Methodology Issues in ML-based Phishing Detection

Preliminary Report

Problems with previous approaches

- 1) Internal validity: Alexa set contains numerous malicious URLs (adware, redirects), yet is assumed “benign” in most papers
- 2) External validity: Phishing URLs are incredibly transient, and frequently go offline permanently, extracting features at the HTML-level is time sensitive. Additionally, a fuller set of canonical HTML pages needs to be captured as malicious sites generally mimick other sites to be effective
- 3) Construct validity: Alexa consists of host names, Phishtank full URLs
- 4) Statistical validity: word vector approaches generate too many singleton "words" as URLs have no spaces or other indicators of word/phrase ending.

Problem: Inflated Accuracy

- Logistic Regression using word vector of URL:
- Base URLs (Alexa) compared to Full URLs (Phishtank) - 93%
- Base URLs compared - 89% accuracy
- Full URLs compared - 84% accuracy
- Hypothesis was that as the construct better resembles real world usage, accuracy will drop
- Across features and algorithms, accuracy converged at approx. 85% for base-to-base, 75% full-to-full. This indicates that URLs along are insufficient except for the 93% case, which resembles a sort of global “allowed domains” list, which is both infeasible and is anathema to the concept of an open internet

Solution

- Convert the URL and HTML-level content into a vector based on salient features
- 1) Internal: Standardize Alexa URLs by dropping malicious URLs
- 2) External: Time-stamped archive of active URLs from phishing attempts, benign site home and login pages
- 3) Construct: Find canonical, full URLs for Alexa sites
- 4) Statistical: Engineer feature vectors that encapsulate the core features of a URL and page

Procedure (and scripts)

- URL-level features are generated out of the full URL and base URL, giving three modalities: Full-to-Full, Base-to-Base, and Both-to-Both
- Full cycle approach that goes from lists of URLs, obtains content, and loads them into a jupyter notebook

Steps – Gathering Content

1) google_canonical_result.py -- Takes a list of URLs (host names) and finds a canonical full URL to associate with that host. Not supported or endorsed by Google nor is Google endorsed. Secondary purpose is to drop suspicious URLs from datasets. Necessary if dataset consists only of host names.

2) get_raw_html.py -- takes list of URLs and attempts to download the HTML file. Stores as CSV. Large file warning. Records all status codes and failures.

3) remove_zero.py -- utility script to drop failed connections from dataset. Can be skipped if NAs are needed in dataset.

Steps – Feature Extraction

4) html2vec.py -- takes CSV with HTML, generates summary features. Saves two CSVs, one that still contains the original HTML, and one that is trimmed.
Features: document length, script length, style length, body length, script-to-body, number of title tags.

5) url_preproc.py -- given a dataset with url(s), generate a feature vector that summarizes core syntax of the URL. Inherits from html-level feature vector. Generates features based on host name (base url) and full url.

Features: number of periods, presence of special symbols (@, -), URL length, IP address (if site responded), number of anchors (#), number of URL parameters, number of queries, number of digits, Shannon Entropy score.

6) jupyter notebook with examples – can drop in other datasets/features

Aggregate feature set

As of 14 Oct 2020:

url, base_url, status, datetime, flag, dataset, batch, xml_doc_length, xml_script_length, xml_style_length, xml_body_length, xml_scriptbody_ratio, xml_num_titles, base_num_periods, full_num_periods, base_spec_symbols, full_spec_symbols, base_length, full_length, ip, full_anchors,base_anchors, full_params, base_params, full_queries, base_queries, full_digits, base_digits, full_entropy, base_entropy

Future direction

- Grow data set larger (more scraping from Phishtank over time to capture trends)
- More features (some are strongly correlated so not worth it)
- HTML-level feature comparisons (I have the data, just need to add it)
- Dynamic, semantic features using topic analysis to capture name brands, common terms, brand names, and other trends
- Obtain more recent version of Alexa set
- Improve scraper and data backend (the csv files are massive, hdf5 may be a better solution)
- Obtain access to other URL sets