Second heap snapshot of www.google.com passed to the Graph Generator which generates heap graph as shown in figure 2.Then resulting graph passed as input to the Graph Filter which only consider Object of type OBJECT and CLOSURE and remaining types of objects are discarded. References of type ELEMENT and PROPERTY are only included other are discarded. The resultant filtered graph is as follows

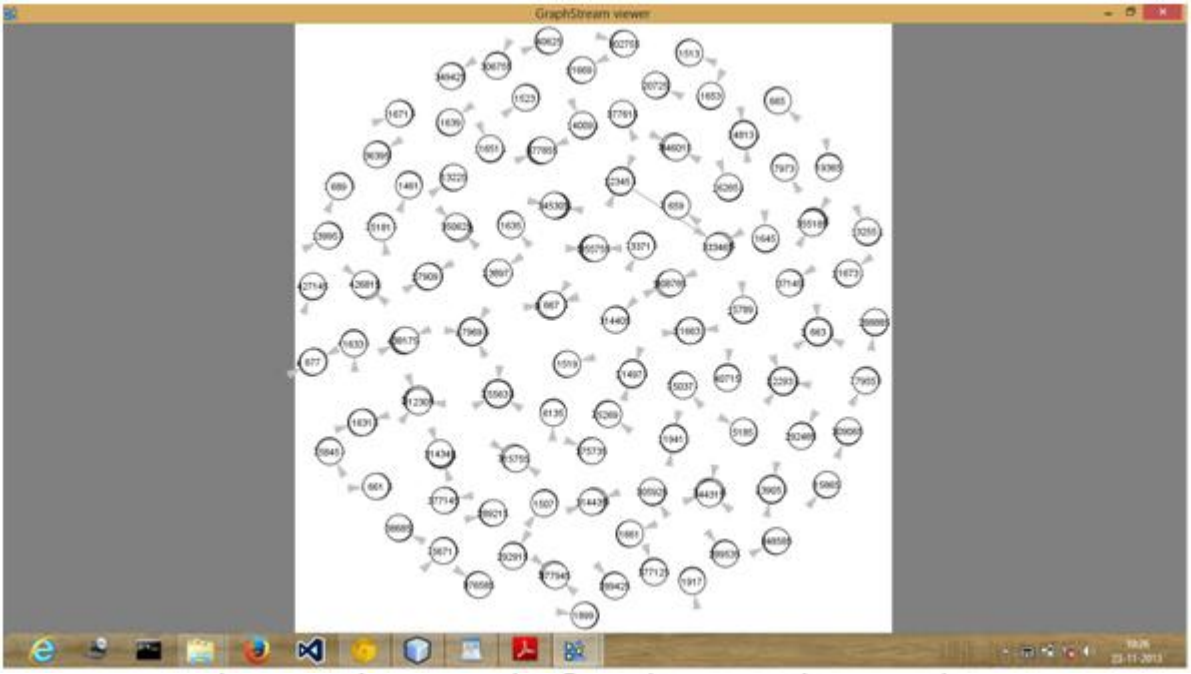

Figure 4: Results of Graph filtered of graph from fig.2 by second module.

In the heap graph all objects are assigned a unique id. Two snapshots which are shown in figure 3 and figure 4 are passed to the Graph Merger which produces single graph and merging done on the basis of object id. The resultant single graph is s follows

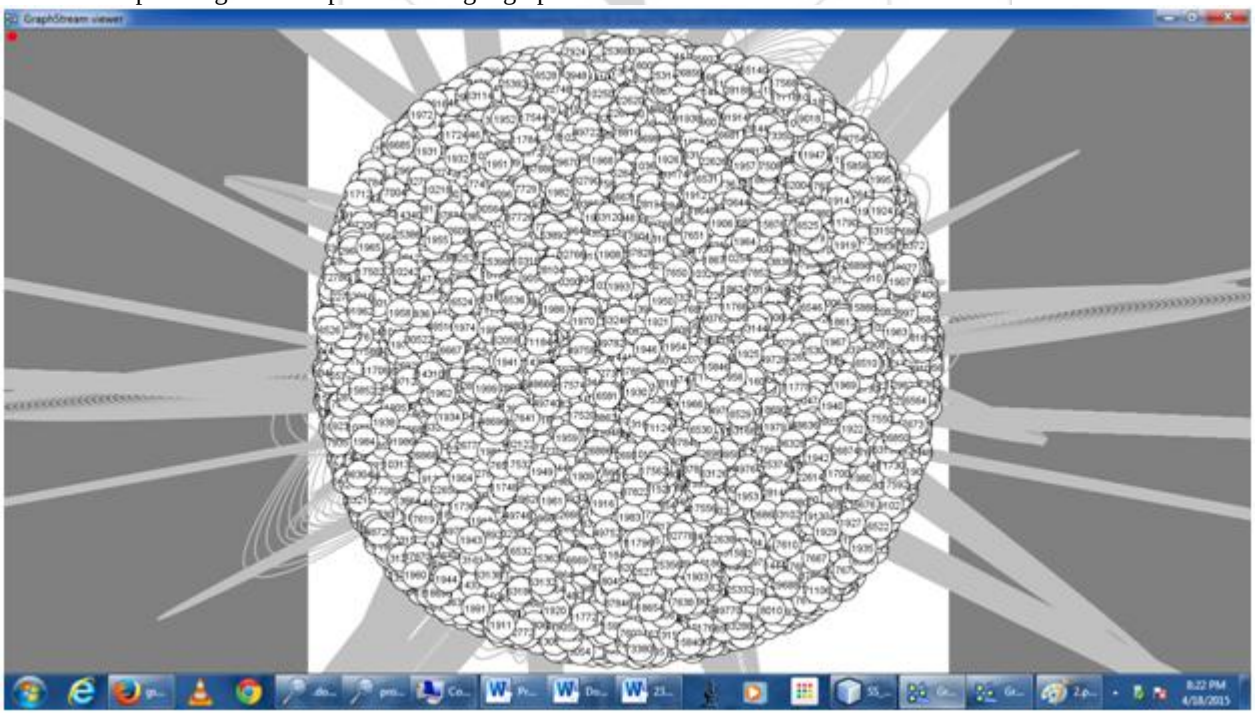

Figure 5: Results of merging of two Graphs from fig.3 and fig 4

Frequent sub graph Mining is applied on single graph generated by Graph Merger as shown in figure 5 which searches subgraph with maximum occurrence in the largest graph. The resultant subgraph of graph displayed in figure 5 is as follows

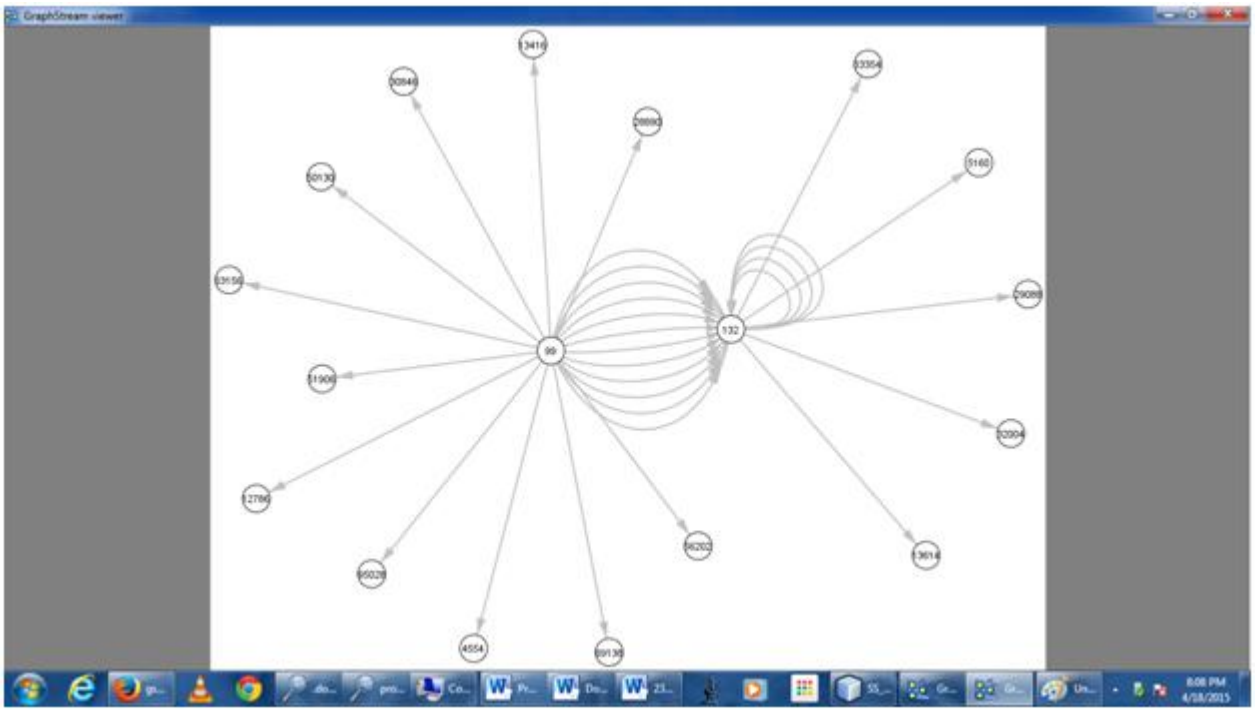

Figure 6: Subgraph selected by subgraph selector module as a software birthmark

6. Conclusion and Future Work

we proposed a robust heap graph based software birthmark system for JavaScript programs. We evaluated our birthmark system using 50 large-scale websites which gives us birthmark through frequent subgraph mining. We will implement modified subgraph detector which will search birthmark of original program into suspected program

References

[1] A. Monden, H. Iida, K. I.Matsumoto, K. Inoue, and K. Torii, "Watermarking java programs," in Proc. Int. Symp. Future Software Technol., Nanjing, China, 1999.
[2] C. Collberg, E. Carter, S. Debray, A. Huntwork, J. Kececioglu, C. Linn, and M. Stepp, "Dynamic path-based software watermarking," in Proc. ACM SIGPLAN 2004 Conf. Programming Language Design and Implementation (PLDI '04), New York, 2004, pp. 107–118, ACM.
[3] X. Wang,Y.-C. Jhi, S. Zhu, and P. Liu, "Behavior based software theft detection," in Proc. 16th ACM Conf. Comput.andCommun. Security (CCS '09), New York, 2009, pp. 280–290, ACM.
[4] G.Myles and C. Collberg, "Detecting software theft via whole program path birthmarks," in Proc. Inf. Security 7th Int. Conf. (ISC 2004), Palo Alto, CA, Sep. 27–29, 2004, pp. 404–415.
[5] 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.
[6] H. Tamada, K. Okamoto, M. Nakamura, A. Monden, and K. I. Matsumoto, Design and Evaluation of Dynamic Software Birthmarks based on API Calls, Nara Institute of Science and Technology, Tech. Rep., 2007. [7] P. Chan, L. Hui, and S. Yiu, "Jsbirth: Dynamic JavaScript birthmark based on the run-time heap," in Proc. 2011 IEEE 35th Annu. Comput.Software and Applicat. Conf. (COMPSAC), Jul. 2011, pp. 407–412.
[7] P. P. F. Chan, L. C. K. Hui, and S.M. Yiu, "Dynamic software birthmark for java based on heap memory analysis," in Proc. 12th IFIP TC 6/TC 11 Int. Conf. Commun. and Multimedia Security (CMS'11), Berlin, Heidelberg, 2011, pp. 94–106, Springer-Verlag.
[8] Yan, X.,Jiawei Han, "gSpan: graph-based substructure pattern mining," in 2002 IEEE International Conference on Data Mining, 2002. ICDM 2003.