compressing connection IDs.

5.5. IP Proxying

For the rare cases where the previous mechanisms are not sufficient, proxying can be performed at the IP layer. This would use a different DATAGRAM_ID and IP datagrams would be encoded inside it without framing. Over TCP, a dedicated stream with two byte length prefix would be used. The server can inspect the IP datagram to look for the destination address in the IP header.

5.6. 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.

5.7. Service Registration

MASQUE can be used to make a home server accessible on the wide area. The home server authenticates to the MASQUE server and registers a domain name it wishes to serve. The MASQUE server can then forward any traffic it receives for that domain name (by inspecting the TLS Server Name Indication (SNI) extension) to the home server. This received traffic is not authenticated and it allows non-modified clients to communicate with the home server without knowing it is not colocated with the MASQUE server.

To help obfuscate the home server, deployments can use Encrypted Server Name Indication (ESNI) [I-D.ietf-tls-esni]. That will require the MASQUE server sending the cleartext SNI to the home server.

6. Operation over HTTP/2

MASQUE implementations using HTTP/3 MUST support the fallback to HTTP/2 to avoid incentivizing censors to block HTTP/3 or QUIC. When running over HTTP/2, MASQUE uses the Extended CONNECT method to

negotiate the use of datagrams over an HTTP/2 stream [I-D.kinnear-httpbis-http2-transport].

MASQUE 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 [RFC7838], but future mechanisms such as [I-D.schwartz-httpbis-dns-alt-svc] can be used if they become widespread.

7. Security Considerations

Here be dragons. TODO: slay the dragons.

7.1. Traffic Analysis

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

7.2. Untrusted Servers

As with any proxy or VPN technology, MASQUE 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 servers that they trust, as a malicious actor could easily setup a MASQUE server and advertise it as a privacy solution in hopes of attracting users to send it their traffic.

8. IANA Considerations

We will need to register:

- the "/.well-known/masque/" URI (expert review)

https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml [5]

- The "masque" and "masque-udp-proxy" extended HTTP CONNECT protocols

We will also need to define the MASQUE control protocol and that will be likely to define new registries of its own.

9. References

9.1. Normative References

[I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-20 (work in progress), April 2019.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-20 (work in progress), April 2019.

[I-D.kinnear-httpbis-http2-transport]
Kinnear, E. and T. Pauly, "Using HTTP/2 as a Transport for Arbitrary Bytestreams", draft-kinnear-httpbis-http2-transport-01 (work in progress), March 2019.

[I-D.pauly-quic-datagram]
Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", draft-pauly-quic-datagram-03 (work in progress), July 2019.

[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/info/rfc2119>.

[RFC7540] 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/info/rfc7540>.

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

[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/info/rfc8174>.

[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/info/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/info/rfc8484>.

9.2. Informative References

[I-D.ietf-httpbis-http2-secondary-certs]
Bishop, M., Sullivan, N., and M. Thomson, "Secondary Certificate Authentication in HTTP/2", draft-ietf-httpbis-http2-secondary-certs-04 (work in progress), April 2019.

[I-D.ietf-tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-03 (work in progress), March 2019.

[I-D.pardue-httpbis-http-network-tunnelling]
Pardue, L., "HTTP-initiated Network Tunnelling (HiNT)", draft-pardue-httpbis-http-network-tunnelling-01 (work in progress), October 2018.

[I-D.schwartz-httpbis-dns-alt-svc]
Schwartz, B. and M. Bishop, "Finding HTTP Alternative Services via the Domain Name Service", draft-schwartz-httpbis-dns-alt-svc-02 (work in progress), April 2018.

[I-D.schwartz-httpbis-helium]
Schwartz, B., "Hybrid Encapsulation Layer for IP and UDP Messages (HELIUM)", draft-schwartz-httpbis-helium-00 (work in progress), June 2018.

[I-D.sullivan-tls-post-handshake-auth]
Sullivan, N., Thomson, M., and M. Bishop, "Post-Handshake Authentication in TLS", draft-sullivan-tls-post-handshake-auth-00 (work in progress), August 2016.

[RFC8441] McManus, P., "Bootstrapping WebSockets with HTTP/2", RFC 8441, DOI 10.17487/RFC8441, September 2018, <https://www.rfc-editor.org/info/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/info/rfc8471>.

9.3. URIs

[1] mailto:masque@ietf.org

[2] https://github.com/DavidSchinazi/masque-drafts

[3] mailto:masque@ietf.org

[4] https://github.com/DavidSchinazi/masque-drafts

[5] https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml

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 like to thank 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