MASQUE

Multiplexed Application Substrate over QUIC Encryption

David Schinazi – Google
dschinazi@google.com

Motivation

Censorship supported by actors with considerable means

VPN block, SNI filtering, ESNI blocking, known DoH server blocking

Any traffic that “sticks out” is blocked, or logged

MASQUE aims to build a mechanism to circumvent these tools

Ideally solution can be deployed by anyone on their personal website

Threat Model

Client and server run trusted software with credentials learned out of band

Network considered harmful, can mount active attacks on both client and server

Requirement: Impossible to detect MASQUE usage

Looks like web (HTTP/3) traffic

Does not stick out

- Same SNI and ALPN
- No uncommon TLS Client Hello extensions or QUIC transport parameters

Requirement: Impossible to detect MASQUE server

Attacker can probe web server

Cannot be able to tell if a web server supports MASQUE

MASQUE server cannot give client any MASQUE information before client authenticates

Requirement: Fallback to HTTP/2 + TLS + TCP

Some networks block QUIC

MASQUE can support all of its features over HTTP/2 over TLS over TCP

Performance can be impacted

Authentication Mechanism

Client initiates HTTP/3 connection to server, validates server certificate

Server is authenticated using Web PKI

Client uses TLS keying material exporter to generate a shared secret, and uses it as a nonce

Client sends HTTP CONNECT to :protocol “masque” with username, OID of signature/hash algorithm and signature/HMAC of the nonce

```
Masque-Authentication: PublicKey u="am9obi5kb2U=";a=1.3.101.112;
s="SW5zZXJ0IHNpZ25hdHVyZSBvZiBub25jZSBoZXJlIHdoaWNoIHRha2VzIDUxMiBiaXRzIGZvciBFZDI1NTE5IQ=="
```

MASQUE Features

If client authentication succeeds, the CONNECT request establishes a bidirectional stream for MASQUE control

Endpoints negotiate supported features and configuration

Feature: HTTP CONNECT Proxy

Client can send HTTP CONNECT requests to the server which will proxy at the transport layer. Client then performs TLS end-to-end to the final server.

Can also be used for any TCP protocol (e.g. SSH)

Feature: DNS over HTTPS

MASQUE server can act as DoH server

Feature: UDP proxying

Client can request that the server proxy UDP datagrams to a given IP and port

Uses QUIC DATAGRAMs with a DATAGRAM IDENTIFIER per (IP, port)

(When falling back to HTTP/2, uses a dedicated HTTP/2 stream)

Allows proxying QUIC, WebRTC, DTLS, etc. over MASQUE

Feature: IP proxying / VPN

Client sends full IP datagrams that the server can forward

Server can optionally NAT datagrams to obfuscate clients to end servers

Traffic Analysis

Currently considered out of scope

QUIC allows grouping multiple STREAM, DATAGRAM and PADDING frames in one encrypted UDP packet

Allows endpoints to obfuscate traffic patterns

Are there good obfuscation techniques that could be relevant here?

Path MTU Discovery

When proxying IP or UDP, MASQUE adds overhead and reduces MTU

Client needs to ensure it does send packets exceeding that MTU and communicate it to the end server

Transport Considerations

Running QUIC over MASQUE implies stacking two congestion controllers

QUIC design principle was to encrypt congestion control information

Are there good solutions here?

Next Steps

Is this work interesting?

Does anyone want to collaborate on code?

Does anyone want to collaborate on standardization?

Should we find an existing or new home for this work?