MP-Shield: A Framework for Phishing Detection in Mobile Devices

G.Bottazzi, E.Casalicchio, D.Cingolani, F.Marturana, M. Piu
Dep. of Computer Science
University of Rome "Tor Vergata " - Italy

***Abstract*—Today, there was an exponential growth of e-services requiring the exchange of personal and sensible information over the Internet. Phishing techniques are emerging as the easiest solution to break the weakest link of the security chain: the end user. Social engineering attacks are deployed by financial/cyber criminals at a very low cost to induce naïve Internet users to reveal user ID, passwords, bank account and credit card numbers. This problem needs to be addressed in the mobile field as well, due to the large diffusion of mobile platforms such as smartphones, tablet, etc. In this paper we propose a novel framework for phishing detection in Android mobile devices which, on the one hand exploits well-known techniques already implemented by popular web browsers plug-in, such as public blacklist search, and, on the other hand, implement a machine learning based engine to ensure zero-hour protection from new phishing campaigns.**

Keywords—mobile phishing, proxy, blacklists, heuristics, machine learning.

I. INTRODUCTION

Social Engineering based attack leverages psychological manipulation of people, tricked into performing actions or disclosing confidential information. Phishing is one of the more known social engineering attack and aims at exploiting weaknesses in system processes caused by users' behavior. Indeed, a system can be secure enough against password theft (e.g. the client-server communication channel is encrypted), but nothing can be done against a naive user threatening the security of the system by revealing her/his password to a fake Web site reached, for example, via an email-embedded HTTP link.

In literature can be found different definitions of phishing that cover partially the phishing attack landscape [1-4]. The broader of these ones, identifies phishing as "a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit" [3].

The evolution of phishing techniques, the switch over malware-based phishing campaigns together with the massive spread of mobile devices for business or personal use, has clearly expanded the threat posed by phishing, especially towards the mobile world, along with people underestimating the mobile cyber-threats and neglecting even basic security measures [5-6]. 17% of all Android apps (nearly one million total) were actually malware in disguise and 36% of all mobile apps, even not malicious by design, did inadvertently harmful things like tracking user behavior [7].

Phishing attack detection in desktop environment has been widely investigated in literature [4]. On the contrary, despite the severity of the phishing problem in the mobile ecosystem, we noticed a lack of literature on the phishing attack detection for mobile devices [8-13] and, even extending the scope into the mobile malware detection, very few works have converged towards solutions running on mobile devices [14-17] and related to malware-based phishing.

Many of the approaches proposed in literature, regardless of their effectiveness, still have a strong verticality and focus on specific aspects such as: attack techniques; existing security context; systems and protocols used to capture data; methodological approaches used for phishing detection (black lists, heuristics, machine learning, etc.); devices on which deploy the developed solution.

To the best of our knowledge, none of the solutions proposed so far, shown a unified approach across different environments (such as mobile and desktop) and across subsets of the above mentioned aspects. This is mainly due to the fact that, each solution often needs a number of prerequisites that are: *i)* difficult to remove, in case some detection technique is covered by other third party software (e.g. antivirus); *ii)* hard to adapt to other contexts, enabling the exploitation of the same detection technique for other threats (e.g. botnets); *iii)* and burdensome to merge in a single tool that can exploit different targets, sources or approaches. Therefore, in this paper, we propose a unified reference model and present the Mobile Phishing Shield (MP-Shield) framework that implements it for the Android platform.

Our unified reference model has three main pillars:

- *Context and platform independence*. The approach must realize a proxy-based architecture, natively interposed to any type of traffic to and from the Internet, able to intercept every network pattern and deployable both on mobile and desktop devices;
- *Adaptability and multimodality/multifunctionality*. Each detection technique implemented (blacklists, heuristics, machine learning, etc.) must be enabled/disabled as needed (e.g. when the same feature is already implemented by third party software such as the antivirus client) and must be effectively extended to other malware detection contexts (e.g. botnets);

- *Scalability*. The approach must be able to manage different rates and types of connections, required by user applications (e.g. the proposed solution must not represent a bottleneck in case of high speed and low latency network connections) and must be able to move the computational intensive client tasks remotely in a cloud platform, in order to reduce workloads.

In particular, the MP-Shield framework, which implements the aforementioned model, has been developed with the following features: *i)* it runs in user mode rather than in root mode; *ii)* it supports different machine learning-based detection techniques, thanks to the deployment of the WEKA engine for mobile platforms, and *iii)* it could be easily delivered to Cloud-based platform to address CPU/memory/energy consuming issues typical of mobile devices and to scale with the number of customers.

The paper is organized as in what follow. In section II we review the literature. In section III are described the MP-Shield framework principles and implementation details. The experimental evaluation is presented in Section IV. Conclusions and future works are discussed in sections V and VI respectively.

II. Related Work

The huge literature on phishing detection techniques is almost completely oriented toward methodologies tailored for desktop/laptop environments (see [4] for an extensive survey). Considering the literature related to the mobile environment that, as mentioned before, is not exhaustive, we can distinguish between work on "traditional" phishing [8-13] (resulting in no particular research trend) and work on Malware-based phishing detection [14]. The lack of literature on this specific subject is a consequence of a variety of reasons spanning from (a) the inevitable convergence of the same threat to any Internet-enabled device (sometimes addressed as the Internet of Things), (b) the difficulty of implementing a performing security solution on a mobile device, (c) the absence of basic security settings on mobile devices (e.g. antivirus and firewall) and (d) the increased variability of attack vectors directed towards mobile devices.

A. "Traditional" phishing detection

Concerning the specific subject of phishing detection on mobile devices, in [6] is proposed a phishing detection taxonomy for mobile environment, trying to depict all possible scenario of phishing attacks and related countermeasures. Leaving aside all possible attacks related to very specific vectors (e.g. SMS, Bluetooth and Vishing), the paper stresses the lack of solutions dedicated to mobile devices other than black/white lists. A risk assessment on mobile platforms has been proposed in [9], where a study conducted on 85 web sites and 100 mobile applications discovered that web sites and applications regularly ask users to type their passwords into contexts that are vulnerable to spoofing. The implementation of sample phishing attacks on the Android and iOS platforms demonstrated that attackers can spoof legitimate applications with high accuracy, suggesting that the risk of phishing attacks on mobile platforms is greater than it has previously been appreciated. In [10] the authors proposed a proof-of-concept defense against spoofing and phishing attacks, exploiting the full-screen touch-based user interface. The developed framework logs user keystrokes and warns the user when potentially sensitive information is about to be entered while running an untrusted application. A phishing scheme for mobile phones is presented in [11], trying to exploit OCR text-extraction tools, in order to verify the legitimacy of a website, comparing the text extracted from a login form with the corresponding second-level domain name (SLD). This stems from the assumption that most well-known enterprises use brand name as the SLD of their official websites which is also used, as an image, within their login forms. However, the presented workflow uses also white lists and heuristics to reduce false positives or to increase the efficiency of the overall system (e.g. legitimate websites always use domain names as verification of their identities while phishers are likely to list IP address in URL to disguise their fake identities). In [12] is proposed a study on the threat posed by the notification systems, typical of smartphones, suggesting also a design principle for a OS-controlled framework for the management of the notification systems used by all the Apps installed on the mobile devices. A further research was aimed at identifying anti-phishing techniques, using the visual similarity analysis, in order to protect mobile users using the Internet via public WiFi hotspots [13].

B. Malware-based phishing detection

With regards to phishing as a mean for spreading malware (malware-based phishing), it was possible to recover some additional scientific material related to malware detection for mobile environments through network analysis techniques [14-17].

In this area, however, regardless of the selected method among the most common phishing detection strategies (e.g black/white lists, heuristics, visual similarity matching and machine learning algorithms), it is important to highlight that none of the proposed solutions is implemented in a software module suitable for mobile devices. Further, only in few cases the anomaly detection techniques used were deployed directly on the device. Most of the systems use a server-side component to send the observed data or to perform the learning process offline and plant the learned models back to the devices for the detection process.

The main difficulties seem to lie in capturing the traffic generated by the mobile devices directly from the device itself that, in the solutions analyzed, is emulated with the tools usually available on desktop environments, such as Tcpdump and Wireshark.

The few examples in which it was made a smart engine on board the device [18], continue to have, however, a strong verticality on the phishing attack being investigated (e.g. application update attack) or on the type of technology used.

Hence, it's clear that modern mobile devices, although highlighting high performance levels of interaction with the Internet, still pose great obstacles for the implementation of algorithms and technologies available in traditional detection systems, due to their reduced processing capabilities.

The approach we are proposing with this research work, instead, is a unique framework, modular in its implementation, on which we can apply as needed any detection/prevention method.

III. THE FRAMEWORK

Because none of the solutions proposed in literature is, at the same time, context and platform independent, adaptable and scalable, in this work we present MP-Shield, a mobile light and modular framework that (a) can be easily customized by users, (b) is adaptable in term of detection strategy and algorithms adopted, (c) is reusable over all compatible platforms, (d) is scalable compared to the rate and type of connections, and because computational intensive tasks can be outsourced to a cloud platform.

MP-Shield is an Android application, implemented as a proxy service on top of the TCP/IP stack, with the aim of inspecting IP packets, originated from and directed to a generic mobile user application, for phishing content. The choice for an Android service instead of a mobile app in the user space, is motivated by the need to run both in foreground and in background, even if a context switch occurs. In Fig.1, is described the high-level view of the software stack in which the MP-Shield is inserted – under the transport (TCP/UDP) handler.

In order to detect potentially malicious traffic, MP-Shield natively intercepts any outgoing and incoming IP traffic of the device, originated by either an HTTP browser, such as Firefox or Chrome, or any other app requiring an Internet connection. HTTP-get requests are intercepted from filtered IP packets with a twofold aim: on the one hand a public blacklist is queried for well-known phishing URLs, while on the other, a set of attributes (e.g. number of sub-domains in the URL, length of the URL, fraction of digits in a URL versus the length of the URL) are extracted by the proxy and exploited to assure zero-hour protection from new phishing campaigns. With regards to the latter case, MP-Shield allows users to select/de-select which attributes to use at run time among a set of formal checks concerning Internet domains and IP address structure.

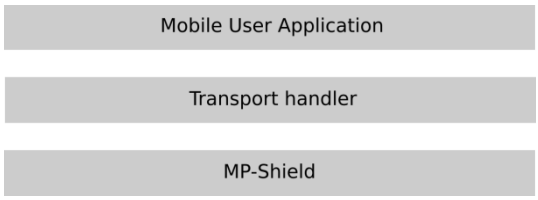

Fig.1. Application stack

It is noteworthy to mention that MP-Shield does not require root privileges for running, which is an important plus since inexperienced users may be unaware of jailbreak procedures needed to get such privileges.

A. Architecture

The MP-Shield high-level architecture is presented in Fig. 2 and is composed of three main packages, namely the *Blacklist API*, the machine learning *Classification Engine* and the *Watchdog*. The first allow querying the Google Safe Browsing service. The second run a set of machine learning algorithms based on the WEKA (Waikato Environment for Knowledge Analysis) framework. The Watchdog is in charge to invoke the other two module to detect if an URL is malicious or not.

To reduce computational overhead on the mobile device and to assure high scalability, the *Blacklist API* and the *Classification Engine* can be enabled/disabled at run time. Moreover, also the set of checks performed by the machine-learning engine can be configured through the application settings panel in order to reach a trade-off between performance and phishing protection. As a consequence, even though disabling some attributes could impact on final classification accuracy, it may allow to better match custom device settings (e.g. mobiles running a third-party antivirus may be already protected against phishing attacks and may not be interested in the activation of certain controls).

MP-Shield inspects network traffic for phishing content by means of the VPN service, a TCP/IP layer proxy provided by the Android libraries (and working without any root privilege). The VPN service analyzes IP packets before relying them to original target hosts. For each unclassified GET request, the proxy passes it to the Watchdog engine which runs asynchronously with respect to the browsing process to reduce latency. Therefore IP packets are duplicated and immediately passed to the virtual interface implemented by the VPN service and transmitted over the Internet. The copied packets are buffered and analyzed by the Watchdog, and target URLs are extracted from the obtained HTTP requests to be classified.

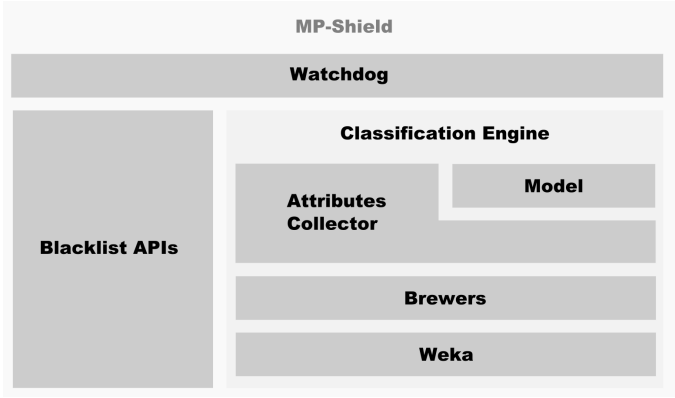

Fig.2. MP-Shield high-level architecture.

Once a new URL is extracted from an IP communication stream, the Watchdog performs two steps in parallel:

Step 1: it uses the URL to query the Google Safebrowsing service via API (Fig.3.a),

Step 2: it passes the URL to the classification engine, which a) extract the set of attributes needed to build a new WEKA instance and b) passes it to the proper classifier model, accordingly to user's preferences (Fig.3.b). The *Attributes Collector* layer is in charge of building the WEKA-compatible instance by computing values for a set of attributes.

Steps 1 and 2 can return a negative answer, that is the URL analyzed does not correspond to a phishing site, or a positive answer that is, a potential illicit URL has been detected. Once a possible threat has been detected, the Watchdog subsystem throws a warning message on the screen to inform the user of

the potential risk. In addition, the warning notification provides both the suspected domain name with the severity score of the threat. To ensure a graceful and comfortable user experience, the warning alert is subject to a context-aware logic which prevents to display it out of the web browsing session.

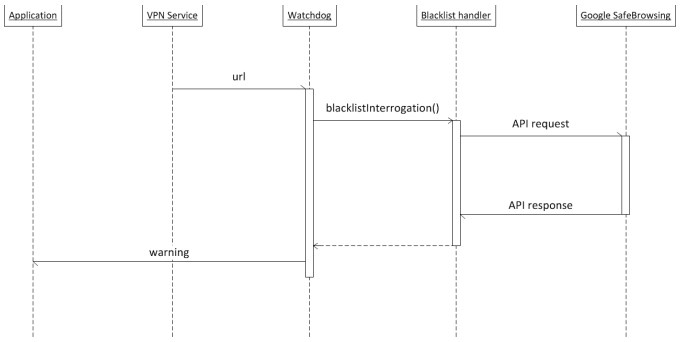

Fig.3.a Google blacklist query - sequence diagram

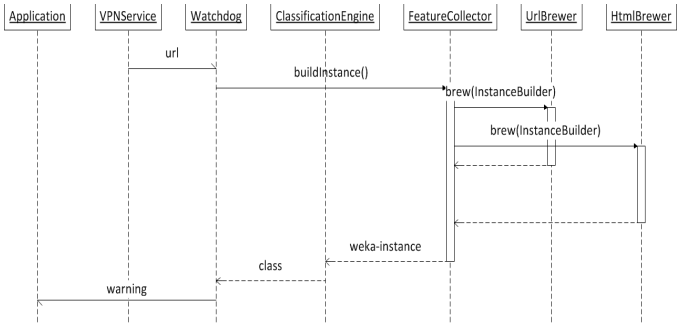

Fig.3.b. WEKA classification - sequence diagram

B. Machine learning module architecture

In Step 2 we adopted a supervised classification approach and related machine-learning model built on a dataset consisting of a uniform distribution of legitimate and phishing URLs, namely extracted from the *DMOZ* directory and from the *PhishTank* website. Due to the limited resources available on a mobile device, compared to desktop environments, we developed a lightweight software that, once trained, is able to rapidly build and classify new test instances. In this regards, since the training model itself can be considered quite stable over time (e.g. it does not require to be constantly updated or modified once compiled) we decided to compile and save it locally on the mobile. To increase efficiency and preserve robustness of the architecture, it is possible to pre-compile the classification model remotely on a cloud platform and upload it to the mobile device on demand.

The Classification Engine consists of three main components (see Fig. 2):

- The *Classification Model* is the pre-compiled model used at runtime to classify new test instances;
- The *Attributes Collector* that generates the WEKA instances;
- The *Brewers,* a set of filters, included in the Feature Collector, implementing the attribute selection and the evaluation logic, used to derive attributes from input objects (e.g. plain strings, HTML pages/forms, css, java-scripts, etc.).

At start up, a pre-compiled classification model is loaded and the Feature Collector is initialized accordingly to current application settings, that specify which security modules to enable, the brewers to be loaded, namely the attributes to be evaluated and other parameters.

The current version of MP-Shield implements two Brewers: an *UrlBrewer* filter, tailored to extract information by adopting static formal heuristics on the intercepted URL string, and an *HtmlBrewer* filter, which gathers information from the fetched page. The *UrlBrewer* computes attributes from a formal string parsing (see Fig. 6 for the complete list of attributes). It splits the input string into basically four main parts: *whole URL*, *host*, *path* and *query*. The host part is split up into its dot-separated components (namely *tokens*) and traced. For each part the brewer will compute substrings' length and counts the number of unconventional characters, such as colon, semicolon, hyphen, underscore, etc. Further derived attributes are evaluated by computing the ratio between parts' length and consonant occurrences to catch possible randomly-generated strings. Other extracted attributes from the URL string are the length of each split part, the occurrences of certain characters on each part, the presence of obfuscated host of IP-based addresses in the main path. On the other hand, the *HTMLBrewer* performs a parsing operation on the fetched web page. Basically it operates on the page's links, specifically the number of links referring to cross-domain resources or the syntactical distance between main referrer, namely the visible and clickable text. Further the number of redirections the page is subject to, the number of outward links and the distance between the inner link and the main host (trying to disclose a web forgery that relies on external legal resources to mimic target service page) are also checked.

The URL string extracted by the Watchdog is forwarded, by the Attributes Collector, to the Brewers that process the input and return the results (the extracted attributes) back to the Attributes Collector that creates a new WEKA instance (see Fig.3.b and 4). The new instance is passed to the WEKA engine that performs the classification against the proper pre-compiled model from the *Model Handler*, which finally categorize the URL as *phishing* or *non-phishing* (see Fig.5).

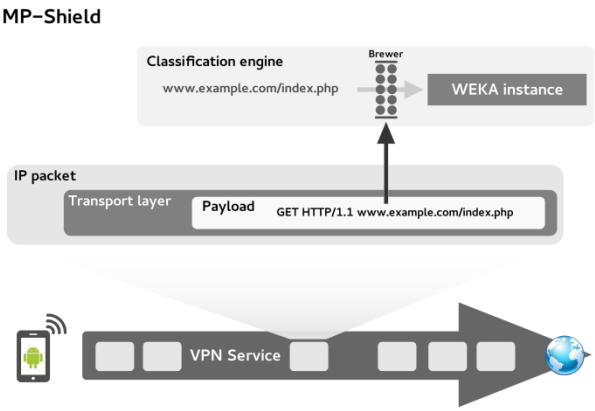

Fig. 4. WEKA's instance creation

Attr 1	Attr 2	Attr 3	Attr 4	…	Attr N	Class
0	1	-	1.56	…	0	Phishing

Fig 5. Example of WEKA instance. Each column corresponds to an attribute and contains the computed value. The Class column contain the classification label computed by WEKA, i.e. phishing or non-phishing.

```
@attribute domain-asn-size numeric
@attribute domain-create-date numeric
@attribute domain-exists numeric
@attribute domain-expire-date numeric
@attribute domain-last-update numeric
@attribute encripted numeric
@attribute has-authority numeric
@attribute has-query numeric
@attribute host-digit-count numeric
@attribute host-hyphen-count numeric
@attribute host-ip-based numeric
@attribute host-length numeric
@attribute host-token-count numeric
@attribute host-underscore-count numeric
@attribute pagerank numeric
@attribute path-at-count numeric
@attribute path-colon-count numeric
@attribute path-digit-count numeric
@attribute path-equals-count numeric
@attribute path-hyphen-count numeric
@attribute path-length numeric
@attribute path-semicolon-count numeric
@attribute path-token-count numeric
@attribute path-underscore-count numeric
@attribute port numeric
@attribute query-at-count numeric
@attribute query-colon-count numeric
@attribute query-hyphen-count numeric
@attribute query-params-count numeric
@attribute query-semicolon-count numeric
@attribute query-underscore-count numeric
@attribute ratio-consonant-url numeric
@attribute ratio-digit-host numeric
@attribute ratio-digit-url numeric
@attribute ratio-host-path-length numeric
@attribute ratio-host-url-length numeric
@attribute subhost-ip-based numeric
@attribute subhost-token-count numeric
@attribute unesc-ampersand-count numeric
@attribute url-length numeric
@attribute exit-link-count numeric
@attribute form-count numeric
@attribute form-get-method numeric
@attribute form-post-method numeric
@attribute fragment-link-count numeric
@attribute inner-link-count numeric
@attribute mean-link-distance numeric
@attribute path-cross-semantic numeric
@attribute redirection-count numeric
@attribute stylesheet-count numeric
@attribute stylesheet-link-distance numeric
@attribute user-agent-change numeric
```

Fig 6. Extract of the .arff file showing the whole set of attributes.

IV. EXPERIMENTAL EVALUATION

Although MP-Shield is developed to intercept both web browsing sessions and background applications traffic we evaluated the performance of our framework in the web surfing case study, considering a user actively browsing web pages on an Android device. We collected a set of realistic web resources' URLs that users may refer to, and such dataset has been used to build the machine learning model used in MP-Shield. Based on WEKA, our classification engine can leverage different machine learning methods (e.g. supervised classification, clustering etc.) and algorithms with different features and capabilities. To narrow down the number of usable algorithms, we focused our attention on the most effective and popular supervised learning implementations (Table I).

TABLE I. MACHINE LEARNING ALGORITHMS

Algorithm	Description
J48	It is a tree-based model which implements the popular C4.5 algorithm which basically reflect a divide and conquer method.
SMO	Sequential Minimal Optimization model implements the Platt's version of algorithm which trains a support vector classifier.
BayesNet	It is the implementation of a Bayesian network model which uses specific search algorithms and quality measures
IBk	The K-Nearest Neighbors (KNN) implementation algorithm.
SGD	Stochastic Gradient Descend is a optimization algorithm to minimize an objective function expressed as sum of differentiable "loss" functions.

The aim of our experimental evaluation is to measure algorithms' performance and to benchmark them to find out the ones that best fit to the case study on the basis of the following metrics:

- True Positive rate (TP_{rate}), which represents the percentage of instances correctly classified as a given class, with respect to actual total of that class. Intuitively it measures the model's sensitivity.
- False Positive rate (FP_{rate}), which represents the percentage of instances improperly predicted as a given class, with respect to the total of that class.
- Accuracy (A), which measures the overall percentage of whole correct predictions.
- Precision (P), which measures the ratio of total number of class' samples.

Let us now define: TP the number of non-phishing instances correctly predicted; FP the number of non-phishing samples incorrectly labeled as phishing; TN the number of phishing instances correctly classified; and FN the number of phishing samples incorrectly predicted as non-phishing. The above metrics are computed as follow:

$$\mathrm{TP}_{\mathrm{rate}} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$$

$$\mathrm{FP}_{\mathrm{rate}} = \mathrm{FP}/(\mathrm{TP} + \mathrm{FP})$$

$$A = (\mathrm{TP} + \mathrm{TN})/((\mathrm{TP} + \mathrm{FP}) + (\mathrm{TN} + \mathrm{FN}))$$

$$P = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$$

TABLE II. MEANING OF TRUE POSITIVE AND FALSE POSITIVE

		Classification	
		Phishing	**Non-phishing**
Real Label	**Phishing**	*TP*	*FN*
	Non-phishing	*FP*	*TN*

Since our goal is to maximize the classification accuracy and the TP_{rate}, we do not consider as a metric the classification algorithm execution time. This choice is motivated by the fact that model's compilation is done once and outside the device, thus not affecting classification performance at runtime.

The selected metrics give an insight on the overall performance of each classification model both on precision and accuracy perspective. While the former assesses the capability to properly recognize all the possible instances, model's accuracy is the ability to correctly classify new instances accordingly to their attributes. We specifically focused on FP_{rate} metric to evaluate model's precision. High values of FP_{rate} reflect a scarce proficiency to model the reality. In order to build a good model, our primary goal is to maximize model accuracy, namely how it suits the reality, and hence to minimize FP_{rate}. On the one hand false negative instances are dangerous as they might compromise whole application's effectiveness, though they might not produce a *direct* negative user experience. On the other hand, false positive samples affects much more in depth the user experience. In fact, warnings are perceived by users as bothersome, as they abruptly interrupt current user's activity.

A. Dataset and Model Training

Each of the algorithms in Tab. I has been subject to a training procedure phase and a performance evaluation phase using a dataset obtained mixing a known verified phishing URLs set from *PhishTank* with a set of legal website, so-called white sites, taken from the DMOZ directory. The final dataset version encompasses almost 86 thousand instances, with 53 attributes (see Fig. 6) each, where 40% are related to phishing URLs. All attributes are designed to be numeric, so that can be easily processed and used for quantitative comparison.

For each algorithm the training procedure is performed directly on the WEKA framework, providing the selected dataset as input. In order to have a baseline performance evaluation trend, each algorithm has been tested with the whole set of attributes and without any preprocessing. Once the model has been built, a test phase is executed on a different input dataset, which contains known labeled samples, so as to evaluate actual model's ability to represent the reality. WEKA's framework provide an evaluator module which automatically performs test procedure providing a confusion matrix which specifies the number of correctly and incorrectly predicted instances by the model against the test dataset (see Table II). Therefore the output confusion matrix has been used to computes the selected metrics, in order to assess punctual algorithms performance.

To optimize the training stage of the learning model, it was necessary to create a large, uniform and possibly unbiased dataset, representing all the available classes (e.g. phishing and non-phishing). Therefore, we created a naive training dataset version mixing up a known verified phishing URLs list from *PhishTank* with a set of legal website, so-called white sites, taken from the DMOZ directory. This very first attempt results in a quite good model showing an average *TPrate* of about 80%, among all the selected algorithms, though it was strongly biased, as a consequence of the structure of URLs provided. Phishing URLs are commonly characterized by longer strings and file paths after the domain. On the one hand this represents a connotative characteristic for the machine learner. Though, the good ones chosen were too "simple" and they were not connotative enough of the vast amount of URLs types that can be found throughout the web. Specifically, the legitimate portion of the URL dataset contains a majority of plain base domain identification, with at most one nesting level in the file path. As a consequence, this likely turned into a biased machine learning model which was not able to correctly classify complex legal URLs, out of the dataset scope. To overcome this issue, we have implemented a crawling system which mimics Internet surfing made by users. This spider bot is in charge of surfing web pages, starting from a predefined pool of URLs, and randomly diving into links, tracing them down. In this way, it was possible to rely on a more realistic set of URLs (link encountered in real conditions). The dataset obtained contains a more uniform distribution of web resources, with respect to the previous version, which takes into account also website sub-pages, single object resources and fragment links.

B. Algorithms performances

Table III summarizes performances achieved by the adopted algorithms, all tested with their default parameter configuration. Figure 7 plots the *TPrate*, *FPrate* and Precision metrics. The metrics are evaluated as weighted average with respect to the actual number of instances for each of the two classes, *phishing* and *non-phishing*. As it is possible to note, J48 has been chosen as the adopted model since the one who shows the best values both of accuracy and precision, nearly to 90%. The algorithm ensures a good capability to catch reality. Unsurprisingly, it also shows the minimum rate of false positive which is one of the main goal in order to minimize bad user experiences, such as several useless and unwonted notifications.

V. CONCLUSIONG REMARKS

Even though inspection tasks were carried out directly by the browser or its plugins, the framework presented in this paper provides a much wider protection. Indeed, MP-Shield can intercept all the outbound traffic of the device, ensuring an increased level of security even with unsecured browsers, as it happens, for instance, running an app-integrated HTML viewer developed with the Android WebView API, without phishing protection capabilities.

TABLE III. PERFORMANCES OF TRAINING ALGORITHMS

Algorithm	*TPrate*	*FPrate*	*P*	A
J48	0.892	0.144	0.891	89.2%
BayesNet	0.781	0.304	0.777	78%
SMO	0.782	0.326	0.783	78.1%
SGD	0.795	0.296	0.794	79.6%
IBk	0.385	0.385	0.128	35%

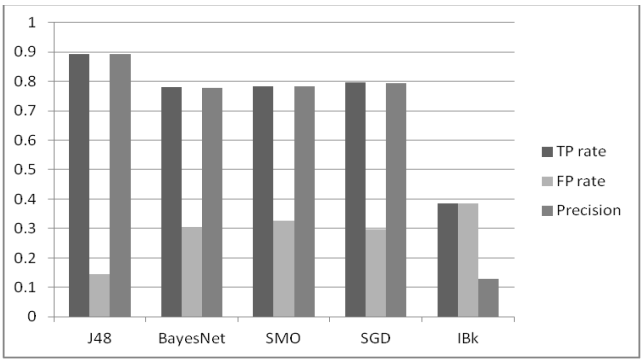

Fig. 7. Comparison of algorithms performance

Furthermore, the approach that we followed, based on modular developments and multiplicity of detection techniques, provides a higher protection level than using a single vertical approach. In fact, as pointed out by the test results, the use of heuristics can improve the level of effectiveness of the tool in the case of the blacklist check fails (False Negative). A wider range of heuristics, with the help of machine learning algorithms, can lead, of course, to a further increase of protection.

Finally, the MP-Shield was designed to provide a good level of protection without disturbing users in their normal device usage and the proxy-based approach we propose can be easily reproduced even on desktop environment, preserving the same benefits.

VI. FUTURE WORKS

The actual architecture implements a pre-compiled model logic, where the classifier is built outside the mobile device. In this way, we achieve a good performance tradeoff since the device is not overloaded by any other task than the classification. The basic assumption is that the model does not require to be constantly updated, neglecting, however, the incrementally-refining capabilities of the learning scheme. In other words the model, once built, is not able to gather new knowledge from the current classifications.

A future version of the MP-Shield framework will provide to user the possibility to interactively indicate how a selected URL has to be considered. In such a way it is possible to implement meta-learning algorithms, able to update at runtime their knowledge. The aim is to form a collaborative community whose knowledge can be shared to progressively refine application's accuracy relying on active users' answers and implementing a data exchange scheme within the community.

Moreover, due to the high level of modularity highlighted by the framework's architecture, other heuristics and black/white lists can be easily developed and added to the ones already implemented. Specifically a further extension to this framework could be the implementation of a multi-blacklist check in order to improve the reliability of detection.

Finally, a future implementation of MP-Shield will provide also semantic heuristics. In particular, the *HTMLBrewer* will be conceived to perform also semantic analysis on the fetched page, looking for semantic inconsistencies, like the employment of static image links rather than text, which might conceal a mismatching between the target URL with respect to the visible use.

ACKNOWLEDGMENT

This research is funded by the SYstem for Prevention and Combat Identity Theft (SYPCIT) project. Contract number HOME/2013/ISEC/FINEC/4000005234.

REFERENCES

[1] Anti Phishing Working Group (APWG), http://www.antiphishing.org/.

[2] PhishThank, What is Phishing, http://www.phishtank.com/what_is_phishing.php

[3] C. Whittaker, B. Ryner, and M. Nazif, “Large-scale automatic classification of phishing pages,” in NDSS ’10, 2010.

[4] Mahmoud Khonji, Youssef Iraqi, Andrew Jones, “Phishing Detection: A Literature Survey”, in IEEE COMMUNICATIONS SURVEYS & TUTORIALS, Vol. 15, No. 4, Fourth Quarter 2013, pp. 2091-2121.

[5] APWG Report, “Phishing Activity Trends Report”, 4th Quarter 2014.

[6] Ezer Osei Yeboah-Boateng, Priscilla Mateko Amanor, “Phishing, SMiShing & Vishing: An Assessment of Threats against Mobile Devices” in Journal of Emerging Trends in Computing and Information Sciences, Vol. 5, No. 4, April 2014, pp. 297-307.

[7] Symantec Report, “Internet Security Threat Report”, Volume 20, april 2015.

[8] Cik Feresa Mohd Foozy, Rabiah Ahmad, Mohd Faizal Abdollah, “Phishing Detection Taxonomy for Mobile Device”, in International Journal of Computer Science Issues (IJCSI), Vol. 10, Issue 1, No 3, January 2013, pp. 338-344.

[9] Adrienne Porter Felt, David Wagner, “Phishing on Mobile Devices”, in Web 2.0 Security and Privacy Workshop (W2SP), May 26 2011, Oakland, CA.

[10] Jie Hou, Qi Yang, “Defense Against Mobile Phishing Attack”, in EECS 588 research seminar, University of Michigan, Winter 2012.

[11] Longfei Wu, Xiaojiang Du, and Jie Wu, “MobiFish: A Lightweight Anti-Phishing Scheme for Mobile Phones”, in 23rd International Conference on Computer Communication and Networks (ICCCN), 4-7 Aug. 2014, Shanghai.

[12] Zhi Xu, Sencun Zhu, “Abusing Notification Services on Smartphones for Phishing and Spamming”, in 6th USENIX Workshop on Offensive Technologies (WOOT12), August 6–7, 2012, Bellevue, WA.

[13] M.Archana, P.M.Durai Raj Vincent , Naveen Kumar Boggavarapu, “Architecture for the Detection of phishing in Mobile Internet”, in International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 2 (3) , 2011, pp. 1297-1299.

[14] Mehedee Zaman, Tazrian Siddiqui, Mohammad Rakib Amin, Md. Shohrab Hossain, “Malware Detection in Android by Network Traffic Analysis”, in International Conference on Networking Systems and Security (NSysS), 5-7 Jan. 2015, Dhaka, Bangladesh.

[15] Anshul Arora, Shree Garg, Sateesh K Peddoju, “Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices”, in 8th International Conference on Next Generation Mobile Applications Services and Technologies (NGMAST 2014), 10-12 Spet. 2014, Oxford, UK.

[16] Dai-Fei Guo, Ai-Fen Sui, Yi-Jie Shi, Jian-Jun Hu, Guan-Zhou Lin, Tao Guo, “Behavior Classification based Self-learning Mobile Malware Detection”, in JOURNAL OF COMPUTERS, VOL. 9, NO. 4, APRIL 2014, pp. 851-858.

[17] Abimael Carrasquillo, Albert E. Maldonado, Eric Santos, José Ortiz-Ubarri, “Poster: Towards a framework for Network-based Malware detection System”, in 35th IEEE Symposium on Security and Privacy, May 18-21, 2014 at The Fairmont, San Jose, CA.

[18] L. Chekina, D. Mimran, L. Rokach, Y. Elovici, B. Shapira, “Detection of Deviations in Mobile Applications Network Behavior”, in Annual Computer Security Applications Conference, 2012, Orlando, FL.