Networking

Autenticazione tramite social network per l'accesso a hotspot pubblici gratuiti.†

Giuseppe Nantista,[a] Andrea Lora,[a] Augusto Pifferi.[a]

In questo rapporto tecnico studieremo la possibilità di autenticare gli utenti di un hotspot pubblico gratuito tramite gli account che essi hanno registrato sui principali social network. Il caso preso in esame fa uso delle API di Facebook per accedere a un hotspot realizzato con hardware Mikrotik, tuttavia il principio alla base del funzionamento è generalizzabile ad altre piattaforme hardware/software e ad altri network che forniscano API di autenticazione. Descriveremo i principi alla base della soluzione realizzata e le limitazioni che essa presenta in termini di effettiva identificazione dell'utente e navigazione anonima.
Keywords: Social hotspot, Facebook, Mikrotik

1 Introduzione

Quando un utente accede a un hotspot pubblico gratuito il fornitore del servizio deve tutelarsi dalle azioni che l'utente compirà, tenendo traccia di tutte le connessioni effettuate e riservandosi la possibilità di ricondurre ogni singola connessione a un individuo. Il metodo più diffuso in questi casi consiste nell'inviare un SMS sul cellulare dell'utente contenente username e password di accesso all'hotspot, associando così le connessioni a un numero di cellulare che, in caso di contestazioni da parte delle autorità giudiziarie, possa essere usato come riferimento univoco all'individuo che ha effettuato le connessioni incriminate. Tale meccanismo prevede da parte dell'utente il fastidio di memorizzare nuove credenziali di accesso al servizio e da parte del fornitore un costo (l'invio degli SMS) non desiderato laddove si offre un servizio gratuito. In altri casi si preferisce identificare *de visu* l'utente tramite l'esibizione di un documento di identità e di conseguenza si provvede ad emettere un ticket individuale con le credenziali di accesso, tuttavia questo caso comporta l'accesso a informazioni riservate e la necessità di impegnare una persona in questa attività.

Demandare a terzi l'autenticazione dell'individuo comporta il già citato vantaggio da parte dell'utente di non dover memorizzare le ennesime credenziali di accesso a un servizio, liberando il fornitore del servizio dagli oneri e dalle responsabilità correlate all'erogazione del servizio stesso.

Questo documento vuole indicare una possibile soluzione mediante l'impiego di un hotspot gateway Mikrotik, di un server web abilitato PHP e di un account sviluppatore Facebook.

2 Operazioni Preliminari

Il primo passo da compiere è la registrazione di un account da sviluppatore di Facebook, è sufficiente accedere con il proprio account all'indirizzo https://developers.facebook.com/ e creare la prima applicazione di tipo "website". Verrà assegnato un "app- id" alla nostra applicazione, della quale dovremo anche specificare l'url alla quale verrà ospitata, opzionalmente è possibile specificare una url per applicazioni mobile. Per iniziare Facebook mette a disposizione un tutorial reperibile nell'area per sviluppatori. [1]

Le applicazioni sono di uso generale, a noi in questa fase interessa la funzione di login, ovvero sia di validazione delle credenziali di accesso, da inserire nella nostra pagina di autenticazione dell'hotspot.

A tal proposito abbiamo configurato una routerboard Mikrotik per svolgere le funzioni di Hotspot e sostituito la pagina di login di default con una che rimanda a una pagina di autenticazione esterna secondo quanto indicato nella wiki di Mikrotik stessa. [2] Nell'hotspot così creato andranno aggiunte alcune destinazioni nel Walled Garden, la raccolta di tutte le destinazioni che un utente non ancora autenticato può visitare; innanzitutto l'indirizzo del nostro web server di supporto e poi due indirizzi generici per raggiungere le pagine di autenticazione di facebook:

```
*facebook*
*akamai*
```

Nella pagina va personalizzato il link di redirect, che dovrà puntare al nostro server web, abilitato PHP, il cui indirizzo coinciderà con quello specificato nel "site url" della applicazione Facebook, in caso contrario otterremo un errore di indirizzo di provenienza errato.

Ora è necessario, affinché un utente possa autenticarsi e navigare, seguire i seguenti passi:

1. L'utente clicca sul pulsante "login with Facebook";

[a] *Istituto di Cristallografia, C.N.R. via Salaria km 29.300, 00015 Monterotondo, Italy*

† apporto tecnico 2015/06 prot. CNR-IC 815 del 24/04/2015

2. Accede alla pagina di login del social network con riferimento alla app in questione, si autentica e autorizza l'applicazione ad accedere alle sue informazioni di base, profilo e email;
3. Ottiene da Facebook un "token" che garantisce che egli si sia correttamente autenticato;
4. Invia questo token al web server che effettua la verifica di validità del token sul sito di Facebook;
5. Se la verifica va a buon fine la routerboard Mikrotik viene istruita a creare una utenza temporanea con una password generata casualmente;
6. Il web server invia una istruzione di redirect verso la pagina di login, inserendo nella url username e password dell'utenza appena creata, cosicché l'utente risulti già in sessione.

Riassumiamo visivamente il flusso di informazioni appena descritto in figura 1.

L'hotspot Mikrotik Mikrotik mette a disposizione un meccanismo guidato per la creazione di un hotspot, che include anche le regole firewall di redirect in caso di utente non autenticato e le eccezioni relative al walled garden. Il redirect iniziale viene rimandato all'url

```
http://ip-dell-hotspot-mikrotik/login
```

La funzione login è hard-coded e invoca l'apertura del file login.html contenuto nella directory radice dell'hotspot; alla funzione login vengono passati alcuni parametri come ad esempio il mac-address e l'ip del client che ha tentato la connessione nonché l'url a cui egli intendeva accedere prima di essere "catturato". Sostituendo il file login.html con quello indicato in [2] è possibile rimandare l'utente a una pagina di login esterna.

Quando, ad autenticazione completata, il web server crea una utenza sull'hotspot Mikrotik fa uso delle API PHP RouterOS, scaricabili liberamente all'indirizzo [3].

Queste funzioni vengono eseguite con una utenza che deve essere preventivamente creata e abilitata all'interno del router con i seguenti comandi:

```
ip service set api disabled=no
/user group add name=gapi policy=api,read,write
/user add name=uapi password=apipasswd group=gapi
```

3 Il web server di supporto

Per realizzare l'infrastruttura proposta in questo progetto è indispensabile un server web con funzionalità PHP abilitate, che interviene in due fasi dell'attività di login dell'utente.

1. *La presentazione della pagina di login*
   La pagina di per se utilizza linguaggio javascript per caricare il pulsante di login, tuttavia l'uso di PHP è fondamentale per "portarsi dietro" i parametri relativi all'utente guest che fa richiesta di accesso e alla url richiesta. In appendice riportiamo una versione semplificata di tale pagina.
2. *L'immissione delle credenziali di accesso*
   Quando l'utente effettua il login su Facebook e ottiene il token relativo alla sua sessione lo comunica al web server, che tramite la pagina "confirm.php", anch'essa riportata in appendice, verifica la validità del token, crea l'utente sull'hotpost e infine rimanda l'utente alla pagina di login immettendo per suo conto i dati di accesso e la url di destinazione.

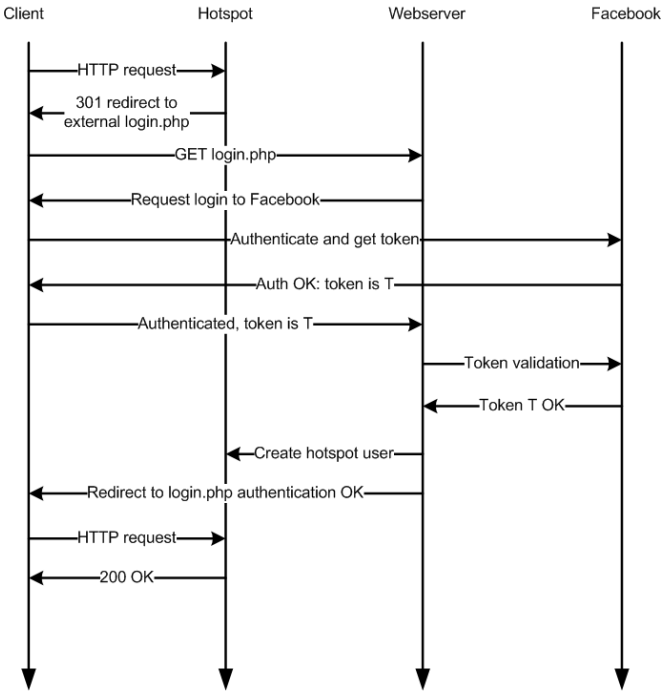

Fig. 1 Diagramma di flusso del processo di autenticazione

4 Affidabilità del sistema

Facebook tiene traccia di un certo numero di domini di posta elettronica che non garantiscono la vera identità di un utente, ad esempio quei domini di "disposable mailbox", tuttavia è possibile raggirare il meccanismo di verifica e creare un account fasullo così da poterlo sfruttare in una circostanza come l'accesso anonimo a una rete, tuttavia riteniamo che il livello di confidenzialità sia sufficiente per la normativa italiana relativa alla tracciabilità delle connessioni degli utenti.

Di recente è stato introdotto il concetto di verifica dell'account, che consente di aggiungere fra i dati personali oltre all'indirizzo email usato per registrare l'account anche un numero di cellulare, il che ricondurrebbe la tracciabilità dell'account all'operatore telefonico che ha rilasciato la scheda sim in questione. In questo senso quindi è possibile consentire l'accesso alla rete solo a quegli utenti che hanno un account verificato da Facebook; il valore in questione viene fornito da Facebook senza che l'utente conceda ulteriori autorizzazioni, il suo nome è "verified" ed è di tipo boolean.

Un'altra limitazione già indicata è la possibilità per un utente non ancora autenticato di navigare all'interno del sito Facebook perché consentito dal walled garden.

Se l'account email dell'utente non è stato confermato Facebook potrebbe anche consentire l'accesso al suo network, tuttavia al momento in cui l'applicazione chiede conferma a Facebook il valore "email" non viene restituito, rendendo di fatto impossibile la creazione dell'utenza temporanea, che usa come username proprio l'indirizzo email dell'utente.

5 Conclusioni

La questione discussa in questo documento tecnico rientra nel discorso della "Open Authentication", una tematica che ha visto la luce nel 2006, quando fu elaborato da Blaine Cook e Chris Messina il primo protocollo di autenticazione aperta, dal nome oauth, pubblicato da Hammer e Lahav nel 2010 come

RFC 5849[4] e recentemente reso obsoleto dal RFC 6749.[5]

L'idea di base[6] è quella di autorizzare terze parti a gestire documenti privati senza condividere la password. La condivisione della password infatti presenta molti limiti a livello di sicurezza, come ad esempio non garantisce supporto per singoli privilegi su determinati file o operazioni, ma rende accessibile l'intero account e il pannello di amministrazione. Inoltre vige l'impossibilità di revocare l'accesso nel futuro se non cambiando la password dell'intero account. OAuth è nato quindi con il presupposto di garantire l'accesso delegato ad un client specifico per determinate risorse sul server per un tempo limitato, con possibilità di revoca.

OAuth presenta alcune limitazioni a livello di sicurezza, il server infatti raccoglierà informazioni riguardanti l'utente, il client e la loro interazione e le terrà in memoria per un limitato arco di tempo, il protocollo inoltre non garantisce confidenzialità né sulle richieste effettuate, né sui contenuti scambiati, ad esempio non garantisce che l'uso delle risorse autorizzate rimanga nell'ambito specificato. Per questo motivo OAuth suggerisce al server di proteggere le risorse tramite il protocollo TLS.

Un ulteriore problema noto è quello del phising. Il client potrebbe indirizzare l'utente and una pagina di accesso fasulla del server per richiedere l'autenticazione e ottenere le credenziali dell'utente.

6 Appendice

Il file login.php

```
    $ip=$_POST['ip'];
    $username=$_POST['username'];
    $linklogin=$_POST['link-login'];
    $linkorig=$_POST['link-orig'];
    $error=$_POST['error'];
    $chapid=$_POST['chap-id'];
    $chapchallenge=$_POST['chap-challenge'];
    $linkloginonly=$_POST['link-login-only'];
    $linkorigesc=$_POST['link-orig-esc'];
    $macesc=$_POST['mac-esc'];
?>

..Omissis...

<script>
//statusChangeCallback viene invocata da checkLoginState
    function statusChangeCallback(response) {
        if (response.status === 'connected') {
            mikAPI(response);
        } else if (response.status === 'not_authorized') {
            } else {
        }
    }

// checkLoginState è la prima funzione chiamata dal pulsante di login
    function checkLoginState() {
        FB.getLoginStatus(function(response) {
            statusChangeCallback(response);
        });
    }

    window.fbAsyncInit = function() {
    FB.init({
    appId   : 'app-id',
    cookie  : true,
    xfbml   : true,
    version : 'v2.2'
    });

    FB.getLoginStatus(function(response) {
        statusChangeCallback(response);
            });
    };

    (function(d, s, id) {
    var js, fjs = d.getElementsByTagName(s)[0];
```

```
    if (d.getElementById(id)) return;
    js = d.createElement(s); js.id = id;
    js.src = "//connect.facebook.net/en_US/sdk.js";
    fjs.parentNode.insertBefore(js, fjs);
    }(document, 'script', 'facebook-jssdk'));

// mikAPI è la funzione creata da noi che chiede il token e lo passa alla pagina confirm.php
    function mikAPI(myResp) {
        console.log('Welcome! Fetching your information.... ');
        FB.api('/me', function(response) {
                                            window.location.replace("http://ip-server-web/confirm.php?token="+
myResp.authResponse.accessToken+"&dst="+"<?php echo $linkorig; ?>");
    });
    }
</script>

...Omissis...

<fb:login-button scope="public_profile,email" size="xlarge" onlogin="checkLoginState();"> </fb:login-button>

...Omissis...
```

Il file confirm.php

```
<?php
    require('routeros_api.class.php');

    $token = $_GET["token"];
    $dst = $_GET["dst"];
    $linkFb = "https://graph.facebook.com/v2.2/me?access_token=".$token."
&fields=id%2Cname%2Cemail&format=json&locale=it_IT&method=get&pretty=0&suppress_http_code=1";
    $result = json_decode(file_get_contents($linkFb),true);
    $email = $result["email"];
    $passwd = substr(str_shuffle('abcdefghijklmnopqrstuvwxyz0123456789'),0,8);
    $nome = $result["name"];
    echo "Benvenuto ".$nome;

    if($email) {
        $API = new routeros_api();
        if ($API->connect('mikrotik-hotspot-ip', 'uapi', 'apipasswd')) {
            $API->comm("/ip/hotspot/user/add", array (
                "name" => $email,
                "profile" => "facebook-prof",
                "limit-uptime" => "00:60:00",
                "password" => $passwd,
            ));
            $API->disconnect();
        }
    }
    header("Location: http://hotspot.local/login?username=$email&password=$passwd&dst=$dst");
    die();
?>
```

Riferimenti

1 https://developers.facebook.com/docs/facebook-login/login-flow-for-web/v2.3.

2 http://wiki.mikrotik.com/wiki/HotSpot_external_login_page.

3 http://wiki.mikrotik.com/wiki/API_PHP_class.

4 E. Hammer-Lahav, The OAuth 1.0 Protocol, April 2010 https://tools.ietf.org/html/rfc5849.

5 D. Hardt, The OAuth 2.0 Authorization Framework, October 2012 https://tools.ietf.org/html/rfc6749.

6 http://it.wikipedia.org/wiki/OAuth.