REFERENCES

[1] Online; accessed 22-Sept-2018. Bypassing non-executable memory, ASLR and stack canaries on x86-64 Linux. https://www.antoniobarresi.com/security/exploitdev/2014/05/03/64bitexploitation/.

[2] Online; accessed 22-Sept-2018. Defeating DEP with ROP. https://samsclass.info/127/proj/rop.htm.

[3] Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. 2009. Control-flow Integrity Principles, Implementations, and Applications. *ACM Trans. Inf. Syst. Secur.*

[4] Periklis Akritidis, Cristian Cadar, Costin Raiciu, Manuel Costa, and Miguel Castro. 2008. Preventing memory error exploits with WIT. In *Proceedings of S&P'08*.

[5] Daniel J Bernstein. 2007. Some thoughts on security after ten years of qmail 1.0. In *Proceedings of CSAW'07*.

[6] Dimitar Bounov, Rami Gökhan Kici, and Sorin Lerner. 2016. Protecting C++ Dynamic Dispatch Through VTable Interleaving. In *Proceedings of NDSS'16*.

[7] Scott Brookes and Stephen Taylor. 2016. Containing a Confused Deputy on x86: A Survey of Privilege Escalation Mitigation Techniques. *International Journal of Advanced Computer Science and Applications.*

[8] Linux capabilities. Online; accessed 23-Sep-2018. http://man7.org/linux/man-pages/man7/capabilities.7.html.

[9] Nicolas Carlini, Antonio Barresi, Mathias Payer, David Wagner, and Thomas R. Gross. 2015. Control-flow Bending: On the Effectiveness of Control-flow Integrity. In *Proceedings of SEC'15*.

[10] Miguel Castro, Manuel Costa, and Tim Harris. 2006. Securing software by enforcing data-flow integrity. In *Proceedings of OSDI'06*.

[11] Hao Chen, David Wagner, and Drew Dean. 2002. Setuid Demystified. In *Proceedings of SEC'02*.

[12] Shuo Chen, Jun Xu, Emre Can Sezer, Prachi Gauriar, and Ravishankar K Iyer. 2005. Non-Control-Data Attacks Are Realistic Threats. In *Proceedings of SEC'05*.

[13] Yueqiang Cheng, Zongwei Zhou, Yu Miao, Xuhua Ding, and Huijie Robert Deng. 2014. ROPecker: A generic and practical approach for defending against rop attacks. In *Proceedings of NDSS'14*.

[14] Mauro Conti, Stephen Crane, Lucas Davi, Michael Franz, Per Larsen, Marco Negro, Christopher Liebchen, Mohaned Qunaibit, and Ahmad-Reza Sadeghi. 2015. Losing Control: On the Effectiveness of Control-Flow Integrity Under Stack Attacks. In *Proceedings of CCS'15*.

[15] Thurston HY Dang, Petros Maniatis, and David Wagner. 2015. The performance cost of shadow stacks and stack canaries. In *Proceedings of ASIACCS'15*.

[16] Shellcodes database for study cases. Online; accessed 23-Sep-2018. http://shell-storm.org/shellcode/.

[17] Mark S Dittmer and Mahesh V Tripunitara. 2014. The UNIX process identity crisis: A standards-driven approach to setuid. In *Proceedings of CCS'14*.

[18] Isaac Evans, Fan Long, Ulziibayar Otgonbaatar, Howard Shrobe, Martin Rinard, Hamed Okhravi, and Stelios Sidiroglou-Douskos. 2015. Control Jujutsu: On the Weaknesses of Fine-Grained Control Flow Integrity. In *Proceedings of CCS'15*.

[19] Henry Hanping Feng, Oleg M Kolesnikov, Prahlad Fogla, Wenke Lee, and Weibo Gong. 2003. Anomaly detection using call stack information. In *Proceedings of S&P'03*.

[20] Yangchun Fu, Junghwan Rhee, Zhiqiang Lin, Zhichun Li, Hui Zhang, and Guofei Jiang. 2016. Detecting Stack Layout Corruptions with Robust Stack Unwinding. In *Proceedings of RAID'16*.

[21] Khilan Gudka, Robert N.M. Watson, Jonathan Anderson, David Chis nall, Brooks Davis, Ben Laurie, Ilias Marinos, Pe ter G. Neumann, and Alex Richardson. 2015. Clean Application Compartmentalization with SOAAP. In *Proceedings of CCS'15*.

[22] Norm Hardy. 1988. The Confused Deputy:(or why capabilities might have been invented). In *Proceedings of SIGOPS'88*.

[23] terry ching-hsiang Hsu, kevin hoffman, patrick eugster, and mathias payer. 2016. enforcing least privilege memory views for multithreaded applications. In *proceedings of CCS'16*.

[24] Hong Hu, Zheng Leong Chua, Sendroiu Adrian, Prateek Saxena, and Zhenkai Liang. 2015. Automatic Generation of Data-Oriented Exploits.. In *Proceedings of SEC'15*.

[25] Hong Hu, Shweta Shinde, Sendroiu Adrian, Zheng Leong Chua, Prateek Saxena, and Zhenkai Liang. 2016. Data-oriented programming: On the expressiveness of non-control data attacks. In *Proceedings of S&P'16*.

[26] Intel. Online; accessed 23-Sep-2018. Control-flow enforcement technology (CET) preview. https://software.intel.com/sites/default/files/managed/4d/2a/control-flow-enforcement-technology-preview.pdf.

[27] Kyriakos Ispoglou, Bader AlBassam, Trent Jaeger, and Mathias Payer. [n. d.]. Block Oriented Programming: Automating Data-Only Attacks. In *Proceedings of CCS'18*.

[28] Bhushan Jain, Chia-Che Tsai, Jitin John, and Donald E Porter. 2014. Practical Techniques to Obviate Setuid-to-root Binaries. In *Proceedings of EuroSys'14*.

[29] Jim Keniston. Online; accessed 23-Sep-2018. Kernel Probes. https://elixir.free-electrons.com/linux/v4.0/source/Documentation/kprobes.txt.

[30] Chung Hwan Kim, Taegyu Kim, Hongjun Choi, Zhongshu Gu, Byoungyoung Lee, Xiangyu Zhang, and Dongyan Xu. 2018. Securing Real-Time Microcontroller Systems through Customized Memory View Switching. In *Proceedings of NDSS'18*.

[31] Gene H Kim and Eugene H Spafford. 1994. The design and implementation of tripwire: A file system integrity checker. In *Proceedings of CCS'94*.

[32] Christopher Kruegel, Darren Mutz, Fredrik Valeur, and Giovanni Vigna. 2003. On the detection of anomalous system call arguments. In *Proceedings of ESORICS'03*.

[33] Volodymyr Kuznetsov, László Szekeres, Mathias Payer, George Candea, R Sekar, and Dawn Song. 2014. Code-Pointer Integrity. In *Proceedings of OSDI'14*.

[34] Long Le. 2010. Payload Already Inside: Data Reuse for ROP Exploits. (2010).

[35] LLVM. Online; accessed 23-Sep-2018. The LLVM Compiler Infrastructure Project. http://llvm.org/.

[36] Kangjie Lu, Chengyu Song, Taesoo Kim, and Wenke Lee. 2016. UniSan: Proactive kernel memory initialization to eliminate data leakages. In *Proceedings of CCS'16*.

[37] Microsoft. Online; accessed 23-Sep-2018. Data Execution Prevention (DEP). https://msdn.microsoft.com/en-us/library/windows/desktop/aa366553(v=vs.85).aspx.

[38] Darren Mutz, William Robertson, Giovanni Vigna, and Richard Kemmerer. 2007. Exploiting execution context for the detection of anomalous system calls. In *Proceedings of RAID'07*.

[39] Ben Niu and Gang Tan. 2013. Monitor Integrity Protection with Space Efficiency and Separate Compilation. In *Proceedings of CCS'13*.

[40] Ben Niu and Gang Tan. 2014. Modular Control-flow Integrity. In *Proceedings of PLDI'14*.

[41] Ben Niu and Gang Tan. 2014. RockJIT: Securing Just-In-Time Compilation Using Modular Control-Flow Integrity. In *Proceedings of CCS'14*.

[42] Ben Niu and Gang Tan. 2015. Per-input control-flow integrity. In *Proceedings of CCS'15*.

[43] Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis. 2013. Transparent ROP Exploit Mitigation Using Indirect Branch Tracing. In *Proceedings of SEC'13*.

[44] AppArmor Project. Online; accessed 23-Sep-2018. http://wiki.apparmor.net/index.php/Main_Page.

[45] Niels Provos. 2003. Improving Host Security with System Call Policies.. In *Proceedings of SEC'03*.

[46] Mohammed Rangwala, Ping Zhang, Xukai Zou, and Feng Li. 2014. A taxonomy of privilege escalation attacks in android applications. *International Journal of Security and Networks* (2014).

[47] Ryan Roemer, Erik Buchanan, Hovav Shacham, and Stefan Savage. 2012. Return-Oriented Programming: Systems, Languages, and Applications. *ACM Trans. Inf. Syst. Secur.* 15, 1, Article 2 (March 2012), 34 pages. https://doi.org/10.1145/2133375.2133377

[48] Jerome H. Saltzer. 1974. Protection and the Control of Information Sharing in Multics. *Comm. ACM.*

[49] Seccomp. Online; accessed 23-Sep-2018. SECure COMPuting with filters. https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt.

[50] Noam Shalev, Idit Keidar, Yaron Weinsberg, Yosef Moatti, and Elad Ben-Yehuda. 2017. WatchIT: Who Watches Your IT Guy?. In *Proceedings of SOSP'17*.

[51] Stephen Smalley, Chris Vance, and Wayne Salamon. 2001. Implementing SELinux as a Linux security module. *NAI Labs Report.*

[52] Chengyu Song, Byoungyoung Lee, Kangjie Lu, William Harris, Taesoo Kim, and Wenke Lee. 2016. Enforcing Kernel Security Invariants with Data Flow Integrity. In *Proceedings of NDSS'16*.

[53] PaX Team. Online; accessed 23-Sep-2018. Pax: the Linux kernel patch for least privilege protection. https://en.wikipedia.org/wiki/PaX.

[54] Victor Van der Veen, Dennis Andriesse, Enes Göktaş, Ben Gras, Lionel Sambuc, Asia Slowinska, Herbert Bos, and Cristiano Giuffrida. 2015. Practical Context-Sensitive CFI. In *Proceedings of CCS'15*.

[55] Jeffrey A Vaughan and Andrew D Hilton. 2010. Paladin: Helping Programs Help Themselves with Internal System Call Interposition.

[56] Hayawardh Vijayakumar, Xinyang Ge, Mathias Payer, and Trent Jaeger. 2014. JIGSAW: Protecting Resource Access by Inferring Programmer Expectations. In *Proceedings of SEC'14*.

[57] David Wagner and R Dean. 2001. Intrusion detection via static analysis. In *Proceedings of S&P'01*.

[58] M. Zalewski. Online; accessed 23-Sep-2018. American Fuzzy Lop. http://lcamtuf.coredump.cx/afl/.

[59] Chao Zhang, Dawn Xiaodong Song, Scott A. Carr, Mathias Payer, Tongxin Li, Yu Ding, and Chengyu Song. 2016. VTrust: Regaining Trust on Virtual Calls. In *Proceedings of NDSS'16*.

[60] Chao Zhang, Tao Wei, Zhaofeng Chen, Lei Duan, Laszlo Szekeres, Stephen McCamant, Dawn Song, and Wei Zou. 2013. Practical Control Flow Integrity and Randomization for Binary Executables. In *Proceedings of S&P'13*.

[61] Mingwei Zhang and R. Sekar. 2013. Control Flow Integrity for COTS Binaries. In *Proceedings of SEC'13*.