SAMP over HTTPS

Mark Taylor (Bristol)

ASTERICS Tech Forum #2
Edinburgh

8 March 2016

`$Id: tlsamp.tex,v 1.11 2016/03/04 14:31:59 mbt Exp $`

TL;DR:

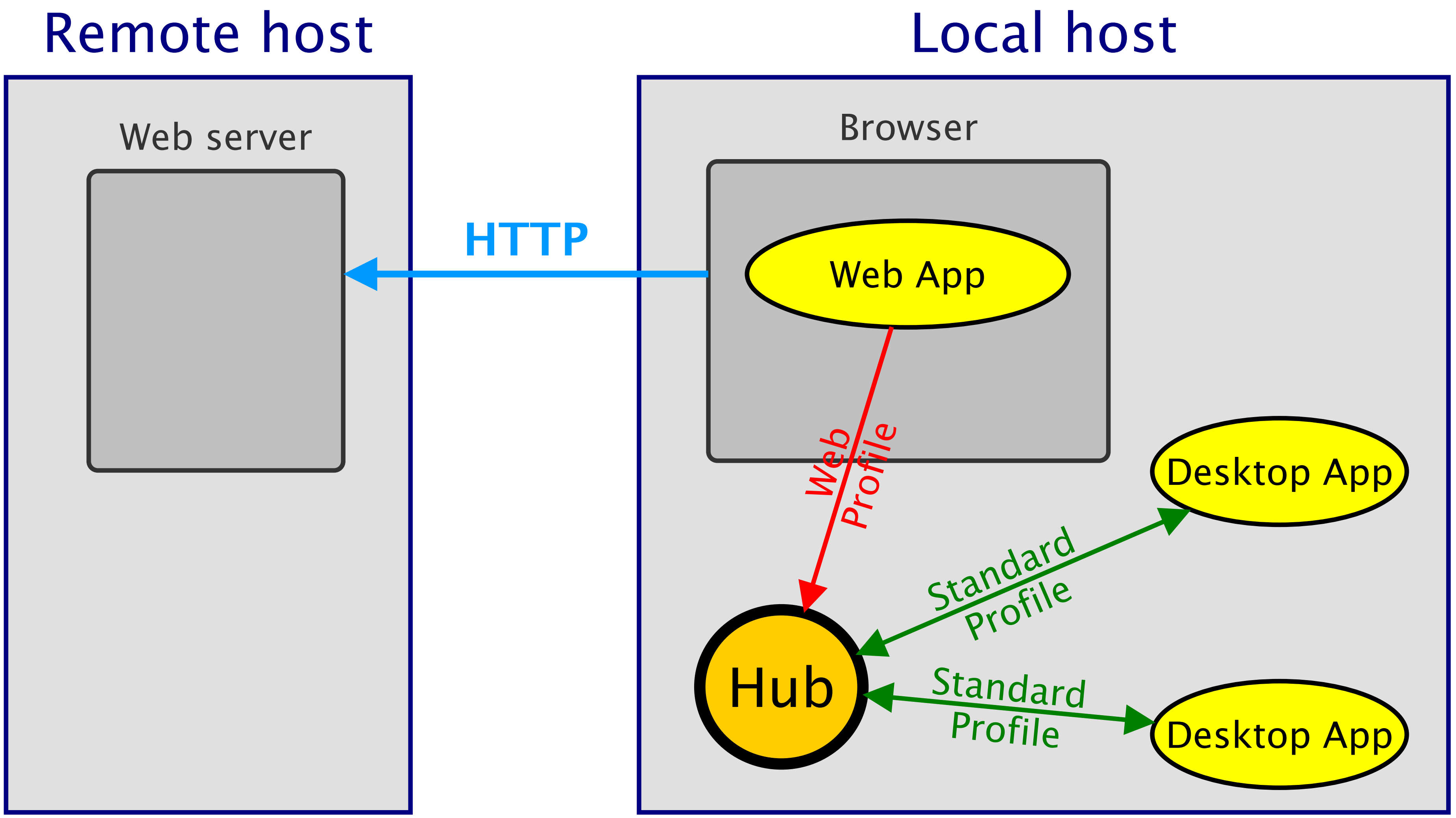
Some people want SAMP functionality from HTTPS-hosted web pages

It's hard ...

... but possible

Outline

- (Web) SAMP refresher
- HTTPS + SAMP: the problem *(abbreviated)*
- Proposed workaround
- Progress report



Simple Application Messaging Protocol

SAMP Refresher

Simple Applications Messaging Protocol

- Allows clients to communicate with each other via a Hub
- Clients can be desktop applications or web applications:

  Desktop application: runs directly on OS with user privileges, can access filesystem
  Web application: runs in a browser (typically HTML+JavaScript), sandboxed

- To make it work, each client has to set up communications with the Hub (not each other)
- The set of rules a client uses for Hub discovery and communication is called the Profile
- Desktop applications use the Standard Profile, web applications use the Web Profile
- Both use XML-RPC over HTTP, but with some differences:

  Standard profile:
    - hub URL is read from lockfile `~/.samp`
    - HTTP communication uses normal user socket

  Web Profile:
    - hub is found at the well-known URL `http://localhost:21012/`
    - HTTP communication uses `XMLHttpRequest` with CORS

  (There are some other differences, but not relevant here)

→ SAMP from an HTTP page works (pretty) well

HTTPS

- HTTPS is HTTP Over TLS
  - RFC 2818, which defines HTTPS, says:

    > **2. HTTP Over TLS**
    > Conceptually, HTTP/TLS is very simple. Simply use HTTP over TLS precisely as you would use HTTP over TCP.

  - TLS = Transport Layer Security ≈ SSL = Secure Sockets Layer
  - Host authentication is mandatory in HTTPS; host requires a trusted certificate
- Some web pages are served over HTTPS
  - Encrypts communications
  - Assures the client that it's talking to the web server it thinks it is
  - Required to support secure authentication
    (e.g. serving restricted data to authenticated users)
  - US Government, ESA?, others? plan to move all services to HTTPS in the near future

HTTPS web page + HTTP SAMP

You might want an HTTPS web application to use SAMP:

- Browser retrieves web page from remote host using HTTPS `https://example.com/query.html`
- Web page JavaScript talks to Hub on localhost using HTTP `http://localhost:21012/`

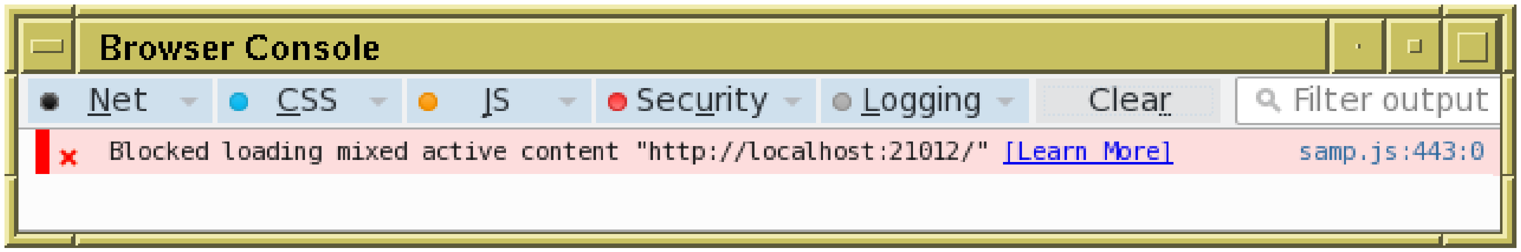
$\longrightarrow$ what's the problem?

HTTPS web page + HTTP SAMP

You might want an HTTPS web application to use SAMP:

- Browser retrieves web page from remote host using HTTPS `https://example.com/query.html`
- Web page JavaScript talks to Hub on localhost using HTTP `http://localhost:21012/`

→ what's the problem?



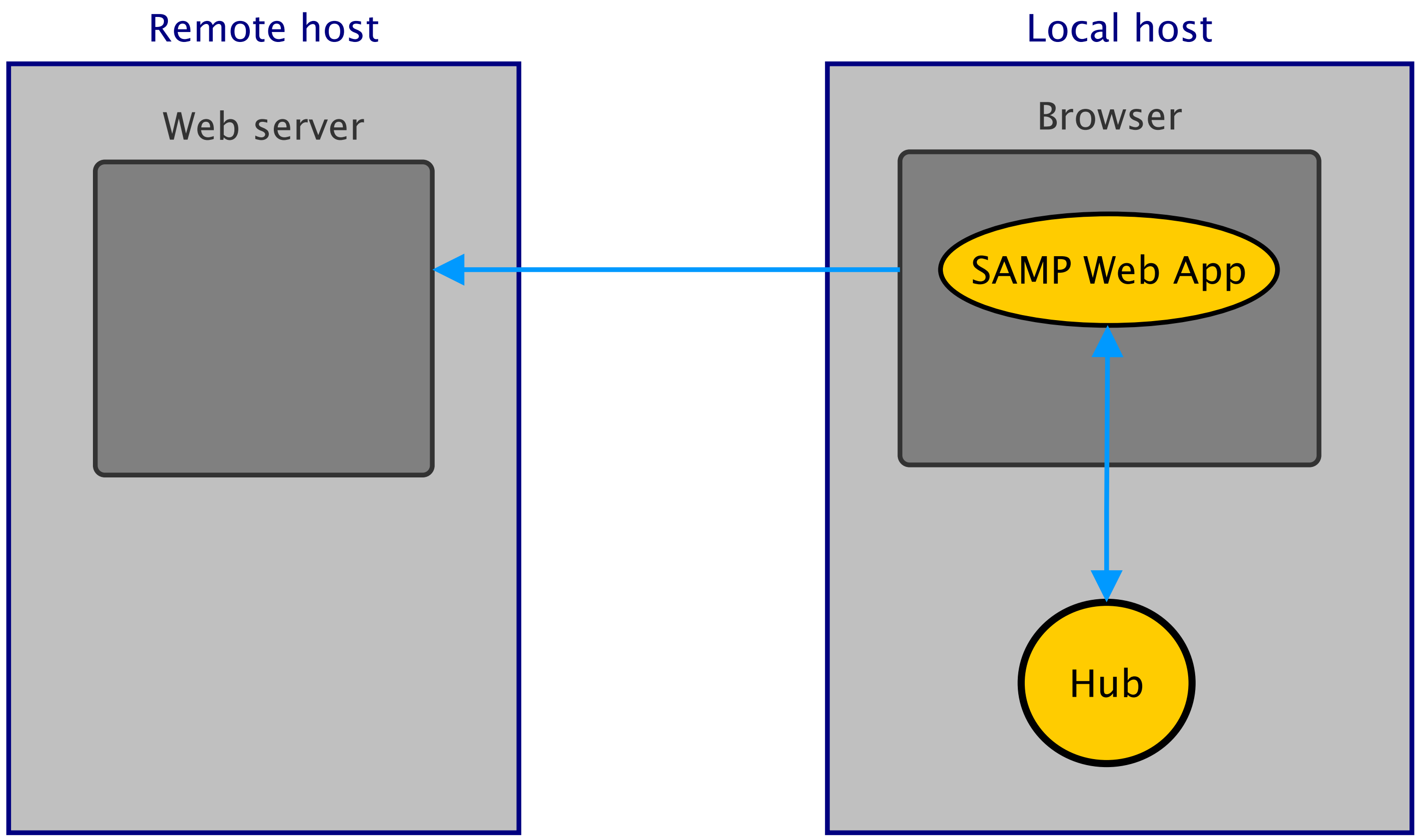
```
Blocked loading mixed active content "http://localhost:21012/" [Learn More]    samp.js:443:0
```

Most browsers block "mixed active content"

- If allowed, pages would be vulnerable to "Man-In-The-Middle" attacks, which would compromise the integrity of the HTTPS communications
- Blocked are *some kinds* of HTTP content within an HTTPS page:

Active:	XMLHttpRequest, javascript, stylesheets, ... *BLOCKED*
Passive:	IMG, video, audio *(grudgingly) ALLOWED*

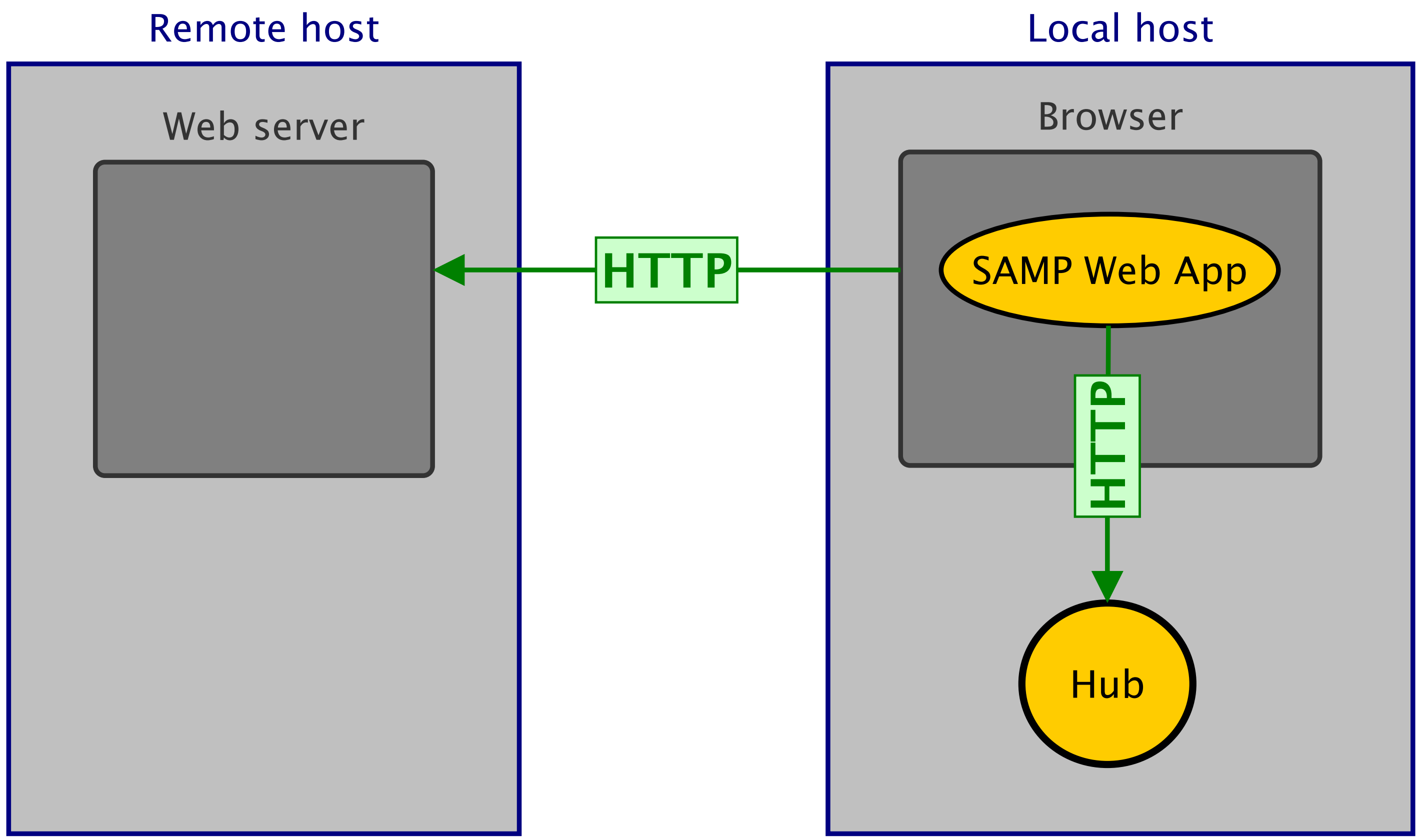
Hub↔Client Communications



Browser retrieves web application from web server

Web application communicates with Hub

Hub↔Client Communications

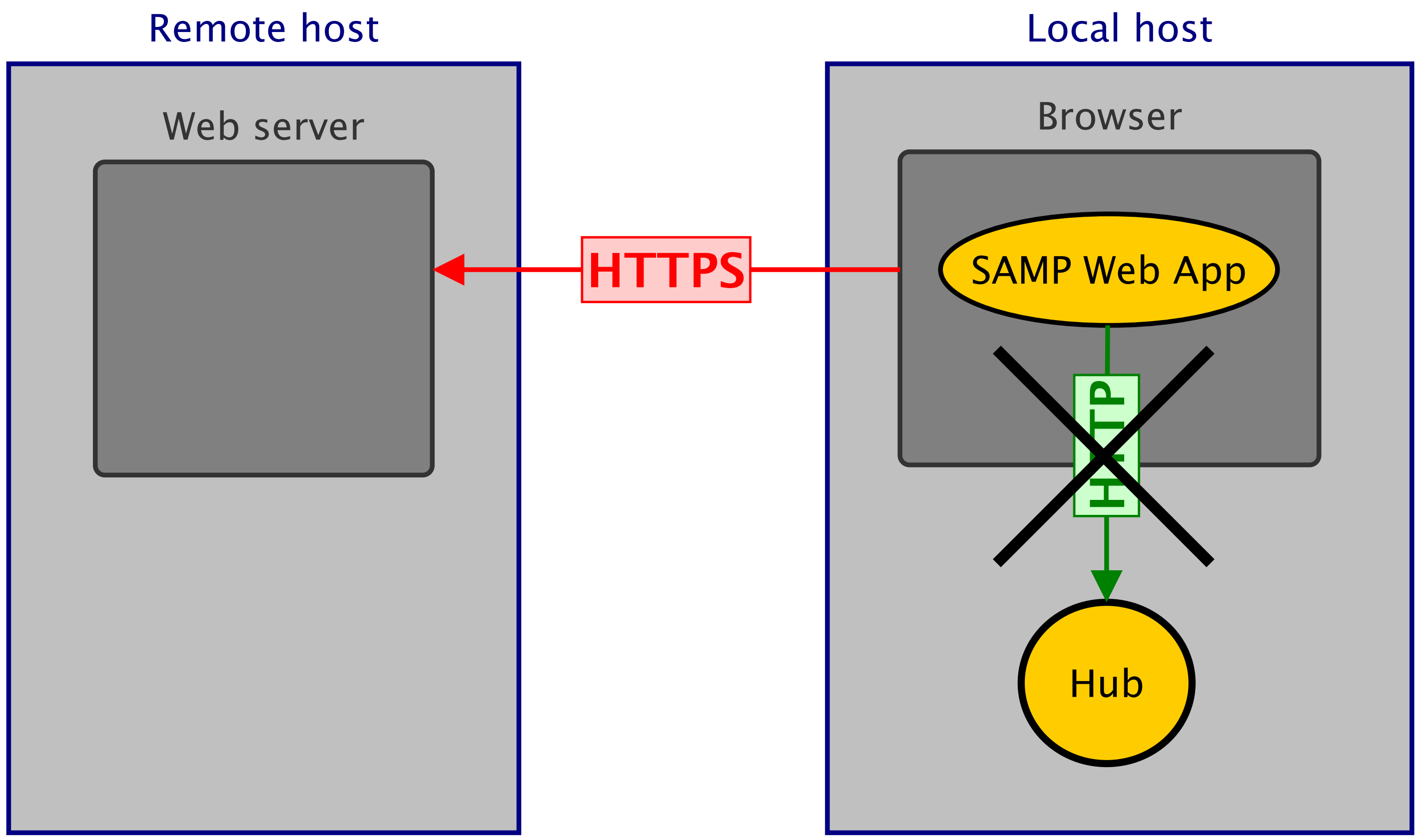


Browser retrieves web application from web server: HTTP

Web application communicates with Hub: HTTP

☺ Normal Web SAMP

Hub↔Client Communications

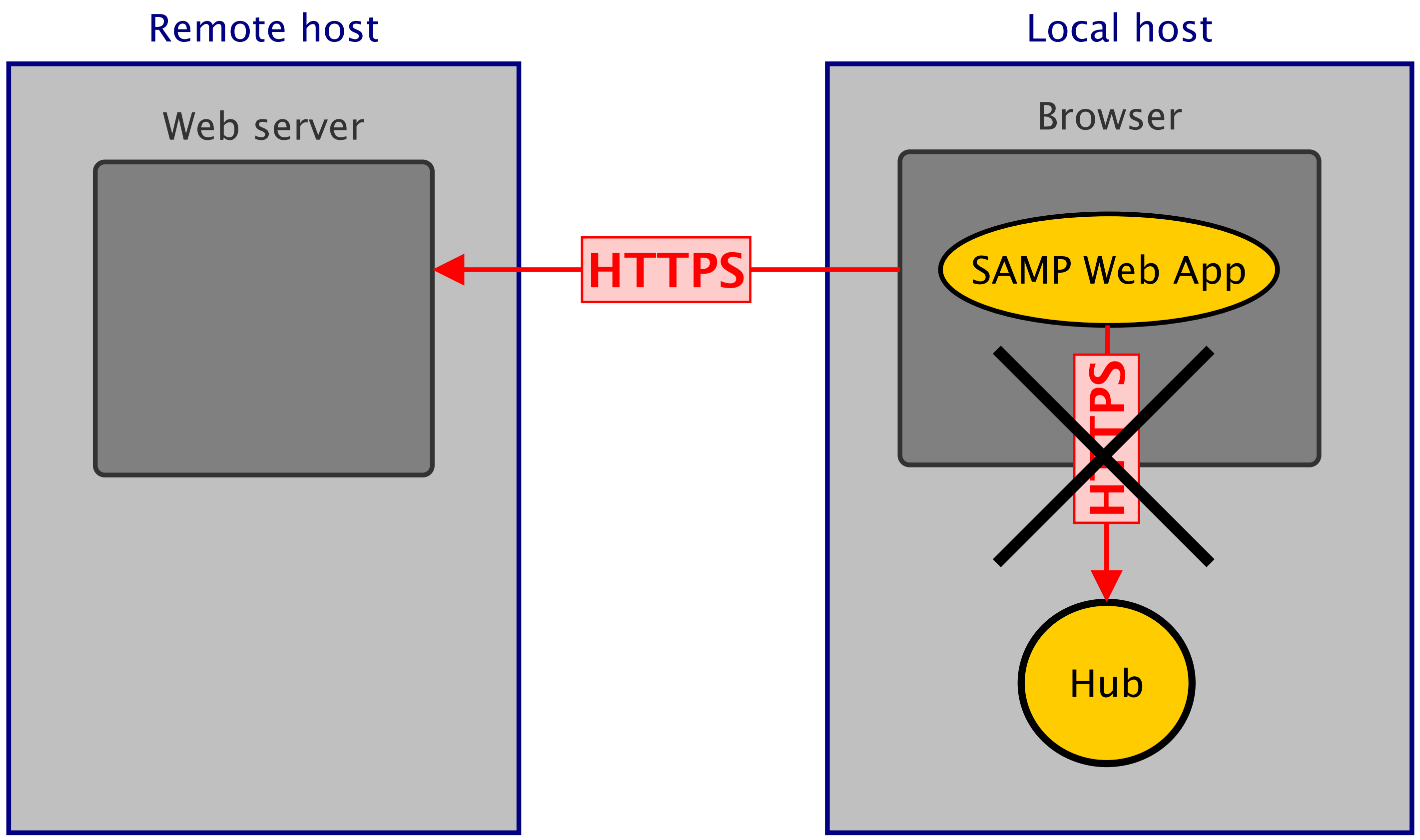


Browser retrieves web application from web server: HTTPS

Web application communicates with Hub: HTTPS

☹ Blocked by browser — Mixed Active Content

Hub↔Client Communications

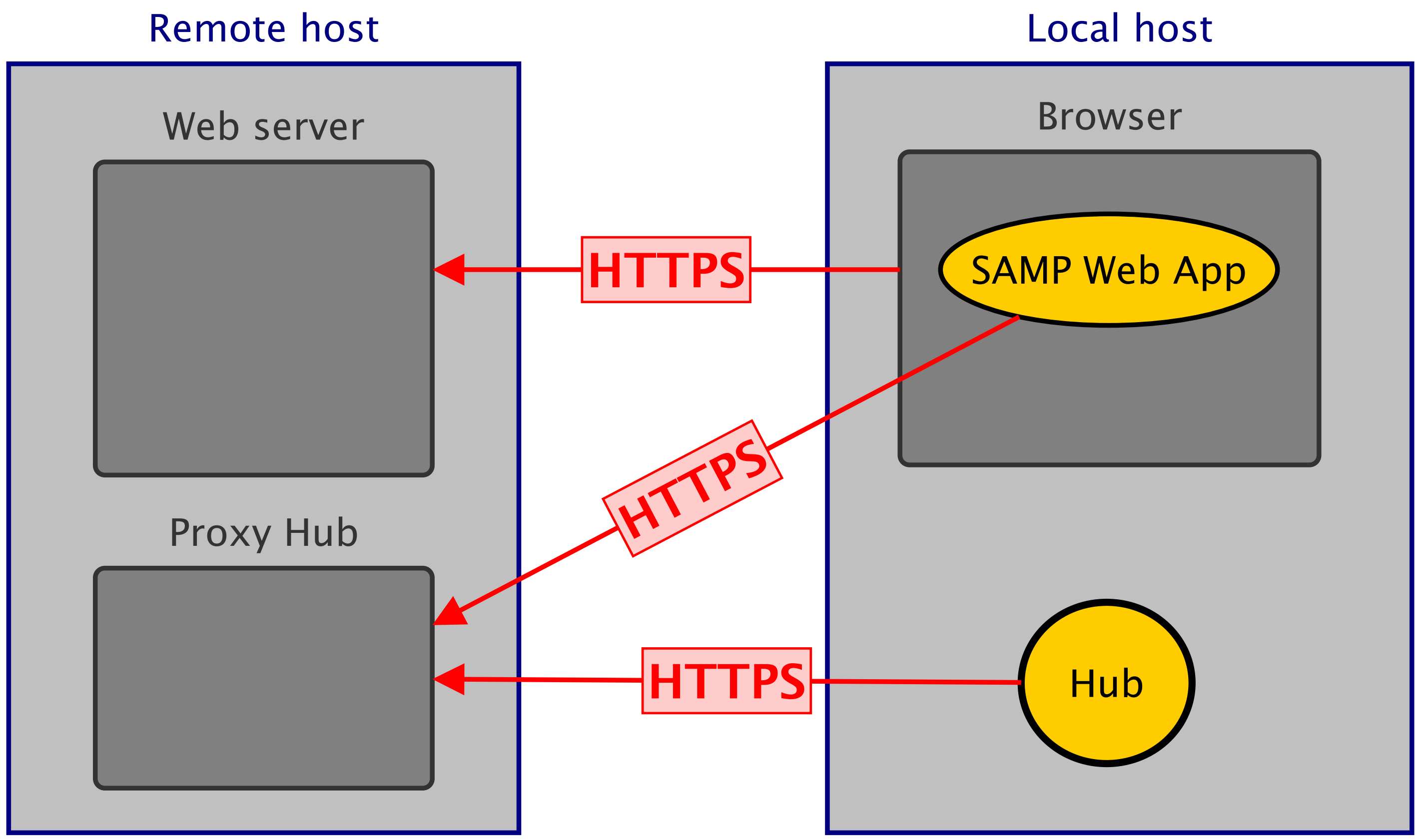


Browser retrieves web application from web server: HTTPS

Web application communicates with Hub: HTTPS

☹ Impossible — localhost security issues

Hub↔Client Communications

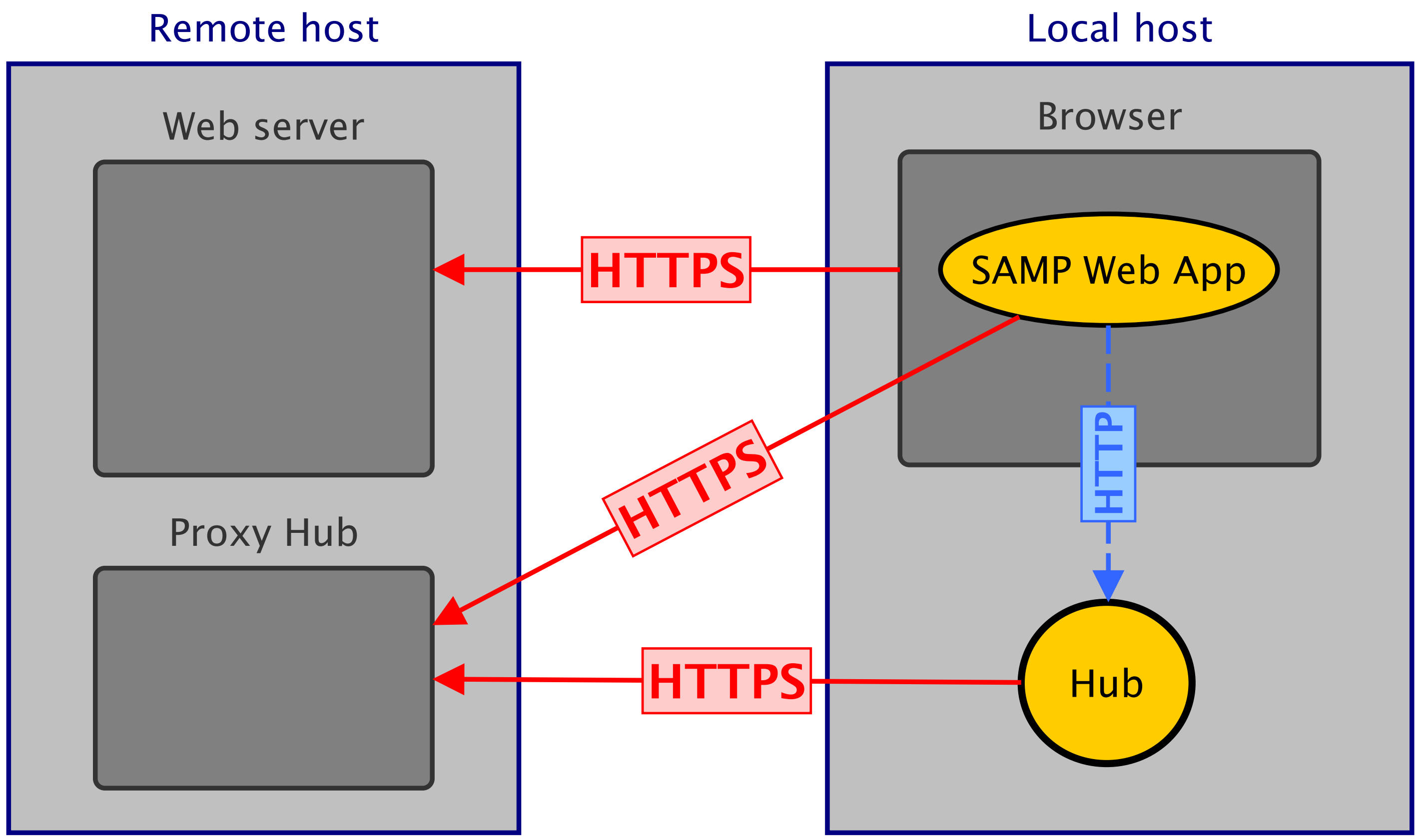


Browser retrieves web application from web server: HTTPS

Web application communicates with Hub: HTTPS via remote server

- OK, but how does hub know to listen?

Hub↔Client Communications



Browser retrieves web application from web server: HTTPS

Web application communicates with Hub: HTTPS via remote server

+ Web app alerts Hub: HTTP *Mixed Passive Content*

☺ Working

Protocol Details

Web application behaviour:

- Knows location of an HTTPS *proxy hub* service (probably on hosting server)
- Makes XML-RPC calls to proxy hub, exactly as if talking to a normal (localhost) hub
- Messages the localhost hub (once? once per XML-RPC call?) using Mixed Passive Content:
  - ▷ Uses well-known hub endpoint (`http://localhost:21013/collect`)
  - ▷ Passes location of remote proxy hub using a well-known parameter (`bouncer`)
  - ▷ Does it by abusing the `<IMG>` element:

```
<IMG src="http://localhost:21013/collect
          ?bouncer=https://andromeda.star.bristol.ac.uk:8080/tlsamp/xmlrpc
          &time=1456918066897"
     width="0" height="0" />
```

Proxy Hub behaviour:

- Collects XML-RPC calls from web application
- Forwards them on request to localhost hub
- Passes the localhost hub's responses back to the web app *(sync: as XML-RPC responses)*

Localhost hub behaviour:

- When the special `/collect` image is requested, asks Proxy Hub for pending calls
- Services such calls (normal hub behaviour)
- Sends call return values to proxy hub *(async: as new XML-RPC calls)*

Implementation Status

Proof-of-concept implementation running:

- Hub: experimental TLS-SAMP Profile for use with JSAMP Hub
- Proxy Hub: example java implementation available in standalone and servlet versions
- Javascript client: `samp.js` library updated, for HTTPS just need extra config like:

```
if (location.protocol === "https:") {
    var proxyHub = baseUrl + "xmlrpc";
    connector.profile = new samp.TlsProfile(proxyHub);
}
```

Available to play with:

- Deployed at: https://andromeda.star.bristol.ac.uk:8080/tlsamp/
- Download web app: http://andromeda.star.bristol.ac.uk/websamp/tlsamp.war

— Basic operations seem to work!

Remaining Requirements

To be done:

- Web application Origin not declared correctly in popup dialogue
- Polling for hub presence not working in javascript library
- Callable client behaviour unreliable
- URL translation not done (non-HTTPS URLs received by web app won't work)
- Security analysis??
- Proxy hub implementation may be inefficient/unreliable
- Various bugs, unreliable operations, ...
- Not documented
- Standardisation work

Open Questions

- Review prototype design decisions:
  - How does client identify itself to proxy hub (just hostname or client-generated token?)
  - When does client message localhost hub (every call or just on registration?)
  - Location of well-known hub port (same as Web Profile or different?)
  - Various endpoint name choices
- Fundamental issues:
  - Security: what can Hub reliably report to user?
  - Use web sockets instead of XML-RPC??

Conclusions

Summary

- OK, it's not impossible ...
- ... but it's ugly and inefficient
  - ▷ SAMP traffic is basically local to the host; this bounces it all via a remote server
- There's a lot of work to take SAMP from HTTP to HTTPS
  - ▷ More options to decide on
  - ▷ Implementation (partly done for Java & js, but quite some tidying left; python??)
  - ▷ Standardisation (new HTTPS Profile to add to standard document)
- This solution may not continue to work indefinitely
  - ▷ Future browsers may disallow Mixed Passive Content (see W3C Mixed Content doc)

Next steps

- Discuss within IVOA
- Improve implementation
- SAMP 1.4 with new HTTPS Profile section
- ... is it worth it?