David Krovich
Paper on OAuth2
CPE538
Summer 2016

Identification, authentication, and authorization are becoming increasing problems as the internet expands into every facet of our existence. The burden of attempting to maintain secure account credentials across a number of resources is a real challenge with a traditional username and password combination. If the same password is used on multiple sites and that password is compromised then all sites using that password are also at risk. If a user maintains unique passwords on different sites then the burden of managing those passwords becomes a problem. Where are those passwords stored? Is that a secure location? Are passwords stored with some sort of encryption? This whitepaper will examine some of the problems associated with identification, authentication, and authorization and explore OAuth2 as a potential solution to this problem.

There are many examples of the failures of today's password management strategies. One famous example is the "Mugged in Madrid" compromise associated with the wife of journalist James Fallows. The wife of James Fallow had her gmail account compromised. The hacker sent email to all her contacts saying that she had been mugged in Madrid and had lost all her credit cards and money and to wire her money. Furthermore the hacker changed all the recovery methods associated with the account. Additionally, the hacker completely deleted and removed from the trash every single email that had been associated with the account. The story eventually has a somewhat happy ending though. Ultimately the account was returned to the rightful owner and the emails were eventually recovered.

Another famous example relates to the compromise of Sara Palin's email account during the 2008 Presidential election. David C Kernell of Tennessee was indicted by a federal grand jury for "intentionally accessing without authorization the e-mail account of Alaska governor Sara Palin." Kernell obtained access by allegedly using a password reset feature and correctly answering security questions associated with the account. Kernell was convicted and spent 1 year in Federal prison and 3 years on probation.

A third example is the story of journalist Mat Honan. In a span of one hour his entire digital life was destroyed. Hackers used a daisy chain approach to ultimately compromise the amazon, apple id, gmail, and twitter accounts. The hackers these accounts to post inappropriate messages on twitter claiming to be Mat Honan. Additionally, hackers used the apple accounts to remotely erase his iphone, ipad, and Macbook computer. He was unable to recover completely from the attack, and lost precious pictures of his daughter that can never be restored.

In recent years enhancements have taken place to help improve username and password security. A very well known initiative was started by Google and is referred to as 2sv (Two Step Verification). With 2sv enabled, there is a two step process when using a new device to access a gmail account. When a new device attempts to log into the gmail account, a verification code is sent via another mechanism to help verify the identify. This secondary device can be a voice call, a SMS message, or email to an alternate email address. 2sv has proven to be an effective mechanism for protecting password accounts and could have potentially thwarted the hacking activities listed earlier in this paper.

While 2sv helps alleviate a lot of problems with password security there are situations that it doesn't apply. As the web becomes increasingly connected it is common to have one site access another on behalf of the account owner. A common way this accomplished is to store username and password credentials on site A, and then site A would use those credentials to access site B. One common example of this is the Bank of America website. Bank of America provides a portfolio feature where you can view all of your account information in one location. In order to do this you must provide username and password credentials for each site and store those credentials on the Bank of America site. This type of strategy is sometimes referred to as the password anti pattern.

The password anti pattern has a number of problems. The first is that there is no way to distinguish between the user and some agent acting on the user's behalf. Since the only identifying information is a username and password there is no way to differentiate who or what is accessing the resource. A second problem is due to the inability to identify the user there is no way to give our a granular level of access. With a username and password combination access is all or nothing. Either the user is let in and gains full access or not. A third problem is that storing passwords on remote servers is typically a bad idea. If that remote server is compromised, or that account on the remote server is compromised, a hacker could potentially gain access to all the other accounts. A fourth problem is that shared passwords are difficult to revoke. In order to change a password it must be updated everywhere that password is stored. With so many problems, it is clear to see a need for a better system of authentication other than the classic username and password combination.

Fortunately there are efforts to create a much better system for the ever evolving and interconnected web. OAuth2 is one such effort. It was invented for the specific purpose of allowing users to log in and control access to accounts without having to share a username and password. A relying party (RP) is a site that would use OAuth2 to offload authentication to an Identity Provider (IDP). An example of this interaction is Huffington Post which uses Facebook authentication for the comment section. In this scenario Huffington Post is the RP and Facebook is the IDP.

Using OAuth2 a site can offload authentication to an IDP such as twitter, google, or yahoo. This can simplify some of the tasks associated with hosting a site such as password resets, account recoveries, etc. All of this is passed to the IDP. Additionally, OAuth2 facilitates authentication without using the password anti pattern. It allows for access tokens to be revoked without the need to change the password with the IDP.

How does OAuth2 accomplish secure authentication without using a username and password? To understand OAuth2 the first thing is to understand the various roles and their function in the OAuth2 framework. A central role in OAuth2 is the Resource Owner. This is typically a human that is the owner of a particular account. The resource owner typically stores resources on a Resource Server (RS). An Authorization Server (AS) is another role that handles verifying the identity of a user and issues access tokens. The last role is the client role which is typically the application or RP that the resource owner is attempting to access. In many implementations the AS and RS functionality are both handled by a single service API provided by the IDP.

OAuth2 access control comes down to the management of tokens. There are two primary activities in OAuth2. The first is acquiring a token and the second is using a token. OAuth2 has 4 different flows that exist to handle the management of tokens. The first flow is the Authorization Code Grant flow. This is the most common OAuth2 flow that is used. This is primarily what users see when they log into news sites that make a user authenticate against a IDP before they can post comments. A second flow is the Implicit Grant flow. This is similar to the Authorization Code Grant flow except the difference is what is returned from the AS. In the implicit code grant the access token is sent directly to the user agent instead of the client itself. This would be useful if the client for some reason can't keep the token secure. Another flow is the Resource Owner Credentials Grant. With this flow the client collects the resource owner's username and password and uses them to get access tokens. Unlike the password anti pattern though these credentials are not stored. They are only used to gain access tokens and from that point on the access tokens are used. The last flow that will be mentioned is the Client Credentials Grant. These are typically credentials that are associated with a specific application and are given when a developer registers their application with a particular IDP. These credentials are often required for an RP to even use an IDP.

Once a token is received using them is straightforward. Tokens are presented to the IDP and the IDP will determine whether or not the tokens are valid. If the token is valid and has appropriate access rights then access will be granted to whatever resource is being requested. If the token is not valid access will be denied. Whenever access to a resource is needed the token is presented. Unlike passwords, tokens can revoked without impacting any other tokens. This allows for a granular level of control that the Resource Owner can use to manage access.

Implementation details of OAuth2 can be up to the IDP but from experience with google and twitter there seems to be some common steps to take. First the application developer must sign up with the developer service on the IDP. This usually involves having an account with the IDP. Next, the developer has to register their account with the IDP. The application must have a name, a URL location, and a callback URL. Upon registration of of this application the IDP will present identification and secret keys to be used by the application to identify itself to the IDP. This is the Client Credentials Grant flow. The developer needs to build these credentials into their application. Once the application is registered and client credentials are handled the developer needs to add a link that will trigger the IDP authentication. This link will visit the IDP and the user will be prompted to log in if they are not currently logged in. If they are logged in they will be prompted as to whether or not they want to grant access to the RP to generate a token. If RP was already authorized and the Resource Owner was already logged in then they would be sent back to the RP. The developer needs to write code to handle this callback from the IDP and figure out what user information they will store locally about the user. Once that is handled the Authorization Code Grant flow would be fully implemented.

Oauth2 is certainly gaining traction as a better way of handling authentication. A recent study set out to determine the most popular IDPs and the number of RPs that relied on them. The study also determined how many of the top IDPs were using OAuth2 in their framework. Among the top 10 IDPs eight use OAuth2 as their primary protocol and nine use OAuth2 for some of their relationships. (Vapen) Here are the top 5 IDPs from that study. The first was facebook.com with 1,293 RPs. The second was twitter.com with 378 RPs. The third was qq.com with 278 RPs. The fourth was google.com with 250 RPs. And the fifth was yahoo.com with 141 RPs.

REFERENCES

Wang, Yating, Ing-Ray Chen, and Ding-Chau Wang. "A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges." *Wireless Personal Communications : an International Journal*. 80.4 (2015): 1607-1623. Print.

Windley, Phillip J. "Api Access Control with Oauth: Coordinating Interactions with the Internet of Things." *Ieee Consumer Electronics Magazine*. 4.3 (2015): 52-58. Print.

Grosse, E, and M Upadhyay. "Authentication at Scale." *Ieee Security &amp; Privacy*. 11.1 (2013): 15-22. Print.

Caton, Simon, Christian Haas, Kyle Chard, Kris Bubendorfer, and Omer F. Rana. "A Social Compute Cloud: Allocating and Sharing Infrastructure Resources Via Social Networks." *Ieee Transactions on Services Computing*. 7.3 (2014): 359-372. Print.

Vapen, Anna, Niklas Carlsson, Anirban Mahanti, and Nahid Shahmehri. "A Look at the Third-Party Identity Management Landscape." *Ieee Internet Computing*. 20.2 (2016): 18-25. Print.

https://www.justice.gov/archive/opa/pr/2008/October/08-crm-910.html

http://www.commercialappeal.com/news/david-kernell-ut-student-in-palin-email-case-is-released-from-supervision-ep-361319081-326647571.html

http://www.theatlantic.com/magazine/archive/2011/11/hacked/308673/

http://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/

https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow