Effective Detection and Security of Source Code Theft based on Software Birthmark

R. Karthikeyan, K.G.S. Venkatesan, R. Saravanan

Department of Computer Science, Bharath University

Chennai-600073, INDIA.

rkarthikeyan1678@gmail.com
venkatesh.kgs@gmail.com
Saravananr1190@gmail.com

Abstract—**Today Internet has become part of the life for each and every person at all the corner of the earth. Web Page has been the main object in the Internet communications. These web pages are made of several bytes of soft codes(software). There are as many open source papers available in the Internet. Software theft(plagiarism) has become a very serious threat. Java script has been a much common language for creating web pages. Apart from the previous methods available, Software Birthmark is being used to detect the code theft. Although it has been used in previous methodologies, here the method produces a result with an accuracy of 100%. We use more effective technique, Improved Graph Selector, Faster Detector. Despite of detection of threats or Hackers, we also provide security by blocking the unauthorized users from entering into the Database. And those unauthorized users can also not use the details stored in the Database.**

Key Words—**Code Theft, Software Birthmark, Plagiarism, Improved Graph Selector, Faster Detector.**

I. INTRODUCTION

"Internet" has been the most curious word, which we come over every day in our life. Today's world moves over exchanging of data through internet. Java script is one of the commonly used languages for creating web pages. Software protection continues to be an important topic for computer scientists. Intruders use many ways to detect the source code and transform the originality to their own model. They create their own Websites, just by hacking the source code from the original web pages. According to a survey from Evans Data in 2008 ***[2]***, over 66% of 3GL and scripting language use, including Java. Hiding of source code includes works only for a certain level. And it has been conquered in an efficient way by the intruders.

Protecting the code from the intruders has become a serious issue for today's researchers and web page developers. Some of the previous methods have been used so as to protect the intruders are listed-Code Obfuscation, Water Marking. "Code Obfuscation" is a methodology in which we use symbols and other variables in a calculated manner in order to replace the original characters. To say technically, Code obfuscation is a semantics-preserving transformation of the source code that makes it more difficult to understand and reverse engineer. However, it only prevents others from learning the logic of the source code but does not protect them from being copied.

While Watermarking is a technique that was well-known and one of the earliest approaches to detect software theft. In order to protect the program, a water mark is being inserted into the original source, that determines the ownership of the program. Watermarking also requires the owner to take extra action (embed the watermark into the software) prior to releasing the software. However, it is believed that "a sufficiently determined attacker will eventually be able to defeat any watermark." ***[3]***

A relatively new but less popular software theft detection technique is software birthmark. Software birthmark does not require any code being added to the software. It depends solely on the intrinsic characteristics of a program to determine the similarity between two programs. In our words, it is to say that every program has their own originality. The behaviour of the program remains same in the copied program also. Thus we can easily determine the Piracy. If once the source code is hacked from the original user page, it has to be used somewhere to create another duplicate page. And the intruder uses his name so as to maintain him as the original user of the web page. In this case, Birthmark has been the main characteristic to determine the originality and detect the intruders and withdraw his permission.

According to Wang *et al.****[4]***, a birthmark is a unique characteristic a program possesses that can be used to identify the program. To detect software theft, the birthmark of the program under protection (the plaintiff program) is first extracted. The suspected program is then searched against the birthmark. If the birthmark is found, it is highly likely that the suspected program (or part of it) is a copy of the plaintiff program.

II. Problem Statement

In software birthmark **[9]**, to help detect code theft of JavaScript programs. A defect may be a distinctive characteristic a program possesses that may be wont to establish the program. we have a tendency to extend 2 recent defect systems that extract the defect of code from the run-time heap.

A birthmark can help them to prove code theft by identifying intrinsic properties of a program. with constant nevus are probably to share a typical origin. Birth marking works above all for code that wasn't protected by tamper-resistant copyright notices that otherwise may prove possession. we tend to propose a dynamic nevus for Java that observes however a program uses objects provided by the Java commonplace API.

To detect theft of Java class files efficiently, we have to date planned an inspiration of Java birthmarks. Since the birthmarks square measure distinctive and native characteristics of each category file, a category file with a similar mar of another may be simply suspected as a duplicate. Performance and tolerance of the birthmarks against subtle attacks had not been evaluated well.. we demonstrate that the proposed birthmarks successfully distinguish non-copied files in practical Java application (97.905%).

There are two categories in software birthmarks. Static birthmarks and Dynamic birthmarks. Static birthmarks are extracted from the syntactic structure of a program. Dynamic birthmarks are extracted from the dynamic behaviour of a program at run-time.**[4]**

Methods such code obfuscation changes only the static behaviour of the program, that is the syntactic structure. While Birthmark uses the characteristic of the program, which uses to protect the Dynamic behaviour of the program. A redesigned heap graph based birthmark for JavaScript is used to make it a scalable and robust solution for detecting software theft. A heap graph is a simple directed graph in which the nodes represent the objects and the edges represent the references between them. The number of hits on the original web page is noted. The hitter's address are also being noted. Drawing the heap graph shows the result of all the hitters over the Original page. Once if the suspect is to be found, a graph Merger is used. That uses Graph Monomorphism so as to eliminate the recursive or repeated objects from the Heap Graph. **[5]**

We use above related works for use and also we work on the birthmark technology. The birthmark mainly deals with the comparison of the two set of codes. The final result of the comparison will be the detection of the duplicate one used. The higher level of security by blocking the user from using the source in any way what so ever is done here in this paper.

III.SYSTEM ARCHITECTURE

The System architecture shows the clear overview of the paper.

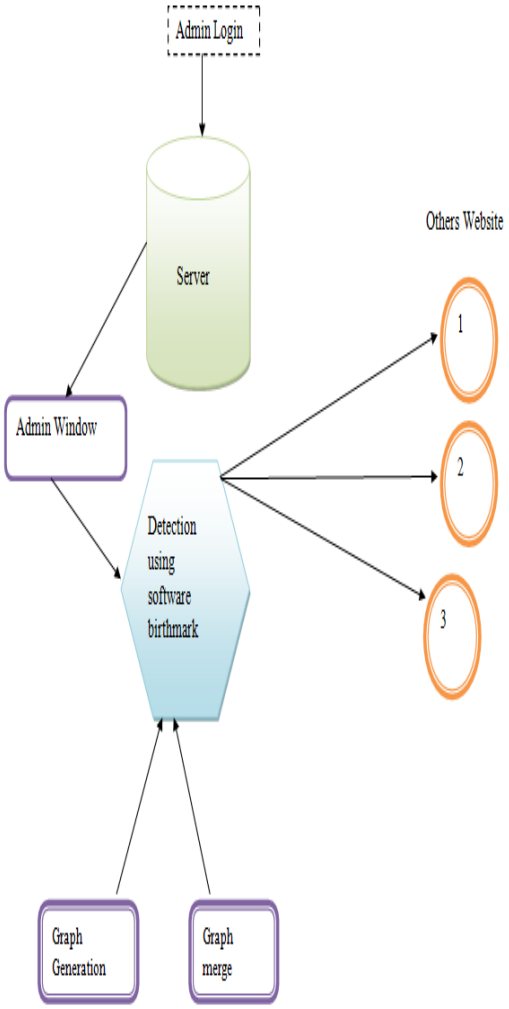

Fig. 1. System Architecture overview of a Detection using software birthmark.

The architecture consists of 5 major parts.

1. Admin Login
2. Admin Window
3. Graph Generator
4. Graph Merger
5. Faster Detector Using Software Birth Mark

Admin Login:

Every Server has their own login details so as to get entered. But there must only one authenticated Admin for every Server. If there exists more than one Admin, there may exist lots of confusion for entering the server. This might also lead to the dangerous threat of even crashing the whole server space. So, for authenticated and much safer use, we use only one Authenticated Admin. The Login window As usual has its own entry details such as User Id, Pass code. May be thee method is traditional and older for use, but gives a bit of security to server.

Admin Window:

Initially Java Script Heap Profiler was used in the referred base paper. That collects all the information of the users, hitting the original server or web page. It just collects identity of the users hitting and keeps track of the objectives they use in the Web Page. JavaScript heap keeps changing due to object creations and garbage collections. Simply to say Admin Window has the control over the Birthmarks [2]. It just manages to find the details, those are collected by the Java Script Heap Profiler as in base paper. Similar to the profiler we can just visit details of all the users hitting the server.

Graph Generator:

Java has an ability to collect all the garbage to it. While excess of garbage leads to loss of data. Which appears to be eliminated in this paper apart from the base paper. Overload is to be reduced. A simple heap graph is drawn through the data collected. A Graph Generator is used so as to generate a heap graph from the collected details. In the collected details, each and every details are being used to draw many Accumulated graphs. This graph appears to be a Larger one so we use a Graph Merger for reducing it to a single graph. And also make it to a more understandable manner.

Graph Merger:

A unique ID assigned to every object in the JavaScript heap by the V8 JavaScript engine. Moreover, the ID of an object does not change across multiple dumps and therefore, can be used to identify the object. The Graph Generator also annotates each node in the heap graph with its object ID. The graph merger takes multiple heap graphs as input and outputs a superimposition of them (one single graph) that includes all the nodes and edges appearing in the input heap graphs.

Sub-Graph Selector :

There are *n* number of graphs generated by the graph generator. Even after filtering unnecessary user's graphs, there are many number of graph. These graphs are later merged into a Single graph by the Graph Merger. The Larger graph has further sub-graphs. These are being selected one by one to check out for the comparison with the Birthmark Original Source program. The Sub-graph selector mainly does the work of selecting the individual objects from the individual windows [3]. Objects mainly refer to the individual users entering the server window. The nodes in the Heap graph are the individual objects, they are nothing but the individual users.

Faster Detector Using Software Birth Mark:

As said earlier in the Sub-Graph Selector, the detection process is being carried out in the detector. Detector which is a main tool in the Base Paper, has some disadvantage of time consumption in the method. The Birthmark of the program of the suspect is compared with the birthmark of the original source program [6]. Since each and every birthmark has their own identity, even the changed template of the suspect's program has the same birthmark of the original source program. This is the reason why this method has an accuracy of 100%. To overcome the previous paper's complaining issues, in this paper we are using a more Faster detector to reduce the time consumption. And increase the efficiency for finding the intruder or suspect's program.

III. Implementation and Technologies

In this paper, I have implemented the concept of "Software birthmark". This does not require any code being added to the software. No syntactic or semantic requirements are added. Or no modulations are made to the original source code. The above matters deal only with the static characteristics of the program. While Birthmark depends solely on the intrinsic characteristics of a program to determine the similarity between two programs. It is to prove that software birthmark is very practical and more effective in detecting software theft that even adopts advanced evasion techniques. Heap graph stores the addresses that hits the original code for every time interval. In software birthmark, to help detect code theft of JavaScript programs. A birthmark is a unique characteristic a program possesses that can be used to identify the program. We extend two recent birthmark systems that extract the birthmark of software from the run-time heap.

The modules used such as Admin Login, Admin Window, Graph generator, Graph Merger are simpler to design, also to understand and are much effective in use. The tools such as Java script, SQL are more common and can be used in common by most of the software developers and users. This makes this paper much feasible to the day today's works. It is also in an more ease of use.

Here I have used the following technologies

- Collections
- JSP
- Servlet
- Thread
- Ms SQL server 2005

JSP :

In our paper we are using JSP to design the application process. JSP pages easily combine static templates, including HTML or XML fragments, with code that generates dynamic content.JSP pages are compiled dynamically into servlets once requested, thus page authors will easily make updates to presentation code. JSP pages can also be precompiled if desired.

Servlet:

In our paper we are using servlet to control the application process. Servlets are modules that run within the server and receive and reply to the requests created by the consumer. Servlet retrieve most of the parameters mistreatment the input stream Associate in Nursing send their responses mistreatment an output stream.

Thread:

In this paper threading concept is very important. A thread could be a successive path of code execution at intervals a program. And each thread has its own local variables, program counter and lifetime. Like creation of a single thread, we can also create more than one thread [8 - 9]. Runnable to make our paper efficient and dynamic. In our paper we are using request process with the help of multi threading concepts.

MS SQL server 2005:

In this paper first we create one database for managing table that table are containing user personal information and perform the data communicate with other table and perform table related to this paper

IV. Evaluation Results and Screen Shots

After checking the practical results of the users hitting the server. We have determined a simple table as under.

TABLE I
ACCURACY OF THE RESULT

	Detection Result	Manual Result	Accuracy
No. of Hitters	12	12	100%

The above table shows the result of the number of hitters found during detection. By checking the result with the manual count of the number of hitters over the server, it produces an accuracy of 100%. This is found to be a much effective one than in the previous papers those produce only 98%.

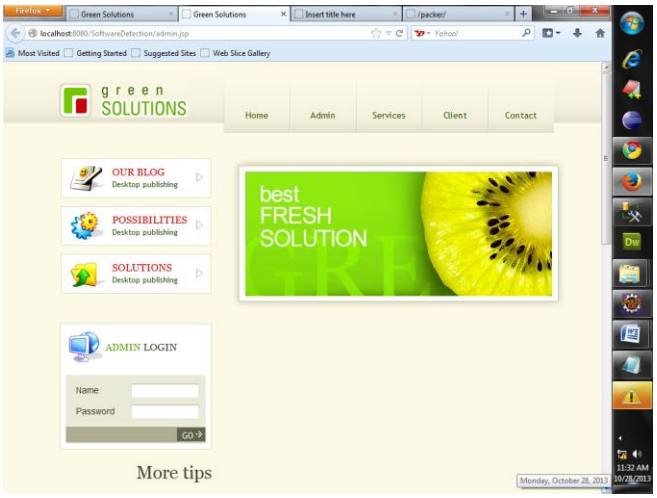

Fig. 1. Login Screen

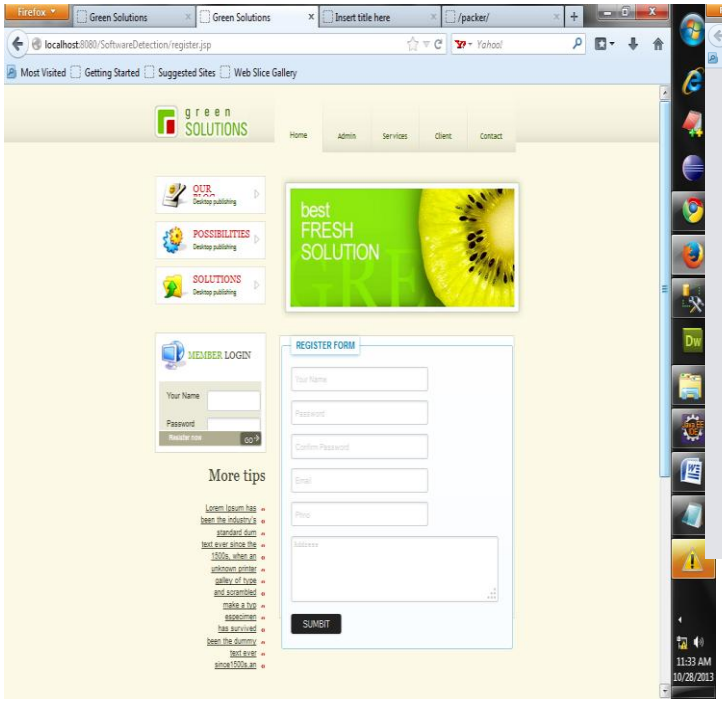

Fig . 2. Registration

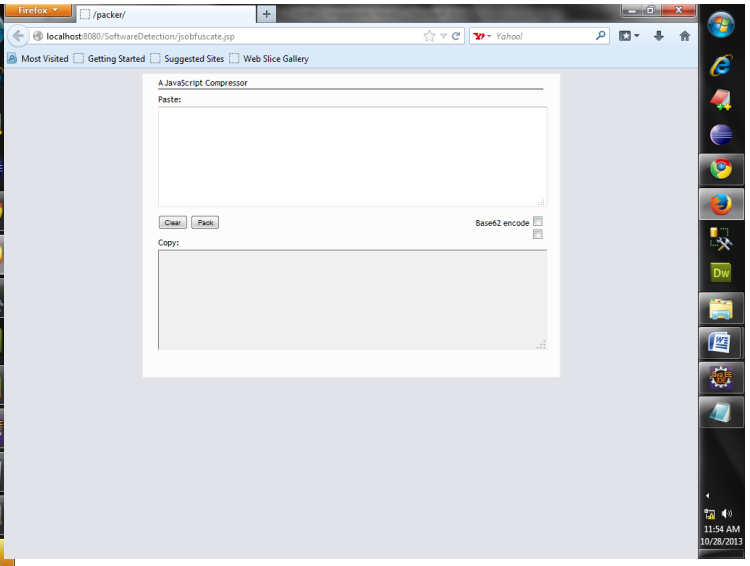

Fig. 4. Graph Generator and Filter

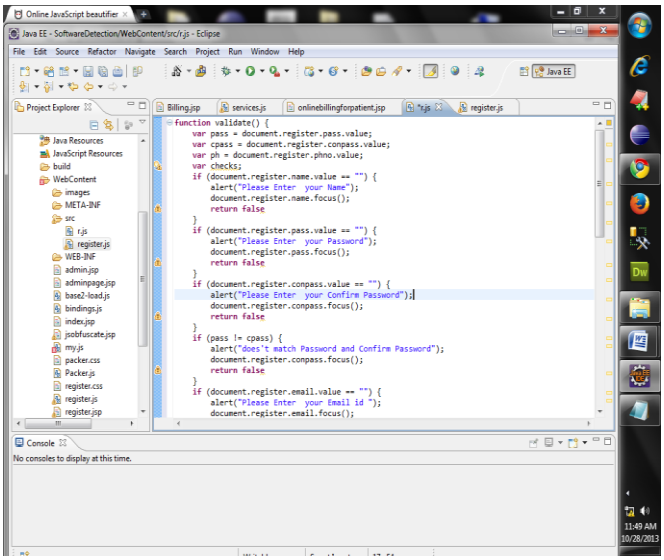

Fig. 3. JavaScript Heap Profiler

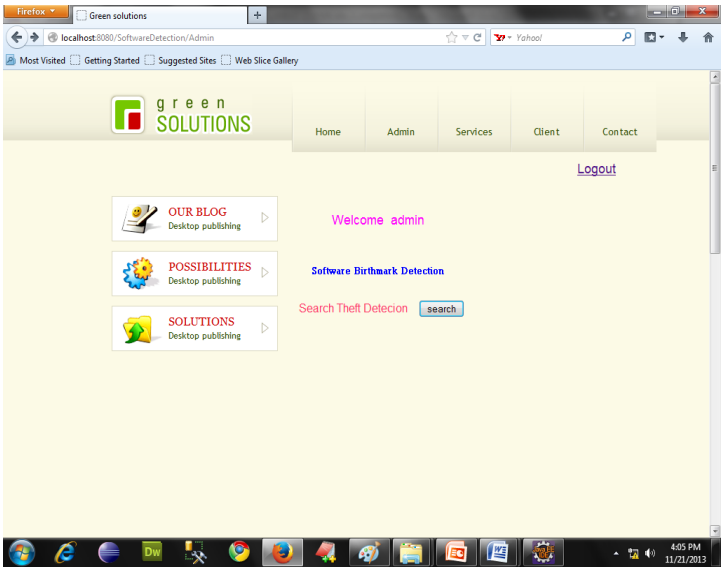

Fig. 5. Detector using Software Birthmark

V. Conclusion

The Main Goal/Objective of the paper is to provide security for the source code of the web pages. The Original developer of the code is affected in many ways by the intruders/hackers technically and also economically. Despite several methods such as Code Obfuscation and Code Watermarking, the intruders keep on breaking the security threats to breach the source code. This makes a heavy loss for the developers and Original users both mentally and financially in an effective way. The main concept used in this paper is "Birth Marking". A Birthmark can help them to prove code theft by identifying intrinsic properties of a program. While code obfuscation and watermarking deal with the static behavior of the source code. To hack in a dynamic mode, it becomes easier for the hackers. The proper system design makes effective Admin control over the birth mark rather in the previous Birth Mark related papers. As discussed earlier, some of the research papers have given a result of 98% over the years. This paper produces a result at an accuracy of 100%. As discussed in earlier in the paper, the control over the database is provided to the administrator. Using this control, the administrator can enter the database and he can restrict the permission of the user from unauthorized user.

Acknowledgment

The author would like to thank the Vice Chancellor, Dean-Engineering, Director, Secretary, Correspondent, Principal, HOD of Computer Science & Engineering, Dr. K.P. Kaliyamurthie, Bharath University, Chennai for their motivation and constant encouragement. The author would like to specially thank Dr. A. Kumaravel for his guidance and for critical review of this manuscript and for his valuable input and fruitful discussions in completing the work and the Faculty Members of Department of Computer Science & Engineering. Also, he takes privilege in extending gratitude to his parents and family members who rendered their support throughout this Research work.

References

[1] *"Heap Graph Based Software Theft Detection,"* Patrick P. F. Chan, *Student Member, IEEE,* Lucas C. K. Hui, *Senior Member, IEEE,* and S.M. Yiu, *Member, IEEE, 2013*

[2] E. Data, JavaScript Dominates EMEA Development Jan. 2008[Online]. Available: http://www.evansdata.com/press/viewRelease. php?pressID=127

[3] C. Collberg and C. Thomborson, "Software watermarking: Models and dynamic embeddings," in *Proc. Symp. Principles of Programming Languages (POPL'99)*, 1999, pp. 311–324.

[4] H. Tamada, M. Nakamura, and A. Monden, "Design and evaluation of birthmarks for detecting theft of java programs," in *Proc. IASTED Int. Conf. Software Eng.*, 2004, pp. 569–575.

[5] Wang,Y.-C. Jhi, S. Zhu, and P. Liu, "Behavior based software theft detection," in *Proc. 16th ACM Conf. Comput. and Commun. Security (CCS '09)*, New York, 2009, pp. 280–290, ACM.

[6] D. Schuler, V. Dallmeier, and C. Lindig, *"A dynamic birthmark for java,"* in *Proc. 22nd IEEE/ACM Int. Conf. Automated Software Eng. (ASE '07)*, New York, 2007, pp. 274–283, ACM.

[7] Detecting Software Theft via Whole Program Path Birthmarks Ginger Myles Christian Collberg {mylesg,collberg} @cs.arizona.edu, University of Arizona, Department of Computer Science , 2004

[8] *"Dynamic Path Based Software Watermarking" by C.Collberg in the year 2001.*

[9] *Dynamic Software Birthmarks to Detect the Theft of Windows Applications*, Haruaki Tamada Keiji Okamoto Masahide Nakamura Akito Monden Kenichi Matsumoto Graduate School of Technology, 8916-5, Takayama, Ikoma, Nara 630-0101, Japan,

+

