SAMP over HTTPS

Mark Taylor (Bristol)

IVOA Interop
Cape Town

10 May 2016

$Id: tlsamp.tex,v 1.13 2016/04/29 09:35:09 mbt Exp $

SAMP + HTTP:

SAMP + HTTP:

Working (Web Profile, since SAMP 1.3)

SAMP + HTTP:

Working (Web Profile, since SAMP 1.3)

SAMP + HTTPS:

SAMP + HTTP:

Working (Web Profile, since SAMP 1.3)

SAMP + HTTPS:

`s/http/https/g` ?

SAMP + HTTP:

Working (Web Profile, since SAMP 1.3)

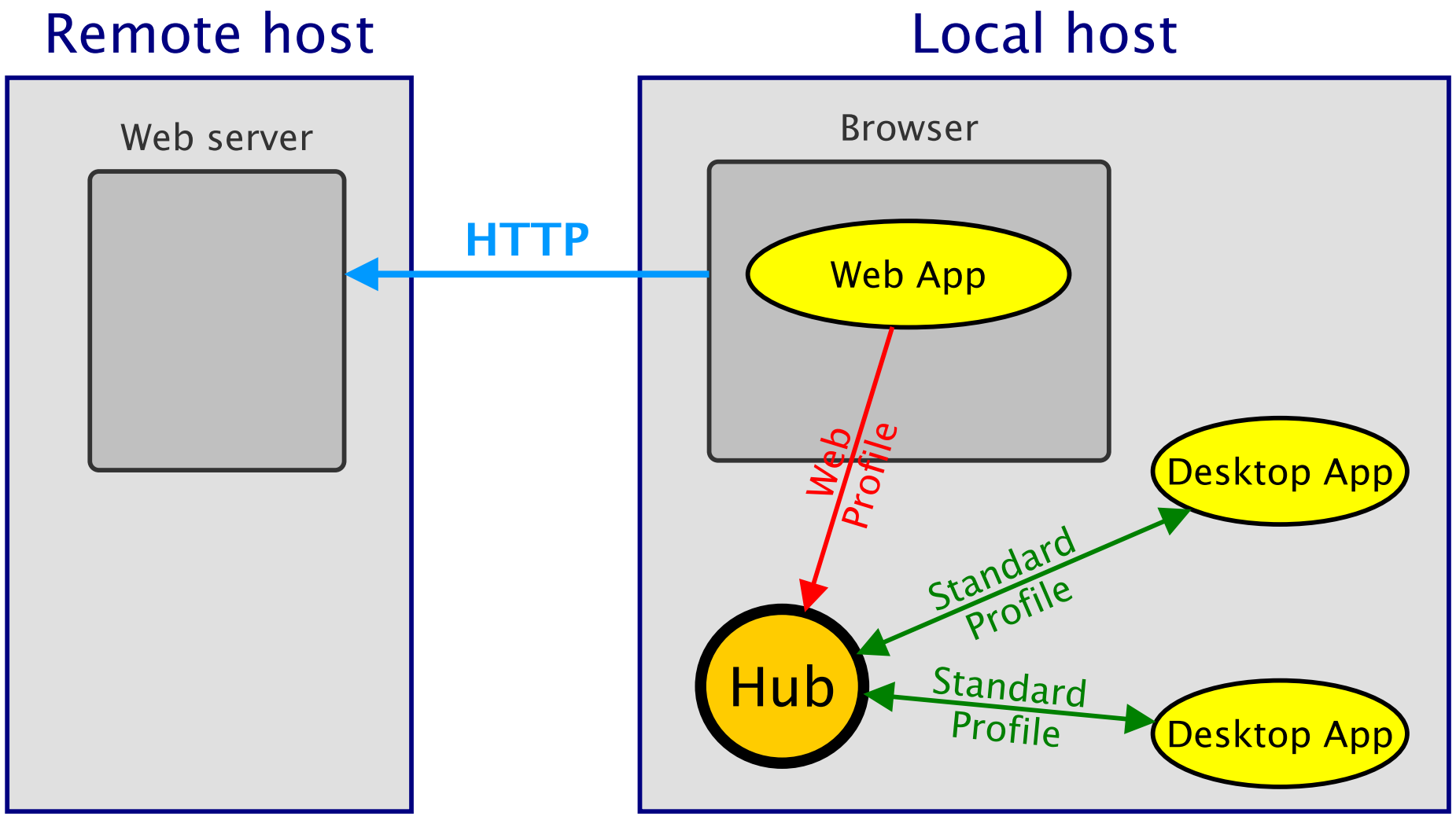
SAMP + HTTPS:

`s/http/https/g` ?

Unfortunately not.

Outline

- (Web) SAMP refresher
- HTTPS + SAMP: the problem *(abbreviated)*
- Proposed solution
- Progress report
- Conclusions
- Next steps?



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/`

→ 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/`

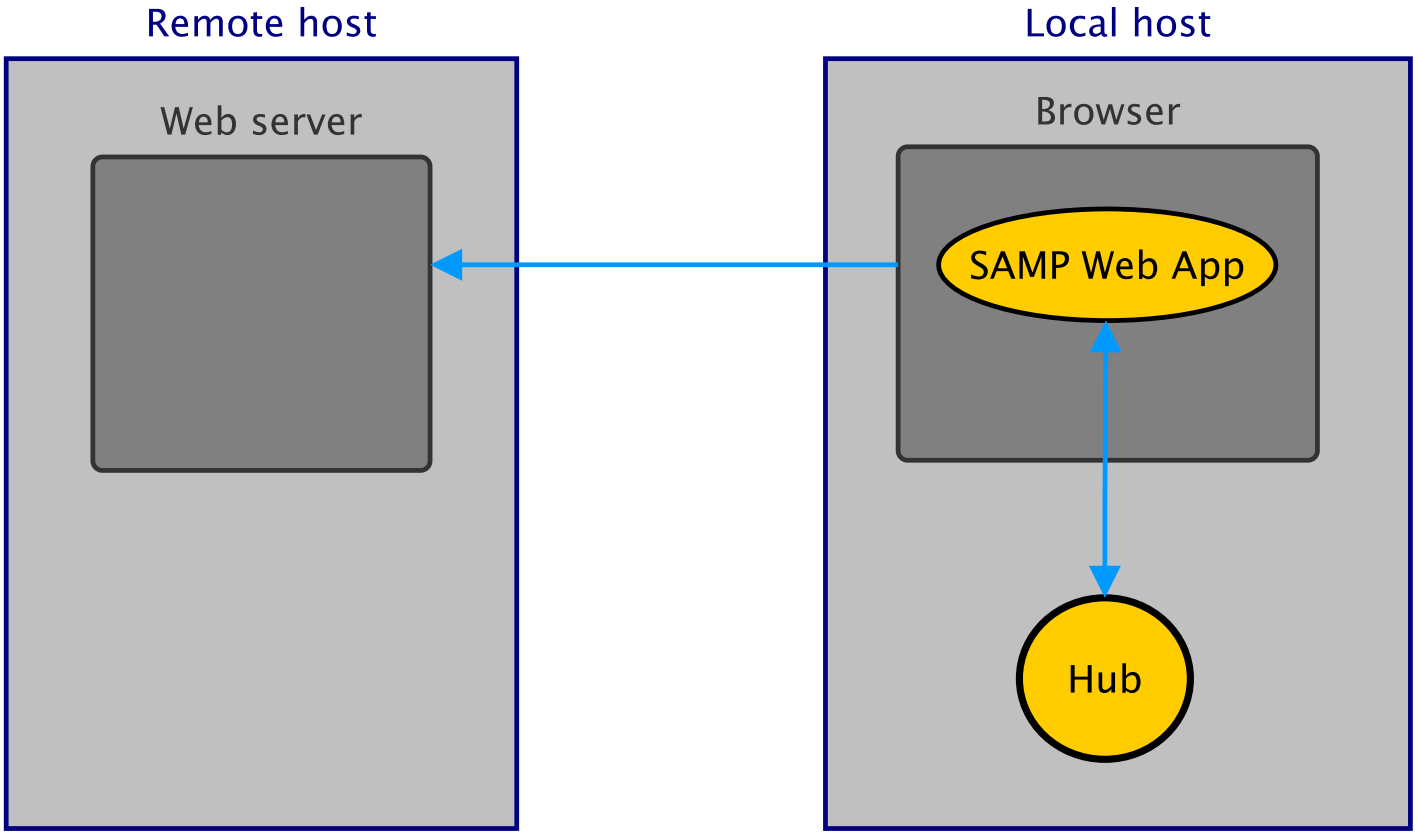
→ what's the problem?

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*

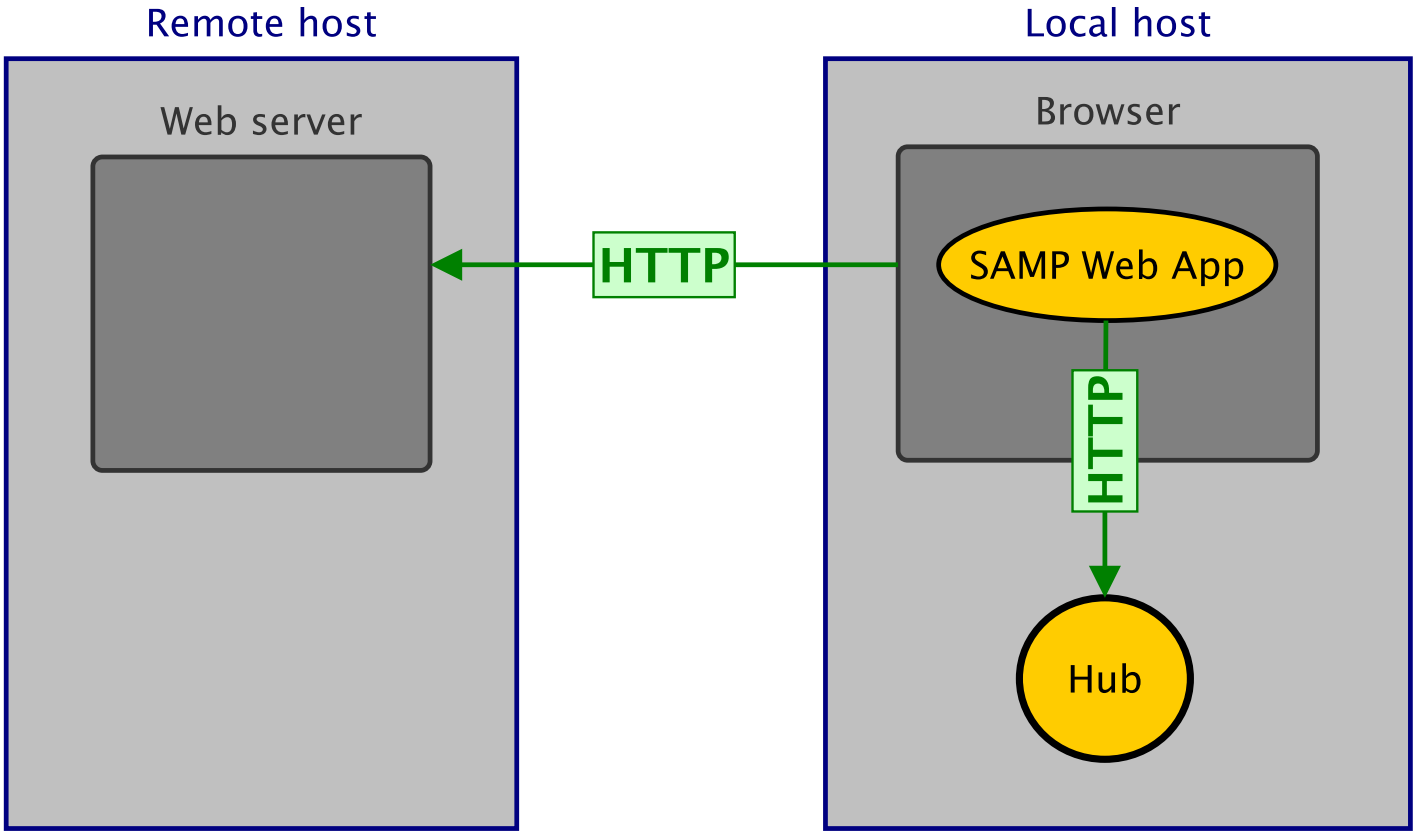
Hub⟷Client Communications



Browser retrieves web application from web server

Web application communicates with Hub

Hub⟷Client Communications

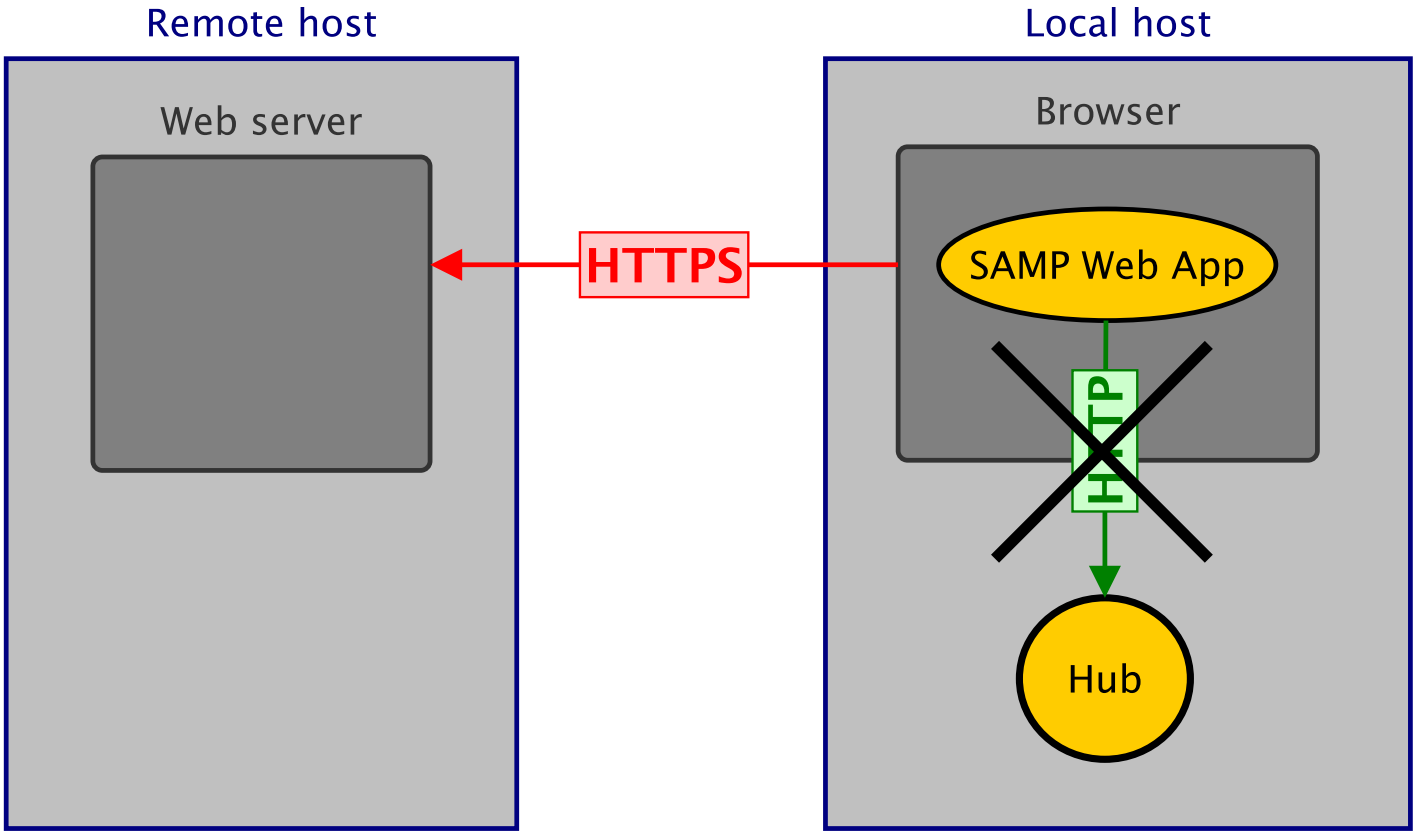


Browser retrieves web application from web server: HTTP

Web application communicates with Hub: HTTP

☺ Normal Web SAMP

Hub⟷Client Communications

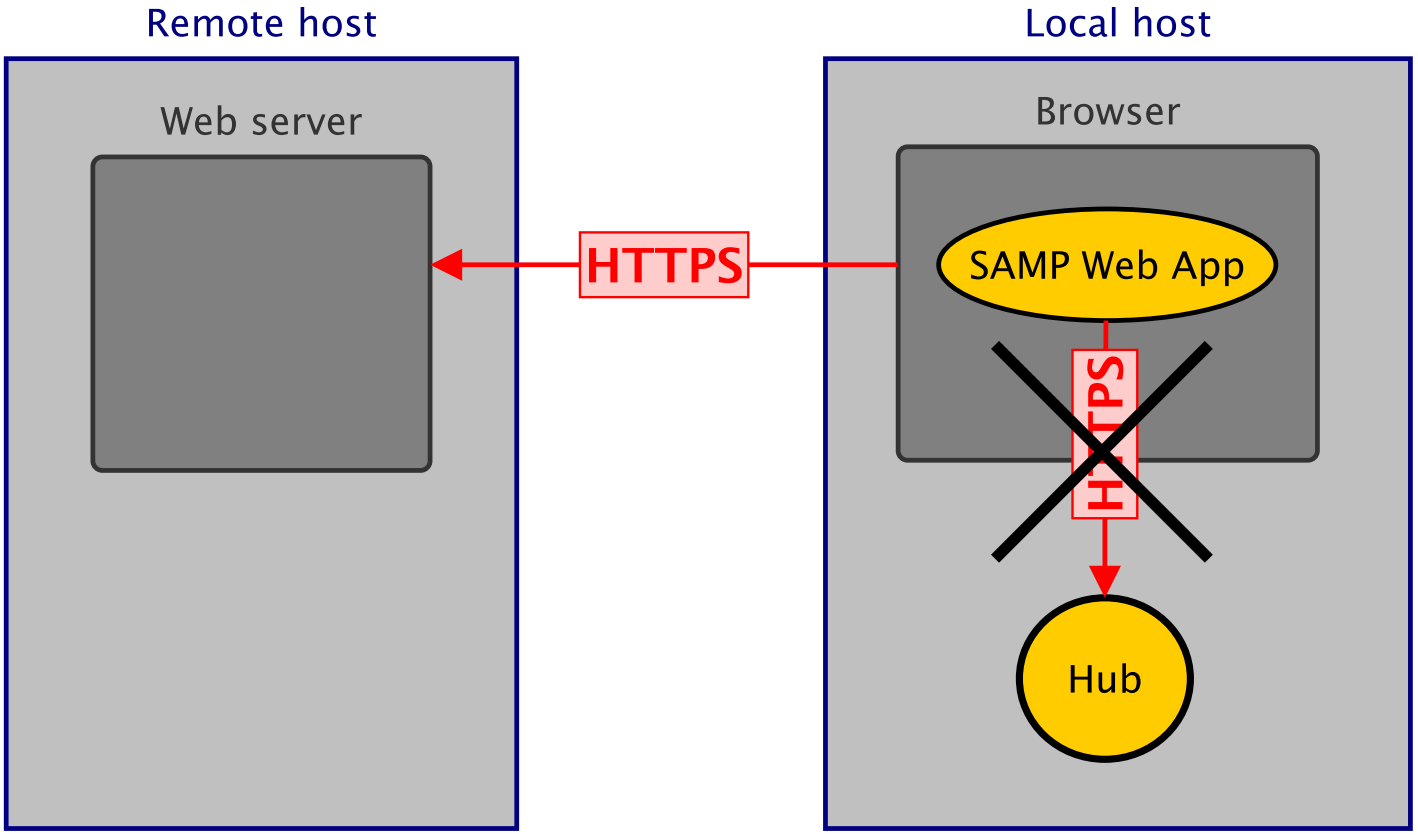


Browser retrieves web application from web server: HTTPS

Web application communicates with Hub: HTTPS

☹ Blocked by browser — Mixed Active Content

Hub⟷Client Communications

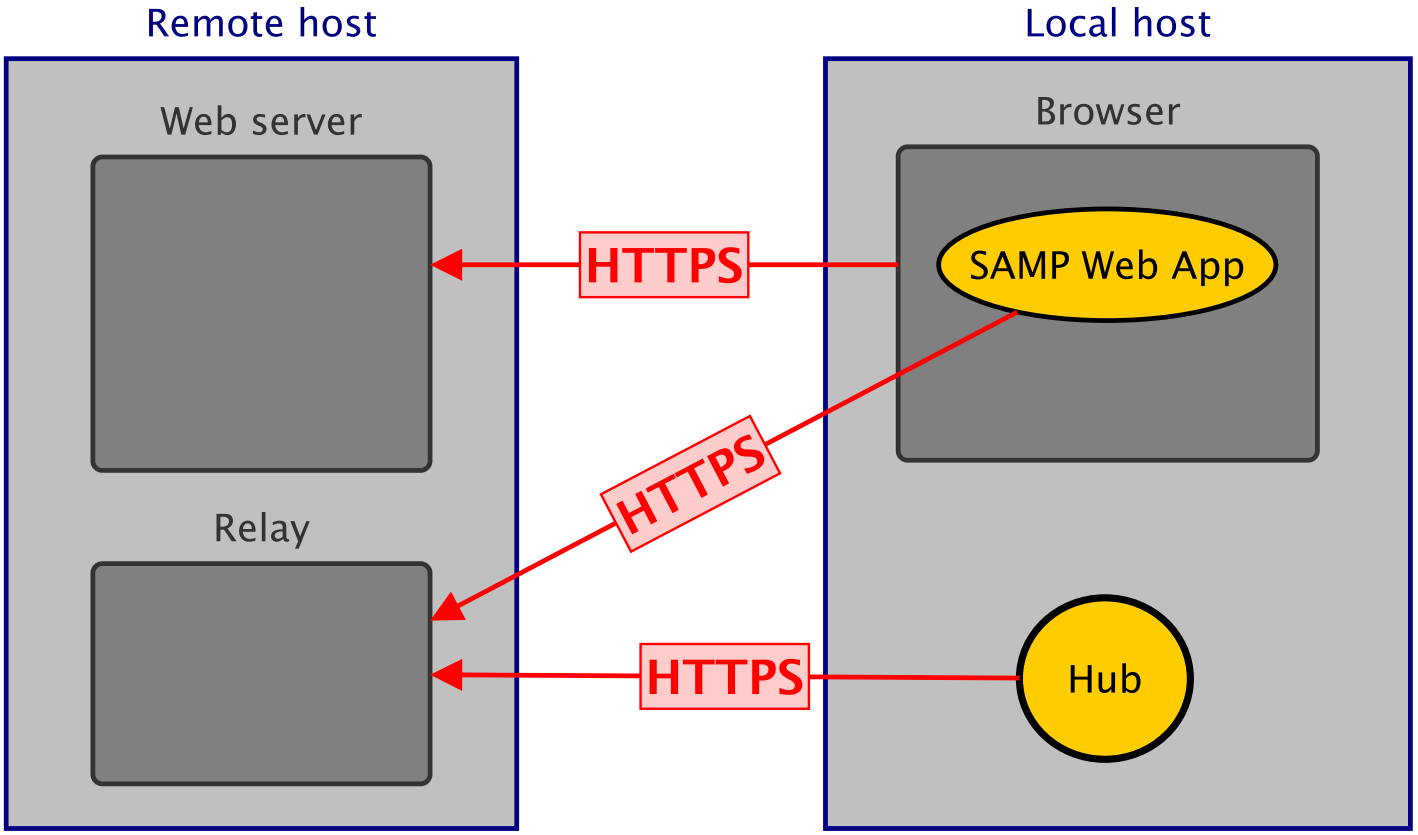


Browser retrieves web application from web server: HTTPS

Web application communicates with Hub: HTTPS

☹ Impossible — localhost security issues

Hub⟷Client Communications

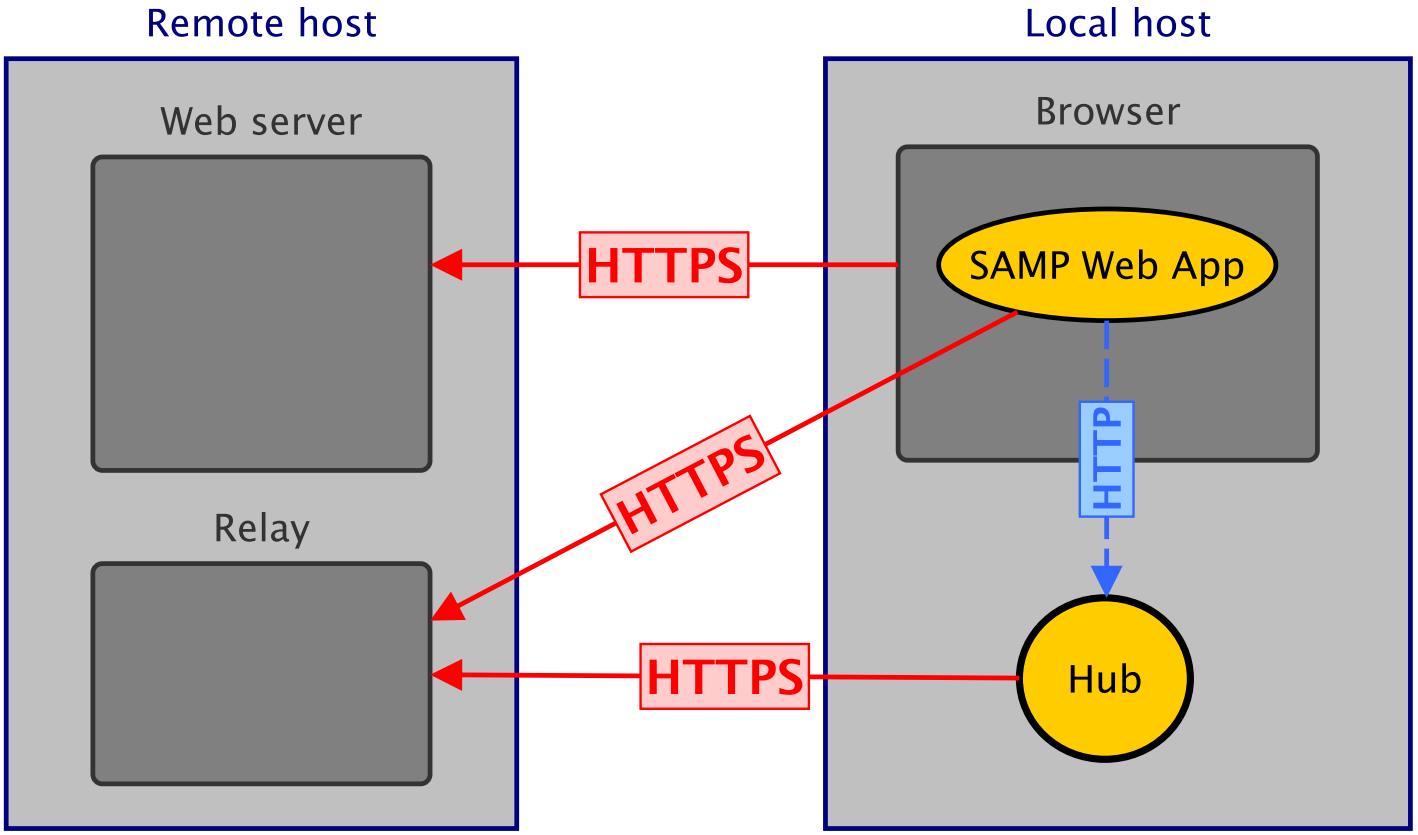


Browser retrieves web application from web server: HTTPS

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 *nudges* Hub: HTTP *Mixed Passive Content*

☺ **Working**

Protocol Details

Web application behaviour (in browser on localhost):

- Knows location of an HTTPS *Relay* service (probably on hosting server)
- Makes XML-RPC calls to Relay exactly as if talking to a normal (localhost) hub
- *Nudges* the localhost hub (once? once per XML-RPC call?) using Mixed Passive Content

Relay behaviour (on external server):

- Exposes an XML-RPC interface just like that of a Web Profile Hub
- Collects XML-RPC calls from web application
- Forwards them on request to Hub
- Passes the Hub's responses back to the web app *(synchronously, as XML-RPC responses)*

Hub behaviour (on localhost desktop):

- Listens on well-known port (21013)
- When the special `/nudge` image is requested, asks Relay for pending calls
- Services such calls (normal hub behaviour)
- Sends call return values to Relay *(asynchronously, as new XML-RPC calls)*

Nudge Hack

Details

- Web client requests an embedded image by changing a DOM `<IMG>` element `src` attribute to some special URL
- The URL can encode additional information, e.g. as query parameters (`?param=value`)
- This smuggles a parameterised message to the hub (the HTTP server)
- The web client JavaScript is notified when the image has been loaded, but has no access to the image content (the image is just displayed in the page)

  $\Rightarrow$ the message is strictly one-way, Hub $\longrightarrow$ Client
- Permitted by browser sandbox only because loaded images are *Mixed Passive Content* ("*optionally-blockable*" in the language of W3C Mixed Content document)

Example

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

- `http://localhost:21013/nudge`: well-known URL
- `relay=...`: location of XML-RPC relay service, known to web client and passed to hub
- `time=...`: cache buster
- `width="0" height="0"`: optionally hide actual (uninteresting) image content

Transport

Problem

- HTTP(S) communication is one directional: localhost $\longrightarrow$ server
- Relaying SAMP calls needs both directions: localhost $\longleftrightarrow$ server

Solutions:

- Current prototype protocol uses XML-RPC with HTTP(S) long polls
  - ▷ Client interface very similar to Web Profile; much code can be reused in hub implementations and web applications
  - ▷ Inelegant, inefficient? Prone to connection exhaustion? See RFC 6202.
- Maybe should use Web Sockets rather than HTTPS long polls
  - ▷ Need an additional layer for RPC over Web Sockets
  - ▷ Obvious choice is WAMP (Web Application Messaging Protocol — see IETF draft)
  - ▷ Architecture nicely matches what SAMP would require
  - ▷ Cleaner design
  - ▷ More efficient? More robust? More straightforward security model?
  - ▷ Would require quite a bit of new standard text and implementation (no longer XML-RPC-based)
  - ▷ Library support available, but big (e.g. jawampa ∼5 Mb; cf. JSAMP ∼0.7 Mb)
  - ▷ Web client code would need more changes from HTTP version

Open Questions

Robustness

- Not widely tested, not all tests successful — don't know why

Security

- SAMP over HTTPS doesn't necessarily mean secure SAMP
  - Notionally private data is now relayed off-host (but over HTTPS)
  - Profile vulnerable to more interference than Web Profile
  - More complicated architecture means more things to get hacked/misunderstood
- Adjustments to current protocol design could help
  - More use of host identification tokens
  - Requires more protocol complexity and more implementation effort

Longevity

- The (essential) *Nudge* hack relies on browsers allowing *Mixed Passive Content*
- W3C intention is to disallow this one day:

> *"Note: Future versions of this specification will update this categorization with the intent of moving towards a world where all mixed content is blocked; that is the end goal, but this is the best we can do for now."*
>
> — W3C Mixed Content document, sec 3

Protocol Status Summary

Feature completeness:

- No URL Translation
  - Localhost-specific URLs (e.g. `file:///...`, `http://localhost...`) sent to HTTPS SAMP clients are unreadable
  - Can't use same approach as for Web Profile; web client can't talk directly to Hub
  - There are ways to do this, but they are both fiddly and inefficient (relaying bulk data over WAN)
  - Few (very few?) web applications actually need this function
- Everything else should work

Changes to make?

- Modify protocol for improved security (more identification tokens)?
- Experiment with Web Sockets/WAMP?

Outlook

- May stop working one day, if future browsers block mixed passive content

Implementation Status

Proof-of-concept implementation running

- Hub: experimental TLS-SAMP Profile for use with JSAMP Hub
- Relay: 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 relay = baseUrl + "xmlrpc";
    connector.profile = new samp.TlsProfile(relay);
}
```

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
- Source code: https://github.com/mbtaylor/tlsamp

Success?

- Works for me ☺
- ... but not for Tom McGlynn ☹

Conclusions

Summary

- Some people want to host SAMP web clients on HTTPS web pages
- OK, it's not impossible ...
- ... but it's ugly and inefficient
  - ▷ SAMP traffic is notionally local to the host; this relays it all via a remote server
- .. and there's a lot of work required:
  - ▷ Adjustments to prototype (security, URL translation; Web Sockets rewrite?? ...)
  - ▷ Implementation (XML-RPC partly done for Java & js; python not started)
  - ▷ Standardisation (new HTTPS Profile to add to standard document)
- The solution may not continue to work indefinitely

Questions:

- *Why* do people want to host SAMP clients from HTTPS?
  - ▷ To support robust authentication?
  - ▷ Political/organisational directive to move to HTTPS?
  - ▷ Fashionable thing to do?
- How many services need to do this? Will the number increase over time?
- Are there other ways round it?
- *Does the requirement justify the effort?*

Next Steps

If we want to take this forward, next steps are:

- Deployment tests
- Review prototype protocol
  - minor adjustments?
  - rewrite using web sockets/WAMP?
- Write/complete implementations (java hub, python hub, javascript client library, relay)
- SAMP 1.4 with new HTTPS Profile section