A Risk-Scoring Feedback Model for Webpages and Web Users Based on Browsing Behavior

MICHAL BEN NERIA and NANCY-SARAH YACOVZADA, Tel-Aviv University
IRAD BEN-GAL, Stanford University

It has been claimed that many security breaches are often caused by vulnerable (naïve) employees within the organization [Ponemon Institute LLC 2015a]. Thus, the weakest link in security is often not the technology itself but rather the people who use it [Schneier 2003]. In this article, we propose a machine learning scheme for detecting risky webpages and risky browsing behavior, performed by naïve users in the organization. The scheme analyzes the interaction between two modules: one represents naïve users, while the other represents risky webpages. It implements a feedback loop between these modules such that if a webpage is exposed to a lot of traffic from risky users, its "risk score" increases, while in a similar manner, as the user is exposed to risky webpages (with a high "risk score"), his own "risk score" increases. The proposed scheme is tested on a real-world dataset of HTTP logs provided by a large American toolbar company. The results suggest that a feedback learning process involving webpages and users can improve the scoring accuracy and lead to the detection of unknown malicious webpages.

CCS Concepts: • **Information systems** → **Clustering**; **Page and site ranking**; **Personalization**; **Web log analysis**; • **Security and privacy** → **Social aspects of security and privacy**

Additional Key Words and Phrases: Machine learning, naïve user behavior, link-based ranking algorithms, spectral clustering, malware detection

ACM Reference Format:
Michal Ben Neria, Nancy-Sarah Yacovzada, and Irad Ben-Gal. 2017. A risk-scoring feedback model for webpages and web users based on browsing behavior. ACM Trans. Intell. Syst. Technol. 8, 4, Article 53 (May 2017), 21 pages.
DOI: http://dx.doi.org/10.1145/2928274

1. INTRODUCTION

The battle against cyber security threats, in general, and malicious webpages, in particular, is becoming increasingly difficult as attackers develop extremely sophisticated techniques to overcome traditional security measures [Perdisci et al. 2010]. The human factor makes this battle even more complex, as the weakest link in security is often not the technology used but rather the users [Schneier 2003]. It has been claimed that most security breaches come from within the organization [Ponemon Institute LLC 2015a]; in some cases, such breaches are caused by malicious employees, yet, in many cases, they are caused by vulnerable (naïve) employees. In recognition of the growing risk due to these users, endpoint security is currently becoming an important priority for organizations [Ponemon Institute LLC 2015a]. The risky behavior of users is

This research was partially supported by the Ministry of Science and Technology of Israel, grant no. 5714272 (principal investigator: Prof. Irad Ben-Gal).
Authors' addresses: M. B. Neria and N.-S. Yacovzada, Department of Industrial Engineering, Tel Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel; emails: {yaacov.michal, nancy.yaco}@gmail.com; I. Ben-Gal, Management Science & Engineering, Stanford University, Stanford, CA 94305, USA; email: bengal@tau.ac.il.

DOI: http://dx.doi.org/10.1145/2928274

likely to emerge in cases where the users do not fully understand the potential risk consequences of their actions, such as browsing risky webpages. Downloading a file, playing a video [Pot 2015], or even viewing a PDF document [Anon 2014; Selvaraj and Gutierrez 2010] from seemingly safe websites can result in malicious software being downloaded unintentionally. This can lead to great damage to the organization. For example, Ponemon Institute claims in their 2015 report that the average 10,000-employee company spends $3.7 million a year dealing with phishing attacks [Ponemon Institute LLC 2015b]. In another report, also from 2015, 703 IT experts were interviewed, and 80% of them reported that drive-by download attacks are frequently seen in their organization's IT networks [Ponemon Institute LLC 2015a]. A potential security solution of managing security permissions based on the employee's position/expertise was found to partially address the risk: CloudLock research [CloudLock 2015] reveals that 52,000 instances of third-party cloud applications were installed in the organization cloud by highly privileged users. Such a scenario is particularly troublesome, given the fact that privileged accounts are highly coveted by cybercriminals. Taking all the above into account, nowadays, organizations are required to put in more effort towards the prevention of threats and risk exposures caused by naïve users. Accordingly, new security solutions are expected to be adaptive and personalized, such that they learn and capture the user's behavior over time.

This article proposes a machine learning scheme for personalized security solutions that focus on learning the browsing behavior of users in the organization. A major contribution in this work is a proposed feedback scheme between two modules – the scoring module for users and scoring module for webpages. The proposed approach aims at preventing vulnerable users in the organization from accessing risky webpages. The presented scheme provides the following advantages over existing conventional security measures: (i) *Behavioral Analysis*: Today's malware developers generate a large number of polymorphic variants of the same malware (e.g., by using executable packing or other code obfuscation techniques) [Perdisci et al. 2010]. As a consequence, anti-virus products (AVs) are challenged to remain up-to-date, and their scanners often generate a high number of false alerts. While malware can change rapidly, user behavior is often more stable and predictable. Accordingly, the proposed method tracks naïve users who can lead to newly recognized risky webpages, without the need for examining the properties, signature or content of the pages. As a result, the proposed scheme is more robust against evolved malware techniques and is relevant for the detection of a large variety of malware types; (ii) *Performance*: Risk scoring of webpages using conventional tools is usually obtained by executing and analyzing the webpage source code, which often requires high computational efforts. On the other hand, the proposed method uses users' behavioral meta-data only, and therefore, performance obstacles can be reduced significantly; (iii) *Accessible Data*: The proposed scheme uses only raw HTTP logs as its input, which in most cases can be easily accessed by the organization's administrator. This overcomes the need for cookie-based information or any other prior knowledge tag on the user's vulnerability/level of expertise.

The proposed approach contains two types of learned entities, web users and webpages. For each of the entities, the model maintains a "risk score" that represents the risk exposure to the organization by that entity. These risk scores are generated by two modules: (i) the *user module* that can be considered as a security-related profiling that learns the user's browsing behavior and (ii) the *webpage module*, which is a risk scoring framework for webpages. Finally, a feedback loop is proposed to represent the interaction between users and webpages.

The main premise is that a user who is more exposed to malicious webpages is more likely to be less "security-aware" and put the organization at higher risk. In a similar manner, it is assumed that a webpage that obtains a lot of traffic from risky users,

e.g., phishing sites, is suspected of being more risky. A dynamic simulator using a real dataset is performed to examine this premise. The experiment shows that the interaction between the user module and webpage improves the scoring accuracy.

The rest of the article is organized as follows: Section 2 briefly reviews some relevant literature. Section 3 states the proposed approach more formally. Section 4 describes the webpage learning module. Section 5 describes the user learning module. Section 6 presents some experimental results based on a real dataset of an American toolbar company. Section 7 summarizes the main conclusions and discusses some directions for further research.

2. RELATED WORK

Malware detection based on user behavior and personalized security systems is a relatively new research area that is not well-categorized in the literature. Risky webpages are defined in this article as pages that contain phishing scams, downloaded unauthorized software/content (e.g., videos or PDF files), hidden malware code or drive-by downloads. Webpages that fetch malicious ads are considered to belong to a different category, which is not targeted by the proposed scheme, as display ads are not necessarily embedded in the webpage itself and often are controlled by Real Time Bidding. In *drive-by downloads*, the user is lured to a malicious webpage that contains code, typically written in JavaScript, which exploits vulnerabilities in the victim's browser or in the browser's plugins. If successful, malware is downloaded on the victim's machine, which, as a consequence, often becomes a member of a botnet. Many of the drive-by download detection tools apply *dynamic analysis* methods, which execute the page content and monitor the processes for malicious activities [Canali et al. 2011]. While these tools are often precise, their analysis is costly. Moreover, human expert assistance is usually required for the final classification of the malware. On the other hand, *static analysis* approaches rely on static factors of the webpage, such as its textual content, features of its HTML and JavaScript code, and characteristics of the associated URL. These techniques are much faster but less accurate than the dynamic methods [Canali et al. 2011].

Drive-by download attacks are very popular among malicious webpages [Provos et al. 2007]. However, there are many other malicious threats to organizations, such as (manually) downloaded Trojan horses and viruses, among others. These threats can often be found in webspam pages. The term *webspam* refers to pages on the World Wide Web that are created with the intention of misleading search engines. Webspam detection algorithms are categorized into three main groups [Spirin and Han 2012]: analysis of the page's *content features*, such as word counts, language models and content duplication. Another group of algorithms utilizes *link-based information* [Page et al. 1999; Gyöngyi et al. 2004; Krishnan and Raj 2006; Sobek 2002; Kleinberg 1999; Lempel and Moran 2000], such as neighbor graph connectivity, link-based trust and distrust propagation, link pruning, graph-based label smoothing and link-based anomalies. The third group includes algorithms that exploit *click stream data* and *user behavior data* [Yuting Liu et al. 2008; Miller et al. 2001; Yiqun Liu et al. 2008; Webb et al. 2008], such as query popularity information and HTTP session information.

Many of the research studies concentrate on the webpage content/characteristics, while only few of them consider aspects of user behavior. The few that do address it (and are categorized as the third group of webspam detection methods) perform their analysis based on user access logs, as this work does. Two known user-based ranking algorithms that apply link-based techniques are *BrowseRank* and *usage of weighted input to HITS*. The *BrowseRank* [Yuting Liu et al. 2008] computes the page importance by constructing a user browsing graph, which is based on user browsing history. In the user browsing graph, webpages are represented by vertices, while the transitions

among the pages are represented by edges. A continuous-time Markov process defined over the user browsing graph, with respect to the time spent in a webpage, and its stationary probability distribution reflect the page's importance. In a similar manner, in the *usage of weighted input to HITS* [Miller et al. 2001], the web server logs are used for graph construction. The authors propose a modification to the adjacency matrix as an input to the HITS algorithm, such that the most frequently followed links play the largest role in determining new authority weights.

Unlike the above-mentioned webspam detection studies, which are mainly based on identifying webspam pages (that are not necessarily harmful to the organization), the suggested approach aims to detect webpages that may put the organization at risk, specifically through naïve users. This is the reason why the proposed approach learns the interaction of users and risky webpages and assesses the risk factors associated with both. This objective is fulfilled by proposing a specific graph that represents the interaction between users and webpages, as described in Section 4. Next, webpages in the proposed graph are scored by well-established link-based algorithms, such as PageRank [Page et al. 1999], Inverse PageRank [Krishnan and Raj 2006], HITS [Kleinberg 1999] and SALSA [Lempel and Moran 2000].

To the best of our knowledge, non-comprehensive work has been conducted on *personalized security systems*. These systems should analyze human behavior factors and human browsing characteristics for the purpose of malware detection. To date, an extensive body of literature exists on the analysis of *malicious behavior detection*. However, most of these studies concentrate on detecting malware writers/users, also referred to as black hats, hackers, or crackers. They analyze the behavior of a program, a computer system or a webpage, rather than analyzing the behavior of the users who consume the content. For example, the assumption that email spammers operate as a global, organized, virtual social network of spammers was explored in Xu et al. [2009]. The method of spectral clustering was applied to a set of spam messages collected under the Honey Pot project for defining and tracking these spammers' social networks. However, this work, similar to many others, did not considered naïve users as factors, which can improve the malware detection task.

Personalized security systems take relevant human factors into account, including the user profile. *User profiling* is obtained by modeling users based on their network browsing behavior. The profiling can often be categorized into three user feedback types: implicit (non-invasive), pseudo-feedback, and the explicit feedback.

Commercialized security profiling schemes have to be non-invasive by relying only on implicit feedback, without disrupting user productivity. In a study conducted by Microsoft [Leontjeva et al. 2013], a holistic approach was presented for the security classification of Skype users, relying only on implicit feedback. Similar to the approach in this work, the authors combined information from diverse sources, such as static users' profiles, time series that represent user activities, and related social connections. As the results demonstrate, fraud classification improves as more of these sources are added. This concept of combining information from diverse sources is applied in this work, but in contrary to focusing on detecting fraudulent Skype users, we aim to detect naïve users over the web and assess their risk.

Although relatively new to cyber security, the integration between behavioral risk scores and system performance was already considered in other fields (e.g., see Kenett et al. [2009] and Bai et al. [2012]). For example, in Bai et al. [2012], the authors proposed a risk-based approach that aims to detect risks associated with a semantic web service with the highest impact on users. The authors analyzed two factors of risk estimation: failure probability and importance. A Bayesian network was constructed to model the complex relationship ontology classes. Kenett et al. [2009] proposed a framework that aims to detect user interface (UI) problems of web services (by tracking

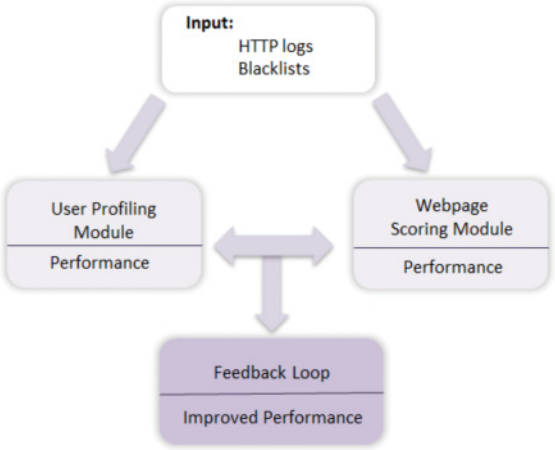

Fig. 1. Proposed feedback loop scheme.

service usability) and to define when an intervention is needed in the system in order to improve its usability. The article presents a framework for identifying user errors due to changes in the service context, and for mitigating such risks. Usability tracking was performed by applying Statistical Process Control and the intervention time was defined by a Dynamic Linear Model. Although we use different tools and have different system objectives, few of the concepts in the above works can be adopted to the proposed approach.

Following all the above-mentioned methods, this work proposes the foundation for a personalized security system based on two modules. First, a non-invasive user profiling module learns the user's behavior and generates a risk score for the user. Second, the webpage learning module scores relevant webpages by also relying on the users' score. Next, once a user wishes to access a webpage, the security system uses behavioral historical data to approximate the potential risk of that specific transaction and react accordingly. More details on each of these modules as well as the integration between them are described in the following sections.

3. PROPOSED APPROACH

The proposed approach considers two types of entities: users and webpages. The model maintains for each of the entities a "risk score" that represents the risk exposure to the organization related to that entity. The main premise is that a user who has more entries to malicious webpages is more likely to be naïve or less "security-aware." This user will probably visit more malicious pages in the future and, therefore, might put the organization at a higher risk, and his risk score should be increased accordingly. On the other hand, tracking his browsing paths can lead to other unknown malicious webpages. These new webpages may help in revealing additional risky users and so on. In a similar manner, it is assumed that a webpage that obtains a lot of traffic from risky users (e.g., phishing sites) has a risk score that should increase as well. This mutual information trading can be seen as a feedback-loop model, as described in Figure 1.

Based on these assumptions, a general scheme is proposed for deriving the personalized risk scores of users and webpages by modeling and analyzing the interactions between them. Figure 1 shows the proposed feedback loop scheme that integrates the two modules: one models the users, while the other models the webpages. Each of these modules generates a "risk score" that is used by the other module. The first module

generates a *user risk score*, which is based on the user's browsing behavior. The second module generates a *webpage risk score*, which is derived by learning the visited webpages and their non-standard linkage characteristics. The two modules share information with each other to improve the overall scoring performance. The proposed scheme produces two types of scores.

User Profiling Score u_i. This module applies an automated identification of security-related profiles and detects the risky users. The proposed approach implements unsupervised machine learning techniques for identifying communities that share "similar behavior" over the cyberspace. Risk patterns associated with naïve users may be evident in different parts of the data. Therefore, a method is proposed for fusing multiple information sources into different feature categories and a technique for the construction of a holistic similarity graph. Next, the users are clustered over this similarity graph based on the Normalized Spectral Clustering method proposed by Ng et al. [2002] and the K-medoids algorithm proposed by Kaufman et al. [1987]. The proposed procedure is as follows:

ALGORITHM 1: User Scoring Module

Input: HTTP logs, blacklists, webpages' risk score $(r_1, \ldots, r_d)$.
Output: users' risk score $(u_1, \ldots, u_n)$.

1. Process HTTP logs
2. Data model:
   i. Initialization: initialize the users' risk profile with respect to known risky webpages
   ii. Construct the similarity matrix $\boldsymbol{F}$ using the user's aggregated features
   iii. Construct the similarity matrix $\boldsymbol{G}$ using the user-webpage graph
   iv. Construct a holistic similarity matrix $\boldsymbol{A}$
      i. Features weighting: solve the linear programming problem to find the optimal weights

$$\boldsymbol{A} = \omega_1 \cdot \boldsymbol{F} + \omega_2 \cdot \boldsymbol{G}$$

3. Learning model:
   i. Cluster the users
4. Users' risk score := users' profile

Webpage score r_j. This module scores the webpages based on their potential malicious reputation while using link-based methods with a specialized browsing graph. Several well-established algorithms are compared for this task: PageRank [Page et al. 1999], Inverse PageRank [Krishnan and Raj 2006], HITS [Kleinberg 1999] and SALSA [Lempel and Moran 2000]. The proposed procedure is as follows:

ALGORITHM 2: Webpage Scoring Module

Input: HTTP logs, blacklists, users' risk score $(u_1, \ldots, u_n)$.
Output: webpages' risk score $(r_1, \ldots, r_d)$.

1. Process HTTP logs
2. Data Model: Construct the browsing graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$
   i. V webpages' initial risk score
   ii. E edges weighted by the transitions of risky users
3. Learning model:
   iii. Rank webpages using the link-base algorithm
4. Webpages' risk score := webpages' rank

The next two sections separately detail each of the autonomous modules without considering the feedback between them at this stage. Each module includes a data model and a learning model. Based on the experimental results, the suggested method is compared to a baseline method. Then, in Section 6, we describe and analyze the results of the feedback loop model. Once these modules are integrated, an improvement in the scoring accuracy is demonstrated.

4. WEBPAGE MODULE: A RISK SCORING FRAMEWORK

4.1. Data Model

The proposed directed *Browsing Graph* consists of vertices representing webpages and edges representing the transitions of users from one webpage to another. An edge represents various types of transitions such as clicking a hyperlink or a bookmark, redirection of the user to another page automatically or simply writing a new URL manually in the address bar. The browsing graph follows the concept used in previous works for webspam detection [Xue et al. 2003; Miller et al. 2001; Yuting Liu et al. 2008]. Multiple edges from one vertex to another collapse into a single edge, implying that it is not a multigraph. The main difference in the proposed graph versus that described in earlier works is the proposed edge weighting (described below) as well as the use of the graph for risk scoring instead of spam ranking. In particular, weights are computed in two ways:

(i) *Without Reference to Users* – The weights of all edges are either 1 if there was a transition between the corresponding webpages or 0 if there was no transition between the corresponding webpages.
(ii) *With Reference to Users* – the edge's weight corresponds to the transitions and the *risk scores* of the users who transitioned between the vertices.

More formally, let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ be a browsing graph, where $\boldsymbol{V}$ is the set of vertices and $\boldsymbol{E}$ is the set of edges. Let $U_{i,j}$ be the set of users who transitioned over the edge (i, j). Denote by $k \in U_{i,j}$ a user k who transitioned from vertex i to vertex j. Let u_k denote the risk score of user k, such that:

$$u_k = \begin{cases} 1, & \textit{if user } k \textit{ is risky} \\ 0, & \textit{otherwise} \end{cases}$$

Note that the index i, j in k and u_k are omitted for the purpose of simplicity of exposition. The risk potential $\tau_{i,j}$ of edge (i, j) is defined as follows:

$$\tau_{i,j} = \begin{cases} \dfrac{\sum_{k \in U_{i,j}} u_k}{|U_{i,j}|}, & \textit{if} \sum_{k \in U_{i,j}} u_k > 0 \\ \varepsilon, & \textit{otherwise} \end{cases}$$

such that edges with transitions of non-risky users have a minor effect (ε). Note that the above calculation is relevant for the case with reference to users' risk score. When the users' risk score is not used, we use $\tau_{i,j} = 1, \forall (i, j) \in \boldsymbol{E}$. Let $I_{isLink}(i, j)$ be an index that indicates whether the edge (i, j) is a hyperlink, i.e.,

$$I_{isLink}(i, j) = \begin{cases} 1, & \textit{if } \exists u_k \in U_{i,j} \textit{ who transitioned by hyperlink from } i \textit{ to } j \\ 0, & \textit{otherwise} \end{cases}$$

where an edge in which at least one user transitioned by a hyperlink is denoted as a hyperlink. Let α be a predefined weight for hyperlink, $\alpha \in [0, 1]$. Then, the weight of edge (i, j) is defined by:

$$\rho(i, j) = \tau_{i,j}((1 - \alpha) + \alpha I_{isLink}(i, j)).$$

When $\alpha = 1$, a *Hyperlink Graph* is obtained: it is a graph where the edges represent physical hyperlinks that are used to transition from one vertex to another. Note that the main difference between this hyperlink graph and the Web graph, which is commonly used in the literature [Page et al. 1999], is that the edges in the former exist if at least one user used the hyperlink, whereas the edges in the latter are not generated with respect to the user's usage, that is, an edge can exist even if no user used the hyperlink. It is well known that links can be created by web developers to mislead search engines [Garcia-molina and Gyöngyi 2005; Spirin and Han 2012]: most of these links are not used and exist in order to maximize the page's score with respect to link-based algorithms. Link farms, honey pots and spam link exchange are all means for manipulating the Web graph to delude these algorithms. Incorporating the hyperlinks used only makes the model less sensitive to such issues. For instance, Yiqun Liu et al. [2008] used the proportion of the used hyperlinks in a page as one of their features for detecting spam webpages. They show that a page in which most of the hyperlinks were not actually used by users is more likely to be spam. This graph structure is used as the baseline data model in this work. It is analyzed under the same two conditions mentioned earlier, that is, with and without reference to the users' risk score.

4.2. Learning Model

Previous studies [Zhen-quan et al. 2010; Huang et al. 2013; François et al. 2011; Kamvar et al. 2003; Wang et al. 2010] showed that link-based webspam algorithms, mainly PageRank [Page et al. 1999], are useful for detecting malicious entities, e.g., pears in P2P networks or botnets. These studies motivate the implementation of such type of algorithms in this work. This section describes a few well-established link-based ranking algorithms that are further examined in later sections.

4.2.1. PageRank. Generally, PageRank [Page et al. 1999] has been used to evaluate the importance of crawled pages over the web. A relevant interpretation of this rank is the probability of staying/entering a webpage during a random walk. In this work, the algorithm is used to find the most risky pages among the "crawled" ones, which are the webpages accessed by the users. Such an approach was used in previous studies [Yuting Liu et al. 2008; Xue et al. 2003].

4.2.2. Inverse PageRank. This method has been used to detect new (unknown) spam pages by the Anti-Trust Rank [Krishnan and Raj 2006] and BadRank [Sobek 2002] algorithms. The Inverse PageRank is executed by applying the PageRank algorithm to a graph with reversed edge directions. Reverse edge directions are based on the intuition that it is very unlikely for spam pages to be pointed to by reliable pages (see Gyöngyi et al. [2004]). The equivalent implication in this work is to use the same approach for detecting new risky pages.

4.2.3. HITS. The algorithm evaluates the importance of webpages using two characteristics: (i) *Authority*—A webpage is considered an authority page if it contributes towards providing information about a specific topic, and (ii) *Hub*—A webpage is considered a hub page if it directs to authority pages on a specific topic. The algorithm is based on the following observation: a good hub points to good authorities, and a good authority is pointed to by good hubs. HITS assigns two scores to each webpage; therefore, the algorithm generates two vectors of scores (an authority vector and a hub vector). In this work, we learn each of the vectors using a different learning method: *HITS auth.* and *HITS hub*, respectively. HITS is usually used to evaluate the importance of webpages that are related to a specific query/topic (typically 1,000–5,000 webpages [Farahat et al. 2006]). However, previous studies use implementations on a larger graph that covers several queries/topics (see Miller et al. [2001] and Xue et al. [2003]). In this

work, the graph is constructed based on some batches of HTTP logs for the detection of risky webpages that are not necessarily related to a specific topic. In addition, the standard adjacency matrix (where 1 represents a link, 0 otherwise) was replaced with a weighted adjacency matrix, in which the entries are defined by the edge's weights, $\rho(i, j)$, as described in the previous section. The use of a non-standard adjacency matrix was presented in earlier studies, such as [Miller et al. 2001; Wang 2002].

4.2.4. SALSA. This algorithm is often considered as an improvement or enhancement of HITS [Kleinberg 1999]. It is a combination of the hub and authority concept of HITS and the random walk of PageRank. The implementation of SALSA in this work is based on the same modifications that were used for the HITS algorithm. While a few previous works used similar browsing graphs for the detection of webspam by implementing PageRank and HITS [Yuting Liu et al. 2008; Xue et al. 2003; Miller et al. 2001; Poblete et al. 2008]), no other work, to the best of our knowledge, used SALSA for this purpose.

Each of the above link-based methods provides a webpage risk score within the range of 0 to 1. In order to compare the performance of the different methods, one can analyze the relative order of a given score rather than the magnitude of the score. Accordingly, each score is transformed to a lower percentile rank. Thus, each webpage's risk score is mapped to the percentage of webpages that have lower or equal scores. Although information may be lost in such a procedure, it is used here for validation purposes and provides a solid comparison of the outputs of several scoring methods, as seen in the following sections.

4.3. Experimental Results

In this section, a naïve procedure was used to define the users' risk score based on the visited webpages (further improvement of the AUC measure was achieved later by applying a feedback loop between the user and the webpage modules, as detailed in Section 6). In addition, due to memory limitations, in all the experiments (Sections 4.3, 5.3, 6), the granularity level of the vertices is a domain (aggregating webpages to their registered domains). Therefore, any reference to "webpage" in these sections is related to "domain." In addition, all experiments include a random selected subset of http logs data that were gathered from an American toolbar company.

A major challenge faced when analyzing a browsing dataset and determining the webpages' risk score is that the true risk level (*ground truth*) of the webpages is unknown. The problem can be defined as a semi-supervised learning problem in the context of a few blacklists of known/suspected webpages, which we use in this work. Note, however, that only 1% of the webpages in this study appeared in the known blacklists. Accordingly, webpages' scores were defined over a [0,1] range, where 0 represents a non-malicious webpage and 1 represents a page suspected of being malicious. The initial weights of the webpages that were not included in the blacklists (i.e., 99% of the webpages) were set to zero, although there is no real guarantee that these webpages are truly non-risky. Based on the data model graphs (i.e., the hyperlink and the browsing graphs with or without the reference to users' risk scores), the above-mentioned algorithms were used to evaluate the webpages' risk scores. The evaluation phase consisted of two aspects: analyzing the known risky webpages and manually analyzing the top 100 scored webpages, as described below:

(i) *Analyzing Webpages that Are Known as Risky (i.e., appear in known blacklists).* A shuffled stratified 10-fold cross validation procedure was applied for this task. In each iteration, the risky webpages of the test set were "hidden," that is, their risk value was assigned to zero, and a graph was reconstructed according to the labeled webpages of only the training set. Next, each of the above-mentioned algorithms

Table I. Average AUC in the Browsing Graph with Reference to Users' Risk Scores versus the Baseline Data Model (Hyperlink Graph without Reference to Users' Risk Scores)

	Hyperlink	**Browsing**	
	Reference to users' risk scores		
Scoring method	**w/o**	**with**	**Lift**
SALSA auth.***	0.524	**0.685**	0.161
SALSA hub***	**0.561**	0.677	0.116
HITS auth.*	0.533	0.628	0.095
HITS hub**	0.553	0.637	0.084
Inverse PageRank*	0.550	0.628	0.078
PageRank	0.496	0.562	0.066

was executed separately over this graph. The AUC (area under the ROC curve) of the test set was measured for each algorithm. The relative score (lower percentile rank, as discussed in Section 4.2) of the "hidden" risky webpages in a descending ordered test set of webpages was examined. The average AUC of all the 10-folds was used as a performance measure for each of the examined algorithms. Note that the evaluation is conservative in the sense that unlabeled risky webpages in the test set with a zero score reduced the AUC performance, even though the algorithm could separate those from the others well.

(ii) *Analyzing Webpages that Are not Known to be Risky*. This task was performed manually by visually examining the first 100 high-score webpages using the best scoring methods. A webpage was visually verified as suspicious if it belonged to one of the following classes: free porn/sex chat/gaming/hacking (mainly breaking phone's OS) or other visually suspicious characteristics (e.g., phishing site).

The tested graph consisted of ∼15,000 webpages (vertices). The following data model hypotheses were analyzed in this Section:

—Reference to the users' risk scores improves the scoring accuracy of the learning models compared to models that do not refer to users' risk scores.
—A browsing graph structure (i.e., not limited to hyperlinks) improves the model accuracy compared to a hyperlink graph.
—An interaction of both factors, a browsing graph and a reference to users' risk scores, further improves the scoring accuracy compared to a baseline-hyperlink graph that does not refer to users' scores.

The conducted experiment confirmed that an interaction does exist between the graph structure and the users' scores factors. Thus, when applying both factors together, they improved the scoring accuracy of all the learning methods (see Table I). The improvement in five out of six scoring methods was found to be statistically significant. For comparison, when applying only one of the considered factors (either the browsing graph or the references to the users' risk scores), only two scoring methods yield better scores, and those that were improved obtained a lower significance level.

Table I presents the lift achieved when using a data model that contains a browsing graph with reference to the users' risk score. Columns represent the used data model, and the best result for each type is shown in bold. Each row represents the scoring method used, and the table entries show the average AUC of the scoring method with the data model type. Statistical significance levels of the data models are provided for each scoring method and are marked as follows: * for the 0.05 level, ** for the 0.01 level, and *** for the 0.001 level. In addition, "with" stands for "with reference to users' risk score," and "w/o" stands for "without reference to users' risk score."

One can observe from Table I that the suggested data model significantly improve almost all scoring methods (excluding PageRank, which seems to be less efficient for

Table II. A Comparison of the SALSA Authority Scoring Based on a Browsing Graph with Reference to Users' Risk Score with Respect to Other Data Model and Learning Method Combinations

T-test version	**Paired T-test**	**Welch' T-test**		
Graph	**Browsing**	**Browsing**	**Hyperlink**	**Hyperlink**
Reference to users' risk scores	**with**	**w/o**	**with**	**w/o**
SALSA auth.	-	0.003	**0.067**	0.000
SALSA hub	**0.639**	0.039	0.015	0.001
HITS auth.	0.010	0.001	0.002	0.001
HITS hub	0.031	0.011	0.007	0.000
Inverse PageRank	0.028	0.009	0.001	0.000
PageRank	0.000	0.000	0.000	0.000

the considered scoring task). Note that the SALSA authority score utilizes the proposed data model most efficiently, with the largest lift and the highest average AUC value. Accordingly, this method is selected as the benchmark for all other cases. Table II describes the improvement in the P-value of the SALSA authority score based on the suggested data model (browsing graph with reference to users' risk score) compared to all the other combinations of data model and scoring method. The only two non-significant combinations are the SALSA hub scores based on the suggested data model and the SALSA authority scores when using the hyperlink graph with reference to users' risk scores (bolded in Table II). All other data model and scoring method combinations are found to significantly underperform in the SALSA authority case when applied to the suggested data model. First, all methods were compared to the SALSA authority on browsing graph with reference to users score. A paired t-test was used since each method was applied on the same folds (samples) in the cross validation process. On the other hand, for the comparison of different graph structures, we used the Welch's t-test for independent (unpaired) samples and unequal variances. The samples are considered independent since the shuffle step was performed before the cross validation process. Our objective was to detect the improvement of the suggested data model; hence we used the two-tailed test.

A similar test was performed over a larger graph that consisted of ∼36,000 webpages. Due to memory limitations, the SALSA algorithm could not be applied. However, similar results were obtained for all the other algorithms, and they supported the strength of the suggested data model.

Next, the top 100 webpages with the best scoring methods were visually evaluated to find suspicious webpages. The SALSA authority scores yielded 46 suspicious webpages, and 27 of them were free porn sites. The SALSA hub yielded 34 suspicious webpages, and at the top list of the HITS hub, there were 31 suspicious webpages. The three scoring methods were chosen based on their high performance in the previous experiments. As a simple naive comparison, a random selection of webpages resulted only in 7 out of 100 webpages that seemed to be suspicious. Thus, all three scoring methods yielded higher results than the random test, and the most substantial advantage was provided by the SALSA authority method.

In summary, the suggested data model, which is based on a browsing graph enriched by the users' risk score, was found to significantly improve the model accuracy. Among the scoring methods, the SALSA authority method provided the best results.

5. USER MODULE: SECURITY–RELATED USER PROFILING

5.1. Data Model

The previous section addressed the risk scoring webpage task that is partially based on users' risk scores. This Section presents an automatic method to generate the users' risk scores based on their browsing behavior. To model the users' behavior, the proposed

method considers several categories of behavior. One such category is the interaction of users with webpages they access, which is represented by a data structure of an incidence matrix (as described in Section 5.1.1). Another category focuses on the interaction of users with risky webpages. A third category examines the browsing behavioral characteristics of users. These browsing characteristics are represented by the vector space of the features. This behavioral model is used to identify communities of users that share similar behaviors over a holistic similarity graph. The holistic graph is constructed by fusing multiple feature categories, where each category is represented by its own matrix. A linear combination of these matrices is obtained by solving a linear programming problem to find the optimal weights of each matrix, resulting in a holistic similarity matrix. As will be described in Section 5.2, a clustering algorithm is then applied over the similarity graph to detect users' communities. The procedure that determines the weights, such that the clustering represents the communities well, is described in Section 5.1.4.

5.1.1. User-Page Incidence Graphs. Users who pose a query to a web search engine or directly access a URL often require specific information needs. Users access webpages that seem to be closely relevant to their intended information needs. Hence, it is reasonable to assume that a frequently clicked set of pages by a user reflects the information that the user finds interesting or relevant. Further, it is observed that users with similar information needs click on a similar set of pages. This behavior forms a click-through bipartite graph. Such a data structure is extremely large and sparse and can be used for collaborative filtering.

Let $\boldsymbol{X}^0$ be a binary incidence matrix that represents the interaction between n users and d webpages, whereby $x^0_{i,j} = 1$ if user i visited webpage j; otherwise, $x^0_{i,j} = 0$. In the same manner, let $\boldsymbol{X}^1$ be an incidence matrix that represents the interaction between n users and $\hat{d}$ known *risky* webpages, where the matrix entries $x^1_{i,j}$ are the webpage risk score r_j generated by a risk scoring framework as described in Section 4. Namely, consider two user-webpage matrices $\boldsymbol{X}^0, \boldsymbol{X}^1$ with the same number of rows n but different number of columns ("features"), d and $\hat{d}$, respectively, where $\hat{d} \ll d$. Finally, note that if several categories of webpages exist, the proposed scheme detailed below can be extended to more than two incidence matrices in a straightforward manner.

5.1.2. Browsing Behavior Features. Each web user can be represented by a behavioral profile reflecting facts such as geographic location, activities on social networks, favorite browser, favorite music genre, IP, mobile phone browsing activity, distribution of browsing activities by time, total time spent on the web, total number of sessions per period, total number of transactions, percentage of browsing volume over weekends, and whether the user uses several browsers, among others. Thus, the user's browsing behavior can be modeled by m aggregated behavioral features $f_1, \ldots, f_m$. In addition, some special engineered aggregated behavioral features can be added, such as the Trx-Magnitude, which is the magnitude of transactions the user executes on average in a session, and CybernautsMeasure, which measures the centrality of internet activities in the user's life. Beyond these features that represent the overall behavioral profile of a user, the proposed scheme also measures some *security-related features*. For example, it measures the "*morning risky activity percentage,*" reflecting the percentage of the user's morning activities in known risky webpages; the "*number of distinct IPs,*" reflecting if the user is frequently changing his IP (which may point to some criminal activities run in the background); and some self-explanatory features such as "*percentage of transactions made from toolbar,*" "*percentage of transactions with missing CUA,*" and "*bot transaction percentage.*"

As indicated, the above scheme generates a feature extraction process that eventually yields m aggregated features. Let $\boldsymbol{Y_{n\times m}}$ be a scaled user-feature matrix, where each feature $\{f_i\}_{i=1}^{m} \in \mathbb{R},\ 0 \leq f_i \leq 1$. The proposed scheme further divides the m features into p categories, such that each category contains a group of features with a similar meaning or with some common ground. In particular and without loss of generality, the implemented scheme divides the features into two categories: a *general features matrix*, denoted by $\boldsymbol{Y}^0$, and a *risky features matrix*, denoted by $\boldsymbol{Y}^1$.

5.1.3. User Holistic Similarity Graph. The next step in the proposed procedure is to operate symmetric and nonnegative similarity functions and kernels on $\boldsymbol{X}^0, \boldsymbol{X}^1, \boldsymbol{Y}^0$, and $\boldsymbol{Y}^1$ in order to map them to proximity matrices $\boldsymbol{F}^0, \boldsymbol{F}^1, \boldsymbol{G}^0$, and $\boldsymbol{G}^1$, respectively. Note that following such a mapping, matrices $\boldsymbol{X}^0, \boldsymbol{X}^1, \boldsymbol{Y}^0$, and $\boldsymbol{Y}^1$ (that are of different dimension) are all mapped to square $(n \times n)$ symmetric matrices. Such a mapping procedure takes care of the sparsity difference between the matrices because $\boldsymbol{X}^0$ and $\boldsymbol{X}^1$ (in which columns represent webpages) are very sparse matrices by nature, while $\boldsymbol{Y}^0$ and $\boldsymbol{Y}^1$ (in which columns represent behavioral features) are often more dense and of lower dimensions. Next, the proposed scheme produces a *holistic similarity matrix* $\boldsymbol{A}$ by a linear combination of the proximity matrices, while multiplying each matrix by its corresponding weight (constant). More formally, denote the weight of features category p by ω_p; then, the *holistic similarity matrix* is computed simply using a linear combination of the corresponding matrices

$$\boldsymbol{A} = \omega_1 \cdot \boldsymbol{F}^0 + \omega_2 \cdot \boldsymbol{F}^1 + \omega_3 \cdot \boldsymbol{G}^0 + \omega_4 \cdot \boldsymbol{G}^1,$$

where the method for automatically determining the weights is given in Section 5.1.4. Now, given a set of n data points and some notion of similarity $w_{a,b} \geq 0$ between all the pairs of data points a and b, the similarity graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ can be defined. Each vertex v_a in this graph represents user a. The users are clustered over this similarity graph based on known clustering algorithms, as described in Section 5.2.

5.1.4. Feature Weighting. In this section, a method for automatically determining the weights $\omega_1, \ldots, \omega_{\mathrm{p}}$ is described. A weight ω_{p} represents the relative impact (importance) of the p-th feature category on detection. A simple linear programming problem is solved to obtain the optimal weights. Optimality is obtained when the clustering algorithm represents the communities over the similarity graph well. Following Shi and Malik [2000], the quality of clustering is measured by the *Normalized Cut* ("*Ncut*") measure, which computes the cut cost as a fraction of the total edge connections to all the vertices in the graph [Shi and Malik 2000]. Note that each feature category is a disjoint subset of all the features extracted from different information sources. Thus, this process can be seen as Feature Subset Weighting - a process that examines how the partitioning (clustering) is affected under different weights and selects the optimal weights. For example, a subset of features is held out ($\omega_{\mathrm{i}} = 0, \mathrm{i} \in [1, \ldots, \mathrm{p}]$) and does not participate in the clustering, or different values are assigned to the weights of different feature categories. For each given combination of weights (e.g., $\omega_1 = \omega_2 = \cdots = \omega_{\mathrm{p}} = \frac{1}{\mathrm{p}}$), the performance of the clustering is evaluated.

An additional procedure is proposed to visualize the effect of weights $\omega_1, \ldots, \omega_{\mathrm{p}}$ by Eigenspace investigation, i.e., visualization of the K first eigenvectors of the Laplacian matrix under different values of $\omega_1, \ldots, \omega_{\mathrm{p}}$. Thus, let A be the holistic similarity matrix (representing the users-graph G(V, E)), and let D denote the degree matrix (a diagonal matrix which contains the degree of each vertex in G). Calculate the symmetric normalized Laplacian matrix $\boldsymbol{L_{sym}} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \cdot \boldsymbol{A} \cdot \boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{I}$ is the identity matrix. Note that the first K eigenvectors of $\boldsymbol{L_{sym}}$ form K "tight" clusters on the surface on the K-sphere [Ng et al. 2001]. Thus, to project K clusters over the similarity matrix $\boldsymbol{A}$, one

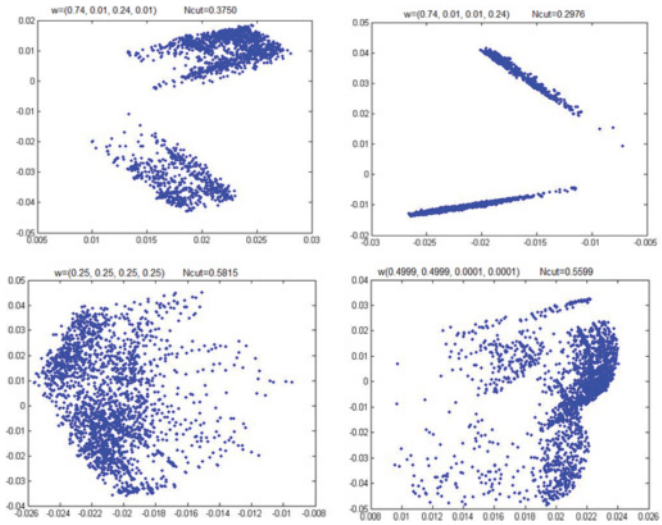

Fig. 2. Investigate the Eigenspace by tuning the ω_p weights. The upper-left plot is obtained by the optimal weights $\omega = (0.74, 0.01, 0.01, 0.24)$, for which the best partition of points is achieved (NCut measure = 0.29). Note that for a uniform weight vector $\omega = (0.25, 0.25, 0.25, 0.25)$, in the lower-left plot, the NCut measure is higher (0.58), and as the plot illustrates, a good partitioning is not achieved.

can consider only the first K eigenvectors of $\boldsymbol{L_{sym}}$ and cluster by rows. The proposed procedure performs a simple search over $\omega_1, \ldots, \omega_p$ and selects the values that following the clustering of $\boldsymbol{L_{sym}}$'s rows yield the tightest (smallest distortion) clusters. Since K is smaller than the number of columns in $\boldsymbol{A}$ ($K < n$), this dimensionality reduction allows us to visualize the eigenvectors in a plot and view the obtained clusters. For the case of two clusters of users, risky and non-risky users ($K = 2$), consider the first two eigenvectors of $\boldsymbol{L_{sym}}$ (i.e., those that belong to the two smallest non-zero eigenvalues) and plot them on a 2-D axis. Repeat this process with different possible values of ω_p, and investigate how the change in one or more of the weights impacts the separation of the Laplacian eigenvectors. For an example of the visualized eigenvectors for different values of ω_p, see Figure 2.

The intuition behind this method is provided by the *Rayleigh-Ritz theorem*. The clustering objective function can be seen as a relaxed optimization (minimization) problem. Using the Rayleigh-Ritz theorem, it can be shown that the solution of this problem is given by the vector v for which the eigenvector corresponds to the second smallest eigenvalue of $\boldsymbol{L_{sym}}$. Another approach for finding the optimal weights in the case of $K > 3$ is obtained by solving the *Normalized Min-Cut* optimization problem to find those weights that minimize the *Ncut* internal validity measure. Thus, solving a constrained nonlinear multivariate problem where, at each iteration, the clustering algorithm is performed based on the holistic similarity matrix $\boldsymbol{A}$ that contains the decision variables (the weights ω_p) and K clusters is as follows:

$$Min\ \ Ncut\,(C_1, \ldots, C_K) = \sum_{i=1}^{K} \frac{cut\,(C_i)}{vol\,(C_i)}$$

$$Subject\ To:$$
$$(1)\ \boldsymbol{A} = \omega_1 \boldsymbol{F}^0 + \omega_2 \boldsymbol{F}^1 + \omega_3 \boldsymbol{G}^0 + \omega_4 \boldsymbol{G}^1$$
$$(2)\ \omega_1, \omega_2, \omega_3, \omega_4 \geq 0$$
$$(3)\ \omega_1 + \omega_2 + \omega_3 + \omega_4 = 1$$

Recall that $w_{i,j}$ is the *similarity* between node i to j. The *weight* between two clusters is defined as $\mathrm{W}(C_l, C_m) = \sum_{i \in C_l, j \in C_m} w_{i,j}$. The cut of a partition is defined as $cut(\mathrm{C}) = \frac{1}{2}\mathrm{W}(C_i, \bar{C}_i)$. Let consider the *cluster "size"* as $vol\,(C) = \sum_{\mathrm{i} \in \mathrm{C}} \deg(\mathrm{i})$ where a *degree* of node i is computed as $\deg(i) = \sum_{j=1}^{n} w_{i,j}$. The optimization process is stopped by a simple convergence rule: when the lowest Ncut is obtained and no further improvement is achieved in the next iteration. This analysis can be enhanced by evaluating each iteration with additional well-known internal validity measures, such as the *Silhouette coefficient* and the *Dunn index* (as performed in the experimental results in Section 5.3). While data fusion can enrich the clustering by revealing hidden patterns that may be evident in different parts of the data, the proposed feature subset weighting also enhances the generalization by reducing overfitting.

5.2. Learning Model

Given a similarity graph G(V, E), where the vertices represent users and edges represent some proximity between the users, a *Community Detection* problem can be solved. Clustering algorithms can be employed to identify groups of users with high behavioral similarity. In this Section, the *Spectral Clustering* (SC) [Von Luxburg 2007] and the *K-medoids* [Zhang and Couloigner 2005] algorithms are examined. This work focuses on the *Normalized Spectral Clustering* method, proposed by Ng et al. [2002], because it is closely related to the well-known graph-partitioning problem. The simplest and most direct way of constructing a partition of the graph G is to solve the minimum normalized cut problem. The normalized version is proposed here because it constrains the "community" sizes and avoids trivial solutions. Unfortunately, introducing the normalization condition results in an NP-complete problem. However, as shown in Von Luxburg [2007], the SC algorithm can be used to solve a relaxed version of this problem (relaxing Ncut leads to normalized Spectral Clustering). Additionally, the SC algorithm obtains data representations in a low-dimensional space and is used to reduce errors from noise factors or outliers.

5.2.1. Determining the Number of Clusters. As with most other clustering algorithms, the optimal number of clusters, K, is unknown. Following the general clustering problem, there are as many clusters as one requires, with no real optimal solution. One practical heuristic suited for selecting K in SC problems is the *Eigen gap heuristic*, which is based on *Spectral Graph Theory* [Chung 1997]. This heuristic is presented and justified in Von Luxburg [2007]. By analyzing the decay of the eigenvalues of the Laplacian matrix, the Eigen gap (also called a *spectral gap*) can reveal the number of natural clusters that exist in the data. The Eigen gap is defined as the difference between two consecutive eigenvalues. One can search for a significant increase in the Eigen gap of the eigenvalues, arranged in increasing order. The applied rule is simple: select the number K such that all the eigenvalues $\lambda_1, \ldots, \lambda_K$ are very small but λ_{K+1} is relatively large. Additional ways of exploring the natural grouping of the data include analyzing the number of connected components and the spectrum of the Laplacian matrix. The number of Laplacian eigenvalues of magnitude zero should be equal to the number of connected components in the Laplacian graph. This implies that one could estimate K simply by counting the number of eigenvalues equal or close to zero. This criterion works when the clusters are well separated and provide a good measure of the natural structure of the data. An additional common heuristic is setting some well-known internal validity index φ (e.g., Silhouette coefficient) as an optimization criterion and searching for the value of K that optimizes φ, which, in our case, involves performing clustering and checking the value of φ under different values of K. The shortcoming of this heuristic is that most indexes φ scale with the number of clusters, while the Eigen

gap heuristic is independent of the final partitions that resulted from the clustering. In our case, all heuristics revealed that the best number of clusters is $K = 2$.

5.3. Experimental Results

In this section, static blacklists of known malicious webpages were used to construct the risky features matrix $\boldsymbol{Y}^1$ as well as construct the users-risky webpages incidence matrix $\boldsymbol{X}^1$. Note that all risky webpages share a score of 1, which remains fixed throughout the analysis. In this section, we analyze the autonomous user module without integrating it with the webpage scoring framework, i.e., without the proposed feedback scheme presented in Section 6.

For the experimental evaluation of the proposed approach, a random sample of 200,000 web transactions that consist of 2,341 distinct users is considered. First, a pre-processing phase is performed to obtain the $(n \times n)$ proximity matrices as described in Section 5.1.3: $\boldsymbol{F}^0$ represents the similarity between users according to the general features matrix, $\boldsymbol{F}^1$ is set according to the risky features matrix, $\boldsymbol{G}^0$ is set according to the user-webpage binary incidence matrix, and $\boldsymbol{G}^1$ is set according to the user-risky webpage binary incidence matrix (consisting of risky webpages only). Note that $\boldsymbol{G}^1$ here is a binary matrix because all risky webpages share the same risk score (as there is no feedback from the scoring framework). The evaluation scheme is executed in three parts. First, it finds the number of clusters, K. Section 5.2.1 presented several heuristics that aim to find an appropriate number of clusters K and reveals that the optimal number is $K = 2$. Second, it finds the optimal weights $\omega_1, \omega_2, \omega_3, \omega_4$ to quantify the impact of each of the different feature categories, as presented in Section 5.1.4. Finally, it compares two clustering methods, namely the K-medoids algorithm and the Spectral Clustering algorithm. For each weight assignment, K value and a clustering method, the resulting clustering was evaluated using known clustering metrics: the Silhouette coefficient, the Connectivity score, the Ncut measure and the Dunn index. The *Ncut* measure was discussed in Section 5.1.4: a lower value implies a better cluster configuration [Shi and Malik 2000]. The *Silhouette coefficient* combines ideas of both tightness (intra-cluster distance) and separation (intra-cluster distance), as introduced in Rousseeuw [1987]: a higher value represents a better cluster configuration. The *Dunn index* has been introduced in Dunn [1974]: if a data set contains well-separated clusters, the distances among the clusters are usually large, and the diameters of the clusters are expected to be small. Therefore, a higher value implies a better cluster configuration. The *Connectivity score* validates whether the clusters capture the local neighborhood structure of users in the graph. A higher connectivity value implies a better capturing of locality.

Different assignments and configurations of the examined algorithm as well as the assigned weights were carried out. Table III presents the most informative configurations or the ones that achieved the best results, while others are omitted due to focus and space limitation. One can see from Table III that in addition to the dimensionality reduction that the Spectral Clustering (SC) algorithm obtains, it performs better than the K-medoids algorithm in terms of all the internal validity measure (Silhouette, Connectivity, Ncut, and Dunn). Arrow signs represent the desired direction for each validity measures.

Figure 3 presents a detailed view on the performance of the SC algorithm under different weights combinations. One can see that combination no. 6, with $\omega = (0, 0, 0, 1)$, achieves the lowest Ncut value (Ncut $= 0.168$), yet at the same time produces the worst silhouette score (0.007) and a poor Dunn index (1). In terms of all the internal validity measures, the best weights combination is given by $\omega = (0.74, 0.01, 0.01, 0.24)$, implying that the best partition is achieved in the case of proper fusion of behavioral features with the user-webpage graph. Additionally, it is seen that the impact of each one of the

Table III. User Module - Clustering Evaluation

Weights Combination	ω_1	ω_2	ω_3	ω_4	Algo.	Silhouette ↑	Connectivity ↑	Ncut ↓	Dunn ↑
1	**0.74**	**0.01**	**0.01**	**0.24**	**SC**	**0.285**	**1.000**	**0.350**	**1.394**
	0.74	0.01	0.01	0.24	K-medoids	0.284	0.998	0.352	1.392
2	0	0	1	0	SC	0.253	0.739	0.495	0.951
	0	0	1	0	K-medoids	0.225	0.697	0.479	0.977
3	0.5	0.5	0	0	SC	0.215	0.913	0.555	1.268
	0.5	0.5	0	0	K-medoids	0.201	0.886	0.564	1.251
4	0.25	0.25	0.25	0.25	SC	0.123	0.843	0.550	1.104
	0.25	0.25	0.25	0.25	K-medoids	0.095	0.906	0.570	1.102
5	0	0	0.5	0.5	SC	0.120	0.729	0.495	0.977
	0	0	0.5	0.5	K-medoids	0.076	0.690	0.541	0.977
6	0	0	0	1	K-medoids	0.009	0.998	0.101	1.000
	0	0	0	1	SC	0.007	0.989	0.168	1.000

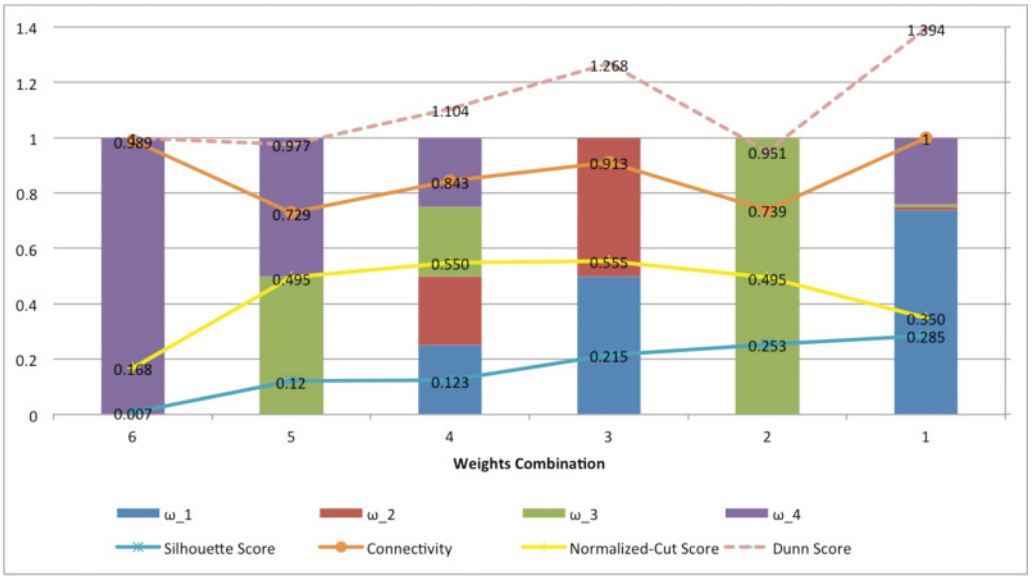

Fig. 3. Spectral clustering performance—the usefulness of the different feature categories.

different sources of information is maximal when combining them to construct a single similarity graph that represents the users' behavior for various feature categories.

The analysis shows the usefulness of the different feature categories, particularly the importance of the general features matrix ($\boldsymbol{F}^0$) with a relatively high weight ($\omega_1 = 0.74$). This fact indicates that in order to better score the users, one should analyze their overall browsing behavior and not only focus on their activity in webpages known to be risky.

6. FEEDBACK LEARNING MODEL EXPERIMENTAL RESULTS

This section describes the experimental results of the entire scheme, integrating the users' scoring module and the webpages' scoring module by a feedback scheme. To test such feedback and risk mitigation over time, a dynamic simulation was performed based on the real HTTP logs dataset provided by a leading American toolbar company, which represents a "real-life" scenario. The simulation was constructed such that new

Table IV. Integration Evaluation (AUC)

Scoring method	Baseline	Iteration 1	Iteration 2	Iteration 3
HITS auth.	0.545	0.55	0.565	**0.654**
HITS hub	**0.549**	**0.571**	**0.574**	0.64
Inverse PageRank	0.545	0.552	0.57	0.61
PageRank	0.509	0.514	0.524	0.577

information (batch of data) was consistently added to the analysis at each iteration. For each such iteration, the users' scoring module learned the users' behavior and generated the updated scores according to the procedure mentioned in Section 5. Then, the webpages' scoring module scored all webpages using these users' risk scores according to the procedure mentioned in Section 4.

Table IV presents an analysis for which the baseline is a hyperlink graph with no reference to the users' risk score. The baseline graph is constructed for each of the scoring methods based only on the first batch of logs. To evaluate the proposed approach, a browsing graph with reference to users' risk score was then constructed at each iteration and for each of the scoring methods (bold numbers denote the highest accuracy in each iteration). The experimental results in Table IV shows that the feedback scheme improves the model accuracy (using the AUC measure) with each new batch of data that is added. Note that such an improvement, although consistent in this study, is not guaranteed in general and can result in over-fitting and a decrease in accuracy. A common phenomenon in continual streaming systems is that historical data can become obsolete over time for a learning algorithm in dynamic environments. In such a scenario, at some point, the learned model is less accurate, especially if it is exposed to accumulated noisy data and less informative signals. Although we support an adaptive learning scheme, the exact definition of the amount of historical data that has to be learned is case-specific and subject to future research.

Let us note that, in this study, three iterations obtained a consistent improvement for all considered scoring methods. Moreover, all these methods, except the Inverse PageRank, obtained better results than those found by the autonomous webpage scoring module experiment (Table I). For example, the HITS authority obtained an AUC of 0.654 compared to 0.628 obtained by the autonomous webpages' scoring module. The experiment does not include the SALSA algorithm and stops after three iterations due to memory limitations (we used a conventional 32GB server). As mentioned earlier, most of the webpages in the dataset were unlabeled, which can result in deteriorated AUC values. This is particularly the case when many risky webpages are unlabeled as risky (i.e., their initial risk score is set to zero), but their predicted risk score is higher. Given a threshold, all unlabeled risky webpages that were ranked above that threshold will count as a "false positive" (type I error), resulting in a decreased AUC value. Finally, the AUC lift is examined with an additional analysis. At each iteration, we constructed a new baseline, which learned both the current batch as well as the former ones (data from previous iterations). Similar to the former analysis, the baseline was defined as a hyperlink graph without reference to the users' risk score. In addition, with each new batch of data, we constructed a browsing graph with reference to the users' risk scores. This procedure was repeated for the four scoring methods presented above. Then, the suggested accumulated data model was compared to the accumulated baseline graph from the same iteration.

Figure 4 shows the percentage of the AUC lift achieved by the suggested feedback scheme.

Except for the HITS auth. method in the second iteration, for all scoring methods, the lift increases with the number of iterations. In particular, the consistent improvement

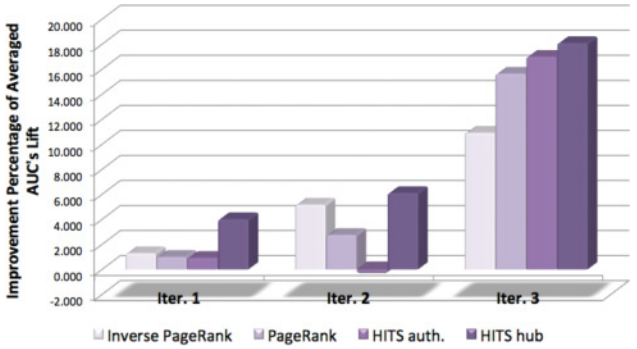

Fig. 4. Improvement percentage of the average AUC lift achieved by the proposed feedback scheme using a browsing graph with reference to the users' risk score.

of the HITS hub method lift is substantial. After three iterations, all scoring methods improved by more than 10% with respect to their initial average AUC; especially, the HITS hub method obtained an improvement of 18% with respect to the baseline AUC value and also yielded the best accuracy in each iteration. In summary, the use of a feedback-learning scheme that uses accumulated data and scores of both risky users and risky webpages is shown to yield the best results in this case study for almost all the considered scoring methods.

7. CONCLUSIONS AND FUTURE WORK

This article proposes an approach for personalized security solutions using machine learning methods. The proposed approach focuses on user behavior aspects to better address cyber threats; it is particularly appealing for analyzing naïve users (employees) within the organization. A feedback scheme is designed for deriving personalized risk scores of users and webpages. These scores are obtained by two modules that interact with each other. First, a user profiling module is proposed, in which implicit feedback from the user is learned, mainly users' behavioral patterns, to assign a corresponding risk score to each user. The second proposed module scores the webpages by their potential malicious reputation using link-based methods while relying on the users' risk scores. The entire scheme operates by monitoring HTTP logs, which can be easily accessed by the organization administrator without further requirement of other resources. Each of the modules can be executed autonomously, but a significance improvement in the model accuracy is obtained once both learning frameworks are integrated with each other. Specifically, a higher malware detection accuracy is obtained when information on "risky" users is used to estimate the webpages' risk scores, and vice versa. A series of experiments that evaluate the suggested scheme was designed and carried out with a real HTTP logs dataset taken from a toolbar company. It was shown that the use of a feedback-learning scheme that relies on accumulated data as well as risk scores of both users and webpages yield the best results in this case study for almost all the considered models.

There are several future research directions to be considered. First, it is beneficial to test the procedure on a large-scale dataset that is partially labeled. Another direction is to profile users by also using browser-fingerprinting techniques. This task will require an enhanced HTTP logs dataset. Accurate tagging of users in a supervised dataset can significantly improve the model performance, if such tagging is available. Another direction for obtaining a more general performance analysis with respect to the different algorithmic parameters can be obtained by applying an experimental design methodology. An interesting direction for risk mitigation evaluation is the use of

the proposed feedback-loop where risky websites above certain thresholds are blocked. In such a case, a scheme which is similar to the outgoing quality limit (AQL) procedure might be applied to bound the risk. Finally, for practical consideration, it is advised to design and implement an incremental update functionality of the data and learning models such that there is no need to repeat the entire learning process over the entire graph whenever new information is available.

ACKNOWLEDGMENTS

We are grateful for the support of our colleagues from the industry and the academia in this project.

REFERENCES

Anon. 2014. PDF Current Threats. 2014. http://www.malwaretracker.com/pdfthreat.php.

Xiaoying Bai, Ron S. Kenett, and Wei Yu. 2012. Risk assessment and adaptive group testing of semantic web services. *Int. J. Softw. Eng. Knowl. Eng.* 22, 5 (2012), 595–620. DOI:http://dx.doi.org/10.1142/S0218194012500167

Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel. 2011. Prophiler: A fast filter for the large-scale detection of malicious web pages categories and subject descriptors. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*. 197–206. DOI:http://dx.doi.org/10.1145/1963405.1963436

Fan RK Chung. 1997. *Spectral graph theory. 92.* American Mathematical Society.

CloudLock. 2015. *The 1% who Can Take Down Your Organization*.

Joseph C. Dunn. 1974. Well-separated clusters and optimal fuzzy partitions. *J. Cybern.* 4, 1 (1974), 95–104. DOI:http://dx.doi.org/10.1080/01969727408546059

Ayman Farahat, Thomas LoFaro, Joel C. Miller, Gregory Rae, and Lesley A. Ward. 2006. Authority rankings from HITS, pagerank, and SALSA: Existence, uniqueness, and effect of initialization. *SIAM J. Sci. Comput.* 27, 4, 2006, 1181–1201. DOI:http://dx.doi.org/10.1137/S1064827502412875

Jérôme François, Shaonan Wang, Radu State, and Thomas Engel. 2011. Bottrack: Tracking botnets using netflow and pagerank. In *NETWORKING 2011: Proceedings of the 10th International IFIP TC 6 Networking Conference*. Springer Berlin Heidelberg, 1–14. DOI:http://dx.doi.org/10.1007/978-3-642-20757-0_1

Hector Garcia-molina and Zoltán Gyöngyi. 2005. Web spam taxonomy. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the web (AIRWeb 2005)*. (Chiba, Japan.)

Zoltán Gyöngyi, Hector Garcia-molina, and Jan Pedersen. 2004. Combating web spam with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*, Volume 30. 576–587.

Junxian Huang, Yinglian Xie, Fang Yu, Qifa Ke, Martín Abadi, Eliot Gillum, and Z. Morley Mao. 2013. SocialWatch: Detection of online service abuse via large-scale social graphs. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIA CCS'13)*. 143–148. DOI:http://dx.doi.org/10.1145/2484313.2484330

Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. 2003. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*. 640–651. DOI:http://dx.doi.org/10.1145/775240.775242

Ron S. Kenett, Avi Harel, and Fabrizio Ruggeri. 2009. Controlling the usability of web services. *Int. J. Softw. Eng. Knowl. Eng.* 19, 5, 2009, 627–651. DOI:http://dx.doi.org/10.1142/S0218194009004362

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM* 46, 5, 1999, 604–632. DOI:http://dx.doi.org/10.1145/324133.324140

Vijay Krishnan and Rashmi Raj. 2006. Web spam detection with anti-trust rank. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web – AIRWeb 2006*. 37–40.

Ronny Lempel and Shlomo Moran. 2000. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Comput. Networks* 33, 1–6 (June 2000), 387–401. DOI:http://dx.doi.org/10.1016/S1389-1286(00)00034-7

Anna Leontjeva, Moises Goldszmidt, Yinglian Xie, Fang Yu, and Martín Abadi. 2013. Early security classification of skype users via machine learning. In *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security (AISec'13)*. 35–44. DOI:http://dx.doi.org/10.1145/2517312.2517322

Yiqun Liu, Rongwei Cen, Min Zhang, Shaoping Ma, and Liyun Ru. 2008. Identifying web spam with user behavior analysis. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the web (AIRWeb'08)*. 9–16. DOI:http://dx.doi.org/10.1145/1451983.1451986

Yuting Liu et al. 2008. Browserank: Letting web users vote for page importance. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*. 451–458. DOI: http://dx.doi.org/10.1145/1390334.1390412

Joel C. Miller, Gregory Rae, Fred Schaefer, Lesley a. Ward, Thomas LoFaro, and Ayman Farahat. 2001. Modifications of kleinberg's HITS algorithm using matrix exponentiation and web log records. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*. 444–445. DOI: http://dx.doi.org/10.1145/383952.384086

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Adv. Neural Inf. Process. Syst.* 2001, 849–856.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. 1999. http://ilpubs.stanford.edu:8090/422.

Roberto Perdisci, Wenke Lee, and Nick Feamster. 2010. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*.

Barbara Poblete, Carlos Castillo, and Aristides Gionis. 2008. Dr: Searcher and Mr. Browser: A Unified Hyperlink-Click Graph. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*. 1123–1132. DOI: http://dx.doi.org/10.1145/1458082.1458231

Ponemon Institute LLC. 2015a. 2015 State of the Endpoint Report: User-Centric Risk. 2015. http://www.ponemon.org/local/upload/file/2015 State of Endpoint Risk FINAL.pdf.

Ponemon Institute LLC. 2015b. The cost of phishing and value of employee training. 2015. https://info.wombatsecurity.com/cost-of-phishing.

Justin Pot. 2015. How to avoid malware when viewing videos on youtube. 2015. http://www.makeuseof.com/tag/youtube-major-source-malware/.

Niels Provos, Dean Mcnamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. 2007. The ghost in the browser analysis of web-based malware. In *Proceedings of the First Conference on 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*.

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65. DOI: http://dx.doi.org/10.1016/0377-0427(87)90125-7

Bruce Schneier. 2003. *Beyond Fear: Thinking Sensibly About Security in an Uncertain World*, Copernicus.

Karthik Selvaraj and Nino Fred Gutierrez. 2010. The rise of pdf malware. 2010. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_rise_of_pdf_malware.pdf.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 8, 2000, 888–905. DOI: http://dx.doi.org/10.1109/34.868688

Markus Sobek. 2002. Pr0 - google's pagerank 0 penalty. BadRank. 2002. http://pr.efactory.de/e-pr0.shtml.

Nikita Spirin and Jiawei Han. 2012. Survey on web spam detection: Principles and algorithms. *ACM SIGKDD Explor. Newsl.* 13, 2, 2012, 50–64. DOI: http://dx.doi.org/10.1145/2207243.2207252

Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Stat. Comput.* 17, 4, 2007, 395–416. DOI: http://dx.doi.org/10.1007/s11222-007-9033-z

Minhua Wang. 2002. A Significant improvement to clever algorithm in hyperlinked environment. 2002.

Yufeng Wang, Akihiro Nakao, and Athanasios V. Vasilakos. 2010. Doubleface: Robust reputation ranking based on link analysis in P2P networks. *Cybern. Syst.* 41, 2, 2010, 167–189. DOI: http://dx.doi.org/10.1080/01969720903584365

Steve Webb, James Caverlee, and Calton Pu. 2008. Predicting web spam with HTTP session information. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*. 339–348. DOI: http://dx.doi.org/10.1145/1458082.1458129

Kevin S. Xu, Mark Kliger, Yilun Chen, Peter J. Woolf, and Alfred O. Hero. 2009. Revealing social networks of spammers through spectral clustering. In *Proceedings of the IEEE International Conference on Communications* 1–6. DOI: http://dx.doi.org/10.1109/ICC.2009.5199418

Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma, and Hong-Jiang Zhang. 2003. User access pattern enhanced small web search. *Microsoft Res.* 2003.

Qiaoping Zhang and Isabelle Couloigner. 2005. A new and efficient k-medoid algorithm for spatial clustering. In *Computational Science and Its Applications–ICCSA 2005*. Springer Berlin Heidelberg, 181–189. DOI: http://dx.doi.org/10.1007/11424857_20

Qin Zhen-quan, LI Kai-bin, and LI Ming-chu. 2010. Webpage security evaluation model based on pagerank and system calls. *J. Chinese Comput. Syst.* 2010

Received January 2016; revised March 2016; accepted April 2016