On the Intrusiveness of JavaScript on the Web

Muhammad Ikram
NICTA and UNSW, Australia
muhammad.ikram@nicta.com.au

Hassan Asghar
NICTA, Australia
hassan.asghar@nicta.com.au

Mohamed Ali Kaafar
NICTA, Australia
dali.kaafar@nicta.com.au

Anirban Mahanti
NICTA, Australia
anirban.mahanti@nicta.com.au

ABSTRACT

Various web components and JavaScripts have been used for collecting personal identifiable information resulting in privacy concerns. Although several privacy preserving tools have been proposed to limit online advertising and tracking their use has been limited and mostly limited to tech-savvy audience. In addition to poor and manual filtering-list maintenance and confusing settings, these privacy preserving tools have, arguably, usability and intrusiveness issues. Among others, their brute-force blockage of all JavaScripts on a website, may result in broken functionalities thus effecting user's web-experience. In this work, we propose a framework to quantify the intrusiveness of JavaScripts with ultimate objective of measuring the usability of privacy preserving tools. We postulate that intrusive JavaScripts carry distinct characteristics that could be used to differentiate them from functional JavaScripts i.e., scripts that are genuinely used for enhancing the user's web experience. We propose a measurement methodology that can automatically separate tracking and privacy intrusive JavaScripts from the functional JavaScripts. Our methodology assumes only partial knowledge of the privacy intrusive JavaScripts.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

JavaScript; privacy; web-tracking; security.

1. INTRODUCTION

Online behavioral advertisement (OBA) is the practice of tailoring advertisements based on user's history and behavior. For this purpose, online advertisers either use customized or third party systems to glean personal identifiable information on the web. In addition to other web-components, trackers and online advertisers use JavaScripts to achieve their tracking or advertising goals. Many web users perceive online tracking intrusive and instances of advertisements annoying [3].

In order to avoid annoying advertisements and privacy intrusive tracking, tools such as NoScripts, Ghostry, Adblock Plus, etc., have been proposed. In this paper we refer to these tools as *privacy preserving tools* (PPTools). PPTools are either based on pre-defined filters or they block all JavaScript executions to limit tracking and online advertising. The brute-force way of blocking JavaScripts by these tools on a website results in broken functionalities thus effecting user's web-experience. We postulate that JavaScripts used for tracking and online advertisements have distinct characteristics that can be used to differentiate them from the functional JavaScripts. In this study we aim to build a framework to quantify intrusiveness of JavaScripts. In order to achieve this goal, we propose a measurement system can to automatically separate tracking and privacy intrusive JavaScripts from the functional JavaScripts. Our methodology assumes only partial knowledge of the privacy intrusive JavaScripts.

We present an overview of our system by discussing data collection and feature extraction methodology in (§ 2). In order to measure the intrusiveness of JavaScripts, we formulate a classifier based only on partial knowledge about tracking JavaScripts, in (§ 3). Our goal is to develop a classifier to extract, based on some quantitative similarity function, tracking JavaScripts from unlabeled JavaScripts data set. We discuss evaluation results in (§ 4).

Once applied, our system results in a quantitative measure of JavaScript similarity which we plan to employ to investigate the usability of PPTools in the future. In contrast to the start-of-the art research, our automated system provides multiple advantages such as flagging tracking JavaScripts based on quantitative measures and resiliency against URL or JavaScript code obfuscations.

2. SYSTEM OVERVIEW

We refer to JavaScripts that enable privacy intrusive tracking or on-line advertisements as *tracking JavaScripts* and the ones which enable users' interactiveness, access on-demand content, improve websites' responsiveness and user friendliness, and allowing genuine user-website operations, as *functional JavaScripts*. The key rationale of the classification methodology we would like to build, is that tracking and functional JavaScripts may have distinct expressions, structures, and program semantics. Moreover, we postulate that functional JavaScripts are similar to one another while being distinct from tracking JavaScripts and vice versa.

Permission to make digital or hard copies of all or part of this work for persona or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CoNEXT Student Workshop'14, December 2, 2014, Sydney, Australia.
Copyright © 2014 ACM 978-1-4503-3282-8/14/12…$15.00.
http://dx.doi.org/10.1145/2680821.2680837.

Given a webpage, our system aims to extract all JavaScripts and catagorise them into either functional or tracking scripts. By running our system on "regular" webpages and then computing the fraction of functional/tracking components while applying PPTools, we can get hints about the usability about the intrusiveness of PPTools (this is left as future work). For a comprehensive analysis, a scalable method to support a large number of webpages is of primary importance. This motivates the use of an automatic classifier. In this paper we focus on measuring the intrusiveness of JavaScripts and classification of JavaScripts into functional and tracking JavaScripts. We first discuss our methodology to collect JavaScripts followed by feature selection, below.

Data Collection: We use Alexa's top listed websites to create a corpus, C, of JavaScripts. After retrieving the document object model (DOM) trees of the landing pages from the top websites, we collect on-site and externally linked JavaScripts by parsing <script src = " " type=" "> and <script type=" "> HTML tags, respectively. Our final corpus consists of several thousand of unique JavaScripts.

Feature Selection: In order to predict similarities, we need a data structure, say X, that captures all of the distinct information about JavaScripts. We call an element of such data structure a $feature$. To derive features, one approach is to evaluate the importance of every term, t, in each JavaScript $\Im \in C$. Here a term is a string of characters belonging to a line of JavaScript code. We then use the boolean term frequency (tf) measure such that $tf(t, \Im) = 1$ if $t \in \Im$ and $t(f, \Im) = 0$, otherwise. Finally, we obtain tf with inverse document frequency (idf) for $t \in \Im$ over the corpus C, as:

$$tf - idf(t, \Im, C) = tf(t, \Im) \cdot \log \frac{|C|}{|\Im \in C : t \in \Im|}$$

So, $tf - idf$ transforms our corpus C to a new data structure, also called feature vector space, X where feature vector x_i corresponds to JavaScript $\Im_i$. We measure the intrusiveness of a JavaScript $\Im_i$, by applying some metrics and similarity functions on feature vector x_i. We explain this in the following section.

3. MEASURING INTRUSIVENESS OF JAVASCRIPTS

In traditional machine learning, a parameterized function f, also called a model, is trained using training data X and y to predict $y' = f(X)$. The parameters of f are usually chosen such that some measure of difference between y and y' is minimized. Based on tracking domain knowledge (e.g., EasyList[1]), we can label a small number of tracking JavaScripts to train f which we call the set of positive examples, P. The set of unlabeled examples, U, might have positive or negatives (functional JavaScripts). Let $S \subseteq X \times L = \{(x_1, s_1), (x_2, s_2), ..., (x_m, s_m)\}$ be the training corpus of size m where $L = \{l, u\}$. Any input sample x_i for which $s_i = l$ belongs to P and the one with $s = u$ belongs to U. In this framework, each input sample x_i is also related to a (possibly unknown) label $y_i \in \{+1, -1\}$, such that if $s_i = l$ then $y_i = +1$ and if $s_i = u$ the y_i can be either positive or negative [2].

The scheme proposed by Elkan and Noto [2] allows us to adapt our problem and parameterize f to any supervised learning algorithm that outputs probabilistic classifiers. Once trained, f provides an estimation of the probability that an example has a certain label [2]:

$$p(y = +1|x) = \frac{p(s = l|x)}{c} \quad (1)$$

where $p(s = l|x)$ is the probability that an input example x is positive and $c = p(s = l|y = +1)$ is the constant probability that a positive example is labeled.

To validate $f(X) : X \to [0, 1]$ we consider a validation V which is drawn from $X \times L$. We let all positive labeled x belong to the set $P \subseteq V \subseteq X$, and estimate the fraction of positive inputs, c, predicted by f:

$$c \approx \frac{\sum_{w \in P} f(x)}{|P|} \quad (2)$$

Finally, we use the estimation probability (Equation 1) to parameterize f with three different machine learning algorithms.

Table 1: Measures performance, F1-score, BiasedSVM, PU Learning, Traditional SVM (TSVM)

|P|	|U|	BiasedSVM	PU Learning	TSVM
10	170	0.491	0.535	0.613
20	170	0.565	0.591	0.676
50	170	0.631	0.662	0.757
85	170	0.695	0.723	0.873

4. EVALUATING RESULTS

To measure the intrusiveness of JavaScripts we apply positive only and unlabeled (PU) learning [2] and BiasedSVM [4] on our test data set. We apply identical experimental setting, as discussed by Elkan et al. [2]. As we do not have full domain knowledge to suggest the fraction of P from U therefore we used traditional soft margin SVM (TSVM) as baseline by manually input training examples. We use radial basis kernel settings with default threshold values (as discussed in [2]). In addition to performance evaluation of classifiers, we evaluate the effects of size of P on overall classification.

We use F1-score $= \frac{2.p.r}{p+r}$[4], where p is precision and r is recall, to compare the performance of classifiers. Table 1 shows the macro-averaged F1-score of $|U|$=170 unlabeled JavaScripts (collected from top 10 websites) for each $|P|$ setting. We observe that TSVM is able to tolerate some noise. While running probability estimation steps in both BiasedSVM and PU learning actually makes the results worse. Compared with traditional SVM, both PU learning and BiasedSVM observe overfitting and is a window for improvement of our future work. Our initial results suggest that PU learning outperforms BiasedSVM in predicting privacy intrusive JavaScripts even if the $|P| \ll |U|$.

5. CONCLUSION AND FUTURE WORK

We have presented a framework to quantitatively evaluate the intrusiveness of JavaScripts. We plan to improve feature selection process to capture program semantics and context of specific piece of code by resorting to program dependency graphs. Building on our current study, we are in the process to improve our understanding about statistical nature of our data set to adopt our scheme for quantitative evaluation of the usability of PPTools.

6. REFERENCES

[1] EasyList. https://easylist-downloads.adblockplus.org/easylist.txt.

[2] Elkan, C., and Noto, K. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Las Vegas, Nevada, USA, 24th August, 2008), KDD '08, ACM, pp. 213–220.

[3] Krishnamurthy, B., Naryshkin, K., and Wills, C. Privacy leakage vs. protection measures: the growing disconnect. In *IEEE Workshop on Web 2.0 Security and Privacy* (Oakland, California, USA, 26th May, 2011), W2SP '11, IEEE, pp. 1–10.

[4] Liu, B., Dai, Y., Li, X., Lee, W. S., and Yu, P. S. Building text classifiers using positive and unlabeled examples. In *Proceedings of the Third IEEE International Conference on Data Mining* (Melbourne, Florida, USA, 19th Nov, 2003), ICDM '03, IEEE Computer Society, pp. 1–8.