Identifikasi dan Deteksi *Phishing Email* Menggunakan *Natural Language Processing*

Dicky - 18217041

Sistem dan Teknologi Informasi - Institut Teknologi Bandung

***Abstract*—Dalam masyarakat kontemporer, *email* menjadi salah satu fitur Internet yang paling banyak digunakan untuk saling berkomunikasi satu sama lain. Namun, masyarakat harus berhati-hati karena penipuan melalui *email* atau yang dikenal dengan *phishing email* telah terbukti menjadi salah satu serangan yang efektif selama bertahun-tahun untuk memperoleh informasi sensitif dari korban. Pada makalah ini, akan dijelaskan 2 pendekatan dalam mengidentifikasi dan mendeteksi *phishing email* yang memanfaatkan salah satu bidang kecerdasan buatan, yaitu *Natural Language Processing* (NLP) dengan cara menganalisis informasi dasar dari *email* untuk memverifikasi kebenaran dari *email* tersebut.**

***Index Terms*—phishing email, Natural Language Processing, PhishNet-NLP, PhishCatch**

I. Pendahuluan

I.1. Latar Belakang

Dalam masyarakat kontemporer, *email* menjadi salah satu fitur Internet yang paling banyak digunakan untuk saling berkomunikasi satu sama lain. Namun, di balik kemudahan yang ditawarkan, masyarakat perlu berhati-hati karena para *social engineer* juga melihat hal ini sebagai suatu peluang bagi mereka untuk melakukan aksi *social engineering*, yaitu sebuah praktik dengan cara memanipulasi psikologi manusia untuk melakukan tindakan tertentu atau membocorkan informasi sensitif.

Phishing adalah salah satu bentuk praktik *social engineering* yang bertujuan mengumpulkan informasi sensitif dari para korban. Kejahatan ini biasanya melibatkan *phisher* mengirim banyak *email* dalam suatu waktu kepada korban yang dituju dan berharap ada korban yang tertipu sehingga mereka secara tidak sadar melakukan aksi tertentu yang dapat membocorkan informasi sensitif mereka.

Phisher ini biasanya menyamar sebagai sebuah situs web populer, bank, atau pihak administrator dari departemen TI. Kemudian mereka akan mencoba membuat korban secara sukarela memberikan informasi sensitif milik pribadi atau organisasi dengan mengklik tautan pada *email* untuk memasukkan informasi pribadi ke situs web palsu yang memiliki tampilan serupa dengan yang asli. Informasi sensitif ini dapat berupa nama pengguna dan kata sandi akun perbankan *online* atau bahkan informasi yang digunakan untuk membuat identitas palsu korban dengan tujuan untuk melakukan aksi kejahatan lainnya [1].

Pada makalah ini, akan dibahas mengenai pendekatan dalam mengidentifikasi dan mendeteksi *phising email* menggunakan *Natural Language Processing* (NLP). Terdapat 2 algoritma yang akan dibahas, yaitu PhishCatch dan PhishNet-NLP yang keduanya memanfaatkan informasi pada *email* seperti *header email, body email,* dan tautan pada *email* untuk mendeteksi kebenaran dari *email* tersebut. Selanjutnya, keduanya akan dievaluasi untuk mengetahui efektivitas dan akurasi dari masing-masing algoritma.

I.2. Rumusan Masalah

Berikut merupakan rumusan masalah dari makalah ini.

1. Bagaimana cara kerja PhishNet-NLP dalam mendeteksi *phishing email*?
2. Bagaimana cara kerja PhishCatch dalam mendeteksi *phishing email*?
3. Bagaimana efektivitas dan akurasi dari PhishNet-NLP dan PhishCatch dalam mendeteksi *phishing email*?

I.3. Tujuan

Berikut merupakan tujuan penulisan dari makalah ini.

1. Mengetahui cara kerja PhishNet-NLP dalam mendeteksi *phishing email*.
2. Mengetahui cara kerja PhishCatch dalam mendeteksi *phishing email*.
3. Mengetahui efektivitas dan akurasi dari PhishNet-NLP dan PhishCatch dalam mendeteksi *phishing email*.

I.4. Metodologi

Makalah ini terbagi menjadi menjadi empat bagian utama, yaitu pendahuluan, studi pustaka, pembahasan, dan penutup. Berikut merupakan metodologi dari makalah ini.

Tabel 1 Metodologi

No.	Bagian	Deskripsi	Isi Bagian
1.	Pendahuluan	Pendahuluan membahas tentang penjelasan singkat mengenai penulisan makalah beserta isi makalah	- Latar Belakang - Rumusan Masalah - Tujuan - Metodologi
2.	Studi Pustaka	Studi pustaka membahas tentang dasar teori yang digunakan dalam pembahasan makalah	- *Natural Language Processing* - TF-IDF - WordNet
3.	Pembahasan	Pembahasan merupakan inti makalah yang mengkaji tentang masalah yang ada pada rumusan masalah	- Cara Kerja PhishNet-NLP Dalam Mendeteksi *Phishing Email* - Cara Kerja PhishCatch Dalam Mendeteksi *Phishing Email* - Evaluasi Kinerja PhishNet-NLP dan PhishCatch Dalam Mendeteksi *Phishing Email*

4.	Penutup	Penutup membahas tentang penyelesaian dari masalah yang telah dijabarkan sebelumnya	- Kesimpulan - Saran

II. Studi Pustaka

II.1. *Natural Language Processing*

Natural Language Processing (NLP) adalah bidang kecerdasan buatan yang memungkinkan komputer untuk membaca, memahami dan memberikan makna dari bahasa manusia. Tujuan dari NLP adalah untuk membuat komputer yang mirip dengan manusia dalam hal pemrosesan bahasa alami untuk melakukan aksi tertentu seperti memproses data dan instruksi. Terdapat 6 teknik NLP yang akan digunakan pada pembahasan kali ini, yaitu *lexical analysis*, *part-of-speech tagging* (POS), *named entity recognition*, *normalization of words to lowercase*, *stemming*, dan *stopword removal*. Tujuan dari *lexical analysis* adalah untuk membagi *email* menjadi kalimat dan setiap kalimat menjadi kata-kata. POS digunakan untuk menandai setiap kata dengan *part-of-speech* yang bersesuaian. *Named entity recognition* menandai kata benda yang disebutkan dalam email, seperti nama orang, lokasi, atau organisasi. Selanjutnya, kata-kata akan dikonversikan menjadi huruf kecil dalam fase normalisasi. Tujuan dari *stemming* adalah untuk mengubah suatu kata menjadi bentuk dasar. Misalnya, kata kerja memahami diubah menjadi paham. Selain itu, terdapat *stopword removal* yang bertujuan untuk menghapus kata-kata umum seperti ini, itu, sebuah, dll. Adapun, teknik NLP semantik seperti *word-sense disambiguation* dan WordNet juga digunakan untuk mencari arti yang tepat dari sebuah kata sesuai dengan konteksnya [2].

II.2. TF-IDF

Dalam pencarian informasi, *Term Frequency-Inverse Document Frequency* (TF-IDF) adalah bobot yang digunakan untuk menentukan kepentingan dari suatu

kata pada suatu dokumen serta dalam sekumpulan dokumen. Kepentingan dari sebuah kata bertambah secara proporsional sesuai dengan frekuensi kemunculan kata tersebut dalam suatu dokumen (*term frequency*) dan berbanding terbalik dengan jumlah dokumen yang memuat kata tersebut pada suatu kumpulan dokumen. IDF mengukur seberapa umum suatu istilah yang muncul pada seluruh kumpulan dokumen. Oleh karena itu, istilah dengan bobot TF-IDF yang tinggi adalah suatu kata yang memiliki frekuensi kemunculan yang tinggi dalam suatu dokumen dan jumlah dokumen yang memuat kata tersebut cukup rendah pada suatu kumpulan dokumen.

II.3. WordNet

WordNet adalah basis data yang menggabungkan fungsi kamus dan thesaurus. WordNet berisi sinonim dan definisi dari sebuah kata. Kata-kata di WordNet disusun secara hierarkis. Blok pembangun dasar WordNet adalah *synset* yang merupakan himpunan semua kata yang mewakili konsep yang sama. Misalnya kata-kata seperti jawaban, balasan dan tanggapan adalah kata-kata yang memiliki suatu konsep yang sama yaitu suatu pernyataan (baik lisan maupun tulisan) yang dibuat sebagai respon atas suatu pertanyaan atau permintaan atau kritik atau tuduhan. Jadi kata-kata ini akan menjadi bagian dari suatu *synset* yang sama, dalam kasus ini PoS menjadi kata benda. Relasi antara sinonim dari suatu *synset* disebut relasi sinonim. Relasi hiponim dan hipernim merupakan contoh relasi lain pada WordNet. Hiponim membantu pengelompokan kata-kata secara hierarkis dan merupakan relasi yang memiliki hubungan subordinasi. Misalnya, mobil adalah hiponim dari kata kendaraan sehingga kendaraan adalah hipernim dari kata mobil. Ketika kita menelusuri lebih jauh tautan hiponim pada hierarki WordNet, kita dapat menemukan penyimpangan konsep aktual dari kata aslinya [3].

III. Pembahasan

III.1. Cara Kerja PhishNet-NLP dalam Mendeteksi *Phishing Email*

PhishNet-NLP adalah sebuah skema komprehensif yang memanfaatkan informasi pada *email* untuk menentukan suatu *email* sebagai *phishing email* atau *legitimate email*. Berikut merupakan *pseudocode* dari algoritma PhishNet-NLP.

Tabel 2 Algoritma PhishNet-NLP [3]

```
Input: SMTP server name, username, password
Output: Label for each email: Phishing or Legitimate
Fetch email from SMTP server
if (new email downloaded) then
    for each email e do
        header h = extractHeader();
        if (h indicates that e is HTML encoded) then
            decodedEmail dE=HTMLDecode(e);
        end
        parsedEmail pE = emailParser(dE);
        headerScore = headerAnalysis(header);
        linkScore = linkAnalysis(links);
        textScore = textAnalysis(text);
        cs = combineScore(headerScore, linkScore, textScore);
        if cs >= 2 then
            Output Label: Phishing
        end
        else
            Output Label: Legitimate
        end
    end
end
```

Langkah pertama adalah dengan melakukan *parsing*, yaitu PhishNet-NLP menerima *email* yang masuk dari SMTP *server* dan membagi *email* tersebut

menjadi 3 bagian: *header*, tautan, dan teks. Jika email dikodekan dalam HTML, dekodekan HTML *body email* ke teks biasa untuk analisis lebih lanjut. Selanjutnya, penjelasan algoritma PhishNet-NLP akan dibagi menjadi 3 bagian, yaitu analisis *header*, analisis tautan, analisis teks, serta kombinasi *Textscore* dan *Contextscore*.

III.1.1. Analisis *Header*

Pada *classifier* ini, analisis dilakukan pada data *header* yang telah diekstraksi. Berikut merupakan penjelasan dari bagian analisis *header*.

1. Ekstrak bagian *From* dan *Delivered-To* dari *header*. Kemudian, ekstrak bagian *Received From*. Lihat bagian *Received From* secara berurutan.
    a. Jika bagian *Received From* dari *email* berisi tanda tangan DKIM, simpan *Signing Domain Identifier* [SDID].
    b. Jika tidak, jika terdapat suatu bagian *Received*-SPF di bawah *Received From*, simpan bagian *Received From*. Selain itu, jika *query* SPF mengembalikan "pass," dan *domain* di bagian *From* menerima alamat IP sebagai pengirim yang diizinkan dalam bagian *Received*-SPF, lakukan NSLOOKUP pada alamat IP ini. Lalu, simpan nama *domain* yang sesuai dengan alamat IP ini dalam variabel SPFQuery.
    c. Jika tidak, simpan bagian *Received From*.
2. Verifikasi data:
    a. Jika bagian *Received From* pertama memiliki nama *domain* yang sama dengan bagian *Form* atau *Localhost* atau setiap akun *forwarding email*. Atau jika NSLOOKUP pada alamat IP pengirim yang diizinkan dalam bagian *Received*-SPF menghasilkan nama *domain* yang sama dengan yang tersimpan

dalam variabel SPFQuery, maka email ini merupakan *legitimate email*.

b. Jika tidak, jika bagian *Received From* pertama memiliki nama *domain* yang sama dengan nama *domain* akun email pengguna saat ini, lihat bagian *Received* berikutnya.
c. Jika tidak, maka email ini merupakan *phishing email*.

III.1.2. Analisis Tautan

Pada *classifier* ini, analisis dilakukan untuk menentukan apakah URL yang ada di *email* mengarah ke situs web yang sah. Hal ini dilakukan dengan mengekstrak semua *domain* dari tautan pada *email* dalam sebuah *array* yang bernama *Domains*. Selanjutnya, LinkAnalysis() *classifier* akan memberikan skor 1 untuk *phishing email* dan 0 untuk *legitimate email* dengan ketentuan sebagai berikut:

1. Jika panjang *Domains* adalah 0 yang menandakan bahwa tidak terdapat tautan, maka *email* ini merupakan *legitimate email*.
2. Jika *email* ini memiliki lebih dari 10 kata yang berbeda, hitung empat istilah teratas dalam *email* menggunakan skor TF-IDF. Nilai IDF suatu kata dapat diperoleh dengan melakukan pencarian pada Google untuk mendapatkan jumlah halaman web tempat kata itu muncul, atau dengan menggunakan corpus NLP standar.
3. Jika tidak, jika jumlah total kata berbeda pada *email* ini kurang dari 10, gunakan Google untuk mencari setiap *domain*. Jika semua *domain* muncul dalam 30 hasil teratas, maka *email* tersebut merupakan *legitimate email*. Jika tidak, maka *email* tersebut merupakan *phising email*.

III.1.3. Analisis Teks

Pada *classifier* ini, analisis dilakukan untuk mengklasifikasikan *email* menjadi dua kelas, yaitu informatif dan dapat ditindaklanjuti. Hal ini dilakukan dengan menganalisis teks dan memberikan *Textscore* pada *email*. Ketika terdapat informasi konteks dari sebuah *email*, PhishNet-NLP akan

menggunakan konteks tersebut untuk menghasilkan *Contextscore*. Konteks *email* didefinisikan sebagai *email* lain yang disimpan pengguna, termasuk *email* yang pernah dikirim dan diterima. Ketika opsi konteks digunakan, maka *Contextscore* dan *Textscore* dapat digabungkan secara logis untuk menghasilkan *Final-text-score*. Jika tidak terdapat informasi konteks, *Final-text-score* = 1 jika *Textscore* ≥ 1 dan *Final-text-score* = 0 jika sebaliknya. Ketika terdapat informasi konteks, hal ini akan mengikuti ketentuan sebagai berikut:

1. Jika *Contextscore* = 1 dan salah satu dari *email* yang menghasilkan skor kesamaan maksimum ditandai sebagai berbahaya oleh pengguna, *Final-text-score* = 1.
2. Jika *Contextscore* = 1 dan semua *email* yang menghasilkan skor kesamaan maksimum ditandai aman oleh pengguna, *Final-text-score* = 0.
3. Jika *Contextscore* = 0, maka *email* tersebut sangat tidak mirip dengan *email* lain di dalam konteks tersebut. Dalam hal ini, *Final-text-score* = 0 jika *Textscore* < 1 dan *Final-text-score* = 0 jika sebaliknya.
4. Jika batas bawah < *Contextscore* < batas atas, maka *email* ini memiliki kemiripan yang sedang dengan beberapa *email* yang terdapat di dalam konteks tersebut. Dalam hal ini, jika *Textscore* < 1, maka *Final-text-score* = 0 dan *Final-text-score* = 1 jika sebaliknya.

Setelah analisis dilakukan, gabungkan skor dari ketiga *classifier* tersebut dengan ketentuan bahwa skor 1 adalah *phishing email* dan skor 0 adalah *legitimate email*. Jika skor gabungan dari ketika *classifier* tersebut ≥ 2, PhishNet-NLP akan memberikan label *phishing email*, jika sebaliknya maka label yang diberikan adalah *legitimate email*.

III.2. Cara Kerja PhishCatch dalam Mendeteksi *Phishing Email*

PhishCatch adalah suatu algoritma yang bergantung pada seperangkat aturan *phishing* untuk mengklasifikasikan *phishing email*. Aturan *phishing* ini dirumuskan berdasarkan analisis terperinci terhadap *phishing email* dan berbagai metodologi *phishing* yang akan menghasilkan beberapa kategori *phishing email*. *Filter* unik akan dikaitkan dengan setiap kategori untuk merumuskan aturan berdasarkan berbagai kombinasi *filter*. Selanjutnya, setiap *filter* akan dikaitkan dengan bobot tertentu yang diperoleh berdasarkan kepentingan setiap *filter* dan pengujian regresi algoritma. Bobot dari setiap *filter* berfungsi untuk menentukan *phishing email* [4]. Berikut merupakan *pseudocode* dari algoritma PhishCatch.

Tabel 3 Algoritma PhishCatch [4]

```
getMail(SMTPserver,username,password)

if (new mail found) then
    for each message
        get recvdFrom from header
        get From from header
        if (message encoded) then
            decode message

        check for text_filters match
            set Phishrank[text_filters]

        if (recvdFrom != From) then
            set Phishrank[received_mismatch]

        find_links in each_email
        if (link found) then
            linkCharacteristics(link)
                compare anchor_tags
            check whois(recvfrom IP)
            check phishtank(phishing_link)
```

```
            check whois(phishing_link)

        if (isPhishing) then
            Alert the user
            Connect to database
            Insert the phishing email details in db

...go to next email

def decode_message()
    Remove HTML encoding

def linkcharacteristics(link)
    if (special characteristics found in link) then
        add link to phishingLink[]
    else if (IP found in link)
         add to phishrank [dotted_quad]
         add link to phishingLink[]
    else if (no_of_folders in link > folder_threshold)
         add to phishrank[no_of_folders]
         add link to phishingLink[]
    else if (no_of_subdomain in link > domain_threshold)
         add to phishrank[no_of_subdomain]
         add link to phishingLink[]
    else if (len(link) > link_threshold)
         add to phishrank[len_of_link]
         add link to phishingLink[]

def whois(link)
     if link == IP address
         contact appropriate whois server
         lookup IP
     else
         get the host name
         contact appropriate whois server
         lookup host
```

```
def phishtank(phishing_link)
     search phishing_link in phishtank
     return found/ not found
def isphishing()
     check phishrules matrix for phishing
         return true if phishing
```

Berikut merupakan penjelasan algoritma PhishCatch.

1. Ambil *email* baru dari *server* SMTP.
2. Ketika ada *email* baru yang masuk, bagi *email* ini menjadi bagian *header* dan *body*.
3. Jika *body email* telah dikodekan sebelumnya, dekodekan terlebih dahulu agar *phishing filter* dapat bekerja dengan benar.
4. Pindai *email* untuk menemukan *text filter* yang telah didefinisikan dalam algoritma. Jumlah *text filter* yang terdeteksi pada *email* akan dicatat dan menjadi bobot dari *filter* itu.
5. Tambahkan bobot dari *filter* ke Phishrank. Phisrank adalah daftar yang berisi pemetaan *phishing filter* dengan bobotnya masing-masing.
6. Periksa kesamaan *domain* antara bagian *Received From* dan *From* yang didapat dari *header email*. Jika kedua bagian tersebut tidak memiliki kesamaan *domain*, dapat diasumsikan bahwa alamat sumber telah dipalsukan sehingga bobot yang sesuai diberikan pada *filter* ketidaksamaan *domain* yang diterima.
7. Jika terdapat *hyperlink* pada suatu *email*, jalankan linkCharacteristics() untuk memindai kemungkinan kesalahan penyajian tautan.
8. Jika terdapat kesalahan penyajian pada tautan, berikan bobot yang sesuai pada *filter* penyandian tautan.
9. Periksa panjang tautan. Jika panjang tautan melebihi batas yang telah ditentukan, berikan bobot yang sesuai pada *filter* panjang

tautan. Demikian pula, jumlah direktori dan jumlah *subdomain* pada *hyperlink* diperiksa untuk diberikan bobot *filter* yang sesuai.

10. Ambil *anchor tag* untuk setiap *hyperlink* di *email* dari kode HTML *email.* Bandingkan setiap tautan dengan *anchor tag* masing-masing untuk memeriksa perbedaan antara *visual link* dan *actual link*. Jika terdapat ketidaksamaan antara *visual link* dan *actual link*, berikan bobot yang sesuai pada *filter* ketidaksamaan tautan.
11. Setelah bobot diberikan pada semua *filter*, aturan PhishCatch yang merupakan kombinasi dari *phishing filter* beserta bobotnya akan dijadikan sebagai pedoman dalam proses penentuan *phishing email*.
12. Selanjutnya, identifikasi *phishing link* di semua *hyperlink* yang ditemukan pada *email* menggunakan sistem peringkat pada *hyperlink*. Prinsip di balik sistem peringkat adalah bahwa pemeringkatan akan dilakukan pada setiap tautan berdasarkan probabilitas bahwa tautan tersebut merupakan sebuah *phishing link.* Probabilitas disimpulkan dengan melihat *filter* yang dipicu oleh *phishing link. Phishing link* yang teridentifikasi kemudian disimpan untuk keperluan pengumpulan informasi serta verifikasi dengan data di PhishTank.
13. Setelah *phishing link* teridentifikasi, lakukan pengumpulan informasi yang lebih banyak terkait serangan *phishing* secara umum dan *phishing link* secara khusus. Tujuan pengumpulan informasi ini adalah untuk menganalisis jenis serangan *phishing* yang terjadi dan memproyeksikan jenis serangan yang dapat terjadi di masa depan. Kemudian, simpan dan perbarui *database* dengan semua informasi itu.
14. Ketika *phishing email* terdeteksi, beri peringatan kepada pengguna untuk tidak mengklik tautan apapun. Lalu, simpan *phishing email* tersebut di *database*.

III.3. Evaluasi Kinerja PhishNet-NLP dan PhishCatch

Evaluasi kinerja dari PhishNet-NLP dan PhishCatch dilakukan dengan menguji kedua algoritma tersebut pada *email* milik phishing corpus [5]. Hasil pengujian dapat dilihat pada tabel 4.

Tabel 4 Evaluasi Kinerja PhishNet-NLP dan PhishCatch

Algoritma	***Effectiveness***	***Accuracy***
PhishNet-NLP	98%	99.3%
PhishCatch	80%	99%

Hasil tes menunjukkan bahwa algoritma PhishNet-NLP berhasil mendapatkan 98% untuk skor *effectiveness*, yaitu PhishNet-NLP berhasil mengklasifikasikan 98% dari *phishing email* dengan benar dan 99,3% untuk skor *accuracy*, yaitu PhishNet-NLP berhasil mengklasifikan 99,3% dari *legitimate email* dengan benar. Di samping itu, hasil tes menggunakan algoritma PhishCatch menunjukkan hasil yang lebih rendah dibandingkan dengan hasil dari PhishNet-NLP. Hal ini dapat dilihat dari hasil tes PhishCatch yang berhasil mendapatkan 80% untuk skor *effectiveness* dan 99% untuk skor *accuracy*. Namun, perlu diperhatikan juga bahwa terdapat perbedaan pada basis data dalam pengujian kedua algoritma tersebut.

IV. Penutup

IV.1. Kesimpulan

Berikut merupakan kesimpulan dari makalah ini.

1. Cara kerja algoritma PhishNet-NLP adalah dengan terlebih dahulu membagi suatu *email* menjadi 3 bagian, yaitu *header*, tautan, dan teks. Selanjutnya, *majority voting* dilakukan pada skor yang diperoleh dari masing-masing hasil analisis *classifier* untuk mendeteksi *phishing email*.

2. Cara kerja algoritma PhishCatch dengan PhishNet-NLP memiliki suatu kesamaan di awal, yaitu dengan terlebih dahulu membagi suatu *email* menjadi 3 bagian. Selanjutnya, pembobotan dilakukan pada setiap *filter* yang sebelumnya telah dikaitkan dengan kategori *phishing* untuk mendeteksi *phishing email*.
3. Evaluasi kinerja algoritma dengan menggunakan *email* milik phishing corpus menunjukkan bahwa PhishNet-NLP memiliki skor *effectiveness* dan *accuracy* yang lebih tinggi dibandingkan dengan PhishCatch.

IV.2. Saran

Pembahasan pada makalah ini menjelaskan tentang cara mendeteksi *phishing email* melalui 3 bagian utama pada *email*, yaitu *header,* tautan, dan *body*. Meskipun demikian, terdapat satu bagian *email* yang belum dibahas, yaitu lampiran *email*. Oleh karena itu, studi lebih lanjut perlu dilakukan untuk mencegah *phishing* melalui lampiran *email* mengingat bahwa penipuan melalui *email* masih sering dilakukan hingga saat ini.

V. Referensi

[1] P. Tianrui, H. Ian, and S. Yuki, "Detecting Phishing Attacks Using Natural Language Processing and Machine Learning," *2018 IEEE 12th International Conference on Semantic Computing (ICSC).*

[2] A. Shivam, K. Vishal, and S. Sithu, "Identification and Detection of Phishing Emails Using Natural Language Processing Techniques," in *Proceedings of the 7th International Conference on Security of Information and Networks,* 2014.

[3] V. Rakesh, S. Narasimha, and H. Nabil, "Detecting Phishing Emails the Natural Language Way," in *Computer Security ESORICS,* 2012.

[4] Y. Weider, N. Shruti, and T. Nagapriya, "PhishCatch – A Phishing Detection Tool," *2009 33rd Annual IEEE International Computer Software and Applications Conference*.

[5] J. Nazario, "The Online Phishing Corpus," 2004.