Abstract—It is well known in the computer security field that Malicious URLs or links are commonly used to mount cyber attacks, creating a greater need for detecting these malicious URLs and warning possible users of their dangers. Email clients are a popular source through which malicious links can permeate their attacks, thus there has been a large effort in incorporating malicious link detection software to help thwart these links from being received through a client. However all of these softwares are limited to surface level analysis of these links given the possible side-effects activating a link may cause to the intended recipient. We propose a model that could be analyzes a specific link and determines with 89.4% accuracy whether this link contains a side-effect. We also discuss possible solutions for links sent through email in order to better this model over time.

I. INTRODUCTION

Proper analysis of emails has now become a necessary feature in order to protect users from possible malicious links. These malicious links are sent to a user’s email address which may be used to mount cyber attacks such as spamming, phishing and malware [2]. In order to detect these attacks, many url scanning softwares evaluate each url through evaluation of the link’s heuristics i.e. textual properties, structures, etc. and even rely on blacklisted urls. While these methods can be successful, they are not robust enough to catch sneaker links that are well disguised as benign. While a clear form of analysis would be to analyze a link by executing it’s request. The scope of the paper prevents this as many links sent in email’s often contain Recipient Side Effects such that the activation of a link causes a change for the user the link was sent to (unsubscribing, confirming and account etc.).

The purpose of this paper is to formally define a Recipient Side Effect and use this property as a predictive task for a given link from an email. We also compare the predictiveness of two different models: Neural Networks and Random Forests, as well as offer guidelines for websites sending recipient side effect links to be better identified by such models. To the best of our knowledge, the analysis of recipient side effect links has not yet been explored.

II. RELATED WORK

To our knowledge, there is no precedent in published literature for identification of Recipient Side Effect links. However the problem of identifying links with malicious intentions has been well explored. While this is outside of our intent, we found studying the existing research in this field beneficial to identifying methods that have been tried and their levels of success.

In a phishing attack, an attacker often models their message off of messages from existing, reputable companies and asks for private information to be entered by an unsuspecting user. Natural language processing has been successfully applied to identify phishing attacks in emails [5]. Verma et al were able to use content like the header of the email, the links within it, and the text of the body of the email to tell the difference between emails that was designed to incite an action, versus a purely informational email.

Feature based approaches were also performed well for identifying malware and phishing schemes[3]. Features that were successful in identifying an attacker versus a benign email included format of the email, punctuation contained in the link, as well as the domain(s) of the sender. The random forest classifier outperformed other classifiers on the task with support vector machines also yielding low false positive and negative rates [3].

Of additional interest to us was work that evaluated system state changes when exposed to malware, identified by executing malware in a contained virtual environment to record its behavioral fingerprint or set of the state changes that result from the execution of malware, including file modification, created processes, and network connections[1]. This method was found to be more consistent than existing anti-virus software classifications[1]. We believe recipient state change relevant for future work to expand upon our methods for identifying Recipient Side Effects.

We did find record of Recipient Side Effect links affecting malware detectors in the wild [8], and thus believe our work to be relevant to continuing mailing list service and malware detector synergy.

III. SETUP

A. Definitions

We will start by forming a more concrete definition of a Recipient Side Effect that will be used in the remainder of the paper for labeling links. We will now refer to a Recipient Side Effect a an RSE for the remainder of the paper.

1) Recipient State: First let us define a Recipient State. A Recipient State represents the state of a single email address as the union of website states that any website using this email as an identifier contains. These website states are defined by the websites themselves and may vary drastically.

Let us look at an example for a better understanding. Assume an email address analysis@mail.com is subscribed to the political and daily newsletters from bbcnews.com and has an account associated with it on facebook.com. Then the website state of analysis@mail.com defined by bbcnews.com would be:

{political newsletter subscriber, daily subscriber}
Likewise, the website state defined by facebook.com would be:

\{account holder\}

Thus the Recipient State of analysis@mail.com would be:

\{political newsletter subscriber: bbcnews.com, daily subscriber: bbcnews.com, account holder: facebook.com\}

It is important to remember the state account holder would be defined by facebook.com and may tie an email address to many identifiers within their domain (friends, profile pictures, event invites, etc.). We keep this definition general in order to allow us to abstract the actual details of a state to allow for a wide range of RSE instances. One recipient state change we do not count are analytic based state changes (websites tracking if you clicked the link, or opened an email). We decided these states were out of scope given analytic data is often kept internal and thus invisible to the user’s known state.

It is also important to note that a recipient state could refer to multiple users, if they are all using one single email address as their identification on a website. In the context of RSE analysis, we focus on an email address as a single being rather than a user.

2) RSE: An RSE is property of a link that is contained in the body of an email. A link is an RSE (RSE = True) if and only if execution of this link results in a change of the recipient state without any further actions required by the executor. A simple example would be an "unsubscribe" link such that a single click on this link results in the user tied to the email recipient to be unsubscribed. Note that the unsubscription of this email address required no further actions by the executor of the link i.e. no "Click here to confirm" links after the response loaded. Throughout our research, we discovered that solely observing links (not executing) made it seemingly impossible to distinguish an RSE link (unsubscribe) and a link that is intended to perform the same state change as a corresponding RSE link, but requires one further action by an executor (unsubscribe with a confirmation). Because of this we chose to also define pseudo-RSE (pRSE).

3) pRSE: A pseudo-RSE (pRSE) can be thought of as a superset of RSE. Thus a pRSE is either:

1) An RSE
2) A link which has the same intention as an RSE, but requires one further action to complete the recipient state change

The important term in the second definition is intention. This will help distinguish links which do not have the intention of performing a recipient state change, but may contain possible recipient state changing actions in their response from links that do have the intention of changing the recipient state but require one more action. A simple example of a link without intention would be a YouTube link. After executing this link, the response (seen in Figure 1) will be the page containing the video, as well as a button "subscribe", based on the definitions above, subscribe would result in a recipient state change, however the intention of the YouTube link was to load the video, thus this link would not be a pRSE.

Fig. 1: Link response page containing a possible recipient state change action "subscribe".

Once again, an example of a pRSE link, which was also mentioned above would be an unsubscribe link that loads a response with an action to confirm the unsubscription. This is a valid pRSE because if all actions required for completing the purpose of the link were combined, it would be identical to the RSE unsubscription link.

B. Constraints

When determining if a link is a RSE/pRSE, we chose to analyze link responses with browser cookies enabled. This allows us to properly label RSE/pRSE links that are tied to accounts (i.e. Facebook state changes that require a user to be logged in). This also allows for a more general model that could be used within email virus detectors that may be executed within a browser who in turn might make use of these cookies during analysis. While current software we aim to incorporate our model into are implemented server side, this will allow for flexibility in the future.

C. Predictive Task

Now that we have defined the properties RSE and pRSE, we can define two predictive tasks:

1) Given an email link, is this link a RSE
2) Given an email link, is this link a pRSE

Using supervised learning techniques, we generate two models to predict these tasks. We generate features for these models based on available information given by an email. It is important to note that an email link only refers to links present in the body of an email. This excludes any resource requests such as image loading urls, given these links are used
to compose an email and could at most track analytic state changes (seeing if a user opened an email) which we earlier explained was out of scope.

It is important to note that our model will favor false-positives. We chose to favor false-positives given a link falsely labeled RSE can still be analyzed by virus detectors (without execution of the link), while a link that is an RSE but labeled not may cause a virus detector to execute the link and cause unwanted recipient state changes.

IV. DATASET

A. Sources

Originally we hoped to create a dataset derived from 1. a combination of our personal emails as well as 2. a publicly available dataset. The best dataset we could find online was the Enron dataset, after analysis of these emails, we came to the conclusion that the dataset would not be beneficial in training our model. This is because the dataset mostly contained outdated links to urls that either no longer existed, or were expired. Our analysis required working links in order to properly label with RSE/pRSE. Thus we chose to not add the Enron dataset to our working RSE/pRSE dataset.

Our second source of personal emails came from six separate email stemming from Yahoo and Gmail clients. These personal emails came from each research in the group and were used for either personal, business or education relations. In addition a small set of emails were retrieved from other participating UCSD Graduate students.

After labeling our separate personal emails, we also came across the issue of repeating email links from common domains. While emails were quantitatively large, the diversity of link were in fact lacking. In order to diversify our dataset and avoid over-fitting the model, we set up new email clients and performed any linking action possible (registering, subscribing, posting, etc.) on the top 500 US Alexa websites. We were able to successfully link email clients to 342.

In conclusion our dataset are a collection of emails from 342 of the top 500 US Alexa websites as well as emails from personal, educational or business clients. All of these emails were sent to either a Yahoo or Gmail client. This was a total of 1571 emails we analyzed. After discarding corrupted email objects our final dataset consisted of 1521 emails, nearly 24 thousand links, 3% of which were classified RSE.

B. Data objects

A script was made using Python’s IMAP library to pull all emails from the described clients and map them into individual data objects. Given our model is focused on analyzing each individual link, it is important to note that in our predictive model, one unit of data corresponds to one link, thus we have a one to many correspondence in the mapping of email to model data. However in order to preserve space, we represented our data with one JSON object per email. The JSON object is made up of the following attributes:

1) **From** (String): The address of the sender of the email

2) **To** (List of Strings): The address(es) of the recipients of the email

3) **Subject** (String): The text of the subject for the email

4) **Cc** (List of Strings): The address(es) of the Cc’ed recipients of the email

5) **Bcc** (List of Strings): The address(es) of the Bcc’ed recipients of the email

6) **Content-Type** (String): ‘text/html’ if the email was available in html, otherwise ’text/plain’

7) **Content-Length** (Int): The length of the body of the email

8) **Urls** (List of Objects): A list of urls that were found within the body of the email (Note each of these corresponding to one unit of data).

   a) **Url** (String): A url found within the body of the email

   b) **RSE** (Boolean): Labeled true if the corresponding url is an RSE OR pRSE.

   c) **pRSE** (Boolean): Labeled true if the corresponding url was not an RSE but fit the definition of a pRSE.

Given the topic of automatically identifying a link as RSE is currently undiscovered, and is in fact what we are trying to accomplish, each of the RSE/pRSE links were required to be manually labeled by a researcher. This was performed by clicking on each link and manually labeling RSE to be true based on the definition listed above and pRSE to be true based on the definition above. It is also important to note that RSE is a super-set of pRSE, i.e. any link classified as RSE is automatically a pRSE as well. While we considered outsourcing this work, we felt a lack of confidence in being able to properly describe all instances of a RSE/pRSE and felt it would be better to work through the emails as a team.

The attributes shown above were chosen to describe the data given they were general features that were available through IMAP and were available for every email encountered (Note some attributes were only available through certain clients or certain types of emails that would not be general enough to be a part of a good predictive feature).

C. Data Pruning

After combining all data object into one data set, we further pruned the set in order to avoid any possible over-fitting of our models. We replaced multiple occurrences of a link (within the same email) with just one instance of that link. We also removed any resource links (i.e. references to images that an email client would serve as part of an email). As mentioned above, these links were out of scope for our predictive tasks, thus should not be included in our dataset. Note that in order to automatically retrieve all links from a body of email, our robust regex needed to account for all possible types of links. While it would have been beneficial to automatically filter out resource links at the data extraction level, the characteristics of resource links were not distinctive enough to differentiate between real links and thus posed a risk of our extractor missing RSE links, which we could not afford.

V. RELEVANT FEATURES

From the manual analysis of the dataset, we chose the following features as relevant to our predictive task:
1) Number of links in email
2) Length of the link
3) Randomness of the link (entropy)
4) Maximum randomness of all forward slash separators
5) Split link on forward slashes, each string is a feature
6) Number of query parameters in a link
7) Visible text of link (to unsubscribe click HERE, optout)
8) Surrounding of the link (analysis of N words/punctuation around link). We can vary N. Match the words to certain set (to unsubscribe click)
9) Presence of user email in link
10) Identifier query arguments (id=, user=, etc.)
11) Presence of more than one recipient (cc,bcc)

VI. MODEL DESIGN

We drew from previous work in phishing email detection in our model design. We wanted to experiment with both natural language processing and feature-driven approaches for RSE classification.

A. Neural Networks

We hypothesize that the action an RSE contains is given context by the words surrounding the link. It is also likely that this context is immediate, in that a string of only N-words is sufficient to determine the action. Consider the example of “To unsubscribe click here.” The problem is analogous to the binary classification of word N-grams, where given a sequence of words, one is asked to make a simple yes or no determination such as writer sentiment (positive or negative) or grammar (correct or incorrect). Neural networks have been used successfully for this task [9], [10].

![Neural network model](image)

We modified a design previously used for text sentiment classification to make our network model shown in Figure 2. The word sequence is fed into an embedding layer that maps each word to a continuous vector for input into the neural network. The neural network consists of a 1-dimensional convolution layer to look for multi-word features. The output of this layer is pooled and fed into a fully-connected layer of rectified linear (ReLU) units. The output of this layer feeds a single sigmoid output neuron that makes the binary classification. Dropout is used to try to prevent over-fitting.

B. Random Forests

Random Forest Classifiers are an ensemble learning method for classification. They consist of a collection of decision trees, which on their own are likely to over-fit on training data. However, an ensemble of decision trees are able to overcome this issue and result in a well qualifying classifier.

C. Support Vector Machines

Support Vector Machines (SVM) are a set of supervised learning methods that can be used for classification. SVMs are effective at trying to find good class boundaries by focusing on support vectors which are data points near the boundary. SVMs are also effective in high dimensional spaces.

VII. EVALUATION

To test the performance of different designs, we reserved 20% of our emails as a test set. As the initial incidence of RSE links was only 3%, we altered this distribution to at least 70% non-RSE to 30% RSE prior to training or testing our models.

A. Neural Network

We formed the word sequences by clipping N-words preceding the link, up to N-words from the text of the link, and made up the remaining words from text following the link, up to maximum length 2N. As the emails contained content other than text, we used a simple replacement policy to try to capture this content. Images were replaced with the word “image”, links with the word “link”, and numerical values with the word “digit”. Email bodies were scraped using BeautifulSoup for HTML-encoded emails or regex for plain text. Due to the different encodings used, we could not process 22% of the links with either of these methods and they were excluded. Then, all word segments were encoded by reverse frequency of incidence in the dataset. We then duplicated random RSE word segments until the training set contained 70% non-RSE to 30% RSE. This was repeated for the test set.

We implemented the neural network using the Keras Python library run on Theano with Cuda used for GPU-acceleration. We chose to use N=5 for a total segment length of 10 words. We used a vocabulary size of 3000 most frequent words. The model was trained using small batches of 100 word segments. Error was determined using binary cross entropy, appropriate for a binary classification problem. Accuracy over 24 training iterations is shown in Figure 3. The model achieved 99.5% accuracy on the training set but only 89.1% accuracy on the test set. The model was highly biased to classifying non-RSE. Only 1.2% of the classifications were false positives, with the remainder of the errors caused by false negatives. It was however able to classify 67.3% of the RSE links correctly. We reason that this error has a lot to due with the inability of
the model to generalize what it learned to the test set. Many of our sources of RSE were singular, thus they could have been assigned to just the test set with no similar examples in the training set. A larger dataset would thus help improve accuracy as the model would find more similar emails to train on in the test set.

We repeated the experiment with the pRSE label, but found our accuracy decreased as shown in Figure 3. This result surprised us because we expected pRSE to be the superset and thus easier to classify. We reason this has to do with the fact that many of our pRSE are attached to images, so our replacement policy may have been too simplistic and not given the network enough information to make the classification. There could have been more inconsistencies in pRSE classification between different people or over time. Additionally our policy for creating word segments can yield the same word segment for links located at the end of an email, where many pRSE are located. Because this approach is text-based, sender-coined words with pRSE pose problems as with too low incidence frequency they get excluded from our vocabulary. For example, one mailing list service uses SafeUnsubscribeTM as the text of the link to unsubscribe from their service.

We experimented with modifying the network architecture to include long short term memory (LSTM) units instead of the fully connected layer. This yielded a minor improvement of 89.4% accuracy on the test set, corresponding to 67.7% RSE correctly classified, shown in Figure 4. LSTM units help the network store information about features it has seen some time back in the sequence. The minor improvement indicates the network is making most of its determination from the presence or absence of features alone, which makes sense for a short sequence.

B. Random Forests and SVM Classifiers

Evaluation of the models was based on estimating Mean Squared Error (MSE) for two validation sets of the data, while the actual training of the model was performed with a training set. However, as these models were too biased predicting pRSE for a 70% non-RSE to 30% RSE distribution, this distribution was further reduced to 55% non-RSE to 45% RSE for which final results are reported.

VIII. Discussion

As could be seen from our classification of RSE it is clear that there is a lack of symmetry between websites and how they handle RSE links. We propose several alternatives that would help differentiate RSE links from non RSE links that can be pursued by website owners. Many websites already implemented some of these alternatives.

1) Replace RSE links in emails with pRSE by adding an additional step (e.g. button click) on the referred page.
Since this behavior is already common, following it would add more uniformity to link behavior. However, it could hurt user experience.

2) Implementing RSE through JavaScript script on confirmation page. In this case after clicking on the link, the page would be loaded containing a JS script which would send a confirmation request to the server after the page fully loads in a browser. This would not hurt user experience since it would look exactly like an RSE, would not require additional steps, and malware analyzers can request HTML page and analyze only static content without running scripts. This is the most convenient option for both user and security companies.

3) Use cookies for user authentication or request login if no cookies are found. This could be used by itself or combined with any of the two previous alternatives. Authentication provides additional security since no one besides the user can perform an RSE, and malware detectors can safely analyze a link. In some cases this approach could be irrelevant since some services do not use user authentication in the first place. For example, a service may subscribe a user to a newsletter using only user token, it performs some action and that is when user receives an email, they can click on the link and open a page in a browser with some program (e.g. browser extension) which can analyze it and predict whether it was RSE or not. This result is sent to the email model which could update itself according to the received information. So, if the email model predicted links as non-RSE and the page model predicted RSE with high probability, the email model would update itself to predict such links as RSE in the future.

IX. FUTURE WORK
There are several directions for future research:

1) Further investigate the propagation process of RSE. Particularly, we want to find when a link click becomes a RSE for different websites and services. It could be the case that when the server gets GET request containing user token, it performs some action and that is when it becomes a RSE. Or, as discussed in the Discussion section, RSE could arise when the browser renders the page and JavaScript makes a request to the server. Also, a link can redirect the user, and if the model can predict if a link is a redirect, we can request the redirected link (get real link which can lead to RSE) and analyze it to predict if the original link leads to RSE/pRSE.

2) Continue to increase the labeled dataset with greater diversification. It would help to significantly increase the accuracy of models and predictions. This could be accomplished via further automation of the labeling process and outsourcing link labeling to services such as Amazon Mechanical Turk. But in the latter case, we would need to come up with a result verification method.

3) Try other models for analysis and combine different models to increase prediction accuracy.

4) Replace supervised learning with unsupervised. This can be done by creating a system consisting of two models: an email model and a page model. The email model could run as a middle man service and would predict for each link whether it is a RSE or not. When the user receives an email, they can click on the link and open a page in a browser with some program (e.g. browser extension) which can analyze it and predict whether it was RSE or not. This result is sent to the email model which could update itself according to the received information. So, if the email model predicted links as non-RSE and the page model predicted RSE with high probability, the email model would update itself to predict such links as RSE in the future.

X. CONCLUSION
This project proposes a novel definition for link Recipient Side Effect (RSE) and demonstrates the feasibility of applying machine learning techniques to classify RSE on a small dataset of 1521 emails. RSE links comprised only 3% of links in this dataset so their frequency was increased prior to training and testing models. We tested both natural language processing and feature driven approaches. We achieved a maximum accuracy of 89.4% with the former which corresponded to a false positive rate of 1.7% and a false negative rate of 32.3%. We were unable to lower the false negative rate through manually selecting features and training random forest and support vector machine classifiers. However, we believe accuracy can be improved by increasing dataset size, combining models, and future support for online unsupervised learning.

REFERENCES

Fig. 7: Confusion Matrix for SVMs