Black Hat USA 2013

Using Online Activity as Digital Fingerprints to Create a Better Spear Phisher

Joaquim Espinhara and Ulisses Albuquerque

70 W. Madison Street, Suite 1050 Chicago, IL 60602 www.trustwave.com

Table of Contents

Introduction

Despite the convoluted title, this document describes a simple approach developed by the authors that can be used to leverage the huge amounts of information from online sources such as social networks, forums, blogs, electronic mailing lists and other similar sources for social engineering purposes.

Through the combination of data mining and natural language processing techniques, it is possible to collect, analyze, index and query large datasets populated from these sources, building user profiles. These user profiles could help pinpointing communication and relationship traits that can then be used to validate further input against a known user identity. This validation can be used for both defensive (i.e., checking if suspect content matches a known profile) and offensive (i.e., supporting the production of content matching a profile) purposes.

This document describes the development process for an automated tool focused on these activities, called µphisher (*microphisher*).

Our Motivation

Many users will post constant updates to their social networks as part of their daily routines. This is information willfully provided by individuals who will in many cases take the extra effort of producing well-crafted texts in order to make their online personas look good. In many situations, it is also publicly available. The authors consider not taking advantage of such a huge, quality source for user information a waste.

Today's existing tools are mostly focused on intelligence gathering and identifying patterns which could be used to predict user behavior. Social networks add a lot of metadata that can improve significantly profiles, including relationships between users, shared interests and communication topics, as well as temporal information. However, building a graph of users that includes all this information is a challenging task without the support of an adequate tool.

Although µphisher could still be considered useful for intelligence gathering purposes, our approach is mostly focused on deceiving victims for social engineering purposes – we attempt to understand communication and relationship patterns and appearances not to anticipate user actions, but rather to mimic them.

Background

In this section the authors describe some of the concepts necessary for a better understanding of the approach taken during the development of µphisher.

Social Networks

Wikipedia defines social networks as[1]:

> *A **social network** is a social structure made up of a set of social actors (such as individuals or organizations) and a complex set of the dyadic ties between these actors. The social network perspective provides a clear way of analyzing the structure of whole social entities. The study of these structures uses social network analysis to identify local and global patterns, locate influential entities, and examine network dynamics.*

The most common social networks revolve around the concept of sharing content between users – status updates, pictures, links and similar information. Their dynamics are closer to friendships, where individuals will use the network as a communication channel for keeping in touch with someone. Facebook is probably the most famous example of such a social network.

Other networks allow users to keep updated about someone who does not necessarily share their interest back at them. Although also mostly focused on content sharing, such networks are more unbalanced in the relationships between users, sometimes mimicking idol/fan relationships. Twitter is such a social network, although other media-sharing networks, such as Instagram, also work similarly.

Finally, there are many specialized networks which focus on specific interests – Last.fm is for music aficionados, TripIt for frequent fliers, Foursquare for those geolocation addicts, and GetGlue for music, TV and movie fans.

[1] https://en.wikipedia.org/wiki/Social_network

Although the information provided by such networks about a user might be of great value to someone interested in the particular specialty, the main consumable for μphisher is user-provided textual content – which is usually limited in more specialized networks. Specialized networks can still provide useful user interaction information, though.

Social Engineering

Once again, according to Wikipedia[2], Social Engineering is defined as:

> *Social engineering, in the context of information security, is understood to mean the art of manipulating people into performing actions or divulging confidential information. This is a type of confidence trick for the purpose of information gathering, fraud, or gaining computer system access. It differs from traditional cons in that often the attack is a mere step in a more complex fraud scheme.*

One of the most commonly used social engineering techniques during security assessments is *phishing*, which consists in "hooking" a particular target into opening a document, clicking on a link or even executing a program file by embedding it into a legitimate-looking email or instant message, or through a website. Due to scheduling limitations, in many situations assessors can get quicker access to critical systems by manipulating their users rather than the systems and applications themselves.

Data Mining

While the use of social networks for social engineering purposes is not a new concept, taking full advantage of the whole amount of data provided by these sources requires more sophisticated approaches than what is currently available. Data mining can be used to explore these huge data sets to discover patterns and associations. This requires understanding the data being collected, as well as what criteria can be used to select relevant records.

The tool calls a set of records sharing common aspects relevant to a given analysis a **work set**. These represent a particular subset of the data collected: e.g., all Facebook status updates mentioning "Black Hat" in the last 3 months. This raw work set is then subject to pre-processing manipulations such as:

- **Data cleaning** consists in removing records which are not relevant to a given analysis (e.g., removing all retweets, removing all Facebook status updates posted from a mobile device);

- **Data integration** consists in correlating identities between different data sources, allowing the μphiser user to perform analysis on records from multiple sources (e.g., user @jespinhara in Twitter is user 'jespinhara' in Instagram);

- **Data normalization** consists in applying transformations to data attributes in order to make records consistent regarding comparison and correlation (e.g., all date and time attributes will be represented using ISO-8601).

[2] https://en.wikipedia.org/wiki/Social_engineering_(security)

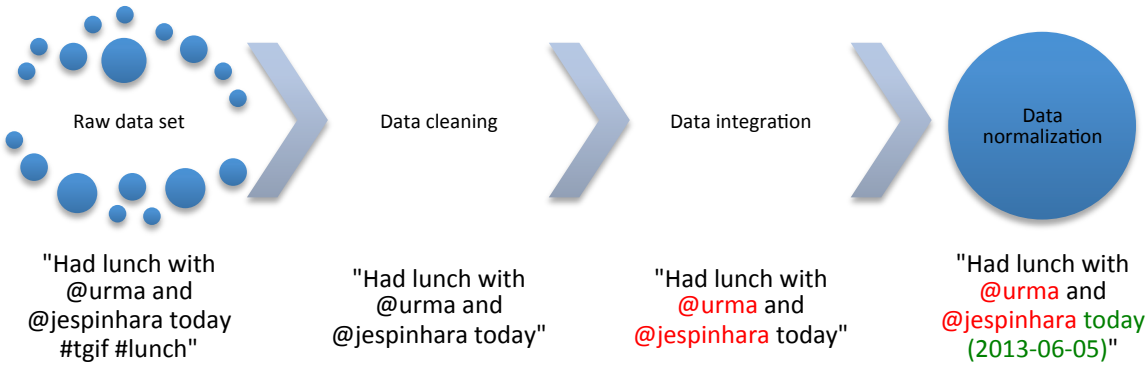

Figure 1: Work set workflow

Pre-processing is crucial in order to cross-reference data between multiple data sources. It allows the μphiser user to apply consistent eligibility criteria on records from different social networks, remove irrelevant entries and correlate identities and temporal information.

The resulting work set goes through data mining functions elected by the μphiser user which properly represent the information needed from the records, such as:

- **Association rule learning**, which consists of identifying relationships between variables (e.g., @jespinhara is commonly referenced alongside @urma when "lunch" is present on the status update);
- **Clustering**, which consists of discovering groups and structures between records which do not share a known common attribute (e.g., all status updates which mention meals during the day)
- **Anomaly detection**, which consists of identifying unusual data records which require further investigation and should most likely be excluded from analysis (e.g., "Hacked by @effffn!!!!111!")

Such techniques are known as **knowledge discovery in database** (KDD).

Natural Language Processing - NLP

Natural language processing is a set of techniques for unstructured text analysis. It has been applied widely in many areas, such as information extraction and information retrieval. Data Mining and natural language processing combined to study social networks is not a new approach, and there is extensive literature available on the topic. The key innovation in the μphisher approach is in using these techniques to support *phishing* attacks.

While natural language processing can be used to generate text as well as extracting information from it, μphisher does not attempt to automatically produce forged content. Because such content when used for social engineering purposes requires fine control over the message being delivered, μphisher instead offers assisted text writing support, suggesting words and phrasing for the user while writing new content.

It is important to emphasize that although NLP can also be used to produce content, our approach involves deceiving targets of social engineering attacks. Thus, producing content which is consistently interpreted as coming from an *unsub* is essentially the same as beating the Turing test, which is obviously a much greater challenge than what the authors propose.

Through data mining it is possible to create work sets and apply natural language processing to those specific records. μphisher uses natural language processing techniques on the textual content of those status updates from various social networks, ranking words alongside their grammar class. This allows the tool to build a dictionary of commonly used terms and their corresponding usage patterns, which is used then used to offers suggestions to users when writing new content.

HowStuffWorks

In this section we describe how the various concepts described previously in this document are combined in μphisher.

Our Approach

By analyzing content produced by a particular subject, our approach allows us the build a profile that can then be used to validate and support the production of additional content that matches this profile. This requires three distinct steps:

- Identifying the subject to profile (the unknown subject, or *unsub*);
- Collecting social network data from the *unsub*;
- Analyzing and building the profile for the *unsub*.

The Unknown Subject

The unknown subject[3], or *unsub*, is the main object of our analysis. It represents an abstract identity, mapped to various social network IDs where content produced by said subject is available. Thus, from the μphisher perspective, the unsub consists of a set of data sources tied to an ID.

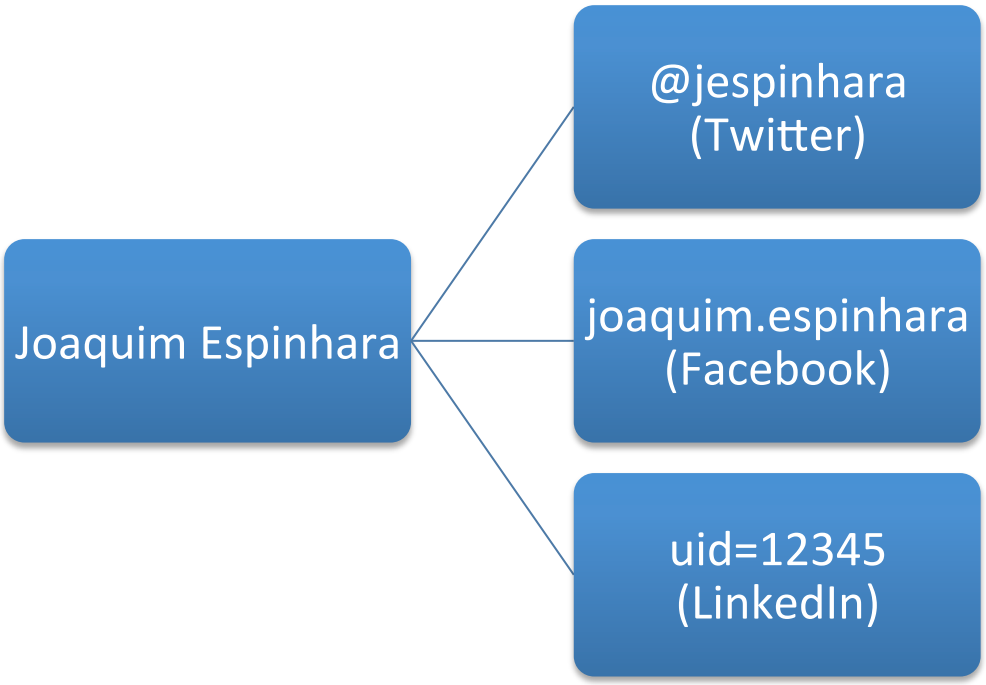

Figure 2: Unsub to social network ID mapping

[3] http://en.wikipedia.org/wiki/Person_of_interest

Data Collection

Once an *unsub* and its entire associated social network IDs are determined, as shown in Figure 2: Unsub to social network ID mapping, all data available from these sources must be collected. This information is stored as-is by the tool, since the authors consider the possibility of selecting entries based on network-specific criteria which might be lost if records were normalized during collection. Additionally, content might become unavailable due to removal or access control restrictions applied by the *unsub*, which might prevent the original content from being retrieved again if needed.

Data collection can be performed either through official APIs, which are offered by all major social network services as part of their efforts to integrate social network data into third-party applications, or through web scraping. Although the official APIs offer much nicer interfaces for automated data collection, they also add both legal and technical restrictions on the data being collected, the most common one being rate limiting. Also, because applications must be registered before being granted access to the APIs, the application ID can easily be identified and blocked by the social network, preventing access.

For Facebook and Twitter, the official APIs support OAuth[4], which allows users to use their actual identities on the social networks when requesting access to data. This can be used to leverage existing relationships between users and *unsubs*, exposing data which might otherwise be not accessible to those outside of the *unsub*'s network.

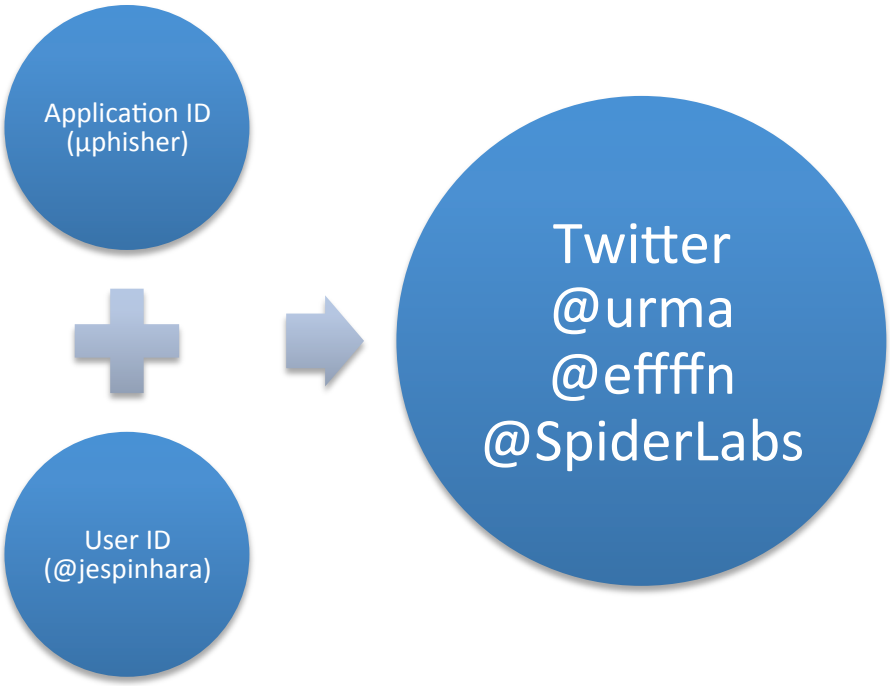

Figure 3: Leveraging OAuth with the Twitter API

Web scraping, or the extraction of content directly from web pages, does not suffer from such restrictions, but must deal with the parsing of HTML and additional data formats used by the web interface of the social networks. It must also deal with authentication (e.g., the storage and transmission of user credentials in the social network application), which is abstracted through OAuth by most APIs.

The official APIs also offer much more control over the querying of user content, such as temporal parameters. When populating a dataset for a given *unsub*, this can be used to obtain every single status update ever produced by this user. For example, the Twitter API method[5] for obtaining tweets from a given user offers the following parameters:

[4] http://en.wikipedia.org/wiki/Oauth
[5] https://dev.twitter.com/docs/api/1.1/get/statuses/user_timeline

Parameter Name	Description
user_id	The ID of the user for whom to return results for.
screen_name	The screen name of the user for whom to return results for.
since_id	Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets that can be accessed through the API. If the limit of Tweets has occured since the since_id, the since_id will be forced to the oldest ID available.
count	Specifies the number of tweets to try and retrieve, up to a maximum of 200 per distinct request. The value of count is best thought of as a limit to the number of tweets to return because suspended or deleted content is removed after the count has been applied. We include retweets in the count, even if include_rts is not supplied. It is recommended you always send include_rts=1 when using this API method.
max_id	Returns results with an ID less than (that is, older than) or equal to the specified ID.
trim_user	When set to either true, t or 1, each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object.
exclude_replies	This parameter will prevent replies from appearing in the returned timeline. Using exclude_replies with the count parameter will mean you will receive up-to count tweets — this is because the count parameter retrieves that many tweets before filtering out retweets and replies. This parameter is only supported for JSON and XML responses.
contributor_details	This parameter enhances the contributors element of the status response to include the screen_name of the contributor. By default only the user_id of the contributor is included.
include_rts	When set to false, the timeline will strip any native retweets (though they will still count toward both the maximal length of the timeline and the slice selected by the count parameter). Note: If you're using the trim_user parameter in conjunction with include_rts, the retweets will still contain a full user object.

While web scraping cannot offer a similar level of parameterization, it removes the single point of failure of the application ID being associated with every request made. Additionally, it does not require the acceptance of usage agreements, which might restrict operations the application might perform against the social network.

Analysis

During the analysis the actual data mining and natural language processing operations are performed on the raw collected dataset. It is also during the analysis that the μphisher user is able to select the subset of records that are to be used in a profile.

The first step during the analysis is to select which entries are to be used as basis for a profile. These criteria should be based on the kind of content that is to be produced. For example, if we are to forge a message about an invitation to lunch, we might be interested in any status updates that mention "lunch", "meal" and were sent between 11:00 and 14:00. While μphisher cannot decide by itself which criteria to use, it does provide an intuitive interface through which the user can change parameters and visualize the resulting work set easily.

After the work set is defined, natural language processing is applied to every entry in order to obtain a grammar classification of all words used. This allows us to distinguish between words with the same written representation but different morphological meanings (e.g., "I am going to take a walk" versus "I walk to work every day"). Words are ranked according the their usage frequency. The end result is a dictionary that allows μphisher to assist user input by offering suggestions while producing forged *unsub* content.

Morphological classification is highly dependent on language support from libraries. This means that natural language processing will work properly on text written on languages not supported by the tools used by μphisher. While English is well supported by most libraries, support for other languages is mostly subpar or non-existent.

The tool

An initial reference implementation for μphisher was created to validate all assumptions made during the conception. Written as a web application, this implementation takes advantage of OAuth[6] for user authentication and uses the official APIs for data collection.

This implementation only supports Twitter as a data source, although multiple Twitter users can be mapped to a single *unsub* identity. This limitation, however, was not as relevant as initially thought due to the way users write content when taking into consideration the technical limitations and purpose of each social network. Twitter in particular affects significantly the words and phrasing of status updates due to its 140-character limitation. This results in content produced for one particular network looking completely different from content written for another network – target audience, technical limitations and subjects vary wildly between them.

The application was written using Ruby on Rails[7], and uses MongoDB[8] for data persistence. Several Ruby gems are used to support related technologies.

When first accessing the application, the user needs to grant it access to his Twitter account in order to authenticate himself against the Twitter API. This approach offers two advantages:

- The application does not use a single user ID for its data collection needs, and thus malicious users cannot hide behind an application ID;
- Using the real user account leverages all the relationships he already has in Twitter – e.g., any *unsub* with protected tweets that has granted access to the user will be accessible.

The μphisher application will redirect the user to Twitter so he can authorize access to his data. As seen on the screenshot, the application only requires read-only access to tweets and user lists.

[6] http://oauth.net/
[7] http://rubyonrails.org/
[8] http://www.mongodb.org/

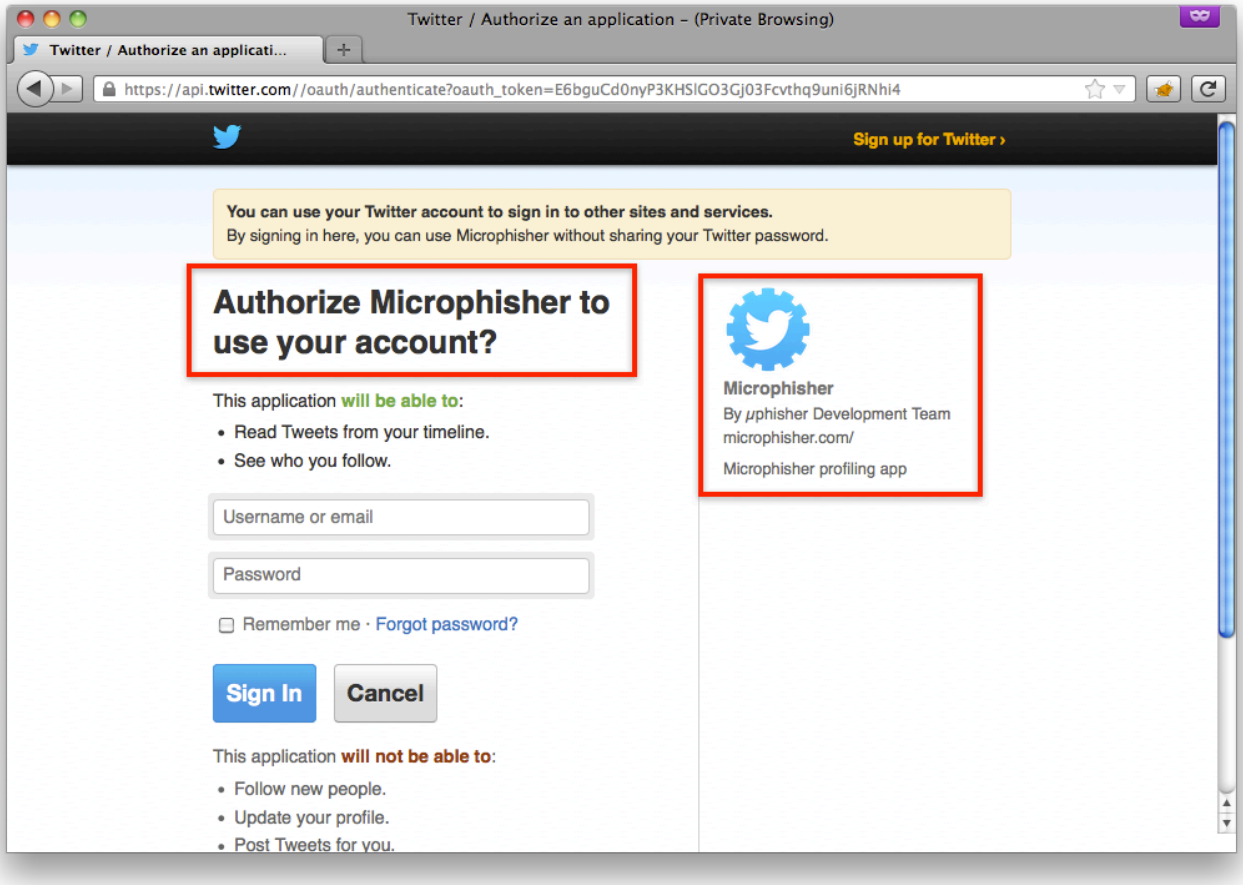

Figure 4: Granting Twitter access to μphisher

Once access is granted, the user can add any *unsubs* he is interested in profiling. Each *unsub* must have at least one associated Twitter user account. After being added, the tweets from each account are fetched using the official Twitter APIs and stored in a MongoDB database in the same format used by Twitter.

```
{
  "coordinates": null,
  "favorited": false,
  "truncated": false,
  "created_at": "Wed Aug 29 17:12:58 +0000 2012",
  "id_str": "240859602684612608",
  "entities": {
    "urls": [
      {
        "expanded_url": "https://dev.twitter.com/blog/twitter-certified-products",
        "url": "https://t.co/MjJ8xAnT",
        "indices": [
          52,
          73
```

```
          ],
          "display_url": "dev.twitter.com/blog/twitter-c\u2026"
        }
      ],
      "hashtags": [
       ],
      "user_mentions": [
       ]
    },
    "in_reply_to_user_id_str": null,
    "contributors": null,
    "text": "Introducing the Twitter Certified Products Program: https://t.co/MjJ8xAnT",
    "retweet_count": 121,
    "in_reply_to_status_id_str": null,
    "id": 240859602684612608,
    "geo": null,
    "retweeted": false,
    "possibly_sensitive": false,
    "in_reply_to_user_id": null,
    "place": null,
    "user": {
      "profile_sidebar_fill_color": "DDEEF6",
      "profile_sidebar_border_color": "C0DEED",
      "profile_background_tile": false,
      "name": "Twitter API",
      "profile_image_url":
"http://a0.twimg.com/profile_images/2284174872/7df3h38zabcvjylnyfe3_normal.png",
      "created_at": "Wed May 23 06:01:13 +0000 2007",
      "location": "San Francisco, CA",
      "follow_request_sent": false,
      "profile_link_color": "0084B4",
      "is_translator": false,
      "id_str": "6253282",
      "entities": {
        "url": {
          "urls": [
            {
              "expanded_url": null,
              "url": "http://dev.twitter.com",
              "indices": [
```

```
                0,
                22
              ]
            }
          ]
        },
        "description": {
          "urls": [
           ]
        }
      },
      "default_profile": true,
      "contributors_enabled": true,
      "favourites_count": 24,
      "url": "http://dev.twitter.com",
      "profile_image_url_https":
"https://si0.twimg.com/profile_images/2284174872/7df3h38zabcvjylnyfe3_normal.png",
      "utc_offset": -28800,
      "id": 6253282,
      "profile_use_background_image": true,
      "listed_count": 10775,
      "profile_text_color": "333333",
      "lang": "en",
      "followers_count": 1212864,
      "protected": false,
      "notifications": null,
      "profile_background_image_url_https": "https://si0.twimg.com/images/themes/theme1/bg.png",
      "profile_background_color": "C0DEED",
      "verified": true,
      "geo_enabled": true,
      "time_zone": "Pacific Time (US & Canada)",
      "description": "The Real Twitter API. I tweet about API changes, service issues and happily
answer questions about Twitter and our API. Don't get an answer? It's on my website.",
      "default_profile_image": false,
      "profile_background_image_url": "http://a0.twimg.com/images/themes/theme1/bg.png",
      "statuses_count": 3333,
      "friends_count": 31,
      "following": null,
      "show_all_inline_media": false,
      "screen_name": "twitterapi"
    },
```

```
    "in_reply_to_screen_name": null,
    "source":                  "<a                  href=\"http://sites.google.com/site/yorufukurou/\"
rel=\"nofollow\">YoruFukurou</a>",
    "in_reply_to_status_id": null
  }
```

The user can use any of the provided attributes, including geolocation, temporal information and associated Twitter user identities, to filter the tweet listing and obtain a work set. It is important to emphasize, though, that although all tweet metadata attributes can be used for filtering, only the actual status update message, contained in the "text" attribute, is actually used in the textual analysis.

After a work set is created, the actual analysis is performed. This process is run asynchronously from the web interface as it can take some time to execute depending on the number of entries present in the work set. After being finished, though, an *unsub* profile will be made available through the interface.

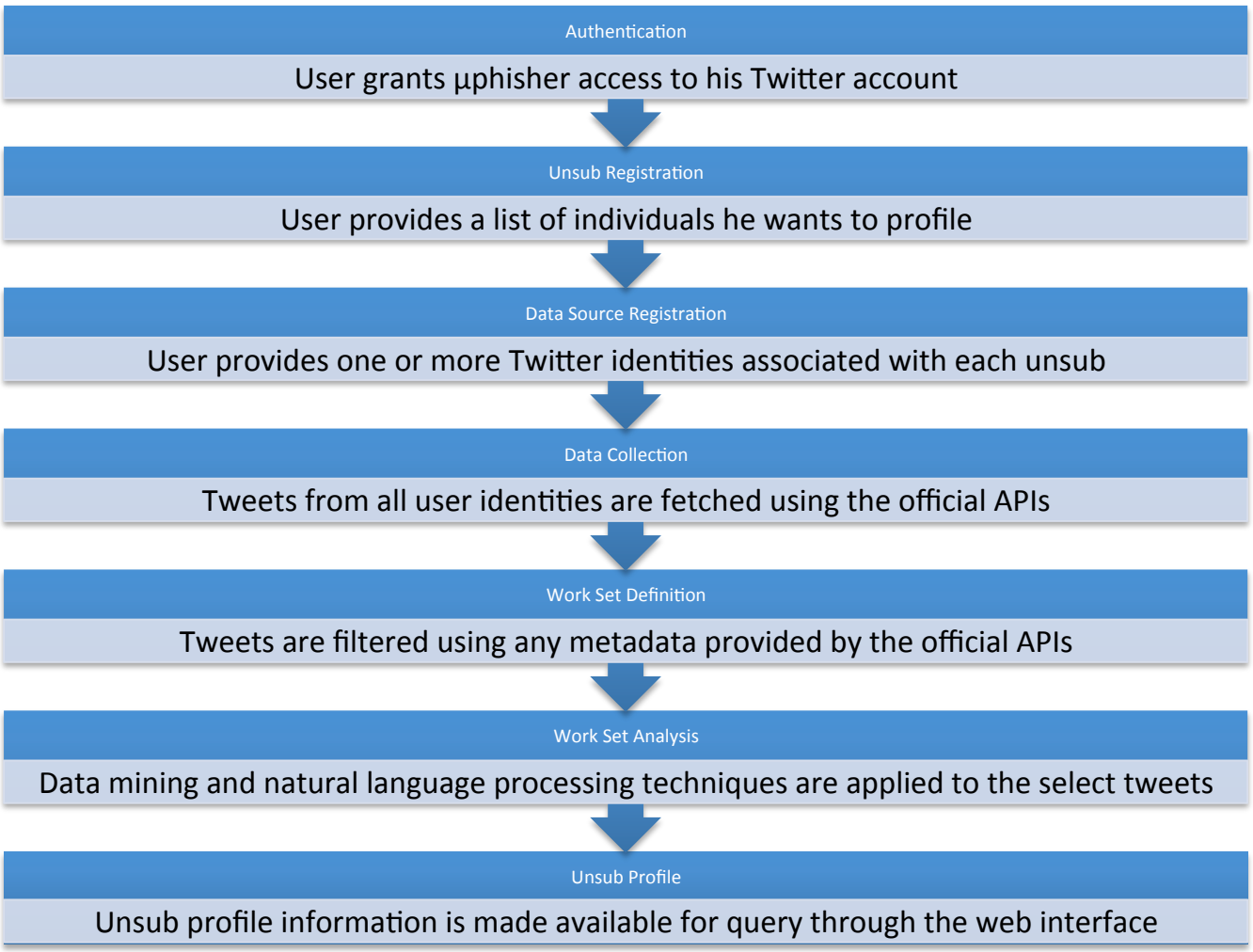

Figure 5: µphisher profiling workflow

An unsub profile can be used for two main purposes:

- Validating textual input against the profile in order to verify the probability of some text being written by an *unsub* (e.g., checking whether a given tweet was indeed authored by a profiled individual);
- Assisting the user in writing content which is similar to what the *unsub* would write regarding word frequency, sentence length, amount of sentences and phrases, amongst other criteria

The second scenario is the main feature of μphisher – being able to support the user in writing text that looks and feels like original *unsub* content is a novel facility for social engineering software.

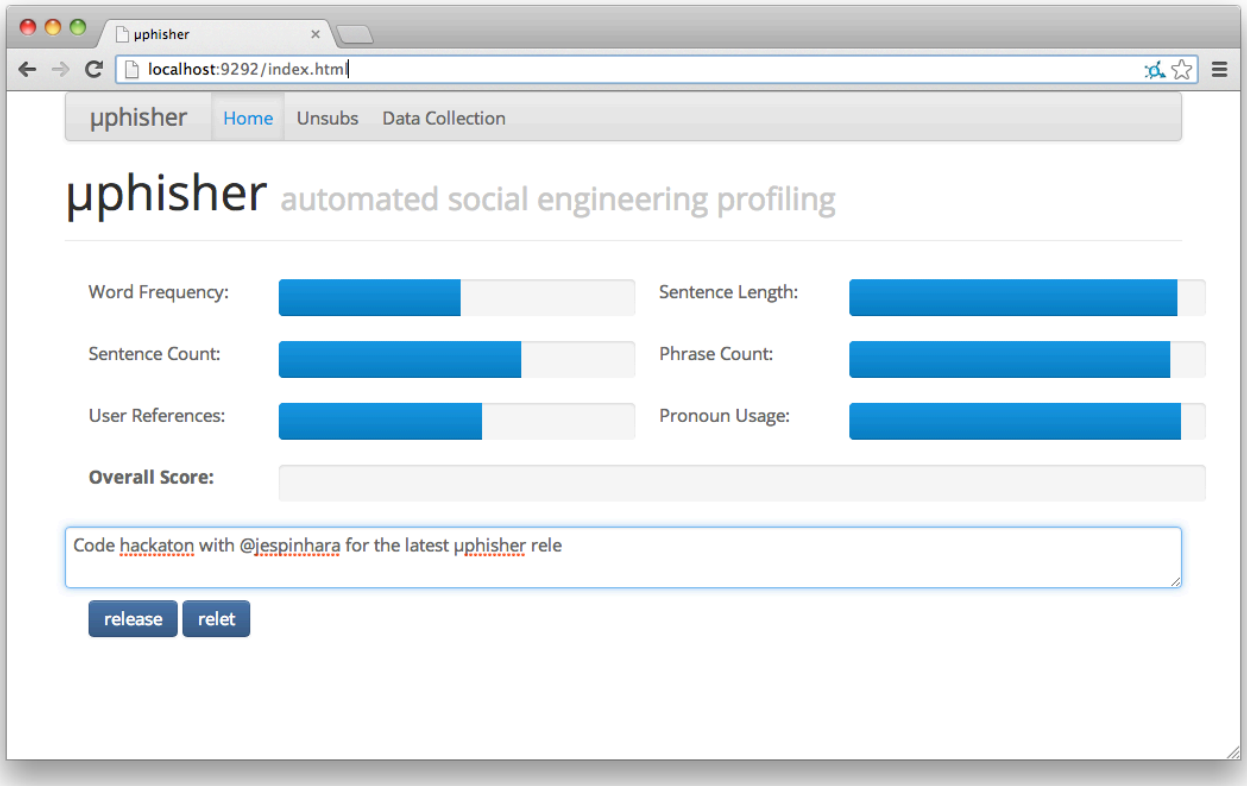

Figure 6: Assisted text input

The screenshot in Figure 6: Assisted text input shows the assisted input for forged *unsub* content. The interface suggests words which are frequently used by the *unsub* in the context of text similar to what is being input, and several metrics about it are show above. This provides the μphisher user instant feedback on the quality of the content being produced.

The code for μphisher is released under the GPLv3 license, and can be downloaded at https://github.com/urma/microphisher

Conclusion

While the use of social networks, data mining and natural language processing for intelligence purposes is not new, the social engineering approach taken by the authors provides a new and unique perspective regarding the production of targeted content. The current set of metrics was chosen due to the availability of supporting libraries and frameworks, but could be easily extended to provide more sophisticated information on any kind of textual content available online.

The provided μphisher implementation is just a reference, and the techniques used should be easily extendable by the community, as the code is being provided under a GPLv3 license. All contributions are welcome.

Future Work

The current implementation can be extended in a number of ways, including:

- Support for additional data sources (e.g., social networks, RSS feeds, blogs, emails, text files);
- Addition of machine learning techniques to improve suggestions for assisted input;

References

The Natural Language Processing Group at Stanford University
http://nlp.stanford.edu/

Introduction to Information Retrieval
Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze

Mining the Social Web
Matthew A. Russell

Natural Language Processing with Python
Steven Bird, Ewan Klein, Edward Loper

Natural Language Toolkit
http://nltk.org/
https://github.com/louismullie/stanford-core-nlp

About Trustwave

Trustwave is a leading provider of compliance, Web, application, network and data security solutions delivered through the cloud, managed security services, software and appliances. For organizations faced with today's challenging data security and compliance environment, Trustwave provides a unique approach with comprehensive solutions that include its TrustKeeper® portal and other proprietary security solutions. Trustwave has helped hundreds of thousands of organizations--ranging from Fortune 500 businesses and large financial institutions to small and medium-sized retailers--manage compliance and secure their network infrastructures, data communications and critical information assets. Trustwave is headquartered in Chicago with offices worldwide. For more information, visit https://www.trustwave.com.

About Trustwave SpiderLabs

SpiderLabs is the advanced security team within Trustwave focused on forensics, ethical hacking and application security testing for our premier clients. The team has performed hundreds of forensic investigations, thousands of ethical hacking exercises and hundreds of application security tests globally. In addition, the SpiderLabs Research team provides intelligence through bleeding-edge research and proof of concept tool development to enhance Trustwave's products and services. For more information, visit https://www.trustwave.com/spiderLabs.php.

Contact

Corporate Headquarters

70 West Madison St.
Suite 1050
Chicago, IL 60602

P: 312.873.7500
F: 312.443.8028

EMEA Headquarters

Westminster Tower
3 Albert Embankment
London SE1 7SP

P: +44 (0) 845 456 9611
F: +44 (0) 845 456 9612

LAC Headquarters

Rua Cincinato Braga,
340 nº 71
Edificio Delta Plaza
Bairro Bela Vista -
São Paulo - SP
CEP: 01333-010 - BRASIL

P: +55 (11) 4064-6101

APAC Headquarters

Level 26
44 Market Street
Sydney NSW 2000,
Australia

P: +61 2 9089 8870
F: +61 2 9089 8989