The Journal of Systems and Software 137 (2018) 67–77

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

New deep learning method to detect code injection attacks on hybrid applications

Ruibo Yan[a], Xi Xiao[a], Guangwu Hu[b,*], Sancheng Peng[c], Yong Jiang[a]

[a] Graduate School at Shenzhen, Tsinghua University, Shenzhen, China
[b] School of Computer Science, Shenzhen Institute of Information Technology, Shenzhen, China
[c] School of Informatics, Guangdong University of Foreign Studies, Guangzhou, China

ARTICLE INFO

Article history:
Received 25 November 2016
Revised 8 September 2017
Accepted 3 November 2017
Available online 14 November 2017

Keywords:
Code injection
Hybrid application
Abstract syntax tree
Deep learning

ABSTRACT

Mobile phones are becoming increasingly pervasive. Among them, HTML5-based hybrid applications are more and more popular because of their portability on different systems. However these applications suffer from code injection attacks. In this paper, we construct a novel deep learning network, Hybrid Deep Learning Network (HDLN), and use it to detect these attacks. At first, based on our previous work, we extract more features from Abstract Syntax Tree (AST) of JavaScript and employ three methods to select key features. Then we get the feature vectors and train HDLN to distinguish vulnerable applications from normal ones. Finally thorough experiments are done to validate our methods. The results show our detection approach with HDLN achieves 97.55% in *accuracy* and 97.60% in *AUC*, which outperforms those with other traditional classifiers and gets higher average precision than other detection methods.

1. Introduction

Mobile applications (apps) are gaining significant popularity as mobile devices are getting more and more popular. The smart phone is not only a communication tool employed by users and business, but also a means of planning and organizing people's work and private life. More and more researchers focus on mobile apps Bagheri et al. (2016). Therefore, mobile security has become increasingly important in mobile computing Xiao et al. (2012). Among all the mobile operating systems, Android is open-source and has accounted for around 85 percent of all smartphone sales worldwide in the beginning of 2016 (Global mobile os market share in sales to end users from 1st quarter 2009 to 1st quarter 2016, 2017). Android is so popular, so in this paper, we mainly focus on Android, but the idea can also be applied to other mobile operating systems.

A hybrid application is one that combines elements of both native and HTML5 applications. There are three main mobile operating systems, Android, iOS and Windows Phone. Traditionally, in order to develop mobile apps running on different systems, developers have to use different programming languages and Application Program Interfaces (APIs), which costs much effort. While using a hybrid application framework which supports cross-platform mobile app development, developers can easily build hybrid apps for different mobile operating systems with HTML, CSS and JavaScript. Hybrid HTML5 application development gains more and more momentum in mobile industry because of its convenience in developing cross-platform apps. There are many hybrid application frameworks, such as Phonegap (2017), which utilize functions of web (2017). It can display web pages and execute JavaScript functions. Since JavaScript code and data can be mixed together in web technology, once the malicious JavaScript code is injected and executed by JavaScript engine through some channels, the code injection attack (Jin et al., 2014b) happens in the hybrid apps. In this paper, our purpose is to judge whether a hybrid application has code injection vulnerabilities. The idea can be applied in mobile application markets, such as Google Play market. If a hybrid application has code injection vulnerabilities, we can tell the enterprise who owns the application to avoid financial loss and user privacy leaking.

The code injection attack also happens in PC web applications, which is called "Cross Site Scripting" (XSS). The code injection attack in hybrid apps inherits the fundamental cause of XSS, but it uses many more channels to inject malicious code than XSS. These channels are unique in mobile devices, including Contact, SMS, Barcode, MP3 metadata, etc. The code injection attack on HTML5-based apps is firstly proposed in Jin et al. (2014b) and explained in depth in Jin et al. (2014a). The work in Jin et al. (2014a) adopted static data-flow analysis to detect these attacks. It got the precision

* Corresponding author.
E-mail addresses: yrb15@mails.tsinghua.edu.cn (R. Yan), xiaox@sz.tsinghua.edu.cn (X. Xiao), hu.guangwu@sz.tsinghua.edu.cn, huguangwu@gmail.com (G. Hu), psc346@aliyun.com (S. Peng), jiangy@sz.tsinghua.edu.cn (Y. Jiang).

https://doi.org/10.1016/j.jss.2017.11.001
0164-1212/

of 97.7%. But it costs much time. We proposed classification methods of machine learning to judge whether an application is vulnerable Xiao et al. (2015). Our previous method reduced the time but only obtained the *precision* of 95.3%.

In this paper, we employ a novel deep learning network, Hybrid Deep Learning Network (HDLN) to detect these attacks. At first, based on our previous work, we extract more features from Abstract Syntax Tree of JavaScript and use three methods to select key features. Then on the basis of feature vectors, HDLN is trained to distinguish vulnerable applications from normal ones by thorough experiments. The results show our detection approach with HDLN achieves the overall *accuracy* of 98.12%, superior to those with other traditional classifiers and gets higher average *precision* than the other detection methods (Jin et al., 2014a; Xiao et al., 2015).

Our contributions primarily lie in the following aspects. First of all, we develop a novel deep learning network, Hybrid Deep Learning Network (HDLN), and use it to detect the code injection vulnerability in hybrid applications. Our method improves both the *precision* and *accuracy* and performs far more better than the method in Xiao et al. (2015). Second, we extract new features from the AST of JavaScript in a hybrid application and use information gain, Chi-square test and document frequency to select key features. Third, we extend our dataset and conduct comprehensive experiments with different feature selection methods and different network structures and the results show the extended features and the feature selection methods can improve the detection performance.

Compared to our previous work (Xiao et al., 2015), at first we add n-grams generated from the AST of JavaScript as new features and use three methods to choose key features. Furthermore, a novel deep learning network, Hybrid Deep Learning Network (HDLN) is used to do the detection. In addition, thorough experiments are conducted to test our new network in this work.

The rest of the paper is organized as follows. In Section 2, we describe the background of code injection attacks. The detection model including HDLN is explained in detail in Section 3. In Section 4, we discuss the results of different experiments and provide an overview of related work in Section 5. At last, we conclude this paper and suggest future work in Section 6.

2. Background

It is well known that there is a dangerous characteristic in the web technology: it allows data and code to be mixed together. So hybrid apps developed by web technologies may have code injection vulnerabilities. In this section, we first describe technologies related to the code injection attack. Then we introduce Abstract Syntax Tree from which we extract features. Finally, we state deep learning neural networks.

2.1. Code injection in hybrid applications

There are three types of mobile applications: Native apps, HTML5 apps and Hybrid apps. Native apps are specific to a given mobile operating system using the specific programming language and Software Development Kit (SDK) that the respective platform supports. HTML5 apps are developed with standard web technologies-HTML5, JavaScript and CSS. This write-once-run-anywhere approach creates cross-platform mobile apps that run on multiple platforms. A hybrid application is one that combines elements of both native and HTML5 applications. Hybrid apps embed HTML5 apps inside a thin native container, combining the best of both the native and HTML apps. There are many frameworks that allow developers to create hybrid apps easily. Among them, Phonegap (2017) is the most popular one which uses the functions of WebView web (2017). There is an API addJavaScriptInterface() in WebView to allow the JavaScript code to call the native Java code. If an application has declared the required permissions, the app can access the system resources by calling the responding native code. PhoneGap does this work for developers, it provides many plugins in JavaScript language for developers to access system resources.

Since it allows data and JavaScript code to be mixed together in the web technology, code injection attacks can happen in hybrid apps. For example, when a user uses a hybrid application to scan a barcode and the content of the barcode is a piece of malicious JavaScript code to constantly send the user's locations to a specific server, the malicious code is injected and executed by JavaScript engine, and the user's location privacy is stolen by attackers. A hybrid application with code injection vulnerabilities should satisfy two conditions: firstly, the hybrid application reads unsafe data from outside or inside the device, such as scanning barcodes, Contacts and so on; secondly, the hybrid application shows the unsafe data with unsafe JavaScript APIs, such as "document.write()".

There are many channels for code injection attacks on hybrid apps. Besides the web channel, there are some data channels unique to mobile devices, such as barcodes, RFID tags and SMS messages, metadata of multimedia files and ID channels. A more hidden code injection attack based on JavaScript coding is proposed in Xiao et al. (2015), the conclusion is that JavaScript Unicode coding, JavaScript hexadecimal coding and JavaScript octal coding are suitable for code injection.

2.2. Abstract syntax tree

In computer science, an Abstract Syntax Tree is a tree representation of the abstract syntactic structure of source code written in a programming language Abstract syntax tree (2017). Each node of the tree represents a construct in the source code. In other words, the AST is a structured description of the source code, which can be used to do static analysis on JavaScript code. Esprima (2017) is a high performance, standard-compliant JavaScript parser written in JavaScript to extract the AST from JavaScript code. Suppose we have the following piece of JavaScript code in List 1, let us see how to generate the AST.

We treat the above JavaScript code as a string and pass it to "esprima.parse". Subsequently, the AST in JavaScript Object Notation (JSON) format is obtained. We program to parse the JSON string and transform it into an abstract syntax tree like Fig. 1. In this tree, there are six paths from the root node to leaves. The six paths are called as *path0, path1, path2, path3, path4*, and *path5. path0* is "Program VariableDeclaration VariableDeclarator Identifier". The remained five paths are in the same format as *path0*. The height of the tree in Fig. 1 is 5 and the width is 3.

2.3. Deep learning neural networks

Artificial neural network is firstly proposed and consequently deep learning frameworks such as convolutional neural network and recurrent neural network are developed. Artifical Neural Network (ANN) model is proposed by McCulloch and Pitts (1943). Artificial neural neworks are typically organized in layers. The neurons of the input layer, hidden layers and the output layer are fully

```
1 var answer = 6 * 7;
2 function test()
3 {
4         alert("HelloWorld!");
5 }
```

Listing 1. JavaScript code.