Security Considerations for Optimistic Use of HTTP Upgrade

Ben Schwartz

HTTPBIS @ IETF 118

Backstory from IETF 117

- “Upgrade” is mechanism to switch from HTTP/1.1 (only) to something else, on the same connection.
- “connect-tcp” makes use of HTTP/1.1 Upgrade.
- “connect-tcp” had to make some design decisions to avoid request smuggling attacks when the Upgrade request is rejected.
- It turns out that “connect-udp” is hypothetically vulnerable to similar attacks
  - only if certain special Capsule IDs are registered at some point in the future.
- There is no clear guidance on how Upgrade protocols should avoid these attacks.
- Working group seemed interested in a draft on this topic.

Highlights from -00

“In some cases, the client might expect that the upgrade will succeed. If this expectation is correct, the client might be able to reduce delay by immediately sending the first bytes of the upgraded protocol "optimistically", without waiting for the server's response. This document explores the security implications of this "optimistic" behavior.”

“When there are only two distinct parties involved in an HTTP/1.1 connection (i.e., the client and the server), HTTP Upgrade introduces no new security issues … However, HTTP connections often involve more than two parties, if the requests or responses include third-party data. … If the third-party data source is untrusted, we call the data it provides "attacker-controlled". The combination of attacker-controlled data and optimistic HTTP Upgrade results in two significant security issues”: [request smuggling and parser exploits].

Impact on Existing Upgrade Tokens

- UPDATE to RFC 9298 (“`connect-udp`”)
  - *“When using HTTP/2 or later, a client MAY optimistically start sending UDP packets in HTTP Datagrams before receiving the response to its UDP proxying request.”*
  - If the request fails, *“the client MUST treat this proxying attempt as failed ~~and abort the connection~~. If the "Upgrade" response header field is absent, the client MAY reuse the connection for further HTTP/1.1 requests; otherwise it MUST abort the underlying connection.”*
- All other registered tokens seem to be OK
  - “`HTTP/0.9`”, “`HTTP/1.0`”, “`HTTP/1.1`” don’t alter the handling of attacker-controlled data.
  - “`HTTP/2.0`” has the “`PRI * …`” Client Connection Preface for exactly this reason.
  - “`TLS`” (RFC 2817) pretty clearly requires the client to wait for Upgrade confirmation.
  - “`WebSocket`”: RFC 6455 says *“Once the client's opening handshake has been sent, the client MUST wait for a response from the server before sending any further data.”*
  - “`connect-ip`”: the draft has been recently been adjusted to match “`connect-udp`” above.

Guidance for Future Upgrade Tokens

“There are now several good examples of designs that prevent the security concerns discussed in this document and may be applicable in future specifications:

- *Forbid optimistic use of HTTP Upgrade (WebSocket).*
- *Embed a fixed preamble that terminates HTTP/1.1 processing (HTTP/2).*
- *Apply high-entropy masking of client-to-server data (WebSocket).*

*Future specifications for Upgrade Tokens **MUST** account for the security issues discussed here and provide clear guidance on how clients can avoid them.”*

Seeking adoption in HTTPBIS

Appendix

Example: Attack on connect-tcp (1/2)

1. Find a user who is using a “connect-tcp” proxy system-wide
   a. over HTTP/1.1
   b. with client certificate authentication
2. Get this user to run an attacker-controlled application.
3. The malicious application requests a TCP socket to a nonexistent target, with an optimistic TCP payload.
   a. draft-ietf-httpbis-connect-tcp forbids optimistic payloads in HTTP/1.1 to prevent this attack.
4. The proxy fails the Upgrade because the target is nonexistent, and interprets the optimistic TCP payload as a request from the authorized client.

Example: Attack on connect-tcp (1/2)

```
GET /?target_host=nonexistent.example
        &tcp_port=443 HTTP/1.1
Host: proxy.example
Connection: Upgrade
Upgrade: connect-tcp

POST /authorized_contacts HTTP/1.1
Host: proxy.example

attacker@mail.example
```

Generated by the proxy client on behalf of the attacker. The proxy client expects it to succeed, but the attacker knows it will fail.

The proxy client thinks this is optimistic TCP payload for nonexistent.example:443, but the proxy origin interprets it as a request for itself.

Example: Attack on connect-udp

1. Register Capsule Type 4175 to mean “UDP packet with DSCP=17”
   a. “Specification Required”. (DSCP=17 is an arbitrary specification here.)
2. Wait for proxies and proxy clients to implement support for this new type.
3. Find a vulnerable user and get them to install a malicious application.
4. Send a UDP packet of length 4948 with DSCP=17 to an invalid IP address.
   a. The system-wide proxy client captures this outbound packet and converts it into a Capsule.
   b. QUIC varint 4175 is `01 010000 01001111` == ASCII “PO”
   c. QUIC varint 4948 is `01 010011 01010100` == ASCII “ST”
   d. Remaining payload is attacker controlled, allowing them to spell “POST /…”
   e. Invalid IP address ensures that the Upgrade will fail, causing any optimistic payload to be interpreted as further HTTP/1.1 content instead.