

Explicating SDKs: Uncovering Assumptions Underlying Secure Authentication and Authorization

Rui Wang^{1*}, Yuchen Zhou^{2*†},
(*Lead authors, †Speaker)

Shuo Chen¹, Shaz Qadeer¹, David Evans² and Yuri Gurevich¹

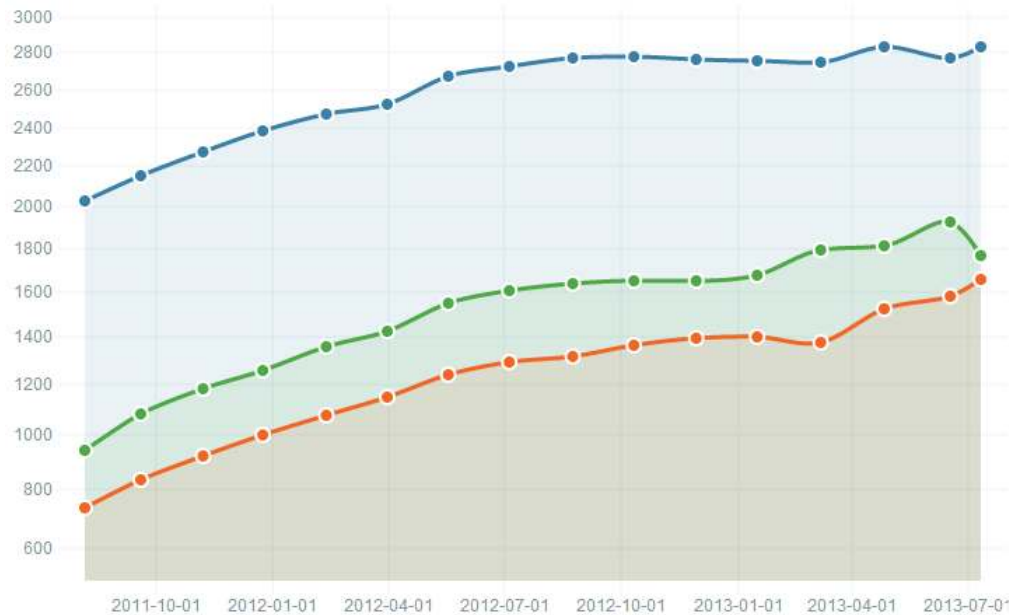
¹Microsoft Research and ²University of Virginia

Microsoft
Research



Social SDK Usage

Statistics for websites using Social SDK technologies



Switch Chart Data  

Top 10k Sites

Top 100k Sites

Top Million Sites

Top 3 Legend

[Unselect All](#)

- Facebook for Websites
- Facebook SDK
- Twitter Platform

We are still building out our sub-categorization lists, tell us if we are missing something here.


[View Pie Chart](#)


chart source: [builtwith.com](#)

Most modern apps are empowered by online services.

Single Sign-On Service

Log in to Pinterest ×

 **Log in with Facebook**

 **Log in with Twitter**


Email Required


Password Required


Forgot your password? Log in


Sign up now


Use a connected account


 **Log in with Facebook**


 **Log in with Twitter**

 **Log in with Google**

 **Log in with AOL**

 **Yahoo**

 **LinkedIn**

 **Hotmail**

Log in to The Huffington Post so you can comment, share, and get your news by email. ×

Username or email

Password [I forgot my password](#)

Log In [Having trouble logging in?](#)

Remember me

[...or create an account](#)

Constructing a URL to the OAuth Dialog

To invoke the OAuth Dialog, redirect the user's browser to a URL of the form:

```
https://www.facebook.com/dialog/oauth/?
  client_id=YOUR_APP_ID
  &redirect_uri=YOUR_REDIRECT_URL
  &state=YOUR_STATE_VALUE
  &scope=COMMA_SEPARATED_LIST_OF_PERMISSION_NAMES
```

Parameters

The OAuth Dialog supports the following parameters which may be passed in the URL string:

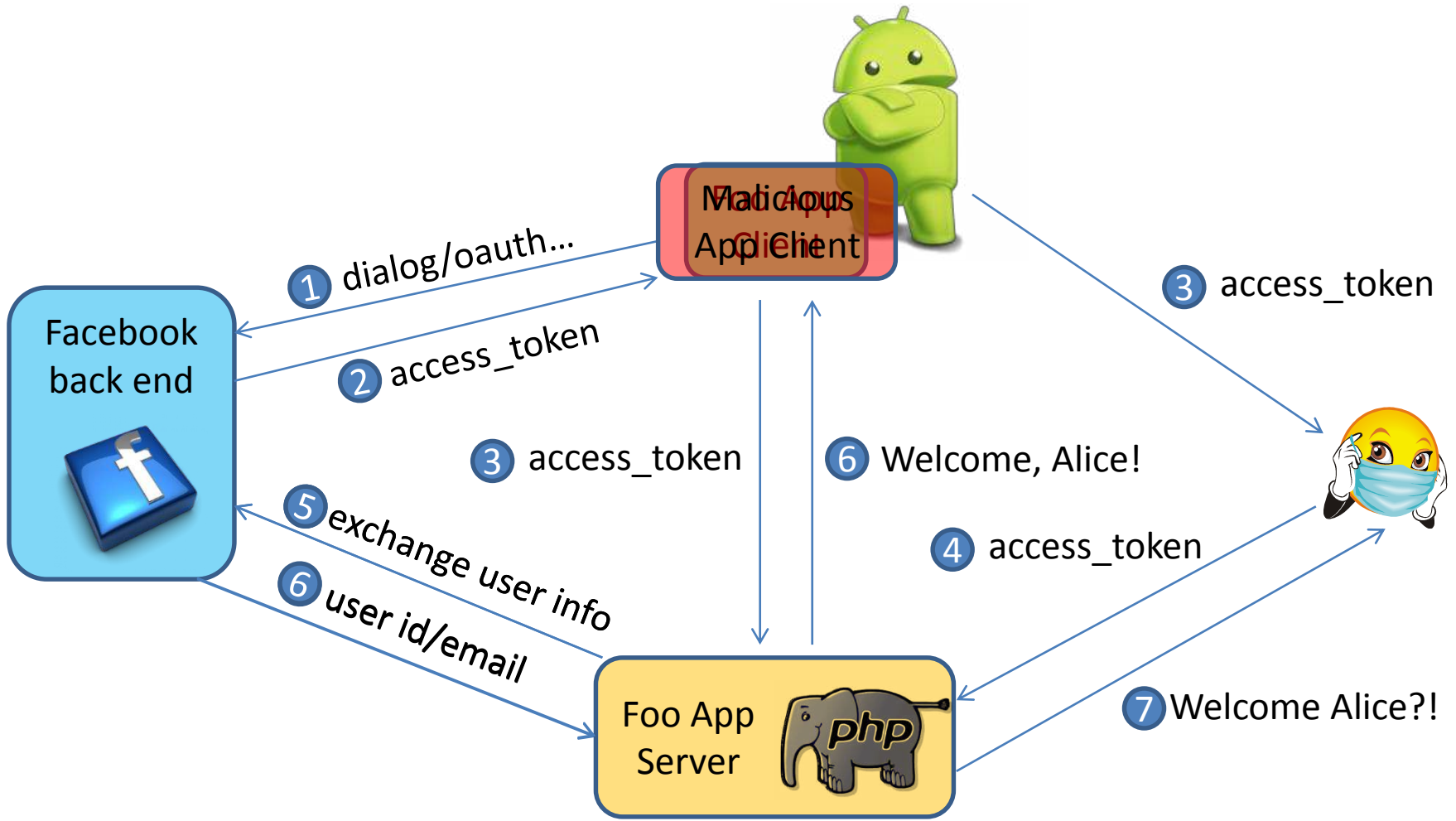
If the developers follow the guides properly, will the application be secure?

<code>scope</code>	No	A comma separated list of permission names which you would like the user to grant your application. Only the permissions which the user has not already granted your application will be shown
<code>state</code>	No	A unique string used to maintain application state between the request and callback. When Facebook redirects the user back to your <code>redirect_uri</code> , this parameter's value will be included in the response. You should use this to protect against Cross-Site Request Forgery.
<code>response_type</code>	No	The requested response type, one of <code>code</code> or <code>token</code> . Defaults to <code>code</code> . If left unset, or set to <code>code</code> the Dialog's response will include an OAuth code which can be exchanged for an access token as per the server-side authentication flow. If set to <code>token</code> , the Dialog's response will include an OAuth user access token in the fragment of the URL the user is redirected to - as per the client-side authentication flow.
<code>display</code>	No	The display mode with which to render the Dialog. One of <code>page</code> , <code>popup</code> or <code>touch</code> . Defaults to <code>page</code> when the user is using a desktop browser or the dialog is invoked on the <code>www.facebook.com</code> domain. Defaults to <code>touch</code> when the user is using a mobile browser or the dialog is invoked on the <code>m.facebook.com</code> domain. No other display type is allowed on <code>m.facebook.com</code> . In <code>page</code> mode, the OAuth dialog is displayed in the full Facebook chrome. In 'popup' mode, the OAuth dialog is displayed in a form suitable for embedding in a popup window. This parameter is automatically specified by most Facebook SDK, so may not need to be set explicitly.

The requested response type, one of `code` or `token`. Defaults to `code`...

***Important:** When using the JS SDK, do not specify `client_id` or `redirect_uri` - these will be set by the SDK.

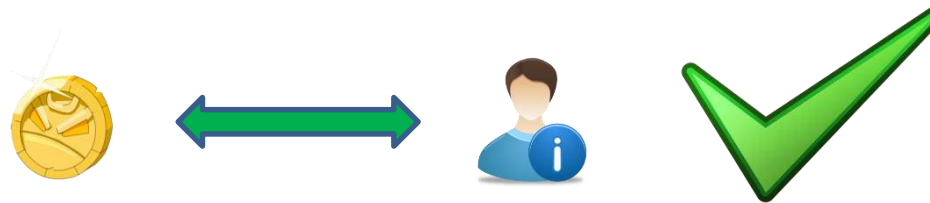
Possible Implementation



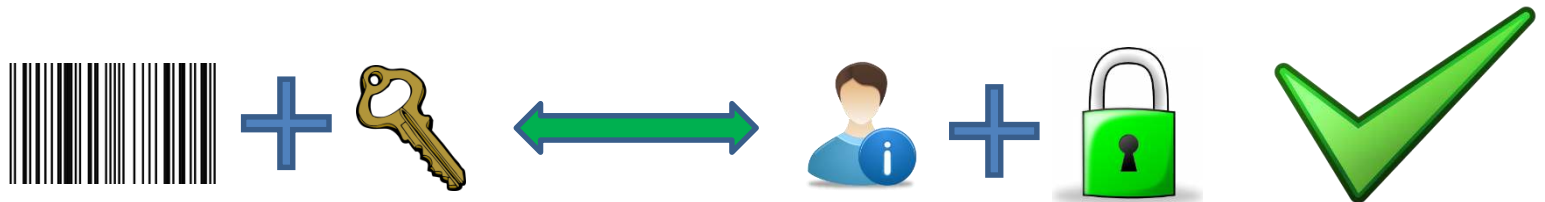
Video Demo

Who's to blame?

Developers?



SDK providers?



Developer guide does not explicitly state that token flow **MUST** not be used for server-side authentication.

Constructing a URL to the OAuth Dialog

To invoke the OAuth Dialog, redirect the user's browser to a URL of the form:

```
https://www.facebook.com/dialog/oauth/?
  client_id=YOUR_APP_ID
  &redirect_uri=YOUR_REDIRECT_URL
  &state=YOUR_STATE_VALUE
  &scope=COMMA_SEPARATED_LIST_OF_PERMISSION_NAMES
```

Parameters

The OAuth Dialog supports the following parameters which may be passed in the URL string:

<code>scope</code>	No	A comma separated list of permission names which you would like the user to grant your application. Only the permissions which the user has not already granted your application will be shown
--------------------	----	---

If developers follow the guides properly,
will the applications be secure?

The requested response type is `token`. Defaults to `code...`

No.

- Not because of vulnerabilities in SDKs.
- Due to *implicit assumptions* about how to use them.

***Important:** When using the JS SDK, do not specify `client_id` or `redirect_uri` - these will be set by the SDK.

Implicit Assumptions

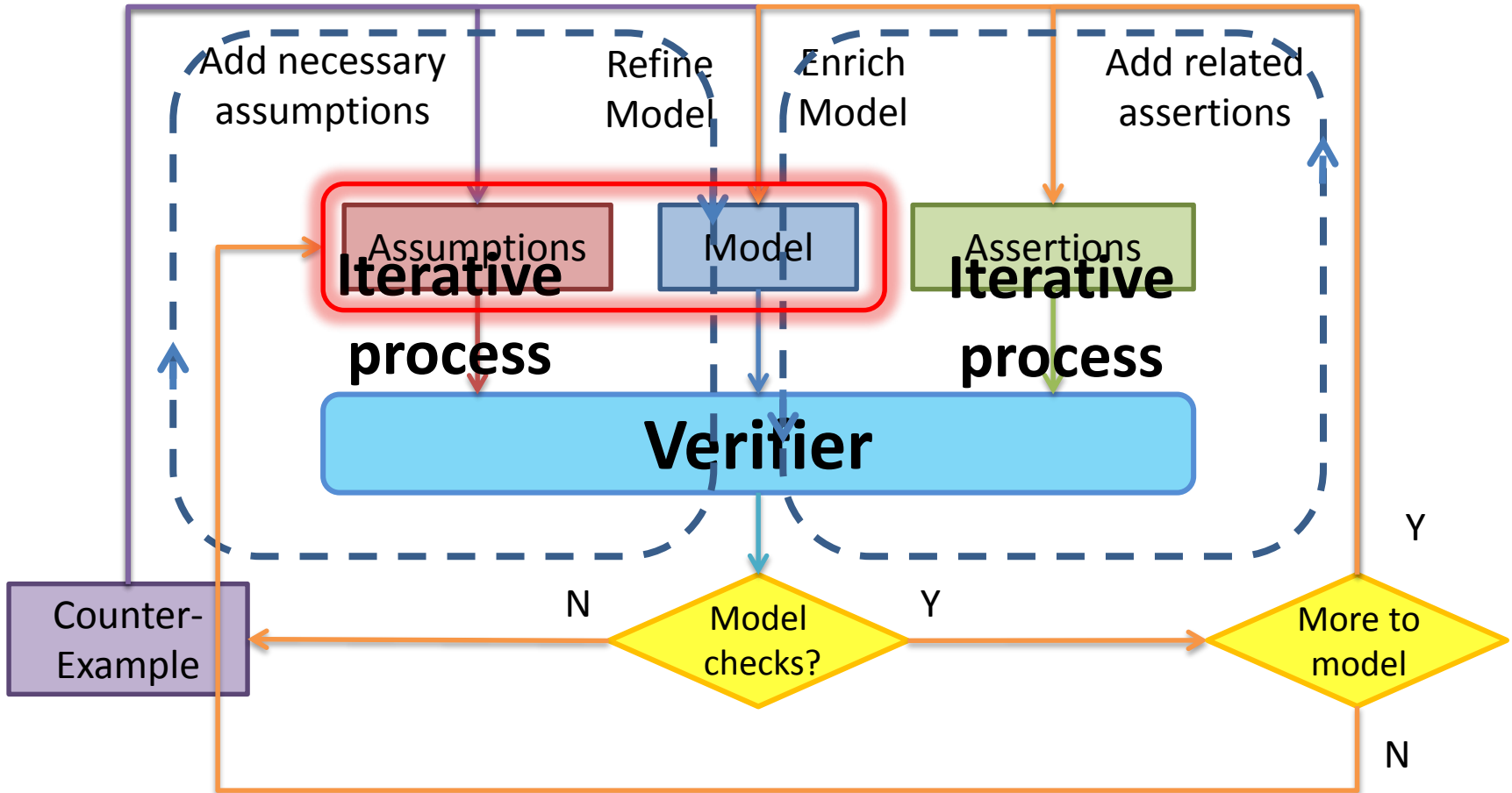
Assumptions that are

- not clearly stated in the SDK's developer guides;
- related to how the SDK should be used;
- **essential for application's security property.**

Goal of this project:

systematically find these implicit assumptions

General Approach

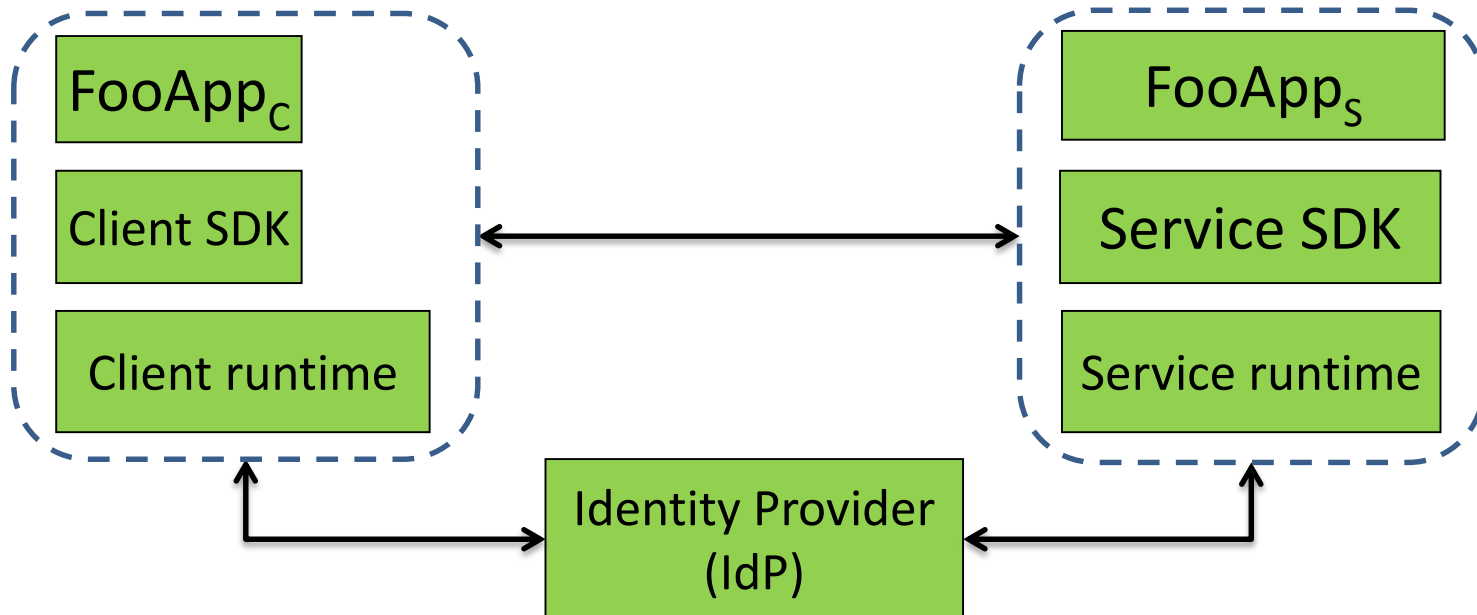


General Approach

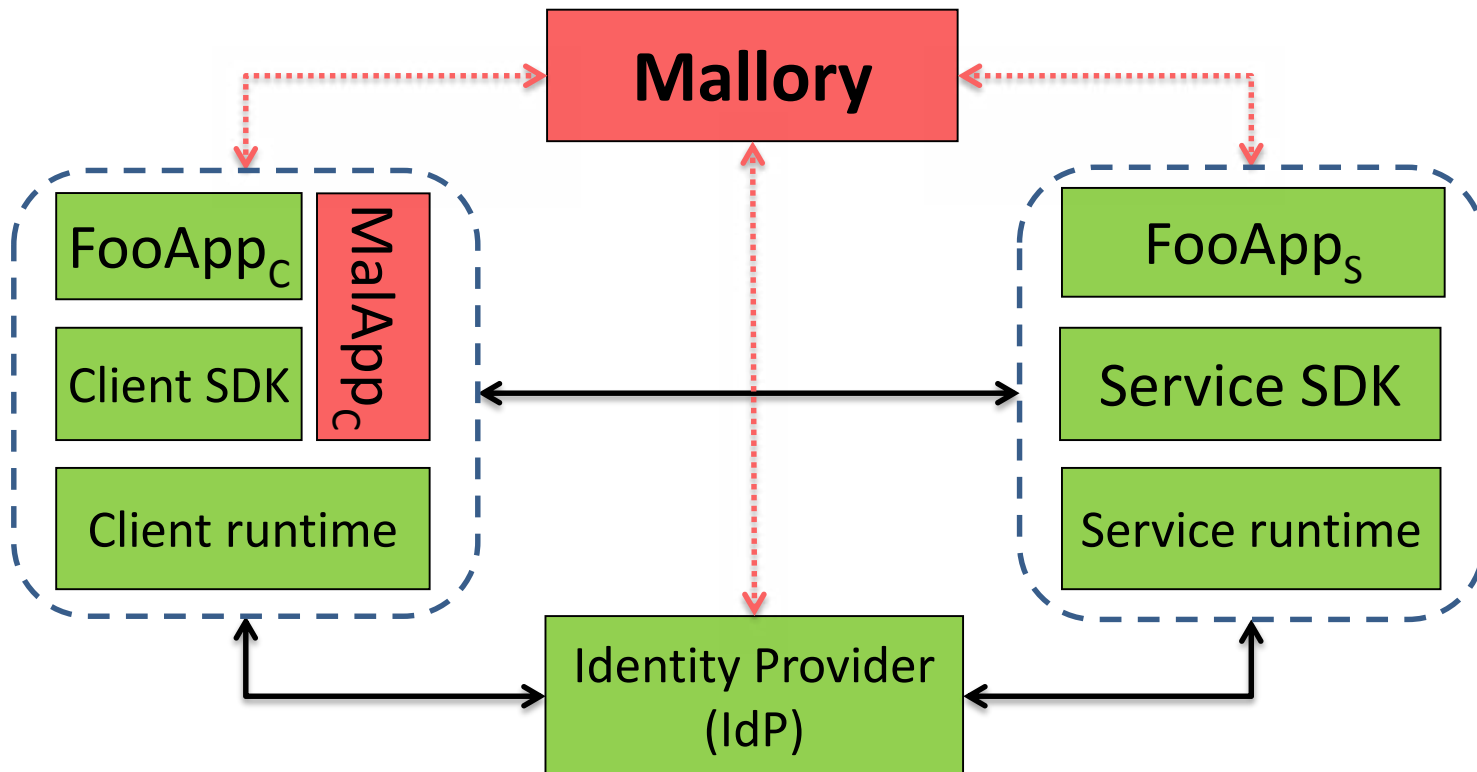
Target: Single-Sign On Systems



System Architecture



Threat model



Desired Security Properties

Authentication

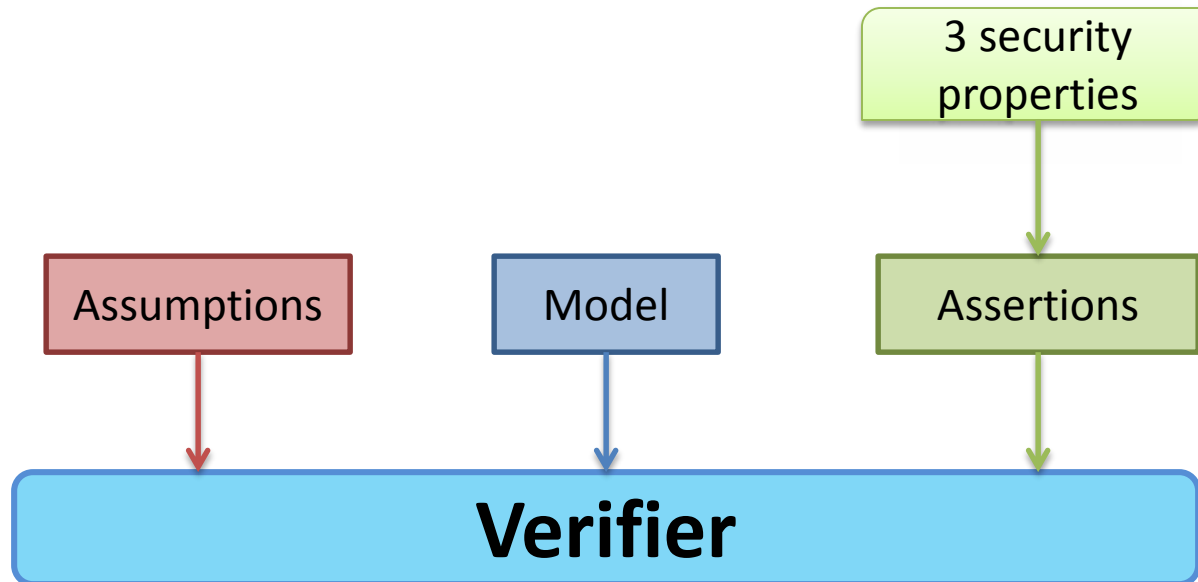
Mallory cannot sign into Foo app as Alice.

Authorization

Mallory cannot access data that Alice have not granted permission to Mallory.

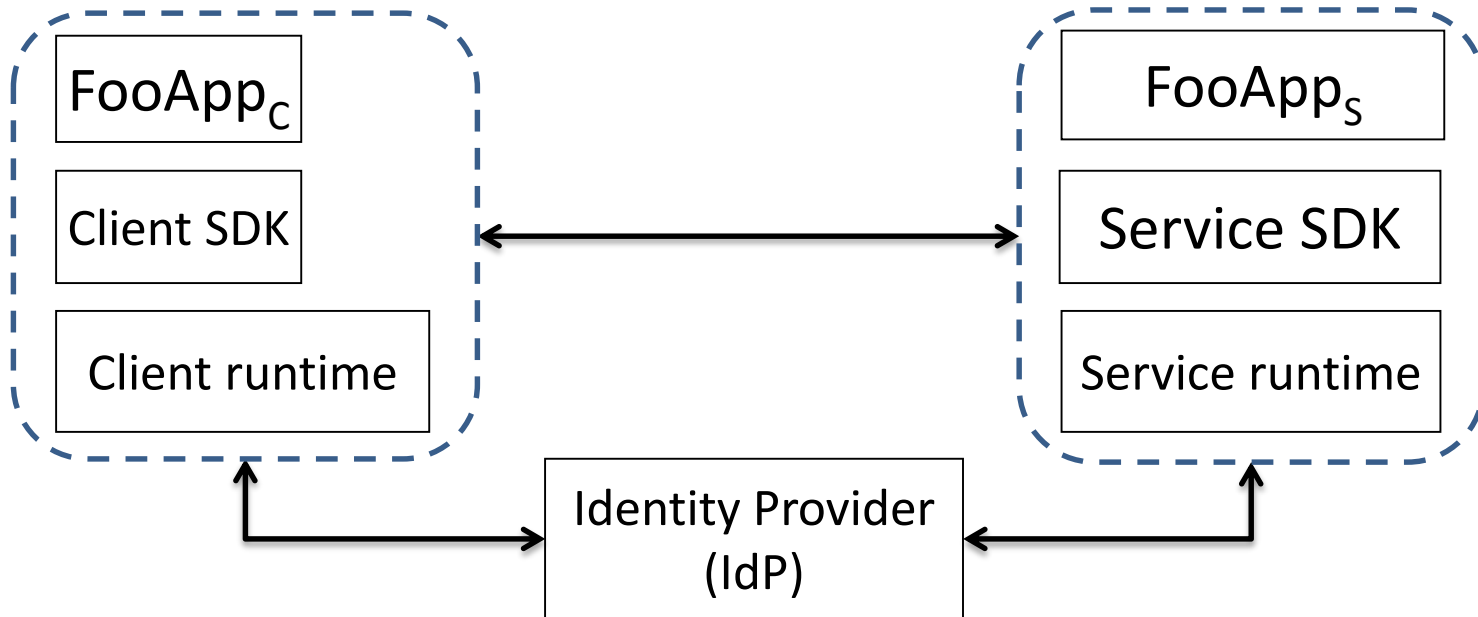
Association

The **user identity**, **user's permission** (authorization result), and **session identity** must represent the same person.

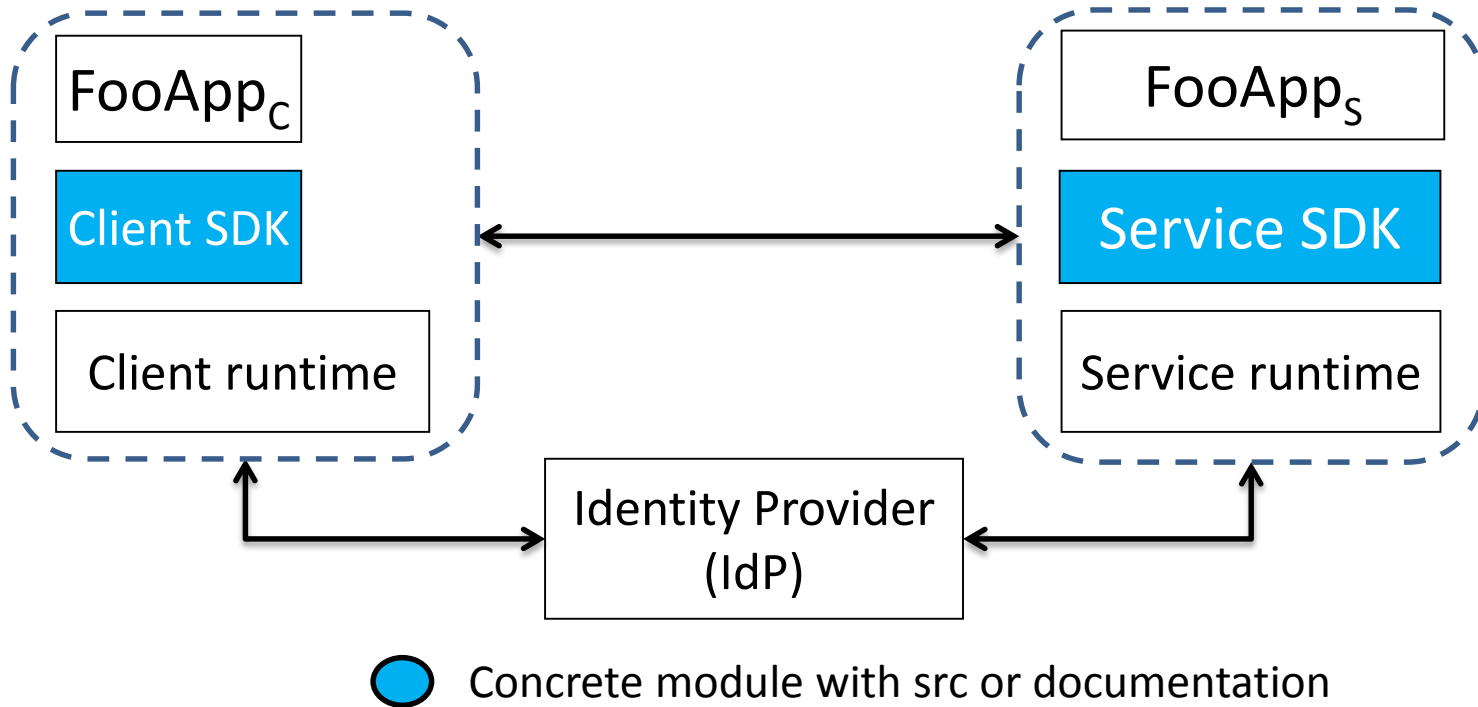


```
assert(logged_in_user != _Alice);
```

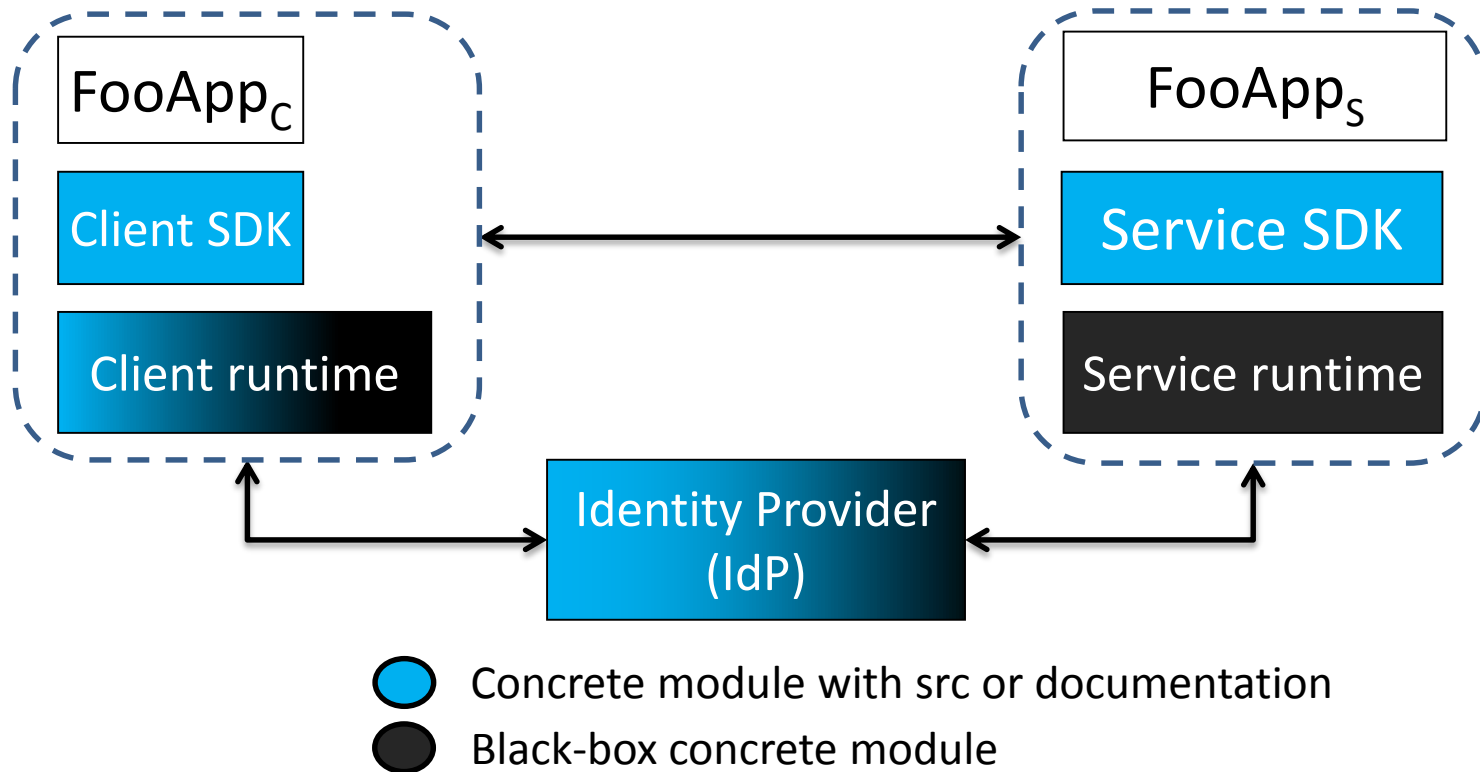
System Architecture



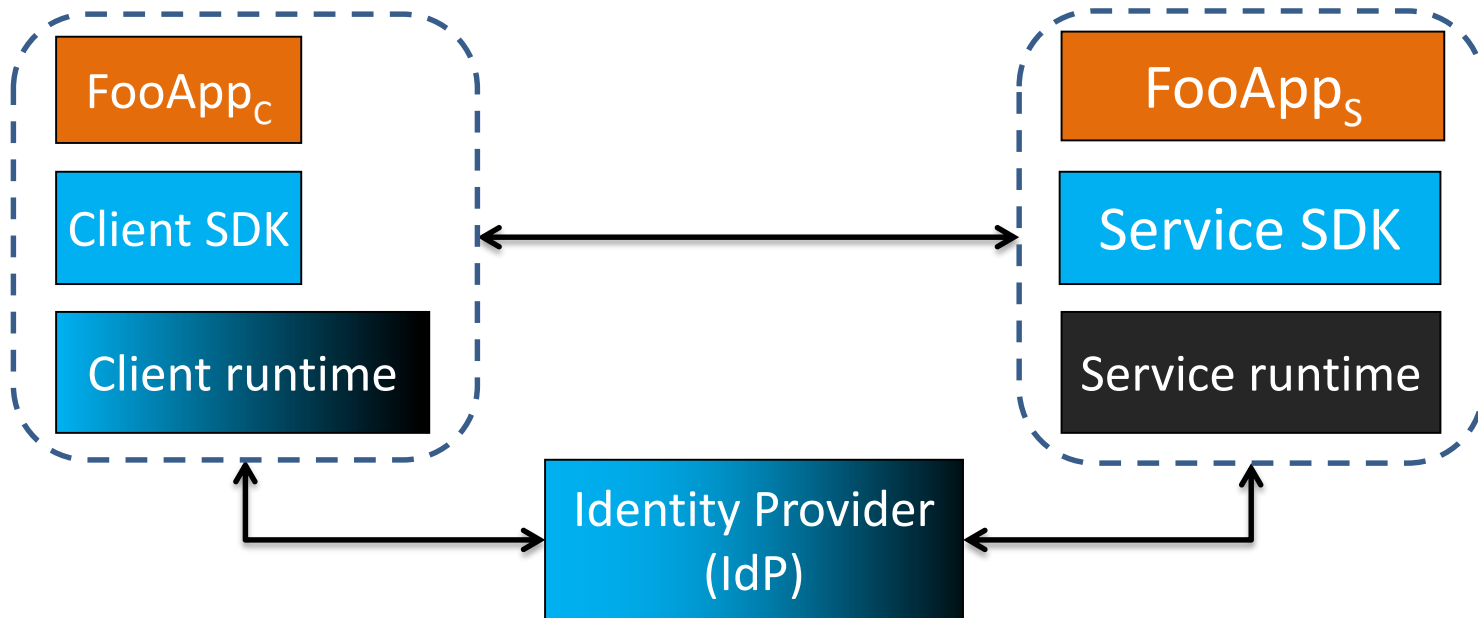
Modeling SDKs



Modeling underlying system layer



Modeling Foo application

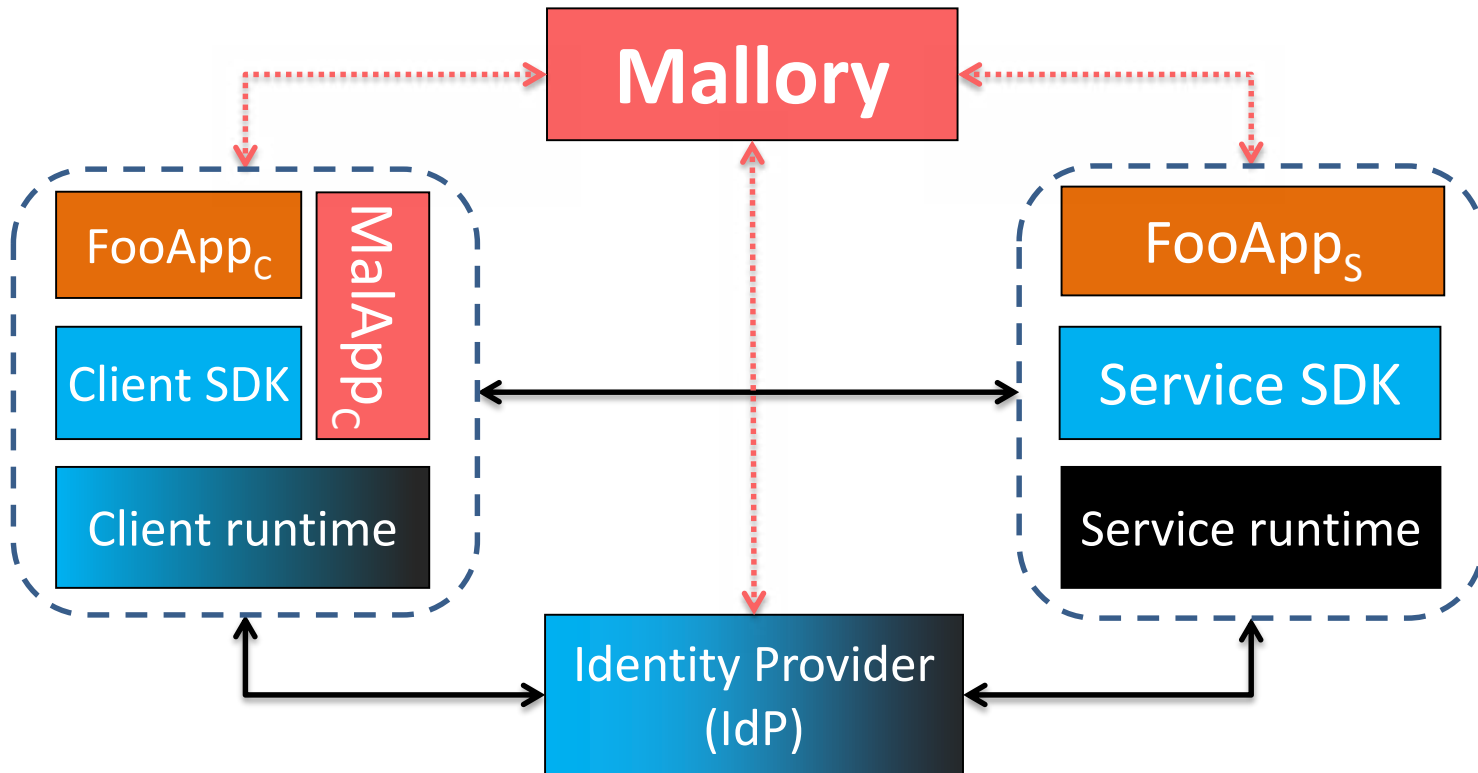


● Concrete module with src or documentation

● Abstract module subject to dev guide

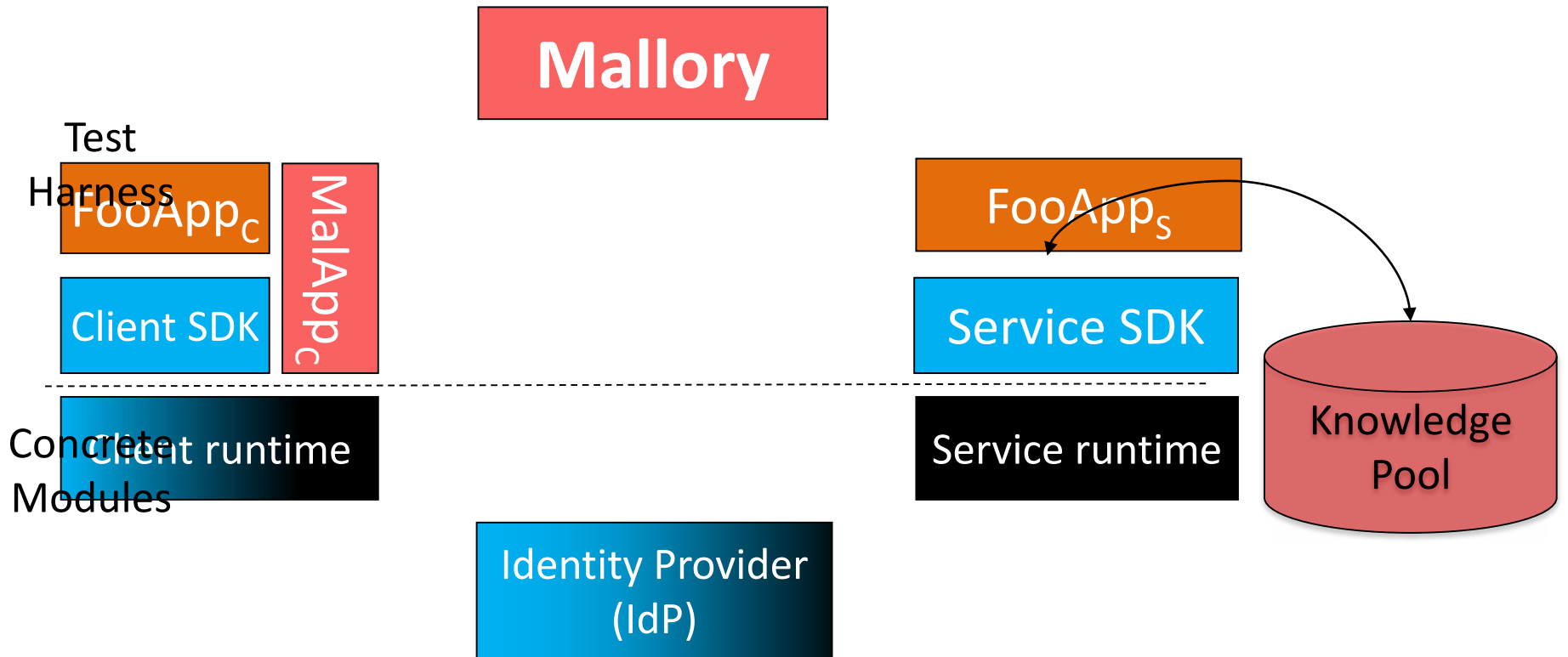
● Black-box concrete module

Modeling Mallory



- Concrete module with src or documentation
- Abstract module subject to dev guide
- Black-box concrete module
- Abstract module subject to knowledge pool

Modeling Mallory

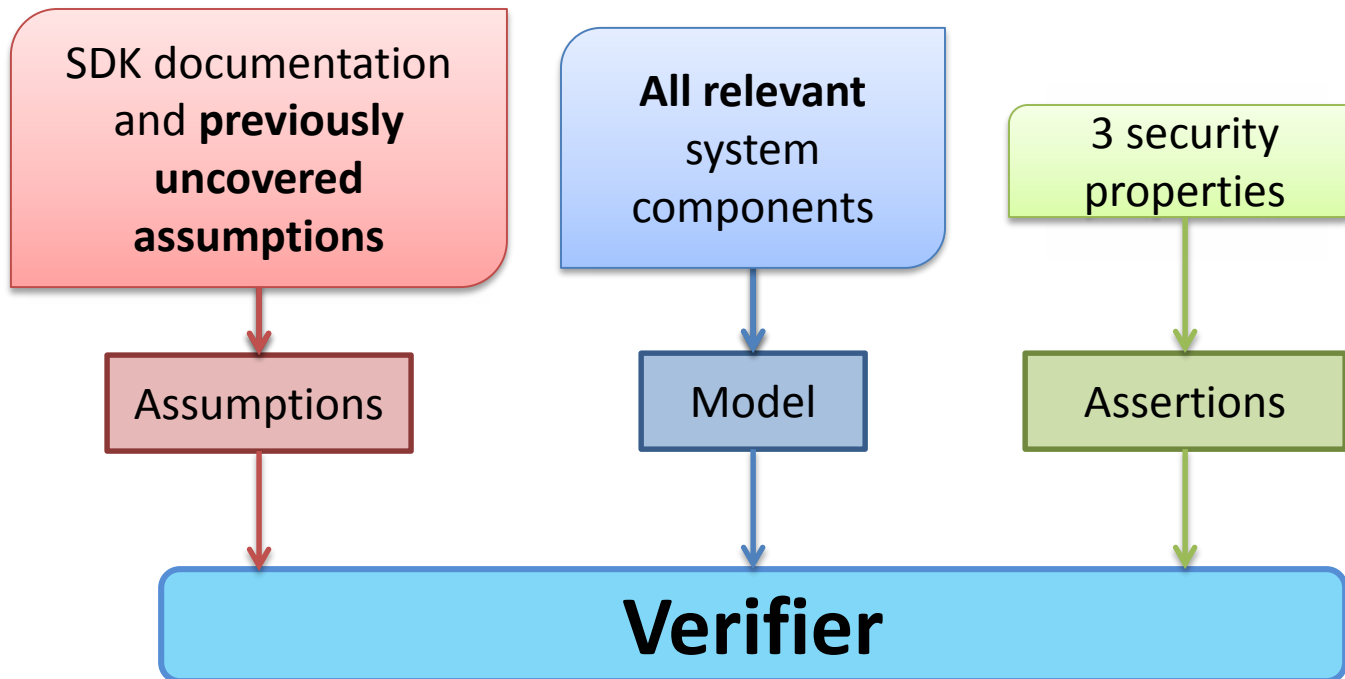


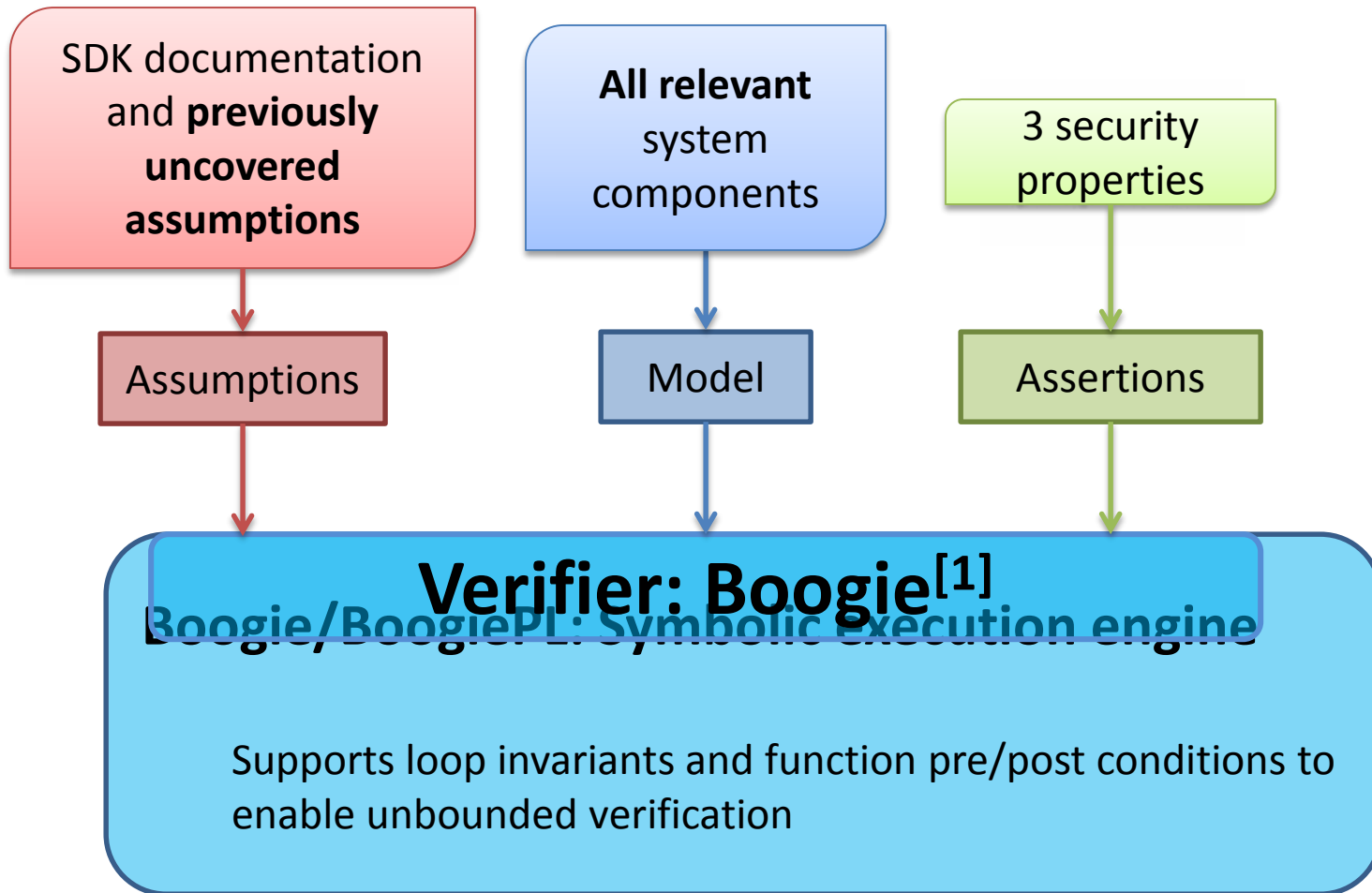
● Concrete module with src or documentation

● Abstract module subject to dev guide

● Black-box concrete module

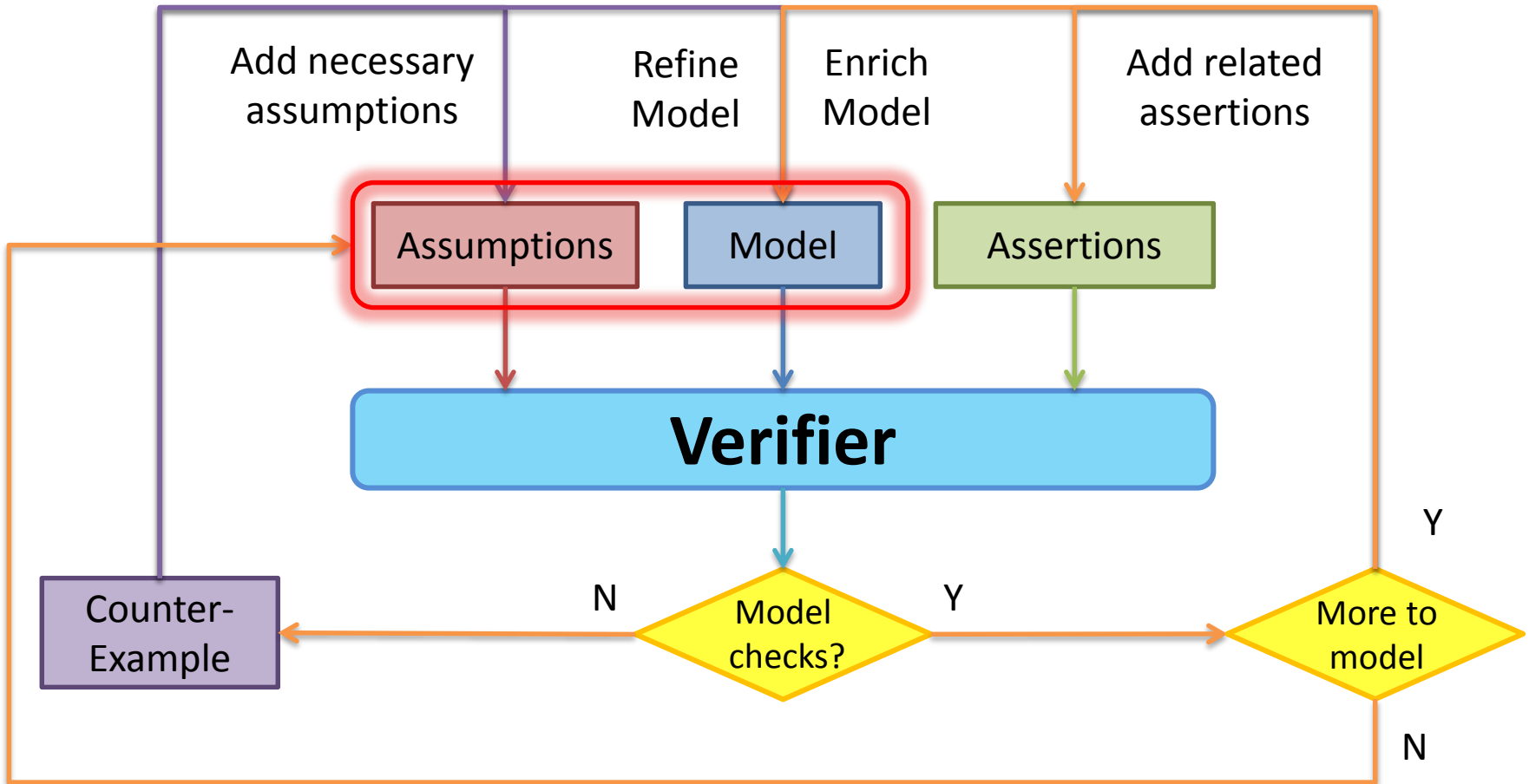
● Abstract module subject to knowledge pool





[1]: Boogie: An Intermediate Verification Language. <http://research.microsoft.com/en-us/projects/boogie/>

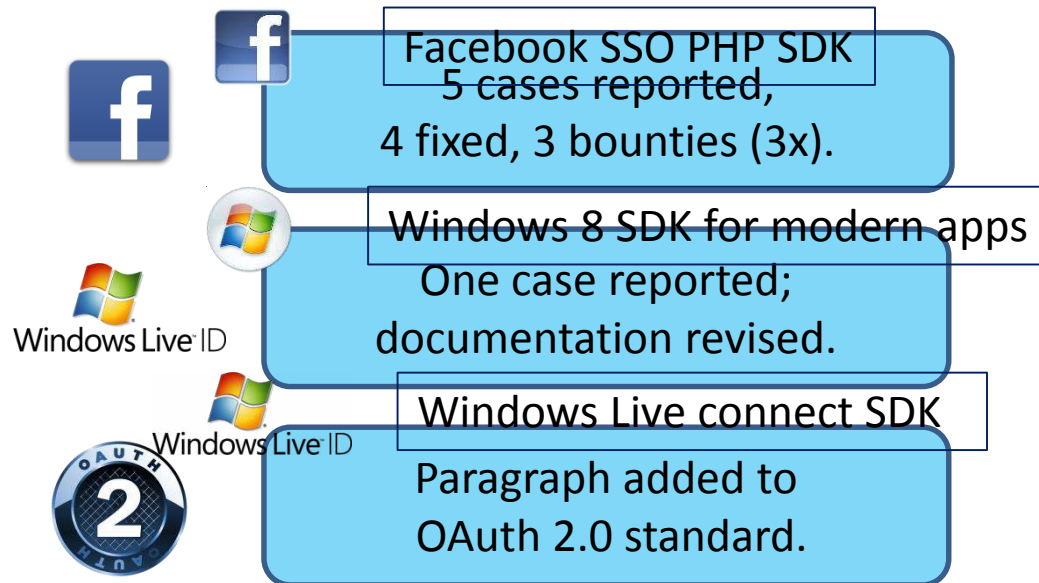
Results



Results overview

Explicated three SDKs: (6 months)

Many implicit assumptions were found:



The diagram consists of three vertically stacked items, each with an icon on the left and a text box on the right. The top item features a large Facebook 'f' icon and a smaller one above the text box. The middle item features a Windows logo icon and the text 'Windows Live ID' below it. The bottom item features a circular OAuth 2.0 logo with the number '2' and the text 'Windows Live ID' to its left.

- Facebook SSO PHP SDK**
5 cases reported,
4 fixed, 3 bounties (3x).
- Windows 8 SDK for modern apps**
One case reported;
documentation revised.
- Windows Live connect SDK**
Paragraph added to
OAuth 2.0 standard.

Majority of tested apps vulnerable due to failure to ensure at least one uncovered assumption.

SDK models

```
protected function getUserFromAvailableData() {
    if ($signed_request) {
        ...
        $this->setPersistentData('user_id',
                               $signed_request['user_id']);

        return 0;
    }
    $user = $this->getPersistentData('user_id', $default = 0);
    $persist_token =
        $this->getPersistentData('access_token');
    $access_token = $this->getAccessToken();
    if ($access_token &&
        !($user && $persist_token == $access_token)) {
        $user = $this->getUserFromAccessToken();
        if ($user)
            $this->setPersistentData('user_id', $user);
        else $this->clearAllPersistentData();
    }
    return $user;
}

public function getLogoutUrl() {
    return $this->getUrl(
        'www', 'logout.php',
        array_merge(array(
            'next' => $this->getCurrentUrl(),
            'access_token' => $this->getAccessToken(), ), ...));
}
```

Facebook PHP Source code

```
procedure {:inline 1} getUserFromAvailableData() returns (user:User) {
    if (IdP_Signed_Request_Records__user_ID[signed_request] != _nobody) {
        ...
        user := IdP_Signed_Request_Records__user_ID[signed_request];
        call setPersistentData__user_id(user);
        return;
    }
    call user := getPersistentData__user_id();
    call persisted_access_token := getPersistentData__access_token();
    call access_token := getAccessToken();
    if (access_token >= 0 &&
        !(user != _nobody && persisted_access_token == access_token)) {
        call user := getUserFromAccessToken(access_token);
        if (user != _nobody) {
            call setPersistentData__user_id(user);
        } else {
            call clearAllPersistentData();
        }
    }
    return;
}

procedure {:inline 1} getLogoutUrl()
returns (API_id: API_ID, next__domain: Web_Domain, next__API: API_ID,
        access_token: int) {
    API_id := API_id_FBConnectServer_login_php;
    call access_token := getAccessToken();
    call next__domain, next__API := getCurrentUrl();
}
```

Boogie PL Model

API models

Constructing a URL to the OAuth Dialog

To invoke the OAuth Dialog, redirect the user's browser to a URL of the form:

```
https://www.facebook.com/dialog/oauth/?
client_id=YOUR_APP_ID
&redirect_uri=YOUR_REDIRECT_URL
&state=YOUR_STATE_VALUE
&scope=COMMA_SEPARATED_LIST_OF_PERMISSION_NAMES
```

Parameters

The OAuth Dialog supports the following parameters which may be passed in the URL string:

Parameter	Required?	Description
client_id	Yes*	Your App ID. This is called <code>client_id</code> instead of <code>app_id</code> for this particular method in order to be compliant with the OAuth 2.0 specification.
redirect_uri	Yes*	Should not be set when using the JS SDK to invoke the dialogs. The URL to redirect to after the user clicks a button in the dialog. The URL you specify must be a URL of with the same Base Domain as specified in your app's settings, a Canvas URL of the form https://apps.facebook.com/YOUR_APP_NAMESPACE or a Page Tab URL of the form https://www.facebook.com/PAGE_USERNAME/app_YOUR_APP_ID
scope	No	A comma separated list of permission names which you would like the user to grant your application. Only the permissions which the user has not already granted your application will be shown
state	No	A unique string used to maintain application state between the request and callback. When Facebook redirects the user back to your <code>redirect_uri</code> , this parameter's value will be included in the response. You should use this to protect against Cross-Site Request Forgery.
response_type	No	The requested response type, one of <code>code</code> or <code>token</code> . Defaults to <code>code</code> . If left unset, or set to <code>code</code> the Dialog's response will include an OAuth code which can be exchanged for an access token as per the server-side authentication flow. If set to <code>token</code> , the Dialog's response will include an oauth user access token in the fragment of the URL the user is redirected to - as per the client-side authentication flow.
display	No	The display mode with which to render the Dialog. One of <code>page</code> , <code>popup</code> or <code>touch</code> . Defaults to <code>page</code> when the user is using a desktop browser or the dialog is invoked on the www.facebook.com domain. Defaults to <code>touch</code> when the user is using a mobile browser or the dialog is invoked on the m.facebook.com domain. No other display type is allowed on m.facebook.com . In <code>page</code> mode, the OAuth dialog is displayed in the full Facebook chrome. In 'popup' mode, the OAuth dialog is displayed in a form suitable for embedding in a popup window. This parameter is automatically specified by most Facebook SDK, so may not need to be set explicitly.

*Important: When using the JS SDK, do not specify `client_id` or `redirect_uri` - these will be set by the SDK.

```
procedure {:inline 1} dialog_oauth(IdPLoggedInUser:User,
client_id: AppID,
redirect_domain: Web_Domain, scope:Scope,
response_type:ResponseType)
returns (r:int, Response_data: int)
modifies Access_Tokens__TokenValue, Access_Tokens__user_ID,
Access_Tokens__Scope;
modifies Codes__user_ID,Codes__App_ID,Codes__Scope;
modifies ...
```

```
{
  var access_token:int, code:int, sr:int;

  ...
  if (response_type==_Token || response_type==_Signed_Request){
    havoc access_token; //it means "access_token := *;"
    ...
    IdP_Signed_Request_signature[sr]:=ValidIdPSignature;
    IdP_Signed_Request_oauth_token[sr]:=access_token;
    IdP_Signed_Request_code[sr]:=code;
    IdP_Signed_Request_user_ID[sr]:= IdPLoggedInUser;
    IdP_Signed_Request_app_id[sr]:= client_id;
  }
  if (response_type==_Token) {
    Response_data:=access_token;
  } else if (response_type==_Code) {
    Response_data:=code;
  } else {
    Response_data:=sr;
  }
  r:=200;
}
```

Facebook Dialog API documentation

Boogie model

Example vulnerability from Facebook SDK

The [examples](#) are a good place to start. The minimal you'll need to have is:

```
require 'facebook-php-sdk/src/Facebook.php';

$facebook = new Facebook(array(
    'appId' => 'YOUR_APP_ID',
    'secret' => 'YOUR_APP_SECRET',
));

// Get User ID
$user = $facebook->getUser();
```

1

Login or logout url will be needed depending on current user state.

```
if ($user) {
    $logoutUrl = $facebook->getLogoutUrl();
} else {
    $loginUrl = $facebook->getLoginUrl();
}
```

3

Example assumption from Facebook SDK

```
procedure {:inline 1} getLogoutUrl()
returns (API_id: API_ID, ...)
modifies ...;
{
  var test_t:int ;
  call access_token := getAccessToken();
  call test_t := getApplicationAccessToken();
  assume(access_token != test_t);
  API_id := API_id_FBConnectServer_login_php;
  ...
  return;
}
```

Assumption (in the model)

Assumption (in plain text):
getLogoutUrl() must not return an
application access token to the
client.

Best outcome:

Facebook fixed their SDK code so this
assumption is not needed in the newer
version.

Example assumption from Windows Live

```
function saveRefreshToken($refreshToken)
{
    // save the refresh token associated with the user
    // id on the site.
}
function handleTokenResponse($token, $error = null)
{
    // save the refresh
    $authCookie = $_COOKIE[AUTHCOOKIE];
    $cookieValues = parseQueryString($authCookie);
    ...
    token associated with the
    user id on the site.
    if (!empty($token->{ REFRESHTOKEN }))
    {
        saveRefreshToken($token->{ REFRESHTOKEN });
    }
}
...

```

associate refresh token with the user ID obtained from the global variable `$_COOKIE`

Two interpretations

associate refresh token with the user ID obtained from the refresh token passed into the function

original Live ID developer guide

```
procedure {:inline 1} saveRefreshToken (refresh_token_index:
int, RP_Cookie_index: int)
modifies RP_Refresh-Token_index;
{
    var user: User;
    //call user := get_User_ID(RP_Cookie_index);
    user := Refresh-Token__user_ID[refresh_token_index];
    if (user == _nobody) {return;}
    RP_Refresh-Token_index[user] := refresh_token_index;
}
```

Example assumption from Windows Live

```
procedure {:inline 1} saveRefreshToken (refresh_token_index: int,  
RP_Cookie_index: int)  
modifies RP_Refresh-Token_index;  
{  
  var user: User;  
  call user := get_User_ID(RP_Cookie_index); !  
  user := Refresh-Token_user_ID[refresh_token_index];  
  if (user == _nobody) {return;}  
  RP_Refresh-Token_index[user] := refresh_token_index;  
}
```

```
LiveConnectServer.bpl<200,23>: inline$login_live_com_oauth20_token_srf$9$Ret  
urn  
RPCallbackPHP.bpl<60,21>: inline$requestAccessTokenByVerifier$3$anon0$1  
RPCallbackPHP.bpl<57,23>: inline$requestAccessTokenByVerifier$3$Return  
RPCallbackPHP.bpl<85,3>: inline$handlePageRequest$3$anon8_Then$1  
RPCallbackPHP.bpl<88,4>: inline$handlePageRequest$3$anon9_Then  
RPCallbackPHP.bpl<34,22>: inline$handleTokenResponse$7$anon0  
RPCallbackPHP.bpl<37,3>: inline$handleTokenResponse$7$anon3_Then  
RPCallbackPHP.bpl<4,2>: inline$get_User_ID$1$anon0  
RPCallbackPHP.bpl<27,2>: inline$saveRefreshToken$7$anon3_Else  
RPCallbackPHP.bpl<21,23>: inline$saveRefreshToken$7$Return  
RPCallbackPHP.bpl<37,3>: inline$handleTokenResponse$7$anon3_Then$1  
RPCallbackPHP.bpl<39,22>: inline$handleTokenResponse$7$anon2  
RPCallbackPHP.bpl<88,4>: inline$handlePageRequest$3$anon9_Then$1  
test_harness.bpl<60,2>: inline$call_an_API_on_foo_service_app_from_foo_app_f  
rom_alice_device$0$anon0$1  
test_harness.bpl<375,3>: inline$foo_client_app_calls$0$anon5_Then$1  
test_harness.bpl<364,23>: inline$foo_client_app_calls$0$Return  
test_harness.bpl<440,3>: inline$takeAction$0$anon5_Then$1  
test_harness.bpl<590,3>: anon2_LoopBody$1  
  
Boogie program verifier finished with 0 verified, 1 error
```

```
D:\Research\Explicated-SDKs-master\models\model_LiveConnectSDK>Bo  
s.bpl specificDecls.bpl LiveConnectSDK.bpl RPCallbackPHP.bpl Live  
pl knowledge_pool.bpl test_harness.bpl  
Boogie program verifier version 2.2.30705.1126, Copyright (c) 201  
ft.  
  
Boogie program verifier finished with 1 verified, 0 errors  
  
D:\Research\Explicated-SDKs-master\models\model_LiveConnectSDK>_
```

Example assumption from Windows Live

```
function saveRefreshToken($refreshToken)
{
    // save the refresh token and associate it with the
    // user identified by your site credential system.
}

function handleTokenResponse($token, $error = null)
{
    $authCookie = $_COOKIE[AUTHCOOKIE];
    $cookieValues = parseQueryString($authCookie);

    if (!empty($token))
    ...
```

new Live ID developer guide

Testing Popular Apps

Showcase






See how companies make their sites personalized and social with Facebook

Facebook showcase applications

← Results for "facebook" 804 apps

All categories Free Sort by highest rating

 La Sierra University ★★★★★ Education Free	 Quran Explorer ★★★★★ Books & Reference Free	 Love Matching ★★★★★ Games Free
 MITalk ★★★★★ Social Free	 Egypt's History In Photos ★★★★★ Photo Free	 Movie Alerts ★★★★★ Entertainment Free
 Bi-Blocks ★★★★★ Games Free	 Student Schedule ★★★★★ Education Free	 Appo ★★★★★ Social Free
 MyPicture ★★★★★ Photo Free	 QPR ClevFoot ★★★★★ Sports Free	 Juventus ClevFoot ★★★★★ Sports Free
 Qualità della Vita ★★★★★ Entertainment Free	 Blastanova ★★★★★ Music & Video Free	 Scribbling Pad ★★★★★ Tools Free
 How to Kiss? ★★★★★ Lifestyle Free	 Flippit ★★★★★ Social Free	 Social Alert ★★★★★ Tools Free
 Qwotee ★★★★★ Books & Reference Free	 News of News! ★★★★★ News & Weather Free	 Negozi di Roma ★★★★★ Travel Free
 Where's My Stuff ★★★★★ Tools Free	 Turtle Graphics ★★★★★ Education Free	 Vid2Pix ★★★★★ Music & Video Free
 Firat Karikatürleri ★★★★★ Entertainment Free	 Pierce The Veil ★★★★★ Entertainment Free	 Greetings: Christmas and New... ★★★★★ Lifestyle Free

Popular Windows 8 modern applications
using Facebook SSO

Testing Results

Test Set	Number of Apps	Vulnerable
Illustrative example	27	21 (78%)
Assumption A1 (in the paper)	7	6 (86%)
Assumption A6 (in the paper)	21	14 (67%)



Conclusion

Missed **implicit assumptions** can lead to many vulnerabilities when integrating third-party services.

What we advocate: SDK providers **explicate SDKs** before release.

- Fix SDK Code
- Develop Automated Checker
- Improve SDK documentation

Thank you!

Visit project website for more info:
http://www.cs.virginia.edu/~yz8ra/ExplicatingSDKs_website/

Models available at: <https://github.com/sdk-security/Explicated-SDKs>

Rui Wang^{1*}, Yuchen Zhou^{2*†},
(* Lead authors, †Speaker)

Shuo Chen¹, Shaz Qadeer¹, David Evans² and Yuri Gurevich¹

¹Microsoft Research and ²University of Virginia

Microsoft[®]
Research

