[9] Wang W, Wang X, Feng DW, Liu JQ, Han Z, Zhang XL. Exploring permission-induced risk in Android applications for malicious application detection. IEEE Trans. on Information Forensics and Security, 2014,9(11):1869−1882. [doi: 10.1109/TIFS.2014.2353996]

[10] Cesare S, Xiang Y, Zhou WL. Control flow-based malware variant detection. IEEE Trans. on Dependable and Secure Computing, 2014,11(4):307−317. [doi: 10.1109/TDSC.2013.40]

[11] Park Y, Reeves DS, Stamp M. Deriving common malware behavior through graph clustering. Computers and Security, 2013,39(Part B):419−430. [doi: 10.1016/j.cose.2013.09.006]

[12] Chandramohan M, Tan HBK, Briand LC, Padmanabhuni BM. A scalable approach for malware detection through bounded feature space behavior modeling. In: Proc. of the 28th IEEE/ACM Int'l Conf. on Automated Software Engineering. 2013. 312−322. [doi: 10.1109/ASE.2013.6693090]

[13] Arzt S, Rasthofer S, Fritz C, Bodden E, Bartel A, Klein J, Traon YL, Octeau D, McDaniel P. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In: Proc. of the 35th ACM SIGPLAN Conf. on Programming Language Design and Implementation. 2014. 259−269. [doi: 10.1145/2594291.2594299]

[14] Alazab M. Profiling and classifying the behavior of malicious codes. Journal of Systems and Software, 2015,100:91−102. [doi: 10.1016/j.jss.2014.10.031]

[15] Saxe J, Berlin K. Deep neural network based malware detection using two dimensional binary program features. In: Proc. of the 2015 10th Int'l Conf. on Malicious and Unwanted Software. 2015. 11−20. [doi: 10.1109/MALWARE.2015.7413680]

[16] David OE, Netanyahu NS. Deepsign: Deep learning for automatic malware signature generation and classification. In: Proc. of the 2015 Int'l Joint Conf. on Neural Networks. 2015. 1−8. [doi: 10.1109/IJCNN.2015.7280815]

[17] Richardson F, Reynolds D, Dehak N. Deep neural network approaches to speaker and language recognition. IEEE Signal Processing Letters, 2015,22(10):1671−1675. [doi: 10.1109/LSP.2015.2420092]

[18] Huo X, Li M, Zhou ZH. Learning unified features from natural and programming languages for locating buggy source code. In: Proc. of the 25th Int'l Joint Conf. on Artificial Intelligence. 2016. 1606−1612.

[19] Balachandran V, Emmanuel S. Software protection with obfuscation and encryption. In: Proc. of the Int'l Conf. on Information Security Practice and Experience. 2013. 309−320. [doi: 10.1007/978-3-642-38033-4_22]

[20] Cheng R, Zhang FG. Obfuscation for multi user encryption and its application in cloud computing. Concurrency and Computation: Practice and Experience, 2015,27(8):2170−2190. [doi: 10.1002/cpe.3399]

[21] He YX, Chen Y, Wu W, Chen N, Xu C, Liu JB, Su W. A program flow-sensitive self-modifying code obfuscation method. Computer Engineering and Science, 2012,34(1):79−85 (in Chinese with English abstract).

[22] Dong H. Research on the detection and protection technology of mobile applications [Ph.D. Thesis]. Beijing: Beijing University of Posts and Telecommunications, 2014 (in Chinese with English abstract).

[23] Lim HI, Park H, Choi S, Han T. A method for detecting the theft of Java programs through analysis of the control flow information. Information and Software Technology, 2009,51(9):1338−1350. [doi: 10.1016/j.infsof.2009.04.011]

[24] Stojanović S, Radivojević Z, Cvetanović M. Approach for estimating similarity between procedures in differently compiled binaries. Information and Software Technology, 2015,58:259−271. [doi: 10.1016/j.infsof.2014.06.012]

[25] Sæbjørnsen A, Willcock J, Panas T, Quinlan D, Su ZD. Detecting code clones in binary executables. In: Proc. of the 18th Int'l Symp. on Software Testing and Analysis. 2009. 117−128. [doi: 10.1145/1572272.1572287]

[26] Tamada H, Nakamura M, Monden A, Matsumoto K. Java birthmarks—Detecting the software theft—. IEICE Trans. on Information and Systems, 2005,88(9):2148−2158. [doi: 10.1093/ietisy/e88-d.9.2148]

[27] Desnos A. Android: Static analysis using similarity distance. In: Proc. of the 45th Hawaii Int'l Conf. on System Sciences. 2012. 5394−5403. [doi: 10.1109/HICSS.2012.114]

[28] Tian ZZ, Zheng QH, Liu T, Fan M, Zhuang E, Yang ZJ. Software plagiarism detection with birthmarks based on dynamic key instruction sequences. IEEE Trans. on Software Engineering, 2015,41(12):1217−1235. [doi: 10.1109/TSE.2015.2454508]

[29] Crussell J, Gibler C, Chen H. Andarwin: Scalable detection of android application clones based on semantics. IEEE Trans. on Mobile Computing, 2015,14(10):2007−2019. [doi: 10.1109/TMC.2014.2381212]

[30] Caliskan-Islam A, Harang R, Liu A, Narayanan A, Voss C, Yamaguchi F, Greenstadt R. De-Anonymizing programmers via code stylometry. In: Proc. of the 24th USENIX Conf. on Security Symp. 2015. 255−270.

[31] Burrows S, Tahaghoghi SMM. Source code authorship attribution using *n*-grams. In: Proc. of the 12th Australasian Document Computing Symp. Melbourne: RMIT University, 2007. 32−39.

[32] Gray AR, Sallis PJ, MacDonell SG. Software forensics: Extending authorship analysis techniques to computer programs. Information Science Discussion Papers Series No.97/14, 1997.

[33] Neme A, Pulido JRG, Muñoz A, Hernandez S, Dey T. Stylistics analysis and authorship attribution algorithms based on self-organizing maps. Neurocomputing, 2015,147:147−159. [doi: 10.1016/j.neucom.2014.03.064]

[34] Burrows S, Uitdenbogerd AL, Turpin A. Comparing techniques for authorship attribution of source code. Software: Practice and Experience, 2014,44(1):1−32. [doi: 10.1002/spe.2146]

[35] Wisse W, Veenman C. Scripting DNA: Identifying the Javascript programmer. Digital Investigation, 2015,15:61−71. [doi: 10.1016/j.diin.2015.09.001]

[36] Ding HB, Samadzadeh MH. Extraction of Java program fingerprints for software authorship identification. Journal of Systems and Software, 2004,72(1):49−57. [doi: 10.1016/S0164-1212(03)00049-9]

[37] Alrabaee S, Saleem N, Preda S, Wang LY, Debbabi M. OBA2: An onion approach to binary code authorship attribution. Digital Investigation, 2014,11:S94−S103. [doi: 10.1016/j.diin.2014.03.012]

附中文参考文献:

[1] 国家计算机网络应急技术处理协调中心.2015 年中国互联网网络安全报告.2015. http://10.3.200.202/cache/11/03/cert.org.cn/9ef7 d80315e96fbe5617b189a9e2715a/2015annualreport.pdf

[2] 艾瑞咨询&安全管家.2013 年移动安全报告.2013. http://www.199it.com/archives/188839.html

[21] 何炎祥,陈勇,吴伟,陈念,徐超,刘健博,苏雯.基于程序流敏感的自修改代码混淆方法.计算机工程与科学,2012,34(1):79−85.

[22] 董航.移动应用程序检测与防护技术研究[博士学位论文].北京:北京邮电大学,2014.

杨昕雨(1991－),女,吉林长岭人,博士生,主要研究领域为软件安全.

徐国爱(1972－),男,博士,教授,博士生导师,主要研究领域为软件安全,信息安全管理,密码学.