PhishStorm: Detecting Phishing With Streaming Analytics

4 authors, including:

Samuel Marchal
Aalto University
32 PUBLICATIONS   **368** CITATIONS

Thomas Engel
University of Luxembourg
310 PUBLICATIONS   **2,812** CITATIONS

Radu State
University of Luxembourg
298 PUBLICATIONS   **1,648** CITATIONS

Some of the authors of this publication are also working on these related projects:

Project   Intel Collaborative Research Institue for Secure Computing View project

Project   data analytics View project

PhishStorm: Detecting Phishing with Streaming Analytics

Samuel Marchal, Jérôme François, Radu State and Thomas Engel

Abstract—**Despite the growth of prevention techniques, phishing remains an important threat since the principal countermeasures in use are still based on reactive URL blacklisting. This technique is inefficient due to the short lifetime of phishing Web sites, making recent approaches relying on real-time or proactive phishing URLs detection techniques more appropriate. In this paper we introduce PhishStorm, an automated phishing detection system that can analyse in real-time any URL in order to identify potential phishing sites. PhishStorm can interface with any email server or HTTP proxy. We argue that phishing URLs usually have few relationships between the part of the URL that must be registered (low level domain) and the remaining part of the URL (upper level domain, path, query). We show in this paper that experimental evidence supports this observation and can be used to detect phishing sites. For this purpose, we define the new concept of intra-URL relatedness and evaluate it using features extracted from words that compose a URL based on query data from Google and Yahoo search engines. These features are then used in machine learning based classification to detect phishing URLs from a real dataset. Our technique is assessed on 96,018 phishing and legitimate URLs that results in a correct classification rate of 94.91% with only 1.44% false positives. An extension for a URL phishingness rating system exhibiting high confidence rate ($> 99\%$) is proposed. We discuss in the paper efficient implementation patterns that allow real time analytics using Big Data architectures like STORM and advanced data structures based on Bloom filter.**

Index Terms—**Phishing Detection, URL Rating, Word Relatedness, Search Engine Query Data, Big Data, STORM.**

I. INTRODUCTION

Phishing is currently one of the most lucrative cybercrime activities. Although accurately evaluating the financial loss caused by phishing is difficult, some surveys have been conducted, suggesting losses of several billion dollars every year. In 2007, Gartner Research estimated a $3.2 billion loss due to phishing scams [1]. Javelin Strategy & Fraud published a report [2] that identity theft led to a loss of $54 billion in 2009, mostly due to cybercrime. More recently, RSA estimated worldwide losses of $687 million due to phishing in the first half of 2012[1].

Various techniques are used to perform phising attacks, ranging from *technical subterfuges* (DNS cache poisoning, e-mail spoofing, Web server takeover, etc.) to *social engineering*. In addition various goals are sought: data, money or credential

stealing through fake Web sites, drive-by download of malware, etc. Despite this diversity, one common feature is the use of obfuscated URLs to misdirect users to fake Web sites or drive-by downloads.

Luring Internet users by making them click on rogue links that seem trustworthy is an easy task because of widespread credulity and unawareness. To cope with this threat, the best strategy is to prevent connection to phishing Web sites by the identification of phishing URLs. Other techniques like taking down phishing Web sites have been proved difficult and inefficient [3] mainly due to short Web site lifetime (around 12 hours [4]) and the use of DNS fluxing [5]. Phishing Web site short lifetime makes the protracted process of reactive blacklisting based on user reports inefficient. In addition the use of different variations in URLs for the same phishing campaign [6] complicates the task of blacklisting, as blacklists must provide a perfect match for a URL. Hence real-time malicious URL detection is a better technique for defeating phishing.

In this paper, we propose an automated real-time URL phishingness rating system to protect users against phishing content: PhishStorm. The underlying method targets identification of phishing URLs that are based on registered domains (malicious or not) that are not related to their targeted brand. To delude their victims, phishers blend many phishing keywords (famous brand, attractive words) into the remaining parts of the URL. Most Internet users are not aware of the DNS hierarchy. Seeing words like *paypal*, *ebay* or *visa* at any level of a URL will make them feel confident that the rogue link actually leads to the official Web site of these brands.

From observation of phishing URLs, we claim that there are few relationships between the registered domain and the rest of the URL. However, the words that compose the rest of the URL (low level domain, path, query) often have many interrelationships. Therefore, our approach evaluates the relatedness of words that compose a URL and highlights the differences between legitimate and phishing URLs. Previously existing solutions [7], [8], [9], [10], are not well suited to evaluating word similarity or relatedness for the Internet vocabulary. These tools, coming from the natural language processing field, usually have no entries for domain names and most of the words that compose a URL. We leverage search engine query data from Google and Yahoo to compute this relatedness.

Based on this, we define the term of intra-URL relatedness. Efficient feature computation methods leveraging distributed streaming analytics techniques and space-efficient data structure are used. These reduce the delay of detecting phishing URLs that has been observed in our previous work [11] and

Samuel Marchal, Radu State and Thomas Engel are with the Interdisciplinary Centre for Security, Reliability and Trust (SnT) of the University of Luxembourg, Luxembourg.
e-mail: {samuel.marchal,radu.state,thomas.engel}@uni.lu
Jérôme François is with INRIA Nancy Grand Est, Nancy, France.
e-mail: jerome.francois@inria.fr

[1]http://blogs.rsa.com/phishing-in-season-a-look-at-online-fraud-in-2012/

Obf. Type	Example
Type I	http://school497.ru/222/www.paypal.com/29370274276105805/ http://paypal.com.eu.compte.client.update.condst.com.br/
Type II	http://www.quadrodeofertas.com.br/www1.paypal-com/encripted/ssl218 http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run
Type III	http://cgi-3.paypal-secure.de/info2/verikredit.html http://paypal-shopping.co.il/
Type IV	http://69.72.130.98/janaseva/https.paypal.com/uk/onepagepaypal.htm ftp://212.13.144.72/SERVICE/PayPal.com/security/alert/paypal.com
Type V	http://tiny.cc/clientID00858JD8 http://goo.gl/HQx5g

TABLE I
EXAMPLE OF OBFUSCATED URLs FOR THE DOMAIN *paypal.com*

permit wider applications such as phishing email or HTTP traffic filtering. We extract 12 features from a single URL which are input to machine learning algorithms to identify phishing URLs. Our technique is assessed on ground truth data of 96,018 URLs leading to a correct classification rate of 94,91%. Finally, a phishingness score is computed for every single URL based on Random Forest classifier.

To summarize the major contributions of this paper:
 – We introduce the concept of intra-URL relatedness depicting the relation between a registered domain and the words that compose the rest of a URL.
 – We leverage search engine query data to establish relatedness between words and show that this is more suited to Internet vocabulary than existing methods.
 – We propose new features based on intra-URL relatedness and build a machine learning based approach relying on these for distinguishing between phishing and non-phishing URLs.

This paper is an extension of [11] with the following additional contributions:
 – We use distributed real-time computation technique (STORM) to infer intra-URL relatedness.
 – We use space-efficient data structures (Bloom filter) to reduce the delay in intra-URL relatedness features calculation.
 – A detailed feature engineering process is performed.

The rest of the paper is structured as follows: we start by presenting URL obfuscation techniques in Section II. We introduce the search engine query data and the metrics used to calculate intra-URL relatedness in Section III. Section IV present applications of PhishStorm and the underlying techniques used for fast features computation. We describe the datasets used for experiments and study differences between legitimate and phishing URLs in Section V. PhishStorm is assessed in Section VI, both for classification and scoring. Limitations of this technique are identified in Section VII, related works are discussed in Section VIII and we conclude in Section IX.

II. PHISHING URL OBFUSCATION

This paper assumes some knowledge about DNS organisation and operation; the reader is referred to [12], [13], [14] for the necessary background.

Phishers usually try to lure their victims into clicking on rogue URLs pointing to phishing sites or drive-by downloads. Different URL obfuscation techniques are used with the aim of hiding the real host, and particularly the registered domain, the only part of the URL that cannot be freely defined. If somebody wants to use a domain *mydomain.tld* and derive several URLs from it: *url1.mydomain.tld*, *url2.mydomain.tld/file*, he has first to register the domain *mydomain.tld* at a domain registrar, ensuring that it cannot be registered by anybody else. Assuming a phisher wants to trap PayPal users, he must use a *domain.tld* other than *paypal.com*, as this domain is already registered by PayPal Inc. The phisher must register a domain name *mydomain.tld* and try to deceive people by blending labels such as *paypal* into the rest of the URL: *login.mydomain.tld/paypal*.

A registered domain consists of two parts: a *main level domain* and a *public suffix*. A *public suffix* (or *ps*) is a domain name suffix under which an Internet user can register a name. It can be just a Top Level Domain like *.com*, *.org* or a combination of level domains like *.co.uk* or *.blogspot.com*. A *main level domain* (or *mld*) is the level domain preceding a *public suffix*. A registered domain is then: *mld.ps*. For instance in *www.paypal.com/login*, *com* is the *ps* and *paypal* is the *mld*.

The different obfuscation techniques consist of blending either the original domain name or phishing keywords into the remaining part of the URL. These keywords are usually the targeted brand, related services of the brand and other attractive words such as *secure*, *login*, *protect*, etc.

Assume a URL formed of a hostname with different level domain (*ld*), a path (*path*) and a query (*key=value*): *http://5ld.4ld.3ld.mld.ps/path1/path2/path3?key1=value1 &key2=value2*. The obfuscation often consists in blending keywords into the path, the query and the low level domain of the hostname (*5ld.4ld.3ld*). In the following we present the most used URL obfuscation techniques [15], with examples given in Table I for the domain *paypal.com*:
 • **Type I: URL obfuscation with other domain:** The *mld.ps* is a real domain name, usually registered by the phisher, while the original domain being phished is part of the path, the query or the upper level domain.
 • **Type II: URL obfuscation with keywords:** Again the

mld.ps is a real domain name, and the brand being phished and related words are part of the path, the query or upper level domain.

- **Type III: Typosquatting domains or long domains:** the *mld.ps* of the URL is the domain being phished but misspelled, with letters or words missing or added, or the domain is pronounced the same way as the original but written differently. The targeted brand can also be combined with other words to create an unregistered domain.
- **Type IV: URL obfuscation with IP address:** the URL's hostname is replaced by an IP address and the brand being phished is part of the path or the query.
- **Type V: Obfuscation with URL shortener:** A URL shortening service is used to hide the name of the real host. Such URLs are not meaningful and are mainly used in phishing attacks targeting services that use this kind of short URL, like Twitter.

We focus on the identification of the four first types of URL obfuscation techniques since our technique relies on natural language processing, which is clearly not suited to shortened URLs. The common feature of these obfuscated URLs is that the brand and some related terms are included in the path, the query and low level domain. These terms are related as these have relationships with the targeted brand and have no obvious relation with the *mld.ps* that is used for phishing. This is the opposite of what happens for a legitimate URL, where all the parts of the URL are normally related. To reveal this difference a relatedness analysis of the different part of a URL is performed.

III. INTRA-URL RELATEDNESS ANALYSIS

The intra-URL relatedness is the quantification of the relatedness among the words composing the different parts of a URL and more precisely between the registered domain and the rest of the URL. Due to the limitations of existing relatedness calculation techniques, we leverage search engine query data to compute it.

A. URL Word Extraction

The examples of obfuscated phishing URL from Type I to IV highlight a global characteristic in URL obfuscation, namely that there is no relation between the *mld.ps* and the rest of the URL. To reveal this, we split the URL in the two parts that are presumed to have no relationship: extract the *mld.ps* and separate it from the rest. As the *ps* may be composed of multiple level domain, we use Public Suffix List[2] to identify it and then retrieve the immediately preceding level domain as the *mld*. For the rest of the URL, a split according to non-alpha-numeric characters is first performed. From extracted parts composed of several words such as *paypalitlogin* in *http://sezopoztos.com/paypalitlogin/us/...* we use a dictionary-based word splitter [16]. For instance, the three words *paypal*, *it* and *login* are extracted from *paypalitlogin* through this process.

[2]http://publicsuffix.org/list/

Based on this splitting two sets are composed: one, called RD_{url} (for Registered Domain), consists just of two elements: $RD_{url} = \{mld, mld.ps\}$. The other, REM_{url} (for REMaining part), is composed of all extracted words from the URL except *mld.ps*. Given *http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run*, the following sets are extracted:

- $RD_{url} = \{sezopoztos, sezopoztos.com\}$
- $REM_{url} = \{paypal, it, login, us, web, src, html, cmd, login, run\}$

The *mld.ps* is not split like the other part to keep the *mld* unmodified, which can be composed of several words.

Assume a type III obfuscated URL such as *http://cgi-3.paypal-secure.de/info2/verikredit.html* . The word *paypal* would be an element of $RD_{phish} = \{paypal, secure, de\}$. If *http://cgi-3.paypal.de/info2/verikredit.html* is a real PayPal URL, we have $RD_{legit} = \{paypal, de\}$ and $RD_{legit} \cap RD_{phish} = \{paypal, de\}$. However with the proposed decomposition of *mld.ps* we have the two lists $RD_{phish} = \{paypal-secure, paypal-secure.de\}$ and $RD_{legit} = \{paypal, paypal.de\}$ giving $RD_{legit} \cap RD_{phish} = \emptyset$. Hence our proposed decomposition emphasizes the difference between the two domains.

Once the two sets are built, the next step is to evaluate the relatedness of their components. It is tempting to compute word similarity or word relatedness with existing tools such as Disco [9]. However this tool, even if efficient in most cases and especially for phishing domain names analysis as shown in our previous work [17], is not necessarily suited to intra-URL relatedness computation.

B. Word Relatedness Evaluation Tools Shortcomings

WordNet [10] is a lexical database containing a collection of english language words. Given a word, WordNet can give a collection of related words. The limitation of this tool is that it is only based on vocabulary that is likely to appear in an English dictionary, whereas Internet vocabulary includes several different languages and many words that are not contained in dictionaries.

Automated techniques and measures have also been developed to evaluate word relatedness. Latent Semantic Analysis (LSA) proposed by Landauer and Dumais [18] or Pointwise Mutual Information (PMI), introduced by Church and Hanks [7], are examples of these techniques. The Normalized Google Distance (NGD [8]) computes the semantic similarity between two words by querying the Google search engine for these words and counts the number of Web pages where they appear together and individually. Disco [9] relies on mutual information evaluation between two words based on corpora.

To prove the limitations of these existing tools, we tested whether two of them are able to find related words for a set of labels. WordNet and Disco are chosen since these are the only ones that are really usable through an interface. The testing set consisted of the RD_{url} extracted from a set of 94 URLs. These URLs come from PhishTank (described in Section V-A) *i.e.* phishing URLs present in PhishTank blacklist are categorised according to the brand they target. As of March 2014, when we made our evaluation, 94 brands and associated URLs were

Brand	*mld*	*mld.ps*
JPMorgan Chase	jpmorganchase	jpmorganchase.com
TAM Airline	tam	tam.com.br
Visa	visa	visa.com
Windows Live	live	live.com
Poste Italiane	poste	poste.it
Wells Fargo	wellsfargo	wellsfargo.com
Blizzard	blizzard	blizzard.com

TABLE II
SUBSET OF MOST PHISHING TARGETED BRANDS WITH *mld* & *mld.ps*

Tool	#*mld*	%*mld*	#*mld.ps*	%*mld.ps*
WordNet	20	21.3%	0	0%
Disco	23	24.5%	0	0%
Yahoo Clues	87	92.6%	68	72.3%
Google Trends	92	97.9%	76	80.9%
Total	94	-	94	-

TABLE III
NUMBER OF LABELS MATCHING AT LEAST ONE RELATED WORD FOR 4 TOOLS

present in this list. A subset of this test set is given in Table II and the result of the test for each tool is given in the two first rows of Table III. The numbers of *mld* and *mld.ps* for which the tested tools can give at least one related word are given in absolute value and percentage terms.

Neither WordNet nor Disco performs well on this test set. These only provide related words for less than 25% of *mld* and never match any *mld.ps*, although the brands and domain names tested are well known. In addition for the *mld*s that match a result, it is usually for brand that is also a meaningful word such as *live* or *visa*. The test proves that current word relatedness tools are not suited to the measure of intra-URL relatedness.

While creating a dedicated corpus to be used with existing methods would be helpful but challenging, word relatedness can be dynamically inferred from search engine query data, as shown in next section.

C. Search Engine Query Data

To perform the evaluation of intra-URL relatedness, we use search engine query data. The reason is that URL obfuscation is a social engineering lure. Phishing URLs target a brand, so clever phishers blend within them the brand and words that Internet users associate with the brand, such as a provided service: *payment* for PayPal. People generally use search engines to access these services. When they make a search, they type some keywords that are typically the brand or the domain name and the service needed like *paypal payment* or *hsbc.com on-line banking*. These word associations reflect the cognitive process of users searching for PayPal or HSBC. Consequently, such words are the ones phishers tend to blend into URLs to trap PayPal and HSBC customers [4].

Hence, mining search engine query data for word relatedness measurement is relevant in a phishing context. To achieve this goal, we use search engine query data from two top-ranked search engines: Google and Yahoo. Both offer services, that, given a term, provide some insights on requesting trends concerning it. These services are respectively *Google Trends*[3] and *Yahoo Clues*[4]. In the context of the paper we define a term t as a set of words w. $\{paypal\}$ and $\{paypal, login, secure\}$ are two examples of terms.

Google Trends shows the relative interest of Google users over time in a term. It depicts the geographic interest for this term and provide related terms according to users' related searches. Google Trends provides the top ten related searches over time as well as the ten rising related searches namely those on which interest has increased recently. This allows us to gather up to twenty related terms for one given term.

Yahoo Clues provides the same kind of services as Google Trends. It offers an insight into the search flows, the terms requested just before (5 terms) and after (5 terms) a term. Like Google Trends it also provides a set of related searches.

Combining both sources can give up to forty related terms for one given term. A result for the queried term $\{paypal\}$ for both tools Google Trends and Yahoo Clues is given in Table IV. The ability of these tools to find related words for phishing targeted *mld* and *mld.ps* is highlighted in Table III. Both tools were tested on the same set of terms used for WordNet and Disco, giving the results in rows 3 and 4. They perform better, with Google Trends being the best at finding related words. However both provide match results for more than 90% *mld* and 70% *mld.ps*, much more than usual similarity evaluation tools tested earlier.

Having a URL *url* and the extracted sets RD_{url} and REM_{url}, Google Trends and Yahoo Clues are automatically requested for each element of the two sets. We define $Term_w$, as the set of terms resulting from the requests of the word w in both Google Trends (related & rising) and Yahoo Clues (related & requests). A subset of $Term_{paypal}$ is given in Table IV with $Term_{paypal} = \{\{paypal, account\}, \{paypal, login\}, ...\}$.

We define four sets of words built from a URL *url*: $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$.

$REL_{set}(url)$ consists of all the words *related* to the words of *set i.e.* words included in terms that are results of requests for elements of *set*. Here *set* is either RD_{url} or REM_{url}. The formulas for these sets are given in Equations (1) and (2).

$$REL_{rd}(url) = \{w \in t \mid t \in Term_{w'}, w' \in RD_{url}\} \quad (1)$$

$$REL_{rem}(url) = \{w \in t \mid t \in Term_{w'}, w' \in REM_{url}\} \quad (2)$$

$AS_{set}(url)$ is the set of words that are *associated* with the words of *set i.e.* the words that appear in a common single term. Assuming a term t composed of three words $\{w_1, w_2, w_3\}$, there is a symmetric *association* relationship between w_1 and w_2, w_1 and w_3, w_2 and w_3. The two sets

---

[3]http://www.google.com/trends/
[4]http://clues.yahoo.com/analysis

Google related	Google rising	Yahoo related	Yahoo requests
$\{paypal, account\}$	$\{amazon, paypal\}$	$\{bill, me, later\}$	$\{paypal, login\}$
$\{paypal, login\}$	$\{paypal, fees\}$	$\{netspend\}$	$\{paypal.com\}$
$\{paypal, credit, card\}$	$\{ebay, uk\}$	$\{suntrust\}$	$\{paypal, buyer, credit\}$
$\{paypal, email\}$	$\{paypal, login\}$	$\{regions\}$	$\{paypal, customer, service\}$

TABLE IV
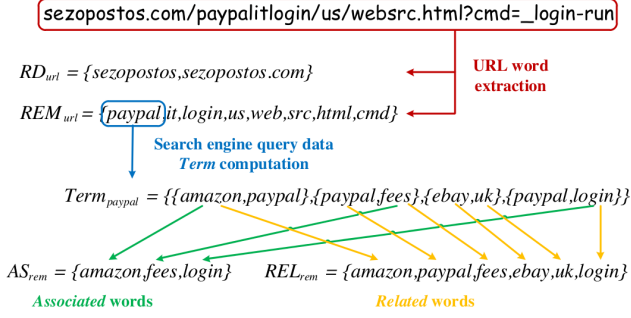EXAMPLE OF TERM RESULTS FROM GOOGLE TRENDS AND YAHOO CLUES FOR $\{paypal\}$

Fig. 1. Word extraction and word set composition for *http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run*

$AS_{rd}(url)$ for RD_{url} and $AS_{rem}(url)$ for REM_{url} are defined in Equations (3) and (4) respectively.

$$AS_{rd}(url) = \{w \in t \mid \exists w' \in RD_{url}, w' \in t, w' \neq w\} \quad (3)$$

$$AS_{rem}(url) = \{w \in t \mid \exists w' \in REM_{url}, w' \in t, w' \neq w\} \quad (4)$$

These four sets are extracted to quantify the relationship between and inside each set RD_{url} and REM_{url}. Assume the URL *http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run*, Figure 1 presents the full process from word extraction to $AS_{rem}(url)$ and $REL_{rem}(url)$ composition based on a subset of $Term_{paypal}$. We have:
$REL_{rem}(url) = \{amazon, paypal, fees, ebay, uk, login\}$
$AS_{rem}(url) = \{amazon, fees, login\}$

D. Feature Computation

Based on the sets defined in the previous subsection, we introduce 12 features characterising intra-URL relatedness and URL popularity. The popularity criteria is based on the search count for components of a URL (registered domain, *mld*, etc.). These features are described in Table V.

The features 1-6 define intra-URL relatedness by calculating the Jaccard index pairwise between the four sets defined in Section III-C ($REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$). The Jaccard index is a long-established metric used to calculate similarity and diversity between two sets A and B. The closer $J(A, B)$ is to 1 the more similar are A and B. These six features quantify the relatedness between the two parts of the URL (*mld.ps* and the rest) through J_{RR}, J_{RA}, J_{AA} and J_{AR}, as these compute Jaccard indexes between sets
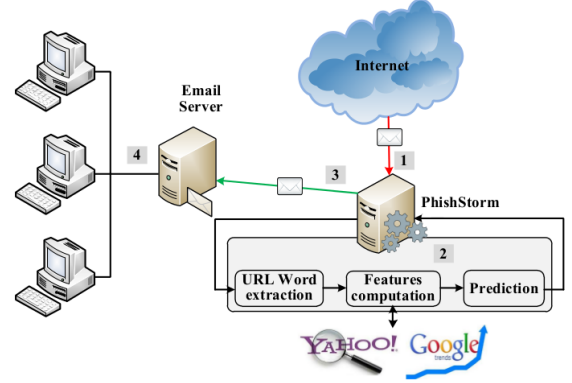

Fig. 2. PhishStorm implementation

extracted from different parts (RD_{url} and REM_{url}). These also measure the relatedness inside each part through J_{ARrd} and J_{ARrem}, as these features are calculated from sets extracted from the same part of a URL.

Features 7-12 reflect the popularity of a URL and its components with the number of words that compose it ($card_{rem}$) and the number of related and associated words found in search engine query data based on these words with $ratio_{Arem}$ and $ratio_{Rrem}$. These two features are weighted by $card_{rem}$. Features $mld.ps_{res}$ and mld_{res} represent the popularity of the registered domain by giving boolean values describing whether the *mld.ps* and *mld* match results while queried in Google Trends and Yahoo Clues. The final feature (*ranking*) is the ranking of the *mld.ps* according to the Alexa[5] Web site ranking list. Alexa gives a ranking for the top 1,000,000 most visited Web sites; if a particular *mld.ps* is not in the list, the value 10,000,000 is considered.

Features 10, 11 and 12 can be considered as relying on the reputation of a domain and not on the intra-URL relatedness. Even if features 10 and 11 are new — *ranking* has been used already in state of the art work — we compare in Section VI classification results with and without these three features to assess the relevancy of intra-URL relatedness features.

IV. PHISHSTORM IMPLEMENTATION

PhishStorm gives a generic solution for phishing URL detection relying on intra-URL relatedness computation. This technique only needs access to search engine query data to operate. Hence the application range of PhishStorm is wide.

[5]http://www.alexa.com/

	Features	Description
1	$J_{RR} = \frac{|REL_{rd}(url) \cap REL_{rem}(url)|}{|REL_{rd}(url) \cup REL_{rem}(url)|}$	Jaccard index b/w $REL_{rd}(url)$ and $REL_{rem}(url)$
2	$J_{RA} = \frac{|REL_{rd}(url) \cap AS_{rem}(url)|}{|REL_{rd}(url) \cup AS_{rem}(url)|}$	Jaccard index b/w $REL_{rd}(url)$ and $AS_{rem}(url)$
3	$J_{AA} = \frac{|AS_{rd}(url) \cap AS_{rem}(url)|}{|AS_{rd}(url) \cup AS_{rem}(url)|}$	Jaccard index b/w $AS_{rd}(url)$ and $AS_{rem}(url)$
4	$J_{AR} = \frac{|AS_{rd}(url) \cap REL_{rem}(url)|}{|AS_{rd}(url) \cup REL_{rem}(url)|}$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rem}(url)$
5	$J_{ARrd} = \frac{|AS_{rd}(url) \cap REL_{rd}(url)|}{|AS_{rd}(url) \cup REL_{rd}(url)|}$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rd}(url)$
6	$J_{ARrem} = \frac{|AS_{rem}(url) \cap REL_{rem}(url)|}{|AS_{rem}(url) \cup REL_{rem}(url)|}$	Jaccard index b/w $AS_{rem}(url)$ and $REL_{rem}(url)$
7	$card_{rem} = |REM_{url}|$	number of words in REM_{url}
8	$ratio_{Arem} = \frac{|AS_{rem}(url)|}{|REM_{url}|}$	ratio of associated words for words in REM_{url}
9	$ratio_{Rrem} = \frac{|REL_{rem}(url)|}{|REM_{url}|}$	ratio of related words for words in REM_{url}
10	$mld_{res} = \begin{cases} 0 & \text{if } |Term_{mld}| = 0 \\ 1 & \text{else} \end{cases}$	whether there is search engine results or not for the *mld* of the URL
11	$mld.ps_{res} \begin{cases} 0 & \text{if } |Term_{mld.ps}| = 0 \\ 1 & \text{else} \end{cases}$	whether there is search engine results or not for the *mld.ps* of the URL
12	$ranking$	Alexa ranking for $mld.ps$

TABLE V
URL FEATURE DESCRIPTIONS

It can operate at different network level to prevent phishing. It can provide a personal protection for users while surfing on the Web if implemented locally as a browser add-on. PhishStorm provides phishingness score for URL and can act as a Web site reputation rating systems, displaying a Web site rating while using a search engine or typing a URL into a Web browser. Centralized phishing protection is an other option as for instance at the Web proxy level of a local company network, filtering HTTP packets sent from URLs identified as phishing.

However, as the main vector of phishing attacks is spoofed emails embedding phishing URLs, we implement PhishStorm as a centralized phishing email detection tool positioned in front of the email server. Nowadays, spam filtering is performed centrally in many organisation and PhishStorm can be added to such process to increase detection performance. Figure 2 depicts the implementation of PhishStorm and the four steps of the phishing email detection process. While incoming emails from the Internet reach PhishStorm (1), potential embedded URLs are extracted therefrom. The system then proceeds to features computation thanks to search engine query data and predicts a phishingness score using machine learning techniques (2). A detection threshold is applied to every predicted score, determining if the email must be forwarded to the email server (3) and then to users (4) with its phishingness score or dropped.

We give in this section a detailed description of the implementation of the features computation process described in previous section.

A. Distributed Word Relatedness Inference

The bottleneck for feature computation is the sequential network communication overhead with the Google and Yahoo servers. However, this bottleneck can be easily removed by leveraging existing Big Data architectures for streaming analytics. For our case, the most relevant architecture is the STORM project[6], which inherently allows to distribute parallel computations over STORM topologies. Because of the transactional support, we have chosen to use the Trident topologies. Within such a topology, nodes perform a processing logic. Nodes are connected with links that indicate how the data is processed. Data is sent within a Stream and STORM can distribute the computation along a sequence of nodes. There are two types of nodes: *Spouts* and *Bolts*. *Spouts* represent the source of data. For our architecture, the *spout* (URL-Spout), depicted in Figure 3, is a URL extraction component that extracts individual URLs. Each individual URL is sent to four different *bolts*. This is done using one of the stream grouping method. Several such methods exist and their working depend on how a *spout* decides to split the output to the connected *bolts*. The *All-grouping* method is the replication to all attached *bolts*. Sem-Bolts in our architecture have a simple function. They connect to the Google and respectively Yahoo servers and retrieve the list of semantically equivalent words as depicted in Figure 3 through a basic STORM implementation. Finally, the intersection among these need to be performed. This is done by the Intersection-Bolt using the method described in Section IV-B.

Using such a simple STORM topology, we can reduce the communication time by a factor of four. Further improvements can be obtained by increasing the degree of parallelism through an increased number of *bolts* as depicted in Figure 4. In this case, the *spout* tokenize the URLs into different words. Each individual word is sent to a URL-Bolt. In this case, the *Field-grouping* method is used. This ensures that a word occurring in several URLs will be always sent to the same URL-Bolt and therefore a caching strategy can avoid repeating the same requests. Each URL-Bolt will furthermore
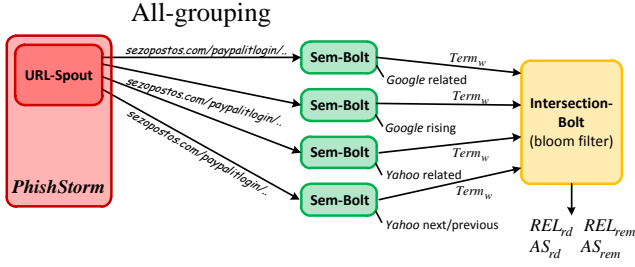

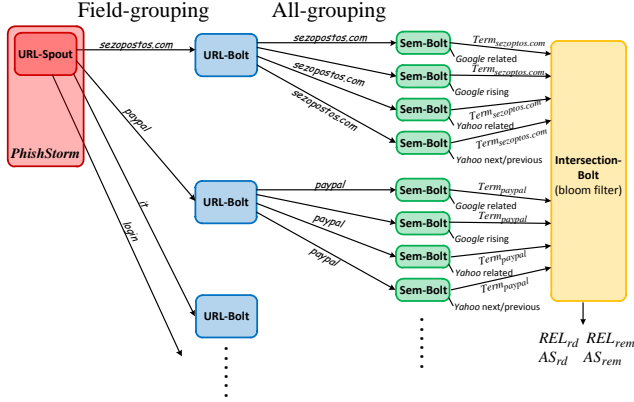[6]https://storm.apache.org/

Fig. 3. Simple STORM topology

Fig. 4. Field-grouping STORM topology

replicate the input to four Sem-Bolts, where each Sem-Bolt is responsible to communicate with a Google, respectively Yahoo server. In this case, we can reduce the communication overhead to the single round trip time between our platform and the Google/Yahoo servers.

Since the topology is defined at compile time, good estimates for the number of URL-Bolts are needed. For this purpose, Figure 5 showing the histogram of individual words per URL is used. This histogram depicts the proportion of URLs being composed of n words. This statistics is obtained by applying the word extraction process introduced in Section III-A on the URL dataset presented later in Section V. We can see that most of URLs is composed of between two and five words. This means that on average the system needs to request the search engine query data for two to five words plus the *mld* and *mld.ps*. Hence to optimize our architecture we set the number of URL-Bolts to seven (5+2), in order that an average URL can be processed in one round.

This method to gather search engine query data leverages a feature computation technique based on set operations that are space-efficient and computationally-efficient.

B. Bloom Filter for Features Computation

Features presented in Section III-D rely mostly on set operations, as for Jaccard Index computation, which requires union, intersection and counting element operations. Moreover, $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$, which are the based set of the features, require intersection operation between the several $Term_w$ sets result from querying
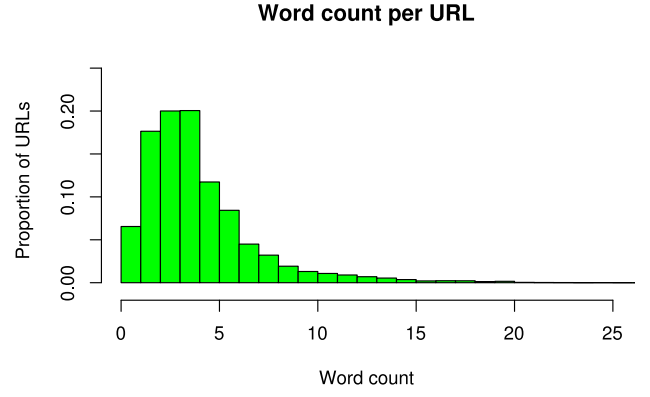

Fig. 5. Repartition of the number of embedded words per URL

Google and Yahoo. As a result we implement all the word set previously defined with an efficient data structure: the Bloom filter [19].

Bloom filters are statistical data structures relying on several hash functions to represent set of elements. This data structure is represented as a bit array and is subjected to false positives for element lookup *i.e.* an element identified as being in the set is not necessarily in the set, but an element identified as not being in the set is surely not in the set. Nevertheless, Bloom filters have the strength to be space-efficient and to have constant complexity for lookup and adding elements. This complexity does not depend on the number of elements the Bloom filter contains and is $O(k)$ with k being the number of hash functions. A Bloom filter can be described with two parameters being its size m in bit, which shall be set in advance, and its number of different hash functions k. Assume a Bloom filter containing n elements, the probability of false positive for testing the fact that an element is in the set is expressed in Equation (5) with P_{error}.

$$P_{error} = \left(1 - \left[1 - \frac{1}{m}\right]^{kn}\right)^k \qquad (5)$$

Computing the three needed operations *i.e.* union, intersection and element count are also subject to error and to some requirements. To perform set intersection and union with Bloom filter the requirement is that both Bloom filters have same size (m) and share the same hash functions [20]. The union of two sets consists in a bitwise OR between the two Bloom filters and the intersection of a bitwise AND. For two Bloom filters BF_1 and BF_2, we have $P_{error}(BF_1 \cup BF_2) < P_{error}(BF_1) + P_{error}(BF_2)$ and $P_{error}(BF_1 \cap BF_2) < min(P_{error}(BF_1), P_{error}(BF_2))$. The element count operation of a Bloom filter containing n elements can be approximated by n^* based on the count of bit X set to 1 as shown in Equation (6).

$$n^* = -\frac{m * ln\left(1 - \frac{X}{m}\right)}{k} \qquad (6)$$

Features	Legitimate dataset						Phishing dataset				
	min	1st Q	median	3rd Q	max		min	1st Q	median	3rd Q	max
$card_{rem}$	0	1	2	4	20		0	3	5	9	58
$ratio_{Arem}$	0	29.462	93	181.5	6097		0	55.412	114.1	172.22	3122
$ratio_{Rrem}$	0	33.833	92	179	5507		0	59.789	113.5	169.75	2826.5
mld_{res}	0	0	1	1	1		0	0	0	0	1
$mld.ps_{res}$	0	0	0	1	1		0	0	0	0	1
$ranking$	1	6655	82260	10e7	10e7		1	10e7	10e7	10e7	10e7

TABLE VI
STATISTICAL VALUES OF FEATURES EXTRACTED FROM LEGITIMATE AND PHISHING DATASET

Hence Bloom filters offer the required operations to compute the features of intra-URL relatedness in a space-efficient and computationally-efficient way, while introducing approximation.

As described before the parameters of Bloom filters (size m in bit and number of hash functions k) must defined in advance. To keep accurate features we propose to set the false positive rate P_{error} to 0.0001 (0.01 %) for $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$. Hence we can deduce m and k by determining the maximum number of elements that will count these sets. To determine this count we refer to Figure 5 showing the word count per URL. We can see that URLs embed from 0 to around 20 words. To have a safety margin we set the maximum word count per URL to 25. Hence every word embedded in a URL is requested to search engine query data tools to gather up to 40 terms per single word, these are composed on average of three words. This gives the umber of element of each set $n = 25 * 40 * 3 = 3000$. The optimal number of bits per element for a Bloom filter is given in Equation (7) and for $P_{error} = 0.0001$ we got $bit/element \approx 19$.

$$bit/element = \frac{1}{ln(2)} * log\left(\frac{1}{P_{error}}\right) \quad (7)$$

$$k = \frac{m}{n} * ln(2) \quad (8)$$

Thus, we have $m = 19 * 3000 = 57000$ bits and we can can deduce the number of hash function to use according to Equation (8). We get $k = 13$ and each set $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$ is set up as a Bloom filter of size 57,000 bits and 13 hash functions. As these have all same size and same hash functions, these can be compared using union and intersection operations to compute features of intra-URL relatedness.

Having described the technique and the implementation of intra-URL relatedness features computation, we build a ground truth dataset of URLs to test their efficacy.

V. DATASETS

To assess the ability of the proposed feature set to be used in supervised classification, we use two datasets. One of these is a malicious dataset, the *phishing dataset*; the other is the *legitimate dataset*. These sets are composed from different sources — as in several phishing detection work [15], [21], [22] — already used in [23], [24].

A. Phishing Dataset

We used PhishTank to build a phishing dataset. PhishTank[7] is a collaborative project to which people can submit phishing emails and Web sites. Suspected phishing URLs are further checked by several people before being confirmed as malicious and added to a blacklist. PhishTank provides lists of valid and active phishing URLs.

We downloaded this list on a daily basis between October 11th and November 10th, 2012 and built a phishing ground truth dataset of 53,089 unique URLs. URLs consisting only of *mld.ps*, *www.mld.ps* or IP addresses without path or query were discarded because it is impossible to calculate the intra-URL relatedness for such URLs, as $REM_{url} = \emptyset$. In addition we already addressed the identification of such phishing domains in [17]. After this selection we had 48,009 extended phishing URLs in the *phishing dataset* meaning less than 10% phishing URLs discarded.

B. Legitimate Dataset

To provide additional learning instances for legitimate URLs, we selected URLs from the Open Directory Project (DMOZ[8]). DMOZ is a directory of the Web containing more than two million URLs. We first discarded URLs consisting only of *mld.ps* or *www.mld.ps*, as for the *phishing dataset*. Then a uniform random selection was made on the rest to keep 48,009 legitimate URLs.

We constructed a balanced dataset (half malicious/half legitimate) of ground truth data composed of 96,018 URLs. We acknowledge that a half/half ratio for phishing and legitimate URLs does not reflect real world repartition. However this dataset is used to assess the efficiency of the search engine query data and the features extracted therefrom, in distinguishing phishing from legitimate URLs through ten-fold cross-validation. And as presented in [25], imbalanced dataset in cross-validation provides misleading results. This URL set with extracted features is publicly available for research purpose[9].

[7]http://www.phishtank.com/
[8]http://www.dmoz.org/
[9]http://secan-lab.uni.lu/images/stories/samuel_marchal/urlset.csv
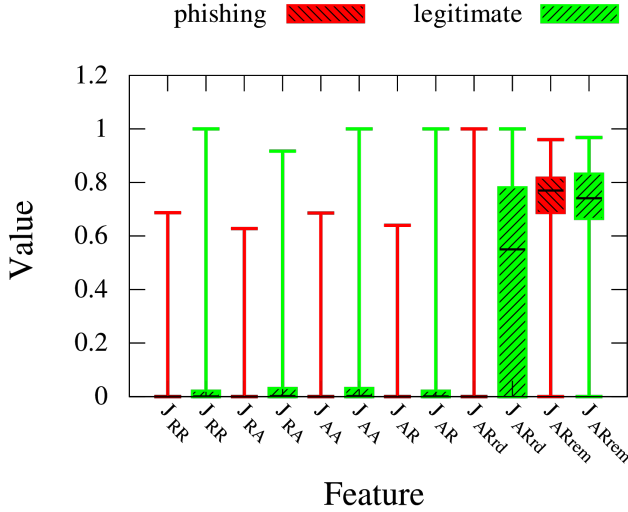
Fig. 6. Box-and-whisker diagram for Jaccard based features (min/max)

Feature	IG	Feature	IG
$ranking$	0.388	mld_{res}	0.178
J_{ARrd}	0.286	J_{RA}	0.133
$card_{rem}$	0.273	J_{RR}	0.125
$ratio_{Arem}$	0.217	J_{AA}	0.123
J_{ARrem}	0.208	J_{AR}	0.12
$ratio_{Rrem}$	0.208	$mld.ps_{res}$	0.07

TABLE VII
INFORMATION GAIN VALUES OF THE 12 FEATURES

C. Features Analysis

The 12 features described in Section III-D were extracted from each dataset. The box-and-whisker diagram of Figure 6 compares the median, 1st quartile, 3rd quartile, minimum and maximum values for each Jaccard-based feature according to the set it is extracted from. The same statistical values are given in Table VI for the six remaining features as these features are either not defined on $[0, 1]$ or are binary values.

There is no significant difference between the legitimate and phishing datasets for features J_{RR}, J_{RA}, J_{AA} and J_{AR} with equal minimum, 1st quartile and median. The values of these features are slightly more scattered for the legitimate dataset, reaching higher values. The 3rd quartile is around 0.02 for legitimate URLs whereas it is 0 for the phishing dataset, while maximum reaches 1 for legitimate, but it does not exceed 0.7 for the phishing dataset. This validates our assumption that the similarity between *mld.ps* and the rest of the URL is more important in legitimate URLs than in phishing ones. In addition we can identify J_{ARrd} as a good discriminative feature. Despite having the same range of value for both kinds of URLs $[0, 1]$ the median and 3rd quartile are an order of magnitude higher for legitimate URLs (0.55/0.78) compared to phishing ones (0/0).

Analysing $card_{rem}$ in Table VI, we conclude that phishing URLs blend more words within them than legitimate ones. Legitimate *mld.ps* and *mld* logically match more results than phishing ones (mld_{res}, $mld.ps_{res}$) when searched for in Google Trends and Yahoo Clues. Finally the ranking of legitimate domains is higher than phishing ones since 31,767 legitimate domains are ranked in the Alexa top one million domains against only 8,081 phishing domains.

Nevertheless it is worth noting that some phishing *mld.ps* have a high ranking, namely 1. This top-ranked domain is *google.com*. The reason why such top ranked *mld.ps* are present in the phishing set is that the Google Docs[10] facility is used as a support for phishing. Phishers steal personal data by

---

[10]http://docs.google.com

creating in it on-line forms that victims are asked to fill out. The use of well-ranked domains as a basis for phishing URLs proves that reputation based features can not be sufficient to identify phishing URLs.

To evaluate the impact of each feature on the classification of phishing URLs, the Information Gain is computed. Assuming S, a set of instances having n features $(x_1, x_2, ..., x_n)$ and a label l, the Information Gain $IG(S|i)$ evaluates the likelihood of deducing l for elements of S given the feature $i \in \{1, n\}$. It is defined in Equation (9), based on the entropy of the dataset S: $H(S)$ and the average conditional entropy of S given the feature i: $H(S|i)$.

$$IG(S|i) = H(S) - H(S|i) \qquad (9)$$

Information gain for the 12 features is given in Table VII, ordered by descending value. The higher the information gain value, the more discriminative the feature. All the features bring information for classification, $ranking$ being the most significant feature mainly due to the fact that most phishing domains have a very poor rank. It is followed by J_{ARrd} which contains a large amount of information as shown in Figure 6. J_{ARrem} is a significant feature despite no clear difference being brought out in Figure 6.

Having established that the feature set is relevant in distinguishing phishing from legitimate URLs, we further assess its efficiency within a machine learning framework.

VI. PHISHING URL DETECTION

To automatically detect phishing URLs, we use supervised classification techniques. We build a feature vector matrix from the dataset presented in Section V. Each feature vector is composed of 12 elements, namely the 12 features described in Section III-D. The predicted variable is 0 for a legitimate URL and 1 for a phishing URL. This gives a matrix of 96,018 feature vectors representing the 96,018 URLs of the testing dataset.

A. URL Classification

Since there is a wide range of supervised classification algorithms, we assessed our dataset according to several classifiers using Weka [26]. Seven classifiers were tested covering tree-based (Random Tree, Random Forest, C4.5, LMT) rule-based (PART, JRip) and function-based (SVM). The classification was made without parameters tuning through a ten-fold cross-validation as a first step to select the most promising approach.
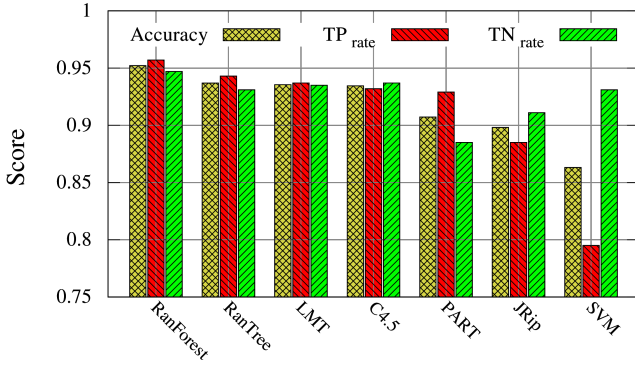
Fig. 7. Classification results for 7 classifiers

Classifier	5th *perc.*	*median*	95th *perc.*	SD
RanForest	94.68	95.22	95.41	0.23
RanTree	93.35	93.81	94.1	0.23
LMT	93.10	93.55	94.01	0.27
C4.5	93.01	93.45	93.87	0.26
PART	89.84	90.62	91.49	0.47
JRip	88.32	89.66	90.48	0.66
SVM	85.23	86.31	87.53	0.62

TABLE VIII
CONFIDENCE INTERVAL FOR CLASSIFICATION RESULTS

Results for accuracy, true positives and true negatives are given in Figure 7 for each classifier. To give additional information about confidence interval of classification results for these classifiers, Table VIII shows the median, 5th percentile, 95th percentile and standard deviation (SD) values for the *Accuracy* of each classifier out of 100 runs. For sake of clarity we define for URLs:

- Phishing classified as phishing: true positives (TP) and $TP_{rate} = \frac{TP}{TP+FN}$
- Legitimate classified as phishing: false positives (FP) and $FP_{rate} = \frac{FP}{TN+FP}$
- Legitimate classified as legitimate: true negatives (TN) and $TN_{rate} = \frac{TN}{TN+FP}$
- Phishing classified as legitimate: false negatives (FN) and $FN_{rate} = \frac{FN}{TP+FN}$

and the accuracy: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

Among the tested classifiers, SVM yields the worst accuracy (86.31%) while being efficient in identifying legitimate URLs (93.1%). Rule-based classifiers have approximately the same performance (around 90%) with disproportionate true positives and true negatives. The best performers are tree-based classifiers, with Random Forest, correctly classifying 95.22% of URLs, being the best. In addition Table VIII shows that these top performer classifiers give accurate results having a standard deviation around 0.25% over 100 runs.

Hence, Random Forest is selected for classification. Random Forest [27] is a classification method that creates a multitude of decision trees during training. During prediction, it outputs a *hard decision* for the class of an instance as the class that has been predicted by the most individual trees.
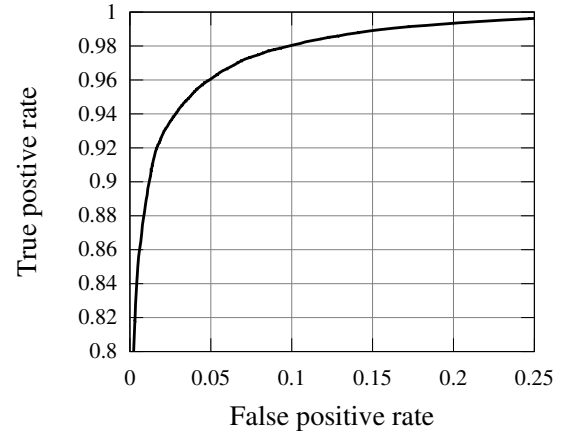

Fig. 8. ROC curve for Random Forest classification

However a *soft prediction* can also be deduced from the combination of results given by individual trees. This *soft prediction* is bounded on $[0, 1]$ and gives a confidence score for the prediction. It is then compared to a discrimination threshold to give the *hard decision*.

We tuned the parameters of Random Forest training in order to achieve better classification. The number of trees to be generated during training was set to 100. The ROC (Receiver Operating Characteristic) curve describing the classification results for the tuned classifier in true positive rate and false positive rate is illustrated in Figure 8. The ROC curve corresponds to the variation of true positives and false positives while varying the discrimination threshold from 0 to 1. To minimize the number of legitimate URLs classified as phishing (false positives) we adjust the discrimination threshold from 0.49 (the value giving the best accuracy) to 0.76. This reduces the accuracy from 95.66% to 94.91% but also decreases the FP_{rate} from 4.13% to 1.44%.

The detailed classification metrics for the Random Forest algorithm with a 0.76 discrimination threshold are given in Table IX. The two first columns represent the rate of well-classified and misclassified instances for each class: TP_{rate}, FP_{rate}, FN_{rate} and TN_{rate}. The *Precision* corresponds to the ratio of phishing URLs classified as phishing with respect to the total URLs classified as phishing such as *Precision* $= \frac{TP}{TP+FP}$. The *F-measure* is defined in Equation (10) with $Recall = TP_{rate}$.

$$F-measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \qquad (10)$$

To show the relevancy of intra relatedness features, we classified with different features the set of URLs. Using only features 1-9 for classification yields an accuracy of 93.48% whereas using reputation based features 10-12 yields 83.97%. While having the best Information Gain as shown in Table VII, feature 12 ($ranking$) and other state of the art features are not sufficient to distinguish between phishing and non-phishing URLs alone. However we show that the proposed feature set yields good results in doing this task. In addition, combining the new proposed features with reputation based features can lead to an improvement in the classification accuracy making

Class	Class. as phish.	Class. as leg.	*Precision*	*F-measure*	*Accuracy*
Phishing	91.27% (TP)	8.73% (FN)	98.44%	94.72%	94.91%
Legitimate	1.44% (FP)	98.56% (TN)			

TABLE IX
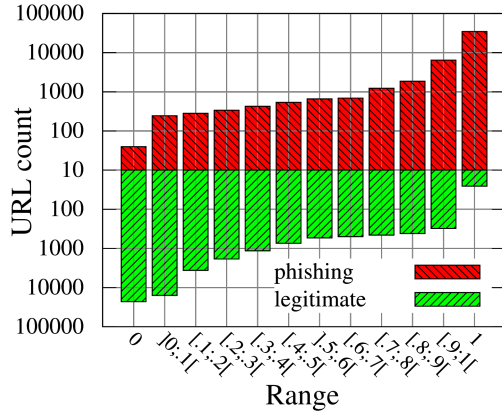DETAILED CLASSIFICATION RESULTS FOR RANDOM FOREST (THRESHOLD = 0.76)

Fig. 9. Phishing and legitimate URL partition according to rating ranges

this work complementary to the state of the art.

Even though this technique, which gives a *hard decision* for URL class, is proved efficient, correctly classifying 94.91% of URLs with only 1.44% of legitimate URLs classified as malicious, we further leverage machine learning to build a reputation system.

B. URL Rating

The *soft prediction* value provided by Random Forest is defined on the range $[0; 1]$. In the previous section a discrimination threshold was fixed to give a *hard decision* on the phishingness degree of a URL. However *soft prediction* values are not uniformly distributed over the range $[0; 1]$ and some sub-ranges of values may be more suitable to providing a highly reliable decision on the phishingness of a URL. Hence, we analysed the *soft prediction* distribution regarding phishing or legitimate URLs.

The *soft prediction* range of value $[0; 1]$ is divided in 12 sub-ranges, two being the exact value 0 and 1 and the ten remaining being ranges of length 0.1: $]0; 0.1[$, $[0.1; 0.2[$, ... , $[0.9; 1[$. The *soft prediction* provided by the tuned Random Forest was computed for all 96,018 URLs of the dataset through a ten-fold cross-validation. We counted the URLs having a score belonging in each sub-range. Figure 9 depicts this count according to the set (phishing at the top / legitimate at the bottom) the URLs come from. The 12 different sub-ranges are on the x-axis and the URL count is on the y-axis in a log scale centered on 10 and increasing in both directions for each class (phishing up/legitimate down).

We can observe that most of the URLs are grouped in each extremity of the range and mostly in the sub-ranges 0, $]0; 0.1[$, $[0.9; 1[$ and 1, which contain a total of 80,630 URLs out of 96,018. In addition the middle values of *soft*

prediction have few of either kind of URLs, usually less than 1,000. This confirms that the *soft prediction* is not uniformly distributed over its range of definition. Considering the two extreme values, very few phishing URLs (40) have the score 0, whereas 22,863 legitimate URLs do. The same happens for a *soft prediction* of 1 where 34,790 phishing URLs have this score against only 26 legitimate. Given that 0 corresponds to legitimate and 1 to phishing, we are able to classify 60.11% of the dataset (57,719 URLs) with an accuracy of 99.89%. URLs getting a *soft prediction* of 0 or 1 are very likely to be either legitimate or phishing URLs respectively. This proves that some ranges of *soft prediction* values are more suited to making a reliable prediction. If we extend the analysis to the range $[0; 0.1]$, it contains 38,741 legitimate URLs and only 288 phishing ones. The range $[0.9; 1]$ is composed of 41,260 phishing URLs and 341 legitimate URLs. Considering these two sub-ranges, these contain 83.97% of the testing dataset and their components are correctly identified as legitimate or phishing with an accuracy of 99.22%.

The *soft prediction* can be used as a *risk score* for a URL. The closer it is to 1, the higher the risk; the closer to 0 the safer the URL. This score is used in PhishStorm to give a confidence rate to every incomming email going through it as depicted in Figure 2. We have demonstrated that such a rating system is reliable in 99.22% of the cases for most of the URLs (83.96%).

While performing our experiments, we timed the process from labels extraction, requesting search engines, features computation to classification decision. With a basic centralized implementation, it took around 112 hours for the set of 96,018 URLs on a single machine (Intel Core i7 processor and 4Gb memory). This gives an average time of 4.2 seconds/URL, most of the time being taken by the requests to search engines. However with the implementation presented in Section IV leveraging STORM for HTTP requests and Bloom filters for set operations the processing time dropped to 20.6 hours. This gives an average processing time of 0.77 seconds per URL which is acceptable for the several applications of PhishStorm.

VII. DISCUSSION

This paper introduces PhishStorm a system relying on search engine query data for phishing URL identification. This technique has some limitations that we identify in this section.

Our technique is not applicable to all types of obfuscated URLs as described in Section II. URLs composed of only a malicious domain, URLs based on shortening services or URLs algorithmically generated are kind of malicious URLs that can bypass our detection technique. This kind of URLs and malicious domains are however widely used in botnet

communication (C&C) or spamming activities [28], such activities does not rely on a social engineering process as phishing does. The main part of URLs used for phishing are meaningful and composed of many terms [4], this is why our technique is relevant in a phishing context.

A limitation of the implementation is that data publicly available through Google Trends and Yahoo Clues is limited. For each requested term only the ten related most popular terms are returned by these tools. Related terms less requested by search engine users do not appear in results while being relevant for intra-URL relatedness computing. For the same reason, some unpopular terms blended in URLs do not match any results. The reason is that Google and Yahoo do not provide data that is not representative enough *i.e.* for terms that are not requested enough by their users. These facts limit the accuracy of intra-URL relatedness computing and is one of the reason why extra features such as $ranking$ are included in the feature set. However, the data provided by these statistical tools from the search engines rely on searches that have been made by users. To the best of our knowledge this data is not biased by commercial interest but a full access to Web search logs would highly improve the relevancy of intra relatedness metrics and, as a result, the classification performance as well. Despite this limited access to data, the results presented in this paper provide strong hints regarding the relevancy of using search engine query data for phishing detection.

A last issue of using on-line tools for inferring intra-URL relatedness is the delay implied by multiple HTTP requests for every term that compose a URL. Even though using streaming analytics techniques such as STORM can lead to a five-fold reduction of the amount of time (0.77 seconds) as presented in Section VI. A solution to speed up the process would be direct access to search engine query data. An alternative solution is to cache a full or partial copy of related searches in a local database and make only HTTP requests when needed. Such a solution can be implemented in the machine hosting PhishStorm. Then PhishStorm can quickly request the local database to compute intra-URL relatedness for every URLs it is subjected to.

VIII. RELATED WORK

For most related work the datasets used for assessment and the implemented techniques of phishing URL detection are not publicly available, making quantitative comparisons impossible.

In recent years, many techniques have been developed to cope with phishing and have focused on the real-time identification of this threat. One approach is to compare the content of presumed phishing Web pages with the original Web page being phished as in [29], [30], [31], [32], [33]. The main shortcoming of such a method is that the site being phished must be first identified. Another is that this approach is limited to rogue Web sites which is just one of several types of phishing (*i.e.* drive-by download attacks). PhishStorm relying only on URL analysis covers a larger scope than the latter.

Passively captured DNS traffic is used to recognise malicious domains in [34], [35]. The technique relies on machine learning algorithms applied to DNS-based features. The limitation of this approach when applied to phishing is that it identifies malicious domains instead of malicious URLs. For obfuscated phishing using URLs based on popular domain names *e.g.* Google Docs as seen in Section V-C, it is inefficient because these domain names are legitimate.

Consequently automated techniques to identify phishing URLs have been developed. Most rely on the extraction of phishing heuristics based only on the URL components. Features such as the length of the level domain, the path, the tokens, and the number of tokens at each level are considered in [23]. Some domain name-related information like ranking, WHOIS information, AS number, blacklisted status, etc. are used in [15], [22], [36]. A lexical analysis is performed to create binary features from each label observed in the URL in [15], [22], [23], [36]. In these techniques, label extraction is performed by splitting according to basic separators (/, ., =, -, etc.) whereas our method is more sophisticated [16]. In addition these approaches need previous knowledge about the exact labels being used in URLs. PhishStorm although relying on labels that compose URLs, only computes from these labels and analyses numerical relatedness metrics. This relatedness can be calculated from previously unseen labels as long as these appear in search engine query data. Moreover none of these methods consider the semantic dimension of labels composing URLs as we do.

More predictive approaches have been developed to cope with phishing. In [37], a tailor made blacklist suited to single machine is proposed, this blacklist is built based on the machine's logs and historical attacks from other machines that are considered as similar. In [24], several phishing URLs are grouped according to common pattern in order to extract a common regular expression. Then new potential phishing URLs are generated according to the variable part of regular expressions. In [38], the authors show how parameters can be tuned in the presence of several hundreds of features.

Recently some solutions considering the semantic dimension of phishing attacks have been proposed. In [39], the content of phishing Web pages is mined. A semantic concept is extracted from every sentence composing the Web page. These concepts are then compared with templates learned from known phishing pages through a machine learning algorithm to determine if the Web page is a phishing one. Approximately the same technique is used in [40] to block phishing emails based on semantic content analysis. The common aspect with our approach is the leveraging of semantic information for phishing detection, a concept close to word relatedness. However previous research targets phishing emails and Web pages content. Semantic relatedness analysis is performed on such content with existing similarity metrics, whereas in our work it is applied to URLs with new similarity metrics based on search engine query data. Our work is complementary to [17], where phishing domain names likely to be registered by phishers are generated in a predictive process based on a natural language model to build predictive blacklists. It is also different from [41], where we determine URL set maliciousness based on semantic analysis for forensic purpose. Here we focus on real-time identification of full phishing

URLs based on machine learning and new features, regardless they are based on malicious or legitimate domains.

Unlike detection oriented phishing approaches, PhishLive [42] searches for malicious URL presence and frequency within the traffic of an edge network in order to be able to scale detection tools such as middleboxes. Therefore, the authors particularly focused on timing characteristics and location of malicious URL access. In addition, they are able to track the HTTP redirection chain to such URLs.

To prevent phishing, a two-factor authentication is promoted in [43] but with a second factor which is completely transparent to the user (stored in his own browser) through the use of javascript API. Alternatively, fighting against phishing attacks can be achieved by targeting the main vectors *i.e.* spamming emails. A comprehensive overview on different data mining approaches for spammin email detection is given in [44].

Search engine query data has already been used for the analysis of search interests over time [45]. By monitoring the variation of interest for terms related to influenza, Ginsberg et al. [46] estimate the magnitude of flu infection for a given geographic region. In [47], Web search logs are used to improve search engine results. However, to the best of our knowledge we are the first to use this data for the purpose of word-relatedness evaluation.

IX. CONCLUSION AND FUTURE WORK

This paper introduces PhishStorm, an efficient phishing URL detection system relying on URL lexical analysis. The approach is based on the intra-URL relatedness. This relatedness reflects the relationship among the words blended into a URL and particularly into the part of the URL that can be freely defined and the registered domain. We leverage search engine query data in order to extract 12 features from a URL characterizing its intra relatedness and its popularity. The proposed features were used in supervised classification on a ground truth dataset of 96,018 phishing and legitimate URLs. This experiment yielded a classification accuracy of 94.91% with a low false positive rate of 1.44%. This experiment was extended to introduce a URL rating system, PhishStorm, to dynamically compute a *risk score* for URLs. The *risk score* on the testing dataset is able to correctly identify 99.22% of the legitimate and phishing URLs for 83.97% of the dataset. We have extended an initial approach [11] towards real-time analytics by leveraging recent Big Data streaming architectures and patterns based on STORM and Bloom filters.

Future work will consist in releasing components of the tools as an add-on for a Web browser such as Mozilla Firefox. In addition, the technique proposed in [17], which is complementary to that introduced in this paper, will be merged to create a phishing detection system with a larger scope of action. We also plan to release the analytics related part in a larger Big Data security analytics stack, which is under current development in our lab.

REFERENCES

[1] "Gartner survey shows phishing attacks escalated in 2007," Gartner Research, Tech. Rep., 2007.

[2] "2010 Identity Fraud Survey Report," Javelin Strategy & Research, Tech. Rep., 2010.

[3] P. J. Nero, B. Wardman, H. Copes, and G. Warner, "Phishing: Crime that pays," in *eCrime Researchers Summit (eCrime), 2011*. IEEE, 2011, pp. 1–10.

[4] "Global Phishing Survey: Trends and Domain Name Use," APWG, Tech. Rep. 1H2014, 2014.

[5] "SSAC Advisory on Fast Flux Hosting and DNS," ICANN Security and Stability Advisory Committee, Tech. Rep. SAC 025, 2008.

[6] "Phishing Activity Trends Report," APWG, Tech. Rep. 2nd Quarter 2014, 2014.

[7] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[8] R. L. Cilibrasi and P. M. Vitanyi, "The Google similarity distance," *Transactions onKnowledge and Data Engineering*, vol. 19, no. 3, pp. 370–383, 2007.

[9] P. Kolb, "Disco: A multilingual database of distributionally similar words," in *Proceedings of KONVENS-2008*, 2008.

[10] G. A. Miller *et al.*, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[11] S. Marchal, J. François, R. State, and T. Engel, "PhishScore: hacking phishers' minds," in *Proceedings of the 10th International Conference on Network and Service Management 2014 (CNSM 2014)*, 2014.

[12] P. Mockapetris, "RFC 1034: Domain Names - Concepts and Facilities," 1987.

[13] ——, "RFC 1035: Domain Names - Implementation and Specification," 1987.

[14] P. Mockapetris and K. Dunlap, "Development of the domain name system," in *Proceedings of the 1988 ACM SIGCOMM*. ACM, 1988.

[15] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the 2007 ACM workshop on recurring malcode*. ACM, 2007, pp. 1–8.

[16] T. Segaran and J. Hammerbacher, *Beautiful data: the stories behind elegant data solutions*. O'Reilly Media, Incorporated, 2009.

[17] S. Marchal, J. François, R. State, and T. Engel, "Proactive discovery of phishing related domain names," in *Research in Attacks, Intrusions, and Defenses*. Springer, 2012, pp. 190–209.

[18] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge," *Psychological Review; Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.

[19] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[20] L. Michael, W. Nejdl, O. Papapetrou, and W. Siberski, "Improving distributed join efficiency with extended bloom filter operations." in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications*, 2007, pp. 187–194.

[21] M. Khonji, Y. Iraqi, and A. Jones, "Lexical url analysis for discriminating phishing and legitimate e-mail messages," in *2011 International Conference for Internet Technology and Secured Transactions (ICITST)*, 2011, pp. 422–427.

[22] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: an application of large-scale online learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 681–688.

[23] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: URL names say it all," in *Proceedings of INFOCOM*. IEEE, 2011, pp. 191–195.

[24] P. Prakash, M. Kumar, R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *Proceedings of INFOCOM*. IEEE, 2010, pp. 1–5.

[25] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *SIGKDD Exploration Newsletter*, vol. 12, no. 1, pp. 49–57, 2010.

[26] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[27] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[29] T.-C. Chen, S. Dick, and J. Miller, "Detecting visually similar web pages: Application to phishing detection," *ACM Transactions on Internet Technology*, vol. 10, no. 2, pp. 1–38, 2010.

[30] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proceedings of the 4th international conference on security and privacy in communication netowrks.* ACM, 2008, pp. 1–6.

[31] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proceedings of the 18th international conference on World Wide Web.* ACM, 2009, pp. 571–580.

[32] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th international conference on World Wide Web.* ACM, 2007, pp. 639–648.

[33] T.-C. Chen, T. Stepan, S. Dick, and J. Miller, "An anti-phishing system employing diffused information," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 1–31, 2014.

[34] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *19th Usenix Security Symposium*, 2010.

[35] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive dns analysis service to detect and report malicious domains," vol. 16, no. 4, 2014, pp. 1–28.

[36] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2009, pp. 1245–1254.

[37] J. Zhang, P. Porras, and J. Ullrich, "Highly predictive blacklisting," in *Proceedings of the 17th conference on Security symposium.* USENIX Association, 2008.

[38] P. Barraclough, G. Sexton, M. Hossain, and N. Aslam, "Intelligent phishing detection parameter framework for e-banking transactions based on neuro-fuzzy," in *Science and Information Conference (SAI), 2014*, 2014, pp. 545–555.

[39] J. Zhang, Q. Li, Q. Wang, T. Geng, X. Ouyang, and Y. Xin, "Parsing and detecting phishing pages based on semantic understanding of text," *Journal of Information & Computational Science*, no. 9, pp. 1521–1534, 2012.

[40] V. Ramanathan and H. Wechsler, "phishGILLNET phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training," *EURASIP Journal on Multimedia and Information Security*, vol. 2012, no. 1, pp. 1–22, 2012.

[41] S. Marchal, J. François, R. State, and T. Engel, "Semantic based DNS forensics," in *Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security (WIFS).* IEEE, 2012, pp. 91–96.

[42] L. Cao, T. Probst, and R. Kompella, "Phishlive: A view of phishing and malware attacks from an edge router," in *Proceedings of the 14th International Conference on Passive and Active Measurement - PAM.* Springer-Verlag, 2013, pp. 239–249.

[43] B. Braun, M. Johns, J. Koestler, and J. Posegga, "Phishsafe: Leveraging modern javascript api's for transparent and robust protection," in *4th ACM Conference on Data and Application Security and Privacy - CODASPY.* ACM, 2014, pp. 61–72.

[44] N. Spirin and J. Han, "Survey on web spam detection: Principles and algorithms," *SIGKDD Exploration Newsletters*, vol. 13, no. 2, pp. 50–64, 2012.

[45] J. Rech, "Discovering trends in software engineering with Google Trend," *ACM SIGSOFT Software Engineering Notes*, vol. 32, no. 2, pp. 1–2, 2007.

[46] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant, "Detecting influenza epidemics using search engine query data," *Nature*, vol. 457, no. 7232, pp. 1012–1014, 2008.

[47] H. Liu, J. He, Y. Gu, H. Xiong, and X. Du, "Detecting and tracking topics and events from web search logs," *ACM Transactions on Information System*, vol. 30, no. 4, pp. 21:1–21:29, Nov. 2012.

Jérôme François is researcher at INRIA (Institut National de Recherche en Informatique et en Automatique) in France in the Madynes team. He's also fellow at SnT, University of Luxembourg.

He was previously a Ph.D. research engineer at INRIA Nancy - Grand Est. He received his Ph.D. on robustness and identification of communicating applications from the University Henri Poincaré in Nancy, France, in December 2009.

His main research activities are focused on network management, particularly network security and configuration. He therefore contributes to anomaly detection, DNS security, network-based fingerprinting and botnet tracking.
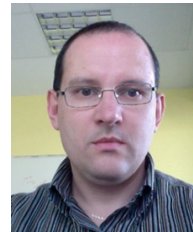
Radu State is a senior researcher with the Interdisciplinary Center on Security and Trust in Luxembourg. He was a former professor at the University of Lorraine and a senior researcher at INRIA Nancy, Grand Est. Having authored more then 100 papers, his research interests cover network and system security and management. He holds a Ph.D and a HDR from University of Lorraine and a M.Sc from the Johns Hopkins University.
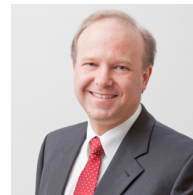
Thomas Engel is Professor for Computer Networks and Telecommunications and Vice-Director of the Interdisciplinary Center for Security, Reliability and Trust (SnT) at the University of Luxembourg. From 1987 to 1995 he studied Physics and Computer Science at the University of Saarbruecken, Germany, where he graduated in 1992 and received the title Dr. rer. nat. in 1996. Prof. Dr. Thomas Engel is member of the Information and Communication Security Panel ICS of NATO and Civil High-Level Expert for Electronic Communications (representing Europe) of NATO CEP/CCPC. He is speaker of the regional group Trier/Luxemburg of the German Society for Computer Science (GI).
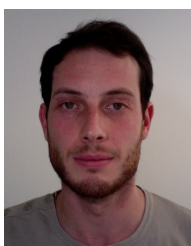
Samuel Marchal received his engineering degree and M.Sc. degree in computer science from TELECOM Nancy, a French leading school in computer science, in 2011. He is currently pursuing a joint Ph.D. degree at the Interdisciplinary Centre for Security, Reliability and Trust (University of Luxembourg) and the University of Lorraine.

His interests lie in web security, network security and intrusion detection techniques.