The Case for Bi-Lateral End-To-End Strong Authentication

by
C. Chandersekaran, Institute for Defense Analyses
William R Simpson, Institute for Defense Analyses

In certain enterprises, the network is continually under attack. An example might be a banking industry enterprise such as a clearing house for electronic transactions, defense industry applications, even credit card consolidation processes that handle sensitive data both fiscal and personal. The attacks have been pervasive and continue to the point that nefarious code may be present, even when regular monitoring and system sweeps clean up readily apparent malware. This Omni-present threat leads to a healthy paranoia of resistance to observation, intercept and masquerading. Despite this attack environment, the web interface is the best way to provide access to many of its users. One way to continue operating in this environment is to not only know and vet your users, but also your software and devices. Even that has limitations when dealing with the voluminous threat environment. Today we regularly construct seamless encrypted communications between machines through SSL or other TLS. These do not cover the "last mile" between the machine and the user (or service) on one end, and the machine and the service on the other end. This last mile is particularly important when we assume that malware may exist on either machine, opening the transactions to exploits for eaves dropping, ex-filtration, session high-jacking, data corruption, man-in-the-middle, masquerade, blocking or termination of service, and other nefarious behavior.

To counter this we devise a system where all active entities (users, devices, and services) are named, registered and credentialed. We assume a single domain or at least a single enterprise where we have control of these details, but will address a federated case later. Credentials include asymmetric encryption keys. All services and devices exercise access controls and use SAML Assertions in their decision process. The requestor will not only authenticate to the service (not the server or device), but the service will authenticate to the requestor. The interface is termed a "Fat" API, or in the case of a browser or presentation system it is a "fat" browser. In the figure below we show the constituent makeup of a service.

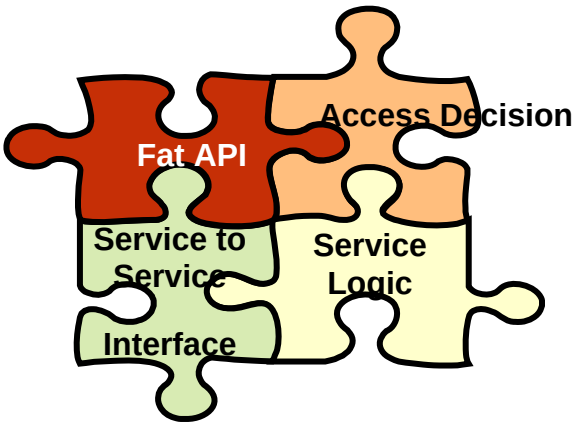

Figure 1 Components of a Service

The FAT API must be plug compatible with the Fat Browser and the Service-to Service Interface as shown in the next figure. It is therefore important that these exercise compatible code segments.

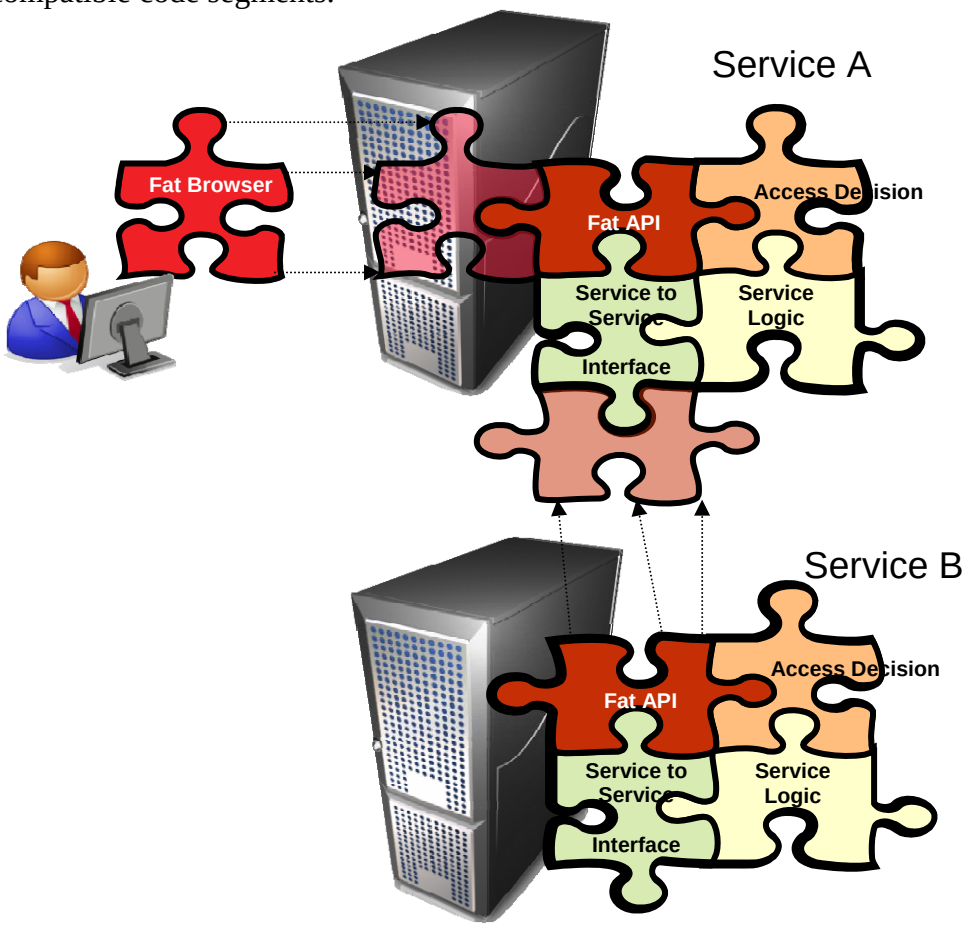

Figure 2 Fat Interfaces Must be Plug Compatible

In the figure below we show two types of Services. The first is an Aggregation Service. This Service calls exposure services aggregates their output and returns the data to the user. The second is an Exposure Service that provides data from an authoritative data source. The "fat" Service call is different between services than between browser and service. The "fat" APIs will also be different for different environments (e.g., .NET or J2EE). The "fat" part of the API consists of:

- Port Listener
- (save data input)
- Bi-lateral End-to-End Authentication
- Consume the assertion package for authorization
- Pass Authorization credentials and initial input to the service

The initiating part on the "Fat" Browser and the Service-to-Service invocation must meet the compatibility issues.

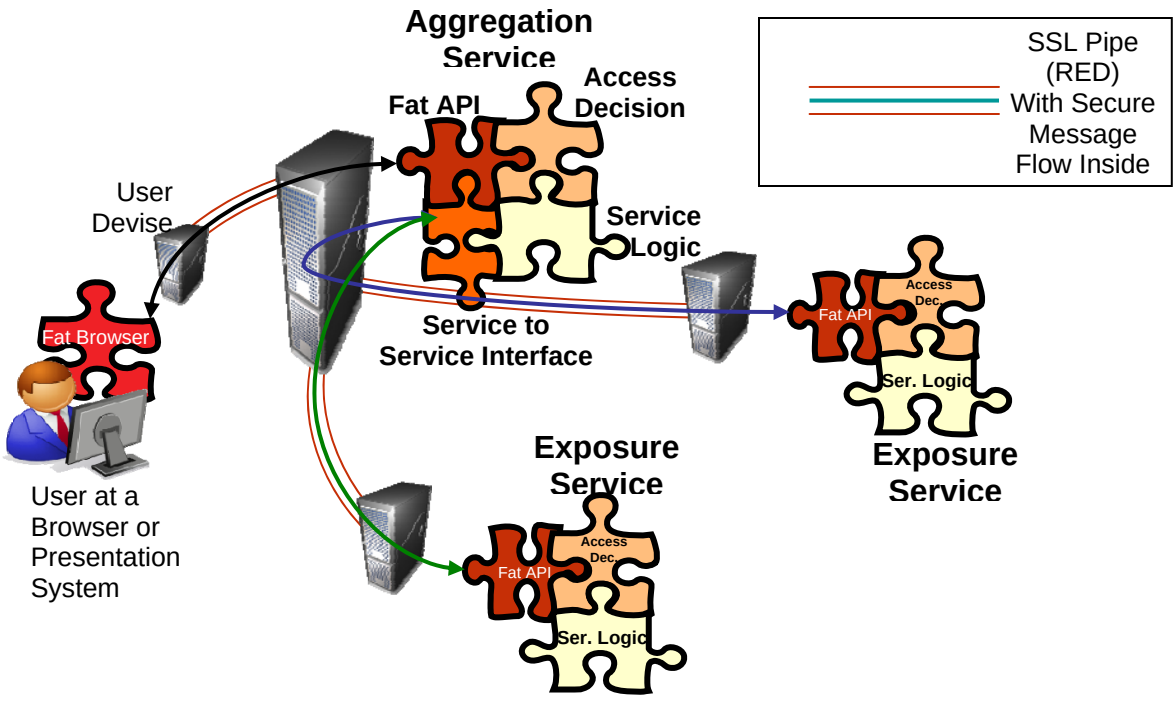

Figure 3 The Steps in Invoking an Aggregation Service

This two way authentication avoids a number of threat vulnerabilities. The requestor will initially authenticate to the server or device and set up an SSL connection to begin communication with the service. The primary method of authentication will be through the use of public keys in the X.509 certificate, which can then be used to set up encrypted communications, (either by X.509 keys or a generated session key). Session keys and certificate keys need to be robust and sufficiently protected to prevent malware exploitation. The preferred method of communication is secure messaging, contained in SOAP envelopes.

Several key pieces are missing to complete this scenario. On the user end we need WS-enabled browser with the ability to communicate with an ID processor and a Security Processor which together form a Security Token Server (STS). The STS will facilitate the exchange of credentials, aid in setting up the initial SSL, and provide the SAML package for consumption. The fat browser may be on a desktop or a mobile device. On the service provider end we need the software to encrypt/decrypt secure message and to consume the SAML package. The latter is not trivial since it must be checked for signature, tampering, timeouts and other factors. If we assume for the moment that the user is tightly bound to the browser, then the user security context is maintained through the device and all the way to the service. This context will assist in attribution and delegation and in monitoring insider behavior activity. The remaining threats of insider activity, ex-filtration of static data and denial-of service (DOS) attacks must be handled by other means, but behavioral modeling, static encryption and dynamic ports and protocols still apply to these threats. Both the fat browser and the fat API are under development.

Several additional features of the STS are needed which the OASIS standards have not addressed. When the communication is across domains, then and STS in each domain is needed and a mutual recognition of signature authority is needed. If they are across enterprise we may need to do a remapping of the SAML assertions, and we need a good process for least privilege, delegation and attribution in each of these circumstances. While WS-Federation standards assist; they do not specifically address attribute pruning, remapping, or multiple STS registered recognition.

The process is not without draw-backs, in that additional cycles are used in the bi-lateral authentication and the double encryption (both SSL and secure messaging). This latter makes it unattractive for some applications where the threat environment is minimal. However, there exist a number of environments where the added security is worth the cycles, or where higher performance cores are available.