Final Project Report

December 9, 2012

Cloud-based Authentication with Native Client Server Applications.

Nils Dussart

0961540

CONTENTS

PROJECT PROPOSAL

PROJECT TITLE

Cloud-based Authentication with Native Client Server Applications.

FACULTY ADVISOR

Professor Munehiro Fukuda

INTRODUCTION

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. Several type of authentication infrastructure exists, that can be classified into 2 main groups on premise and in the cloud. Large enterprises often use a centralized directory that provide authentication to their users. A very commonly used protocol in those designs is Kerberos. For the cloud, several large Identity Providers (IdPs) are provided authentication mainly for web services but also for a wide variety of application. Most of those scenarios use the OAuth or OpenId protocols.

In both scenarios, we also see a multitude of applications that provide their own authentication mechanism. In those cases the authentication is managed by the application itself. This means the application often provides its own custom authentication protocol, the own list of users and passwords all in a secure way. Such designs do not scale well in the enterprise space since it doesn't scale well and creates a management challenge from a compliance and auditing standpoint. Since the application owners are often responsible for securing their application specific security expertise is required (Threat modeling, Intrusion testing) which is not always available.

PROJECT SCOPE AND INDIVIDUAL STUDENT LEARNING GOALS

My project goal is to analyze the possibility of using cloud identity providers as potential identity provider to achieve authentication with on premise client-server applications.

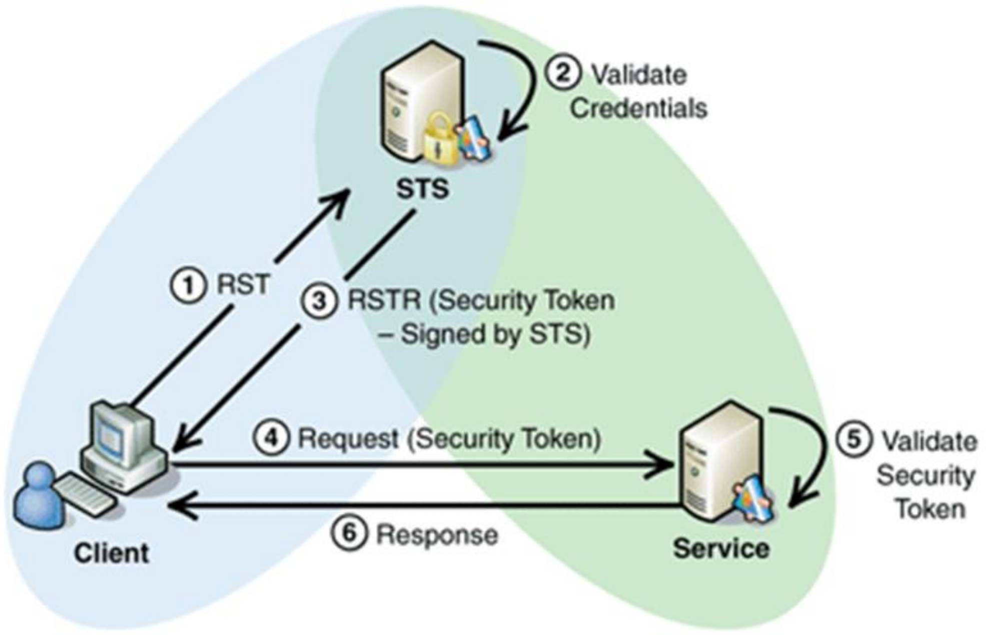
The main objectives are to:

- Identify how to make client-server applications leverage federated login
- Build a proof of concept using and FTP client & server developed in Java
- Identify the necessary modifications to the FTP RFC to make this possible
- Identify the pros and cons of such designs and their limitations if any

ARCHITECTURE DIAGRAMS

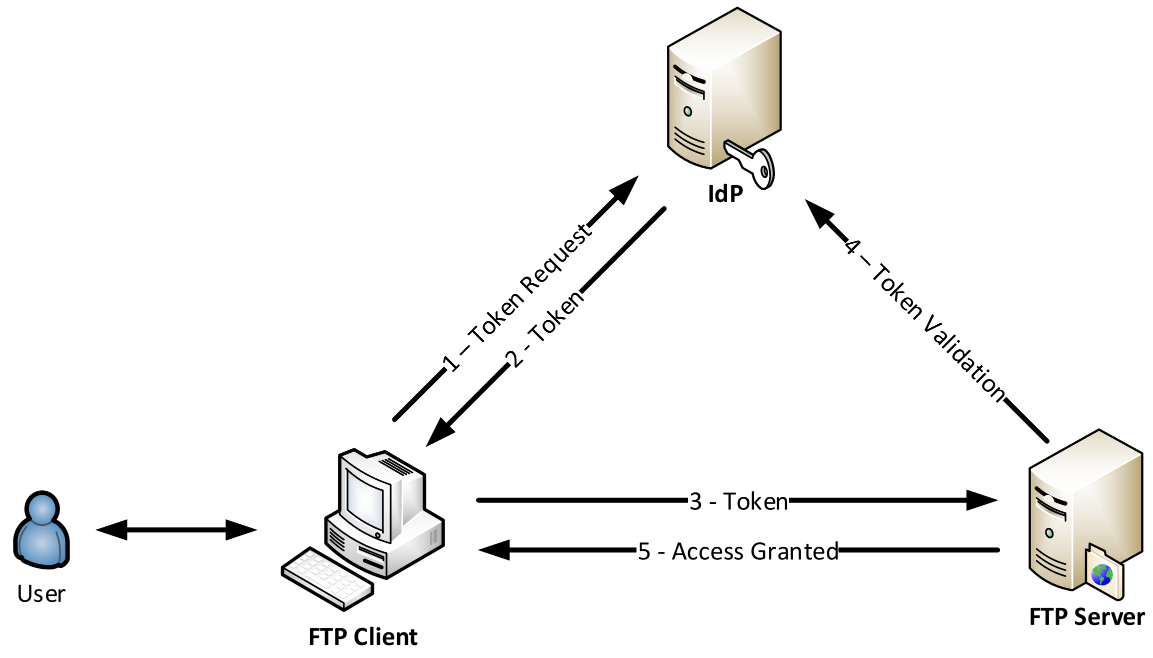
The following diagram describes the usual flow from a web service hosted in a cloud environment. In this case the token is basically relayed between the identity provider (STS) and the service itself. Since this flow happens through the client browser, the client itself has no operation or task to perform in the authentication process itself since everything happens in the backend.

This flow is important to my project since it provides a description if the usual architecture used between 2 different web services roles. Our objectives as part of this project will be to intercept the token and take action on it in the premise of the client rather than in the cloud.


② Validate Credentials
STS
① RST
③ RSTR (Security Token – Signed by STS)
④ Request (Security Token)
⑤ Validate Security Token
⑥ Response
Client
Service

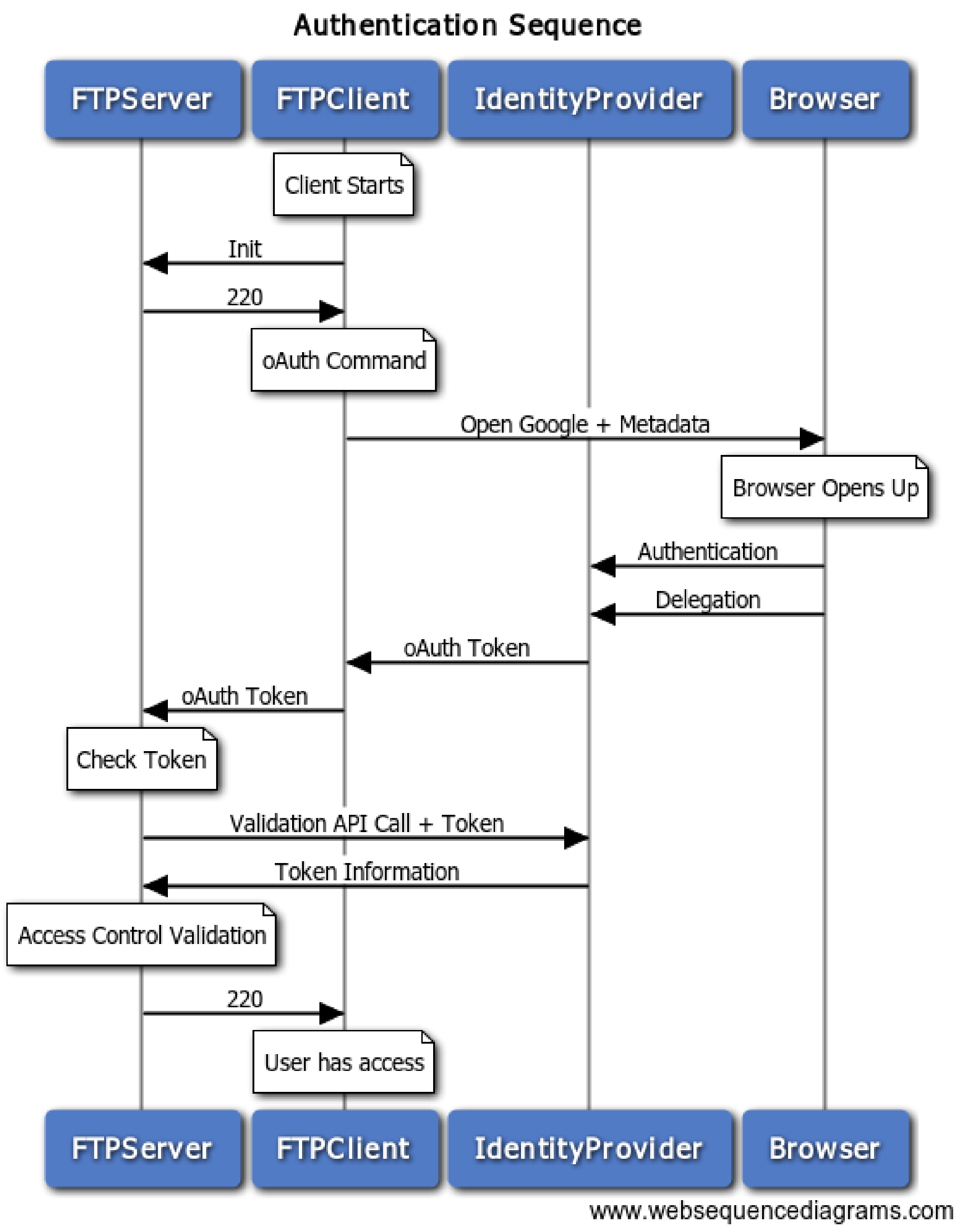
HIGH LEVEL ARCHITECTURE

The project consists into developing a proof of concept where a native client server application will authenticate against an identity provider located in the Cloud. The client will have to obtain an authentication token from that identity provider and then transfer it to the FTP server. The server once in possession of the token will have to verify the authenticity of the token and then make an authorization decision based on the token if it's coming from a trusted identity provider.



AUTHENTICATION SEQUENCE

The following diagram provides the authentication sequence that will be used by all the different players in this project. There are two specific things worth noticing as part of that diagram. First during the authentication sequence with the IdP, the username and password are never in contact with the FTP client and always stay in the context of the Browser process. This provides great security benefit by segregating the secrets from the application itself. Second the validation of the token is done by call an IdP REST API to which delegation is given during the authentication sequence.



NEW FTP COMMAND INTRODUCED

The following new commands are introduced to the FTP server as part of this work:

- OAUTH (OAuth related command)
- REIN (Reinitialize)

OAUTH COMMAND

This new command is non-standard and was implemented for the specific need of this project. This command allows a client to specify that it wants to authenticate using OAuth. This command will trigger the browser session which is required for the user to authenticate. Once the token is returned from the IdP to the client, the client will then transfer it to the server. Once the server is in possession of the token it will validate that the token is valid.

REIN COMMAND

This command will terminate the USER session and purge all account information. It will allow any transfer in progress to be completed. The session will be reset to the default settings and the connection is left open. This is identical to the state immediately after the connection is opened. A USER command may be issued to login or through the new OAUTH command. This command is a standard command defined by the FTP RFC.

DETAILED IMPLEMENTATION

The main challenge of this project was to work with an existing OAuth library. For this specific project we've used the OAuth Java Library from Google that provides a great of abstraction to the authentication process. My first goal was to wrap this library into in a class to simply the overall token request process and totally abstract it from the developer. The OAuth class provided as part of this project provides an easy way to obtain and retrieve a token but also to validate it.

This class is used by both application delivered as part of this project, the FTP Client and the FTP Server. While both applications are using the same class, their use of the class is very different. In the case of the FTP Client, a token will be obtain from Google through an authentication takes place in a browser. The token will then be transferred to the server. The server, on reception of the token, will create a new instance of this specific class using a specific constructor using the token pass by the client. Finally the server will use the class method allowing to validate the token itself.

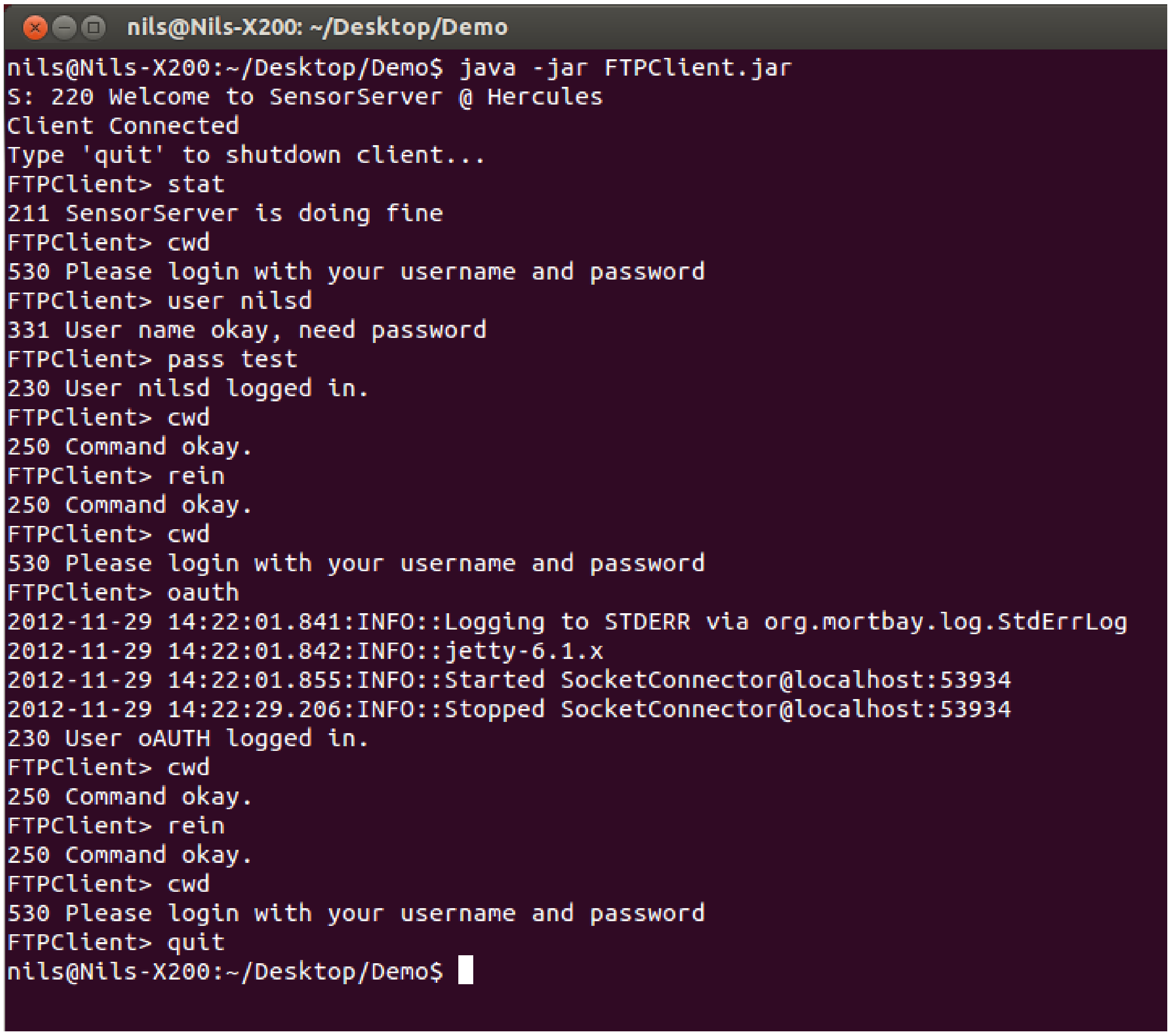
The new OAuth command will not protect the token during the transfer to stay compliance with the original design of the FTP standard. This is identical to the password command which also does not protect the password. Instead the client will have to setup an SSL connection with the server using the "AUTH TLS" command before calling an authentication command. This scenario works and has been tested as part of this project.

SCREEN CAPTURES

The following snapshots show the connection log between the FTP client and the FTP server used for the project.

FTP CLIENT

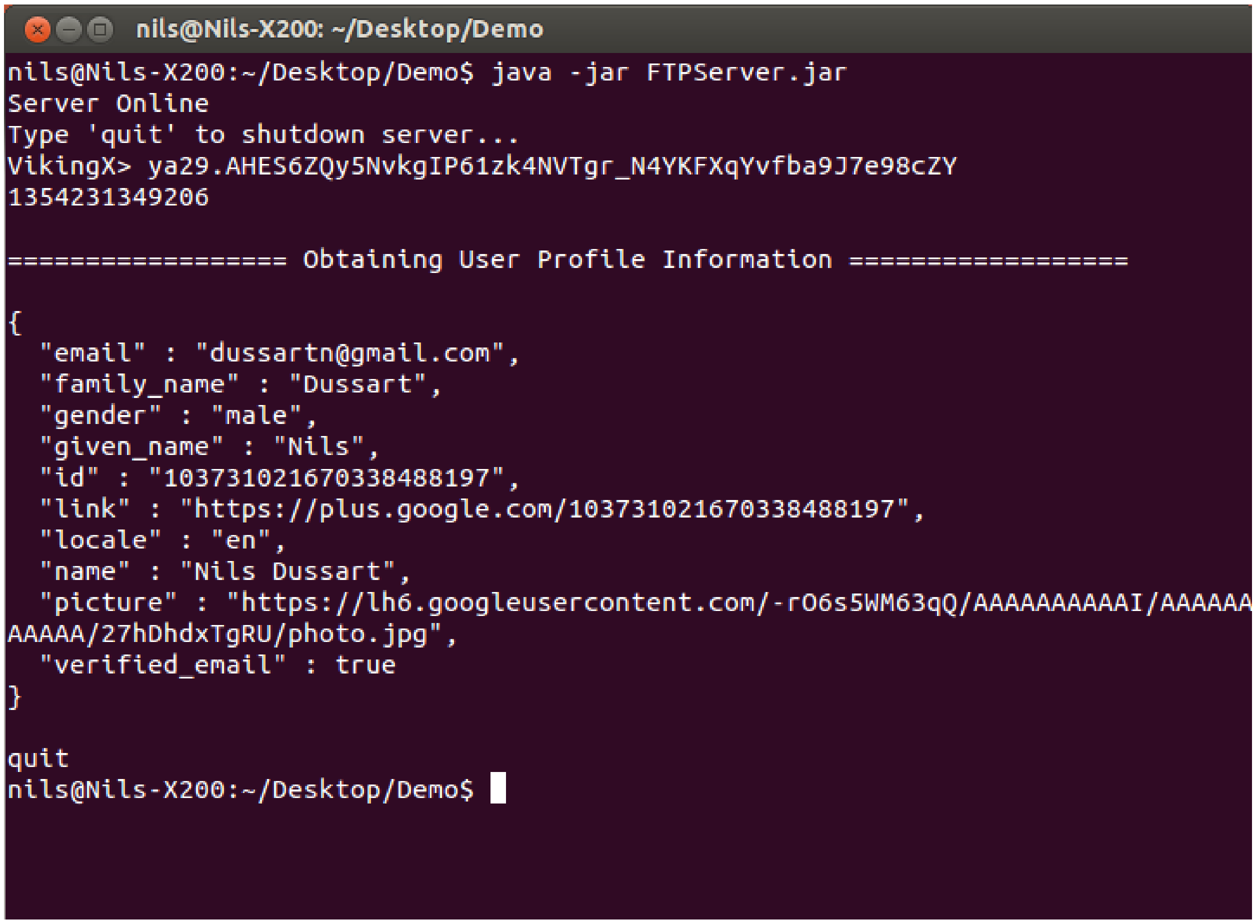
The following screen shot shows the authentication process for the legacy method and the new OAuth command. CWD command allows us to verify in which cases the client has been granted access or not. The REIN command used in these examples is the equivalent of a logoff.



```
nils@Nils-X200:~/Desktop/Demo$ java -jar FTPClient.jar
S: 220 Welcome to SensorServer @ Hercules
Client Connected
Type 'quit' to shutdown client...
FTPClient> stat
211 SensorServer is doing fine
FTPClient> cwd
530 Please login with your username and password
FTPClient> user nilsd
331 User name okay, need password
FTPClient> pass test
230 User nilsd logged in.
FTPClient> cwd
250 Command okay.
FTPClient> rein
250 Command okay.
FTPClient> cwd
530 Please login with your username and password
FTPClient> oauth
2012-11-29 14:22:01.841:INFO::Logging to STDERR via org.mortbay.log.StdErrLog
2012-11-29 14:22:01.842:INFO::jetty-6.1.x
2012-11-29 14:22:01.855:INFO::Started SocketConnector@localhost:53934
2012-11-29 14:22:29.206:INFO::Stopped SocketConnector@localhost:53934
230 User oAUTH logged in.
FTPClient> cwd
250 Command okay.
FTPClient> rein
250 Command okay.
FTPClient> cwd
530 Please login with your username and password
FTPClient> quit
nils@Nils-X200:~/Desktop/Demo$
```

FTP SERVER

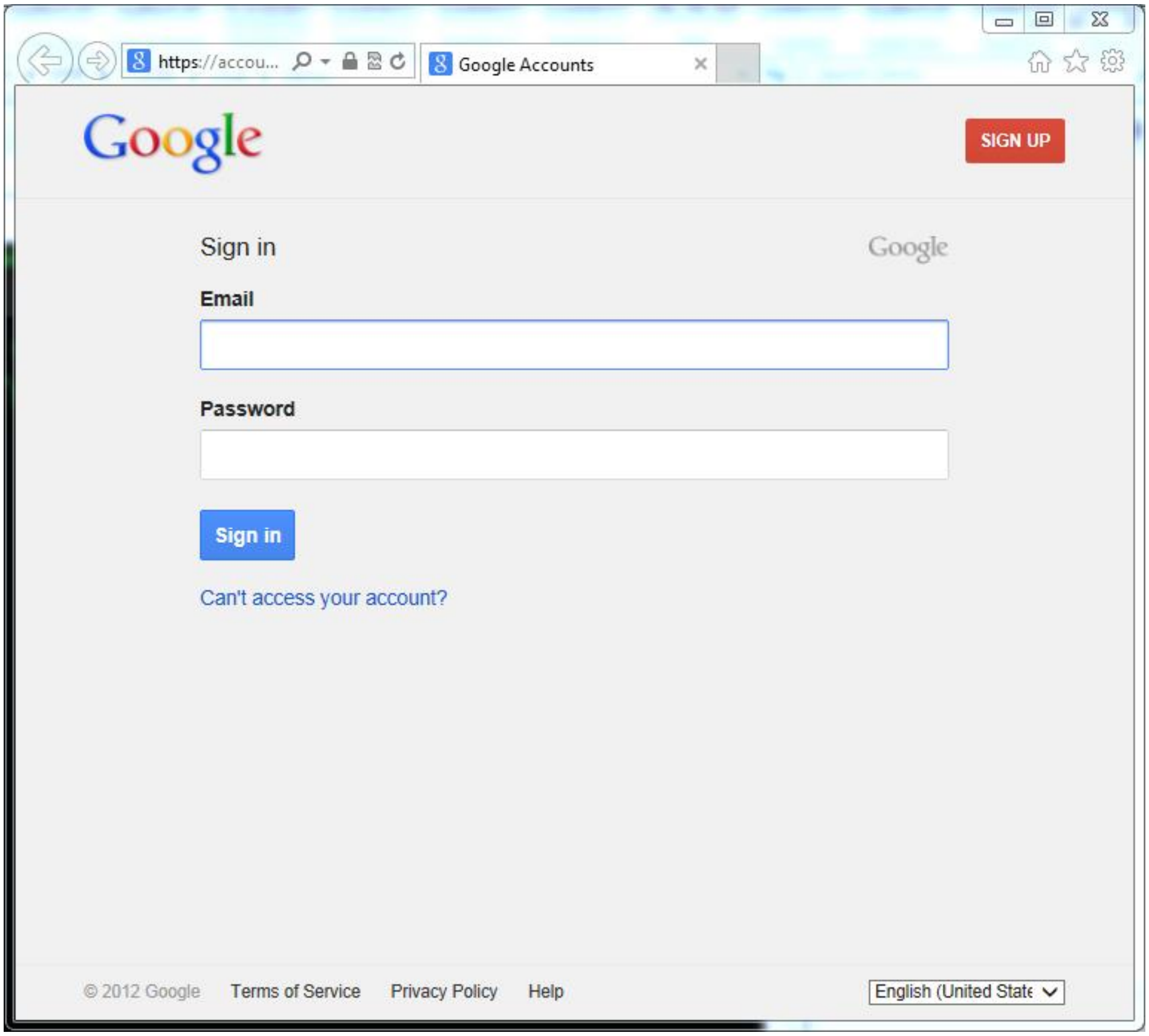
The following screen shot shows the some debugging information displayed in the FTP Server console. In this specific case upon receiving the OAuth command and an associated token, the information is then displayed on the screen. Finally once the token is validated against the Google servers, the user profile information obtained from Google using the token is also displayed at the screen. This information could potentially be used by the FTP Server to make an authorization decision.



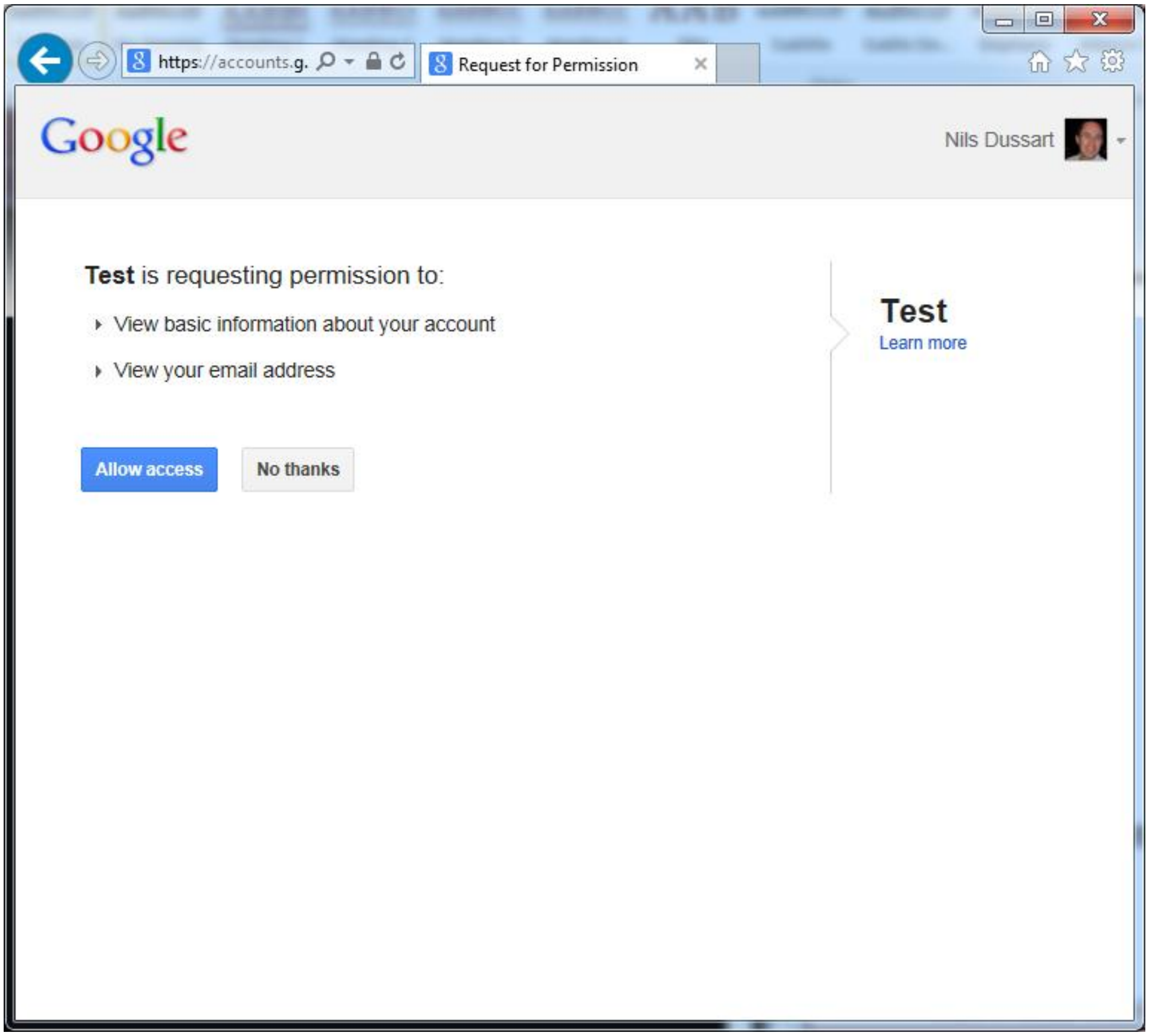
nils@Nils-X200: ~/Desktop/Demo

```
nils@Nils-X200:~/Desktop/Demo$ java -jar FTPServer.jar
Server Online
Type 'quit' to shutdown server...
VikingX> ya29.AHES6ZQy5NvkgIP61zk4NVTgr_N4YKFXqYvfba9J7e98cZY
1354231349206

================== Obtaining User Profile Information ==================

{
  "email" : "dussartn@gmail.com",
  "family_name" : "Dussart",
  "gender" : "male",
  "given_name" : "Nils",
  "id" : "103731021670338488197",
  "link" : "https://plus.google.com/103731021670338488197",
  "locale" : "en",
  "name" : "Nils Dussart",
  "picture" : "https://lh6.googleusercontent.com/-rO6s5WM63qQ/AAAAAAAAAAI/AAAAAA
AAAAA/27hDhdxTgRU/photo.jpg",
  "verified_email" : true
}

quit
nils@Nils-X200:~/Desktop/Demo$
```

GOOGLE AUTHENTICATION SCREEN



GOOGLE DELEGATION AUTHORIZATION SCREEN

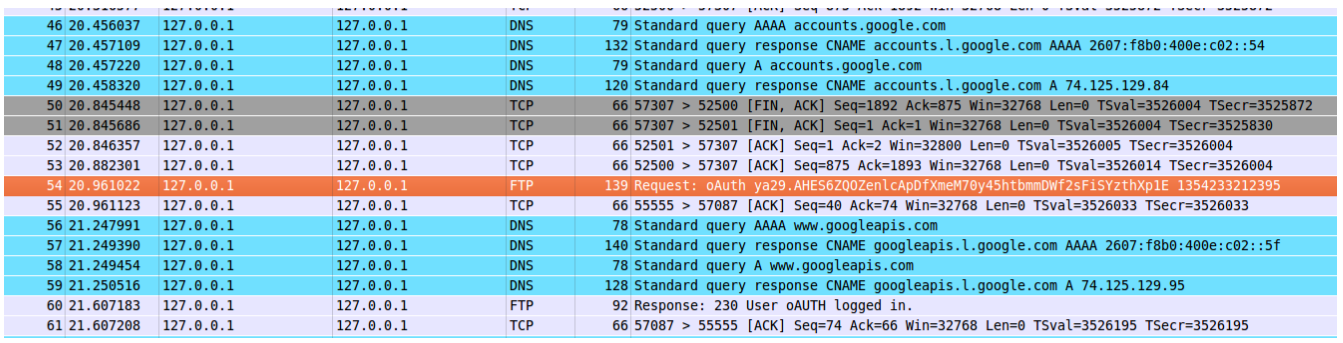


Test is requesting permission to:

- View basic information about your account
- View your email address

NETWORK TRACES

The following snapshot shows a network trace of a standard FTP connection without TLS. As we can see all the control command are in the clear. We can also capture the token which is highlighted in orange in the following trace.



```
46 20.456037  127.0.0.1  127.0.0.1  DNS   79 Standard query AAAA accounts.google.com
47 20.457109  127.0.0.1  127.0.0.1  DNS  132 Standard query response CNAME accounts.l.google.com AAAA 2607:f8b0:400e:c02::54
48 20.457220  127.0.0.1  127.0.0.1  DNS   79 Standard query A accounts.google.com
49 20.458320  127.0.0.1  127.0.0.1  DNS  120 Standard query response CNAME accounts.l.google.com A 74.125.129.84
50 20.845448  127.0.0.1  127.0.0.1  TCP   66 57307 > 52500 [FIN, ACK] Seq=1892 Ack=875 Win=32768 Len=0 TSval=3526004 TSecr=3525872
51 20.845686  127.0.0.1  127.0.0.1  TCP   66 57307 > 52501 [FIN, ACK] Seq=1 Ack=1 Win=32768 Len=0 TSval=3526004 TSecr=3525830
52 20.846357  127.0.0.1  127.0.0.1  TCP   66 52501 > 57307 [ACK] Seq=1 Ack=2 Win=32800 Len=0 TSval=3526005 TSecr=3526004
53 20.882301  127.0.0.1  127.0.0.1  TCP   66 52500 > 57307 [ACK] Seq=875 Ack=1893 Win=32768 Len=0 TSval=3526014 TSecr=3526004
54 20.961022  127.0.0.1  127.0.0.1  FTP  139 Request: oAuth ya29.AHES6ZQOZenlcApDfXmeM70y45htbmmDWf2sFiSYzthXp1E 1354233212395
55 20.961123  127.0.0.1  127.0.0.1  TCP   66 55555 > 57087 [ACK] Seq=40 Ack=74 Win=32768 Len=0 TSval=3526033 TSecr=3526033
56 21.247991  127.0.0.1  127.0.0.1  DNS   78 Standard query AAAA www.googleapis.com
57 21.249390  127.0.0.1  127.0.0.1  DNS  140 Standard query response CNAME googleapis.l.google.com AAAA 2607:f8b0:400e:c02::5f
58 21.249454  127.0.0.1  127.0.0.1  DNS   78 Standard query A www.googleapis.com
59 21.250516  127.0.0.1  127.0.0.1  DNS  128 Standard query response CNAME googleapis.l.google.com A 74.125.129.95
60 21.607183  127.0.0.1  127.0.0.1  FTP   92 Response: 230 User oAUTH logged in.
61 21.607208  127.0.0.1  127.0.0.1  TCP   66 57087 > 55555 [ACK] Seq=74 Ack=66 Win=32768 Len=0 TSval=3526195 TSecr=3526195
```

This shows that the connection should always be protected when transferring a token. The token should be considered a secret.

PROS AND CONS OF THE SOLUTION

The solution provided several advantages:

- User experience is positive since it requires little training on the user behalf. The login screen is a standard Google login screen that most users already know how to use. Also the fact the users can use a commonly used username and password reduces the friction of the authentication process.
- From a development standpoint little expertise is required. All of the OAuth specific calls have been abstracted in a new Java class that totally abstracts the authentication process from the developer.
- The time required from the FTP Server to perform the token validation is minimal.
- The FTP server does not have to maintain a list of password and even more important does not have to secure it anymore. All of this is now maintained and managed by the IdP.
- Significant advantages from a scalability and reliability standpoint. Indeed it is well known that IdPs such as Google and Microsoft have SLAs that are probably higher than what most enterprises achieve for their internal services.
- At the client, the segregation between the secrets and the application itself provide great security benefits.

The solution has several disadvantages:

- OAuth is a framework not a protocol and because of this interoperability does not work well between IdPs. For the application to work with another IdP supporting OAuth, modifications are required to the code because of specific customization in the authentication process.
- Cloud frameworks and protocols such as OpenId and OAuth were initially design for web services and as such, native applications are not primary scenario for those providers.
- The token is a secret that still needs to be protected especially during transport. Developers will have to be train to consider this a security risk that needs to be mitigated.

FUTURE AREA OF INVESTIGATION

Several future areas of investigation have been considered during this project:

- Access Control should be provided by an external service or entity. One of the main benefits of this project is to isolate the application from the user credentials both on the client and the server. On the client, the authentication takes place in the browser process and the token is returned directly to the FTP Client and as such the browser is never in contact with the token. For the server, this means that no passwords need to be managed anymore but on the overhand the authorization still need to take place. This means the server still needs to maintain a list of account (email address) that should be granted access. The logical next step here would be for the server, once the token is validated, to query another service using that same token and query if the user should be granted access. Doing this would allow to segregate not only the authentication from the application but also the authorization.

- It would be interesting to invest more time to identify ways of making the application support more IdPs and protocols. For example, an area of investigation could be to figure out ways of making the same application work with a different IdP using OAuth like Facebook for example. Since the protocol can be customized by providers, there would be value in figuring ways of making the authentication calls more generic.

- More time needs to be spent to find ways of making the solution enterprise ready. For this project all accounts were standard Google accounts but to make this valuable for the enterprise centralized management of accounts would be critical and another area of possible investigation.

LITERATURE REVIEWS/SYNOPSES

RFC959 - FILE TRANSFER PROTOCOL (FTP)

Description of the document

This document is the official specification of the File Transfer Protocol (FTP). It contains all the original commands and expected states that are required to build a compliant FTP client and FTP server.

The document describes 2 specific commands of interest to my project, the USER command and the PASS command. These 2 commands provide a way to pass a user and password argument from the client to the server to provide authentication.

The user name command allows passing a string identifying the user to the server. The user identification is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the control connections are made.

The password command (PASS) allows passing a string which specifies the user's password. This command must be immediately preceded by the user name command.

Specific interest

Interestingly enough the document states the following:

Since password information is quite sensitive, it is desirable in general to "mask" it or suppress type out. It appears that the server has no foolproof way to achieve this. It is therefore the responsibility of the user process to hide the sensitive password information.

The original rfc959 didn't provide a way to protect the password at all. The password was simply transferred in the clear. In the case of my project the token we will pass between the client and server will not need to be protected since it does not contain any secret. Also the token will be signed which protects against alteration.

There are no specific command in the RFC that allows to pass something else than a password. This means we will have to re-use the existing command but find a way to pass a flag to tell the server we will pass a token or create a new non-standard FTP command.

RFC2228 - FTP SECURITY EXTENSIONS

Description of the document

This document defines extensions to the FTP protocol describe in the previous page. These extensions provide strong authentication, integrity, and confidentiality on both the control and data channels with the introduction of new optional commands, replies, and file transfer encodings. The following new optional commands are introduced in this specification:

- AUTH (Authentication/Security Mechanism),
- ADAT (Authentication/Security Data),

- PROT (Data Channel Protection Level),
- PBSZ (Protection Buffer Size),
- CCC (Clear Command Channel),
- MIC (Integrity Protected Command),
- CONF (Confidentiality Protected Command), and
- ENC (Privacy Protected Command).

Specific interest

It is clear from the specification that these commands were not meant to provide a secure way to transfer the password nor provide new methods to authenticate. Instead the document defines authentication as the action of establishing a secure connection between the client and the server and authorization as the action of verifying the user name password to gain access to the file system.

This RFC provides clear security improvements but based on the provided definitions it proposes only to the authentication not the authorization. This means that no improvements are provided as part of this RFC to improve the authorization of users.

Like previously mentioned the transfer of the token in the clear is not an issue since it doesn't contain any secret. This means that this RFC and the commands it provides are not required to implement an authorization model based on a cloud identity provider.

OAUTH WEB AUTHORIZATION PROTOCOL

Description of the document

The document provides a quick historical overview of the OAuth protocol which began as a community effort among numerous companies that provide Internet services. These organizations recognized the need to solve a specific type of identity management problem and developed the first version of a mechanism for doing so. The document describes the use of the protocol in the context of web services and specifically in the scenario where one service needs to interact with another on the behalf of the user.

The goal was to avoid having to ask user to turn their credentials to every service that needed to interact with a specific service. Turning over credentials gives that service full access to the user's credential without any limitations. The OAuth protocol addresses this problem by providing an alternative mechanism through which users can authorize specific actions, and only those specific actions, without giving unrestricted or permanent access.

Specific Interest

Unfortunately the document seems to be mostly about the OAuth delegation model. No mentions are made about the authentication itself and how it can be achieved. The article does mention that the protocol is indeed used by Google, Yahoo, Facebook, and Twitter which makes it a protocol of choice for my project. I will still research Open ID since it seems it is still used by different major services including Google and Yahoo and understand what the clear difference where.

Interesting in the articles was the definition provided for client, authorization server and access token which will be useful for my research.

Authorization server — the service that verifies the resource owner's credentials and performs the authorization checks. This is often the same as the resource server (as it is in both examples), and is always in its trust domain.

Access token — a piece of data the authorization server creates that lets the client request access from the resource server. This is the authorization credential the client will use in place of the resource owner's own credentials.

Client — the service asking for authorization. Note that this is the OAuth transaction's client, not the end user's client program.

INSIDE THE IDENTITY MANAGEMENT GAME

Description of the document

This document describes the identity space as fragmented and incoherent. Multiple standards exists for managing authentication and authorization but so far no global solution exist from identity standpoint that scale all the necessary needs of the Internet.

As the web is evolving so are the need of new generation of web application such cloud-based services, social websites and mobile platforms. The document starts by providing an historical view of the identity space. Different standards exists but the main ones are:

OASIS supplies SAML and Web services (WS-*) as well as several other standards

IETF provides several relevant standards such as HTTP, TLS and OAuth.

The open source community is all also contributing to this space. The best example is OpenID.

There are also a multitude of other projects in both the private and public sector with specific requirements and needs.

The document then provides descriptions of several key protocol like OpenID and OAuth. The high level goal of both protocols is to let users create and asset an identity that is accepted by a wide variety of service. This would provide SSO since the user only needs to remember one password. The document define some key roles:

The end user (data subject)

The service provider (relying party, or RP)

The identity provider (IdP)

Specific interest

The document describes the limitation and current issues associated with OpenID and OAuth. Several implementation of OAuth exists which creates implementation issues. OpenID while still being worked seems to be less and less used. On the other hand, no mention is made in this document of how oAuth is used in authentication scenario and how credentials are passed to the identity provider.

It seems that OAuth is clearly the protocol I should use for my project but unfortunately this document is once again focus specifically on the delegation scenario between 2 web services which is not aligned with the requirements of my project.

THE NATIONAL STRATEGY FOR TRUSTED IDENTITIES IN CYBERSPACE

Description of the document

The document is making the case that password need to be retired since password centric attacks are increasingly common. The author also explains that reliance on weak password technology has been a growing attack vector that threatens to erode confidence in the online world. Human factors are also contributing to his arguments since users have a tendency to use the less complex password possible. Finally passwords don't scale well since the average Internet user in the USA needs to keep track on average of 30 passwords.

The article explains that alternative technologies are needed to replace passwords as the primary method of online authentication. An example is made of the USA DoD and their implementation of the Common Access Card (CAC) to securely manage identities and provide highly secure credentials.

Finally the author concludes by explaining the US government's National Strategy for Trusted Identities in Cyberspace (NSTIC) focuses on working in partnership with the private sector to facilitate the adoption of online identification technologies that are secure and trusted.

Specific interest

This document confirms that Internet and government based identity providers provide large benefits from both a security standpoint and from a user experience standpoint. While the document doesn't provide specific details for the technical implementation of my project, it confirms the assumption that Identity based providers are a technology that will continue to be used widely for years to come. Finding ways for legacy application to interact with those new solutions will allow leveraging their advantages.

TAKING STEPS TO SECURE WEB SERVICES

Description of the document

This document describes at the high level the standards and protocols available to organization to protect their services. The article covers securing web services but only from an enterprise point of view. It states that SOAP functions as transport mechanism for XML messages by letting programs running on one OS communicate with programs running on another, using HTTP and XML to exchange data. However, the World Wide Web original SOAP version provided no security.

To feel that gap, the Oasis organization created a new standard called SAML. "SAML defines a vendor-neutral way to express security information in an XML format". "It defines the schemas for the structure of documents that include information related to user identity and access or authorization rights." Basically SAML defines a specific token format that be used to prove identity, contain identity information and other security attributes. SAML can also indicate the authentication that must be used with a message, such as a password, Kerberos ticket, hardware token or X.509 certificate. Finally SAML allows single sign-on, since it possible for a web services to transfer the token to another site for authentication as long as the other site recognize the same authentication authority.

In addition to defining a token format, the Oasis organization created a set of new protocols to allow Web services to work with several different security models via SOAP extensions. These protocols allow applications to attach security data to the headers of a SOAP messages. These protocols, for example, provide ways of encrypting or signing messages including SAML tokens. This is critical in ensuring that the token was not altered. One specific protocol is worth mentioned for my project, WS-Trust provides a model for establishing both direct and brokered trust relationships for secure interaction. It basically defines how to acquire a token, validate it and exchange it.

The article concludes that one major obstacle for the adoption of those protocols and standards is the fact they rely on a PKI infrastructure or some enterprise directory services. This is not ideal consumer based scenarios.

Specific interest

This document was very interesting for my project since it provides the overview of how enterprise web services are enabling authentication using the specific Oasis protocols and standards. The goal of my project was mainly to interact with a consumer based identity provider such as Google but with time the difference between enterprise directory and public identity provider seems to get blurrier and blurrier. I will look into those standards especially SAML tokens and WS-Trust as possible solutions to my project.

THE VENN OF IDENTITY: OPTIONS AND ISSUES IN FEDERATED IDENTITY MANAGEMENT

Description of the document

This article provides an explanation of what a federation is and provides details about the architectural challenges and the associated security and privacy issues. The main goal of federation is to provide SSO to end users while using services or resources located elsewhere. While there are many benefits, there are significant security risks because the architectures share information across domains.

The author then describes several attacks that are plausible with federation. For example, he describes that user authentication is one of those weak links in the identity chain. Most site relies on username and an associated password pair which are well known to be weak and susceptible to phishing attacks.

The author provides and other example based on the fact that for web services provider's federated identity is less expensive than implementing a high-quality authentication infrastructure because it offloads the authentication task to an identity provider. However, SSO can magnify the costs of a stolen password because it expands the scope of malicious activity. Most SSO protocols offer ways to mitigate this risk. For example, they might limit to a minute or less the valid lifetime of the security token; some protocols also offer a single logout (SLO) feature that offers users near-simultaneous sign-out of all SSO-accessed Web sites.

The article then list several Architectural challenges:

- Identity provider discovery: services need a method to discovery an identity provider that they should trust and use. The configuration might be static or manual in which case the user must specify the location of the identity provider.
- Identifier schemes: this describes how to represent the same identity across different identity providers.

There are currently 3 main federation protocols, SAML, Open ID and Info Card. The author provides a quick summary of the protocols with some pros and cons.

Specific interest

This article was interesting from a thread modeling standpoint. The different threats it describe will influence my choice of protocol for my project. Man in the middle attacks and replay attacks are clearly possible and should be considered as part of my design.

The article doesn't provide enough information to make design decision on Open ID versus SAML. While it lists some pros and cons the article does not go deep enough in the reasons why one group of users might use Open ID while the other might use SAML.

IDENTITY MANAGEMENT AND TRUST SERVICES: FOUNDATIONS FOR CLOUD COMPUTING.

Description of the document

This specific article describes how a campus or an educational organizational should deal with Identity Management & Trust Services. The author makes a case that IT organizations will move from providing IT services locally to becoming an integrator of IT services. Those services will be provided locally but also outside of the institution. Because of this, the article states that institutions should plan for shared services and must understand the essential role that identity management and trust services play in making integration possible.

The challenge is to integrate these different services in a coherent and effective manner. Issues such as authentication, access control, and the user experience when moving from one service to another need to be accounted for to be successful. The author then describes that institution have been missing a way to integrate these external service offerings and offer a comprehensive approach that should focus on the following three activities:

1. Developing an identity management system;
2. Creating a standard set of attributes for each person;
3. Enabling external access through a federation.

Completing those 3 specific activities will allow institutions to develop a common standard to authentication that will address their needs.

Specific interest

While this article didn't have a direct impact on my project, it does bring interesting background on the subject that I'm researching. The author makes a good case at defending the fact that each institution should use a common set of standards to enable a common identity management for their services. The article mentions briefly the Oasis set of standards and more specifically SAML which is on the protocols I am considering using.

SYSTEMATICALLY BREAKING AND FIXING OPENID SECURITY: FORMAL ANALYSIS, SEMI-AUTOMATED EMPIRICAL EVALUATION, AND PRACTICAL COUNTERMEASURES.

Description of the document

This document does an in depth analysis of the OpenID protocol and provides a threat modeling exercise. The analysis reveals that the protocol does not guarantee the authenticity and integrity of the authentication request. The results of the investigation suggest that many OpenID-enabled websites are vulnerable to a series of cross-site request forgery attacks. The author proposes as part of the article some simple and scalable mitigation technique for OpenID-enabled websites, and an alternative man-in-the-middle defense mechanism for deployments of OpenID without SSL.

Specific interest

This article provides a good overview of OpenID and its limitations and weaknesses. Since OpenID is one of the protocols I'm contemplating for my project, the article did provide a great of information that I've found useful especially in the first parts of the article.

A BILLION KEYS, BUT FEW LOCKS: THE CRISIS OF WEB SINGLE SIGN-ON.

Description of the document

This research document does a deep analysis into the different solutions that provides WEB SSO on the web and why there are not more adopted widely. The paper provides a technical description of several solution offered currently on the Web of interest to us is the mention of federated scenarios through SAML and the OpenID protocol.

The article then lists the incentives that are currently lacking to make this attractive to the different players in this field:

- Lack of incentives for Relying Parties
- Lack of driving forces from users.
- Lack of driving forces from Identity providers

The author provides several recommendations to address the lack of adoption of those technologies. One of them is to build a business incentive for relying parties to trust identity providers. The second is improve the user experience when using such technologies.

Specific interest

This document provided with very good insight at the current space and as to why some of the technologies I want to use as part of my project are not leveraged more. The recommendation to build identity support into the browser is a very good idea and is well aligned with my goal of analyzing the user experience when leveraging my proof of concept.

BIBLIOGRAPHY OF YOUR LITERATURE REVIEWS

Geer, David. "Taking Steps to Secure Web Services." *Computer* 36.10 (2003)*/z_ebsco_a9h/.* Web.

Grant J.A. "The National Strategy for Trusted Identities in Cyberspace: Enhancing Online Choice, Efficiency, Security, and Privacy through Standards." *IEEE Internet Comput IEEE Internet Computing* 15.6 (2011): 80-4. */z-wcorg/.* Web.

Leiba B. "OAuth Web Authorization Protocol." *IEEE Internet Comput IEEE Internet Computing* 16.1 (2012): 74-7. */z-wcorg/.* Web.

Lynch L. "Inside the Identity Management Game." *IEEE Internet Comput IEEE Internet Computing* 15.5 (2011): 78-82. */z-wcorg/.* Web.

Maler E., and Reed D. "The Venn of Identity: Options and Issues in Federated Identity Management." *IEEE Secur.Privacy IEEE Security and Privacy* 6.2 (2008): 16-23. */z-wcorg/.* Web.

Suess, Jack, and Kevin Morooney. "Identity Management and Trust Services: Foundations for Cloud Computing." *EDUCAUSE Review* 44.5 (2009): 24,26, 28, 30, 32, 34, 36, 38, 40, 42. */ericresearch/.* Web.

Sun S.-T., et al. "A Billion Keys, but Few Locks: The Crisis of Web Single Sign-on." *Proc.New Secur.Paradigms Workshop Proceedings New Security Paradigms Workshop* (2010): 61-71. */z-wcorg/.* Web.

Villegas, D., et al. "Cloud Federation in a Layered Service Model." *Journal of Computer and System Sciences* 78.5 (2012): 1330-44. */z-wcorg/.* Web.

TECHNICAL DOCUMENTATION

RFC2228 - FTP Security Extensions - http://tools.ietf.org/html/rfc2228

RFC959 – File Transfer Protocol (FTP) - http://tools.ietf.org/html/rfc959

[OAUTH-WG] JSON Web Token (JWT) Specification Draft - http://www.ietf.org/mail-archive/web/oauth/current/msg04407.html

SAML 2.0 Session Token Profile Version 1.0 - http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/saml-session-token-v1.0.html

The OAuth 2.0 Authorization Framework - draft-ietf-oauth-v2-31 - http://tools.ietf.org/html/draft-ietf-oauth-v2-31

APPENDICES/ADDITIONAL DOCUMENTATION

Federated Login for Google Account Users - https://developers.google.com/accounts/docs/OpenID

UX research on Desktop Apps using federated login and/or OAuth - https://sites.google.com/site/oauthgoog/UXFedLogin/desktopapps

SAML Single Sign-on with Desktop Apps – Enabled by OAuth - http://blog.superpat.com/2009/11/12/saml-single-sign-on-with-desktop-apps-enabled-by-oauth/

SAML Single Sign-On (SSO) Service for Google Apps - https://developers.google.com/google-apps/sso/saml_reference_implementation

SOURCE CODE