[16] out-of-bounds read due to signedness error, https://download.lighttpd.net/lighttpd/security/lighttpd_sa_2011_01.txt.

[17] Apache httpd Vulnerability Workaround, http://mail-archives.apache.org/mod_mbox/httpd-users/201408.mbox/%3CCAC=HunseOneq3nOoVbSSADMPVgTpDeihHYOu+95b66fLUT2Qow@mail.gmail.com%3E.

[18] Squid Range Headers Vulnerability Workaround, http://www.squid-cache.org/Advisories/SQUID-2014_2.txt.

[19] Huang, Z. and Tan, G. (2019) Rapid Vulnerability Mitigation with Security Workarounds. In *Proceedings of the 2nd NDSS Workshop on Binary Analysis Research*, BAR '19. doi:http://dx.doi.org/10.14722/bar.2019.23052.

[20] Shacham, H. (2007) The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*: 552–61.

[21] Yee, B., Sehr, D., Dardyk, G., Chen, J.B., Muth, R., Ormandy, T., Okasaka, S. *et al.* (2009) Native client: A sandbox for portable, untrusted x86 native code. In *2009 30th IEEE Symposium on Security and Privacy*: 79–93. doi:10.1109/SP.2009.25.

[22] Wartell, R., Mohan, V., Hamlen, K.W. and Lin, Z. (2012) Securing untrusted code via compiler-agnostic binary rewriting. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12 (New York, NY, USA: ACM): 299–308. doi:10.1145/2420950.2420995, URL http://doi.acm.org/10.1145/2420950.2420995.

[23] Pappas, V., Polychronakis, M. and Keromytis, A.D. (2012) Smashing the gadgets: Hindering return-oriented programming using in-place code randomization. In *2012 IEEE Symposium on Security and Privacy*: 601–615. doi:10.1109/SP.2012.41.

[24] v. d. Veen, V., Göktas, E., Contag, M., Pawoloski, A., Chen, X., Rawat, S., Bos, H. *et al.* (2016) A tough call: Mitigating advanced code-reuse attacks at the binary level. In *2016 IEEE Symposium on Security and Privacy (SP)*: 934–953. doi:10.1109/SP.2016.60.

[25] Xu, Z., Zhang, Y., Zheng, L., Xia, L., Bao, C., Wang, Z. and Liu, Y. (2020) Automatic hot patch generation for android kernels. In *29th USENIX Security Symposium (USENIX Security 20)* (USENIX Association): 2397–2414. URL https://www.usenix.org/conference/usenixsecurity20/presentation/xu.

[26] Schuster, F., Tendyck, T., Liebchen, C., Davi, L., Sadeghi, A. and Holz, T. (2015) Counterfeit object-oriented programming: On the difficulty of preventing code reuse attacks in c++ applications. In *2015 IEEE Symposium on Security and Privacy*: 745–762. doi:10.1109/SP.2015.51.

[27] Wahbe, R., Lucco, S., Anderson, T.E. and Graham, S.L. (1994) Efficient software-based fault isolation. In *ACM SIGOPS Operating Systems Review*, **27**: 203–216.

[28] Tan, G. (2017) Principles and implementation techniques of software-based fault isolation. *Foundations and Trends in Privacy and Security* **1**(3): 137–198.

[29] Abadi, M., Budiu, M., Erlingsson, U. and Ligatti, J. (2005) Control-flow integrity. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*: 340–353.

[30] Wang, G., Chattopadhyay, S., Gotovchits, I., Mitra, T. and Roychoudhury, A. (2019) oo7: Low-overhead defense against spectre attacks via program analysis. *IEEE Transactions on Software Engineering* **47**(11): 2504–2519.

[31] Aleph One (1996) Smashing the stack for fun and profit. *Phrack Magazine* **7**(49).

[32] Ghavamnia, S., Palit, T., Benameur, A. and Polychronakis, M. (2020) Confine: Automated system call policy generation for container attack surface reduction. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*: 443–458.

[33] Duan, R., Bijlani, A., Ji, Y., Alrawi, O., Xiong, Y., Ike, M., Saltaformaggio, B. *et al.* (2019) Automating patching of vulnerable open-source software versions in application binaries. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019* (The Internet Society).

[34] Miyani, D., Huang, Z. and Lie, D. (2017) BinPro: A Tool for Binary Source Code Provenance. *arXiv* .

[35] Tian, Y. and Ray, B. (2017) Automatically diagnosing and repairing error handling bugs in c. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017 (New York, NY, USA: ACM): 752–762. doi:10.1145/3106237.3106300, URL http://doi.acm.org/10.1145/3106237.3106300.

[36] Marinescu, P.D. and Candea, G. (2009) Lfi: A practical and general library-level fault injector. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*: 379–388. doi:10.1109/DSN.2009.5270313.

[37] Jana, S., Kang, Y.J., Roth, S. and Ray, B. (2016) Automatically detecting error handling bugs using error specifications. In *25th USENIX Security Symposium (USENIX Security 16)* (Austin, TX: USENIX Association): 345–362. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/jana.

[38] Kang, Y., Ray, B. and Jana, S. (2016) Apex: Automated inference of error specifications for c apis. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ASE 2016 (New York, NY, USA: ACM): 472–482. doi:10.1145/2970276.2970354, URL http://doi.acm.org/10.1145/2970276.2970354.

[39] (2018), System Error Codes, https://docs.microsoft.com/en-us/windows/desktop/debug/system-error-codes.

[40] Red Hat Bugzilla – Bug 758624, https://bugzilla.redhat.com/show_bug.cgi?id=758624.

[41] Shoshitaishvili, Y., Wang, R., Salls, C., Stephens, N., Polino, M., Dutcher, A., Grosen, J. *et al.* (2016) SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis. In *IEEE Symposium on Security and Privacy*.

[42] (2018), Talos: a software tool that automatically generates and instruments SWRRs into target applications using static program analysis. , https://github.com/huang-zhen/Talos.

[43] Nethercote, N. and Seward, J. (2007) Valgrind: A framework for heavyweight dynamic binary instrumentation. In *The 2007 ACM Conf. on Programming Language*

Design and Implementation (PLDI): 89–100.

[44] Aho, A.V., Lam, M.S., Sethi, R. and Ullman, J.D. (2007) *Compilers: principles, techniques, & tools* (Addison Wesley).

[45] Scrapy | A Fast and Powerful Scraping and Web Crawling Framework, http://scrapy.org.

[46] (2018), Programming reference for Windows API, https://docs.microsoft.com/en-us/windows/desktop/api/index.

[47] (2018), Windows Symbol Packages, https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugger-download-symbols.

[48] (2018), INSTALLING DEBUGINFO PACKAGES, https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/developer_guide/intro.debuginfo.

[49] (2018), Debug Symbol Packages, https://wiki.ubuntu.com/Debug%20Symbol%20Packages.

[50] Buck, B. and Hollingsworth, J.K. (2000) An API for Runtime Code Patching. *The International Journal of High Performance Computing Applications* **14**(4): 317–329. doi:10.1177/109434200001400404, URL https://doi.org/10.1177/109434200001400404. https://doi.org/10.1177/109434200001400404.

[51] Huang, Z. and Lie, D. (2014) Ocasta: Clustering configuration settings for error recovery. In *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '14 (Washington, DC, USA: IEEE Computer Society): 479–490. doi:10.1109/DSN.2014.51, URL http://dx.doi.org/10.1109/DSN.2014.51.

[52] Microsoft Windows WebViewFolderIcon ActiveX Control setSlice() Integer Overflow Vulnerability, https://tools.cisco.com/security/center/viewAlert.x?alertId=11787.