How Leaked Twitter API Keys Can be Used to Build a Bot Army
Speaker

Rahul Sasi
Founder, CEO - CloudSEK

Vishal Singh
Senior Security Engineer - CloudSEK
Let's Talk Numbers

1.6 M+ Hardcoded Sensitive Tokens Found at BeVigil
Agenda

01 Key Findings & Source

02 Threats

03 Remediation
KEY FINDING & SOURCE
Our own security search engine

Step 1: Collection of Mobile Apps
Step 2: Decompiling Apps
Step 3: Building Regexes
Step 4: Find Regex Matches on Large Scale
Collection of Mobile Apps

User submissions

Google Play Store
Decompiling Apps

Open Source Android Decompilers, like JadX

Open Source Tools such as
• APKTool
• JD-GUI
THE TOUGHEST OF IT ALL

Consumer Key:
\([tT][wW][i][tT][eE][rR]\{\(0,30\)\}\{'\s\'}\{1,5\}\b\([0-9a-zA-Z]\{25\}\)b\"\s\}\{0,1\}\)

Consumer Secret:
\([tT][wW][i][tT][eE][rR]\{\(0,30\)\}\{'\s\'}\{1,5\}\b\([0-9a-zA-Z]\{50\}\)b\"\s\}\{0,1\}\)

Access Token:
\([tT][wW][i][tT][eE][rR]\{\(0,30\)\}\{'\s\'}\{1,5\}\b\([0-9]\{5,19\}\-\[0-9a-zA-Z]\{30,44\}\)\b\"\s\}\{0,1\}\)

Token Secret:
\([tT][wW][i][tT][eE][rR]\{\(0,30\)\}\{'\s\'}\{1,5\}\b\([0-9a-zA-Z]\{45\}\)b\"\s\}\{0,1\}\)

We have build our own RegeEx and grabbed & tested from mulitple sources which detects the Hardcoded keys, tokens, and secrets of the apps.
RegEx Matches on an extensive scale
Data Analysis

Uncovered **3207 apps**, leaking **Twitter API keys**, that can be utilized to **gain access** to or to **take over** Twitter accounts.
The Twitter API uses access controls such as:

- **App-Based Authentication**
  - Not Tied to an User Session

- **User-Based Authentication**
  - Tied to an User Session
For this, an OAuth 2.0 Bearer Token is used. This can be obtained by passing the API Key and Secret through the POST oauth2/token endpoint. Only 2 keys are required.
For this, the OAuth 1.0a authentication mechanism is used. This requires an Access Token combined with Access Secret. All 4 Keys are Required for this Authentication.
Naming Convention For Keys

### Client Credentials
- **Key**
  - Alternate name used would be
  - API Key
  - Consumer API Key
  - Consumer Key
  - Customer Key
  - oauth_consumer_key

### Secret
- Alternate name used would be
- App Key Secret
- API Secret Key
- Consumer Secret
- Consumer Key secret
- oauth_consumer_secret

### Token
- Alternate name used would be
- Access token
- Token
- resulting oauth_token

### Secret
- Alternate name used would be
- Access token secret
- Token Secret
- resulting oauth_token_secret
Don't Complicate API Keys

Summary

Issues

VULNERABILITIES

STRINGS

MANIFEST SCANNER

ASSETS

APKID

Don't Complicate API Keys

Strings

Severity : Rule : Description

LOW : Generic API Key : Sensitive

LOW : Google API Key : Sensitive

...gomodule">GBPaymentMercadoPagoModule</string>
  <string name="title_gbpaymentsandboxmodule">GBPaymentSandboxModule</string>
  <string name="title_gbpaymentstripemodule">GBPaymentStripeModule</string>
  <string name="twitter_consumer_key">289886</string> 8pbrwDGs1Al
  <string name="twitter_consumer_secret_key">qedqRuwRbLSB8LwCyoi</string>
  <string name="wm_api_key">SnBLu5wCyo</string>
  <string name="wm_api_secret">SnBLu5wCyo</string>
  <string name="wm_webz..."
Impact of Four Keys Leaks

User-Based Authentication:

- Read DMs
- Retweet
- Like
- Delete
- Remove followers
- Follow any account
- Get account settings
- Change display picture
What about the Remaining Apps?

2977 apps were leaking only 2 keys [App-Based Auth]:
- Consumer Key
- Consumer Secret
What are Twitter Webhooks?

The use of Twitter webhooks requires OAuth 1.0a which is sometimes also referred to as "user context authentication" which allows to make API requests on behalf of a Twitter user. You will need Access to Premium/Enterprise Twitter API to use Webhooks.
TA-DA Moment

WHAT A SURPRISE.

Notifications

All

followed you
function getTwitterProfileData() {
    var deferred = $.Deferred();
    var options = {
        consumerKey: '05pl',
        consumerSecret: '3u',
        accessTokenKey: '4X',
        accessTokenSecret: 'DgV',
        callbackUrl: "https://twitter.com";
    }
    var oauth = OAuth(options);
    oauth.get('https://api.twitter.com/oauth/request_token', function(data) {
        var requestParams = data.text;
        // cb = cordova.InAppBrowser.open('https://api.twitter.com/oauth/authorize?' + data.text
        cb = window.open('https://api.twitter.com/oauth/authorize?' + data.text, '_blank', 'loc
        cb.addEventListener('loadstop', function (loc) {
            if (loc.url.indexOf("https://twitter.com") > -1) {
                var verifier = '';
Where Problems Lies?

1. **Security Pipeline**
   Pain of setting up a proper mobile app security testing pipeline while development.

2. **Awareness**
   Lack of awareness on the scope/impact of the Hardcoded secret.

3. **Budgeting**
   Companies not spending much on doing proper security testing on mobile apps - compared to web apps.
Defending against Attacks

Mitigation

- Standardizing Review Procedures
  Ensure accurate versioning. Publication requires the code base to be examined, reviewed, and approved prior to versioning. Complying with standardized procedures prevents key exposures.

- Hiding Keys
  Variables in an environment are alternate means to refer to keys and disguise them. Variables save time and increase security. Adequate care should be taken to ensure that files containing environment variables in the source code are not included.

- Rotate API keys
  Rotating keys can help reduce the threat posed by leaked keys. Unused keys reduce the severity of invalidation. It is recommended to rotate keys every six months as existing keys get deactivated while new ones get generated.
Cyclic Process

- Standardizing Review Procedures
- Mitigation
- Hiding Keys
- Rotate API keys
Thank you!

https://cloudsek.com/