Attack and Defense the OAuth based SSO systems

Jianjun Ye, Yu Ding, Tongxin Li, Huilin Zhang, Xinhui Han
Peking University
Beijing, China
{yejianjun, dingelish, litongxin, zhanghuilin, hanxinhui}@pku.edu.cn

ABSTRACT
In this paper we show our findings in web based single-sign on (SSO) systems. We show how to steal the access tokens in SSO systems by a motivating example. Then we explain how to detect this kind of attacks automatically.We analyze the current mitigation techniques and their weakness. At last, we give a solution to this kind of attacks.

Keywords
OAuth; Redirect Mechanism; Single Sign-On;

1. INTRODUCTION
More and more sites apply web based single-sign on (SSO) techniques to provide better user experiences. With OAuth based SSO authentication, users can login to *relying parties(RP)* with the credentials of *Identity Providers(IdP)*. SSO authentication releases the burden of remembering different passwords and reduces the risk of password leakage.

Redirect mechanisms are widely used in recent sites, such as forums, search engines and advertisement systems. Generally, redirect page redirects the user's browser to a given URL. In a forum, when the user wants to reply a thread before login, the browser will jump to the login page with a `redirect` argument. After login succeeds, the browser is redirected to the previous thread automatically. The redirect mechanism makes it convenient for both users to post/reply and also reduces the complexity of the sites.

Vulnerability exists in many OAuth based SSO authenticate systems. The RP redirect the user's browser to the IdP login page with a `redirecturl` argument. After authentication, the user's browser will return to the url indicated by the `redirecturl` argument with security tokens. In most cases, the IdP does not verify the redirect url for security. Thus the attacker can tamper the redirect url and steal the security tokens.

Previous works seldom focuses on the redirect mechanism. Adam et al. define the Execution-After-Redirect attack [1] and develop a system to automatically discover this kind of vulnerabilities. San-Tsai et al. analyze the implementation of SSO systems and reveals many security issues [4]. However, these papers do not concern about the token leakage in OAuth based SSO systems.

In this paper, we give an example to reveal the token leakage vulnerability at first. Then we analyze the root cause of this vulnerability and show the design of an automatic vulnerability detection tool. Also we give a solution to eliminate this kind of vulnerabilities. TO sum up, our contributions are:

- We discover a kind of token leakage vulnerabilities in OAuth based SSO systems.
- We show the design of a vulnerability discover tool which can semi-automatically detect vulnerable sites.
- We analyze the root cause of this kind of vulnerabilities and give the solution to improve SSO system security.

2. A MOTIVATING EXAMPLE
Figure 1 shows the normal authentication in (a) and the attack in (b). When a user wants to visit vul.com and looks at his own profile, he first visit the profile page (step 1 in (a)). Then vul.com finds out that he needs to login and sends an auth request (step 2 in (a)). The user then chooses to login with FB account and navigate to auth.com to login (step 3 in (a)). The URL of the login page is called 'key URL' , which contains two redirect URLS. The first return URL indicates the RP and the second return URL indicates the final page after the authentication. Next, after login succeeds, the user's browser returns to the first redirect URL with session token (step 4 in (a)). Finally, the user visit the profile page with session token.

Figure 1(b) reveals the attack. First the attacker makes victim to visit a malicious 'key URL' (step 1 in (b)). This can be done by variety of ways such as spamming. Then the victim's browser will be redirected to auth.com for authentication (step 2 in (b)). Next, the vulnerable redirect page in vul.com redirects the victim's browser to the malicious site with security token (step 3 in (b)). The attacker uses the malicious site to get the security tokens and gain access to the victim's accounts.

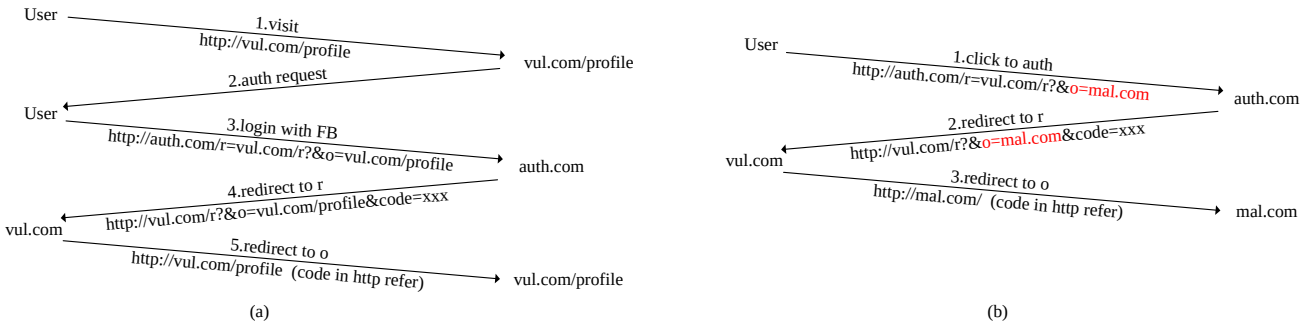

Figure 1: (a) shows the OAuth based SSO authentication progress. (b) shows the attack. In (b), the attacker tamper the first `returnurl` with a malicious URL. After login, the user's browser will visit the malicious URL with security token.

3. DETECTING OAUTH-REDIRECT VULNERABILITIES

There are two return URLs in the whole attack: the first return URL which indicates the RP page, and the second return URL which indicates the final target. To trigger an attack similar to 1(b), there is only one prerequisite: RP can redirect to arbitrary page with security token. To detect such kind of vulnerabilities, we only need to check if the login page of RP can redirect to arbitrary URL with security tokens.

Some mitigation techniques has been deployed in some sites such as sohu.com. In the SSO system in sohu.com, there is no such key URL containing two return url at a time. Instead, the login page invokes `set-cookie` API to store the return page in cookie. After login succeeds, the login page fetch the return URL from the cookie and redirect the user's browser accordingly. In this case, the attacker needs to tamper the first redirect URL (RP url) to achieve the same goal. To detect such kind of vulnerabilities, we need to check if the IdP can redirect to arbitrary URL with security tokens.

Though it is easy to launch such attacks, it is difficult to automatically discover such vulnerable sites. Modern sites have different structures and login pages are also various from each other. So it is difficult to automatically locate such login pages in RP and IdP. We use a semi-automatic approach to overcome the difficulty. First we use web crawler to collect potential RP pages. This is done by simple regular expression matching. Second, we manually check if the collected RP pages are real RP pages. Next, we automatically verify if the RP and IdP can redirect to arbitrary pages. Finally, we manually examine if the arbitrary redirect can be exploited.

4. NEW MITIGATION STRATEGY

Current mitigation techniques can be classified into three classes: browser defense/RP defense/IdP defense. Chrome Safe Browsing [2] use a blacklist to protect user from being redirect to malicious URLs. The blacklist is dynamically updated every day. On RP side, sohu.com invokes 'set-cookie' APIs to reduce attack surface. Freewheel company uses dynamically updated blacklist in the servlet to prevent the user from being attacked. On IdP side, facebook.com has a whitelist to ensure that the security token can only be returned to trusted sites [5]. However, we think that the blacklist based approach can be bypassed by modern URL polymorphic techniques such as URL shorten tools [3] and the whitelist based approach lacks of scalability.

Our mitigation technique is quite simple. We suggest that the IdP should return to a fixed page in RP. The fixed page should be a key part of the SSO protocol. Also, the RP should only redirect to the page in the same site.

5. CONCLUSION

In this paper, we discuss a new kind of attack in OAuth based SSO systems. The root cause of the vulnerability is the lack of URL check on RP side and the IdP side. We give an example to show the attack step by step. Also we show how we detect such kind of vulnerable sites semi-automatically. To this end, we give a new mitigation strategy, which can be used in practical OAuth systems.

6. REFERENCES

[1] A. Doupé, B. Boe, C. Kruegel, and G. Vigna. Fear the ear: Discovering and mitigating execution after redirect vulnerabilities. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 251–262, New York, NY, USA, 2011. ACM.

[2] Google. Chrome safe browsing. `http://www.google.com/transparencyreport/safebrowsing/`.

[3] Google. Google url shortener. `https://goo.gl`.

[4] S.-T. Sun and K. Beznosov. The devil is in the (implementation) details: An empirical analysis of oauth sso systems. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 378–390, New York, NY, USA, 2012. ACM.

[5] C. Warren. Another security flaw gets the heartbleed treatment, but don't believe the hype. `http://mashable.com/2014/05/02/oauth-openid-not-new-heartbleed/`.