# SNIP

draft-thomson-tls-snip-01

IETF 111, TLS Working Group

# Secure Negotiation of INCOMPATIBLE Protocols

Problem:

Two incompatible protocols achieve the same basic goal

How do you know that the “best” one was negotiated?

Incompatible protocols:

X and Y are incompatible if you can’t attempt X and get Y

e.g., HTTP/3 and HTTP/2 are incompatible

# Protocol Overview

Client says that it supports the extension

Server lists incompatible protocols

If the client prefers a listed protocol, it might be an attack

Up to the client how they react

# TLS Bits

```
ProtocolName IncompatibleProtocol; /* ProtocolName from ALPN [RFC7301] */
struct {
  select (Handshake.msg_type) {
    case client_hello:
      Empty;
    case encrypted_extensions:
      IncompatibleProtocol incompatible_protocols<3..2^16-1>;
  };
} IncompatibleProtocols;
```

# Changes in -02

Remove protocol authentication scope concept

- Old design was too complicated, likely not deployable
- Client only considers servers that share IP and port number
- …not IP protocol (UDP port == TCP port for this purpose)
- Limited applicability, but more likely to actually be applicable

# Key Points

Intended for use with any discovery method (SVCB, Alt-Svc, A records…)

- No reliance on authentication, even if discovery is authenticated
- **TLS handshake is authoritative**

Server chooses between compatible protocols (ALPN),
Client chooses between incompatible protocols (SNIP)

- Client reaction to errors will depend on client policy

Minimal constraints on server deployments

- Only services that share IP and port need coordination


ADOPT