E References Cited

[1] M. Abadi, M. Budiu, Ú. Erlingsson, and J. Ligatti. Control-flow integrity principles, implementations, and applications. *ACM Transactions on Information and System Security*, 13(1), 2009.

[2] P. Akritidis, C. Cadar, C. Raiciu, M. Costa, and M. Castro. Preventing memory error exploits with WIT. In *Proceedings of the 29th IEEE Symposium on Security and Privacy (Oakland)*, pages 263–277, Oakland, CA, May 2008.

[3] I. Anati, S. Gueron, S. P. Johnson, and V. R. Scarlata. Innovative Technology for CPU Based Attestation and Sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, pages 1–8, Tel-Aviv, Israel, 2013.

[4] A. Baumann, M. Peinado, and G. Hunt. Shielding applications from an untrusted cloud with haven. In *Proceedings of the 11th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 267–283, Broomfield, Colorado, Oct. 2014.

[5] S. Bhatkar, D. C. DuVarney, and R. Sekar. Address obfuscation: An efficient approach to combat a broad range of memory error exploits. In *Proceedings of the 12th Usenix Security Symposium (Security)*, pages 105–120, Washington, DC, Aug. 2003.

[6] D. Bigelow, T. Hobson, R. Rudd, W. Streilein, and H. Okhravi. Timely rerandomization for mitigating memory disclosures. In *Proceedings of the 22st ACM Conference on Computer and Communications Security (CCS)*, Denver, Colorado, Oct. 2015.

[7] T. Bletsch, X. Jiang, and V. Freeh. Mitigating code-reuse attacks with control-flow locking. In *Proceedings of 27th Annual Computer Security Applications Conference (ACSAC)*, pages 353–362, 2011.

[8] C. Cadar, D. Dunbar, and D. R. Engler. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *Proceedings of the 8th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 209–224, San Diego, CA, Dec. 2008.

[9] P. Carbin. Intel Identity Protection Technology with PKI (Intel IPT with PKI), May 2012. White Paper, Technology Overview.

[10] M. Castro, M. Costa, and T. Harris. Securing software by enforcing data-flow integrity. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 147–160, Seattle, WA, Nov. 2006.

[11] M. Castro, M. Costa, J.-P. Martin, M. Peinado, P. Akritidis, A. Donnelly, P. Barham, and R. Black. Fast byte-granularity software fault isolation. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP)*, pages 45–58, Big Sky, MT, Oct. 2009.

[12] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[13] A. Chaturvedi, S. Bhatkar, and R. Sekar. Improving attack detection in host-based ids by learning properties of system call arguments. In *Proceedings of the 26th IEEE Symposium on Security and Privacy (Oakland)*, Oakland, CA, May 2005.

[14] S. Checkoway and H. Shacham. Iago Attacks: Why the System Call API is a Bad Untrusted RPC Interface. In *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 253–264, Houston, TX, Mar. 2013.

[15] X. Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and

D. R. Ports. Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, ASPLOS XIII, pages 2–13, Seattle, WA, USA, 2008. ACM.

[16] V. Chipounov, V. Kuznetsov, and G. Candea. S2e: a platform for in-vivo multi-path analysis of software systems. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Newport Beach, CA, Mar. 2011.

[17] M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. In *Proceedings of the 1st India software engineering conference*, pages 5–14. ACM, 2008.

[18] B. Coppens, I. Verbauwhede, K. De Bosschere, and B. De Sutter. Practical mitigations for timing-based side-channel attacks on modern x86 processors. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (Oakland)*, pages 45–60, Oakland, CA, May 2009.

[19] C. Cowan, C. Pu, D. Maier, H. Hintony, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proceedings of the 7th Usenix Security Symposium (Security)*, San Antonio, TX, Jan. 1998.

[20] S. Crane, C. Liebchen, A. Homescu, L. Davi, P. Larsen, A.-R. Sadeghi, S. Brunthaler, and M. Franz. Readactor: Practical code randomization resilient to memory disclosure. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland)*, San Jose, CA, May 2015.

[21] S. Crane, S. Volckaert, F. Schuster, C. Liebchen, P. Larsen, L. Davi, A.-R. Sadeghi, T. Holz, B. D. Sutter, and M. Franz. It's a trap: Table randomization and protection against function reuse attacks. In *Proceedings of the 22st ACM Conference on Computer and Communications Security (CCS)*, Denver, Colorado, Oct. 2015.

[22] S. Davenport and R. Ford. SGX: the good, the bad and the downright ugly, Jan. 2014. https://www.virusbtn.com/virusbulletin/archive/2014/01/vb201401-SGX.

[23] Ú. Erlingsson and F. B. Schneider. SASI enforcement of security policies: A retrospective. In *Proceedings of the New Security Paradigms Workshop*, 1999.

[24] Ú. Erlingsson, M. Abadi, M. Vrable, M. Budiu, and G. C. Necula. XFI: Software guards for system address spaces. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 75–88, Seattle, WA, Nov. 2006.

[25] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 627–638, Chicago, Illinois, Oct. 2011.

[26] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM, 2012.

[27] B. Ford and R. Cox. Vx32: Lightweight user-level sandboxing on the x86. In *Proceedings of the 2008 ATC Annual Technical Conference (ATC)*, pages 293–306, Boston, MA, June 2008.

[28] Y. Fu, Z. Lin, and K. Hamlen. Subverting systems authentication with context-aware, reactive virtual machine introspection. In *Proceedings of the 29th Annual Computer Security Applications Conference (ACSAC'13)*, New Orleans, Louisiana, December 2013.

[29] T. Garfinkel, B. Pfaff, and M. Rosenblum. Ostia: A delegating architecture for secure system call

interposition. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2004.

[30] C. Giuffrida, A. Kuijsten, and A. S. Tanenbaum. Enhanced operating system security through efficient and fine-grained address space randomization. In *Proceedings of the 21st Usenix Security Symposium (Security)*, pages 475–490, Bellevue, WA, Aug. 2012.

[31] J. Hiser, A. Nguyen-Tuong, M. Co, M. Hall, and J. W. Davidson. Ilr: Where'd my gadgets go? In *Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland)*, pages 571–585, San Francisco, CA, May 2012.

[32] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, pages 1–8, Tel-Aviv, Israel, 2013.

[33] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[34] Intel. Graphics Drivers Blue-ray Disc* Playback On Intel Graphics FAQ. http://www.intel.com/support/graphics/sb/CS-029871.htm, 2008. Accessed: 05/04/2015.

[35] Intel. Intel Software Guard Extensions Programming Reference (rev1), Sept. 2013. 329298-001US.

[36] Intel. Intel Software Guard Extensions Programming Reference (rev2), Oct. 2014. 329298-002US.

[37] E. Kang and D. Jackson. Designing and analyzing a flash file system with alloy. *Int. J. Software and Informatics*, 3:129–148, 2009.

[38] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall. A conundrum of permissions: installing applications on an android smartphone. In *Financial Cryptography and Data Security*, pages 68–79. Springer, 2012.

[39] S. T. King and P. M. Chen. Subvirt: Implementing malware with virtual machines. In *Proceedings of the 27th IEEE Symposium on Security and Privacy (Oakland)*, Oakland, CA, May 2006.

[40] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. sel4: Formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 207–220, 2009.

[41] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna. On the detection of anomalous system call arguments. In *Computer Security–ESORICS 2003*, pages 326–343. Springer, 2003.

[42] S. Kumar and E. H. Spafford. A software architecture to support misuse intrusion detection. 1995.

[43] C. Lattner and V. Adve. Llvm: A compilation framework for lifelong program analysis & transformation. In *Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization*, CGO '04, Palo Alto, California, 2004. ISBN 0-7695-2102-9.

[44] X. Leroy. A formally verified compiler back-end. *Journal of Automated Reasoning*, 43(4):363–446, 2009.

[45] N. Li, Z. Mao, and H. Chen. Usable mandatory integrity protection for operating systems. In *Proceedings of the 28th IEEE Symposium on Security and Privacy (Oakland)*, pages 164–178, Oakland, CA, May 2007.

[46] Y. Li, J. McCune, J. Newsome, A. Perrig, B. Baker, and W. Drewry. MiniBox: A Two-Way Sandbox for x86 Native Code. In *Proceedings of the 2014 ATC Annual Technical Conference (ATC)*, pages 409–420, Philadelphia, PA, June 2014.

[47] S. McCamant and G. Morrisett. Evaluating SFI for a CISC architecture. In *Proc. USENIX Security Sym.*, 2006.

[48] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An Execution Infrastructure for TCB Minimization. In *Proceedings of the ACM EuroSys Conference*, pages 315–328, Glasgow, Scotland, Mar. 2008.

[49] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig. Trustvisor: Efficient tcb reduction and attestation. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pages 143–158. IEEE Computer Society, 2010. ISBN 978-0-7695-4035-1. doi: 10.1109/SP.2010.17. URL http://dx.doi.org/10.1109/SP.2010.17.

[50] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, pages 1–8, Tel-Aviv, Israel, 2013.

[51] G. C. Necula. Proof-carrying code. In *Proc. ACM Principles of Programming Languages*, pages 106–119, 1997.

[52] G. C. Necula, S. McPeak, S. P. Rahul, and W. Weimer. Cil: Intermediate language and tools for analysis and transformation of c programs. In *Compiler Construction*, pages 213–228. Springer, 2002.

[53] P. Ning, Y. Cui, D. S. Reeves, and D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security*, 7(2):274–318, 2004.

[54] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2003.

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

[56] B. Parno. Bootstrapping trust in a "trusted" platform. In *Proceedings of the 3rd Conference on Hot Topics in Security (HotSec)*, pages 9:1–9:6, 2008.

[57] R. Perez, R. Sailer, L. van Doorn, et al. vTPM: virtualizing the trusted platform module. In *Proceedings of the 15th Usenix Security Symposium (Security)*, pages 305–320, Vancouver, Canada, July 2006.

[58] D. E. Porter, S. Boyd-Wickizer, J. Howell, R. Olinsky, and G. C. Hunt. Rethinking the library os from the top down. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 291–304, Newport Beach, CA, Mar. 2011.

[59] N. Provos. Improving host security with system call policies. In *Proc. USENIX Security Sym.*, 2003.

[60] N. Provos. Improving host security with system call policies. In *Proceedings of the 12th Usenix Security Symposium (Security)*, Washington, DC, Aug. 2003.

[61] J. Rutkowska. Thoughts on Intel's upcoming Software Guard Extensions (Part 1), Aug. 2013. http://theinvisiblethings.blogspot.com/2013/08/thoughts-on-intels-upcoming-

software.html.

[62] J. Rutkowska. Thoughts on Intel's upcoming Software Guard Extensions (Part 2), Sept. 2013. http://theinvisiblethings.blogspot.com/2013/09/thoughts-on-intels-upcoming-software.html.

[63] N. Santos, H. Raj, S. Saroiu, and A. Wolman. Using arm trustzone to build a trusted language runtime for mobile applications. In *ACM SIGARCH Computer Architecture News*, volume 42, pages 67–80. ACM, 2014.

[64] F. B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000.

[65] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich. VC3: Trustworthy Data Analytics in the Cloud using SGX. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland)*, San Jose, CA, May 2015.

[66] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni. A fast automaton-based method for detecting anomalous program behaviors. In *Proceedings of the 22st IEEE Symposium on Security and Privacy (Oakland)*, pages 144–155, Oakland, CA, May 2001.

[67] A. Seshadri, M. Luk, N. Qu, and A. Perrig. Secvisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 335–350, Stevenson, Washington, USA, 2007. ISBN 978-1-59593-591-5.

[68] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh. On the effectiveness of address-space randomization. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 298–307, Washington, DC, Oct. 2004.

[69] U. Steinberg and B. Kauer. NOVA: A microhypervisor-based secure virtualization architecture. In *Proceedings of the ACM EuroSys Conference*, pages 209–222, Paris, France, Apr. 2010.

[70] K. Sun, J. Wang, F. Zhang, and A. Stavrou. SecureSwitch: BIOS-assisted isolation and switch between trusted and untrusted commodity oses. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2012.

[71] L. Szekeres, M. Payer, T. Wei, and D. Song. Sok: Eternal war in memory. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (Oakland)*, pages 48–62, San Francisco, CA, May 2013.

[72] P. Team. PaX address space layout randomization (ASLR), 2003.

[73] L. Van Doorn. Hardware virtualization trends. In *ACM/Usenix International Conference On Virtual Execution Environments: Proceedings of the 2 nd international conference on Virtual execution environments*, volume 14, pages 45–45, 2006.

[74] D. A. Wagner. Janus: An approach for confinement of untrusted applications. Master's thesis, U. California at Berkeley, 1999.

[75] J. Wang, A. Stavrou, and A. K. Ghosh. Hypercheck: A hardware-assisted integrity monitor. In *Recent Advances in Intrusion Detection, 13th International Symposium, RAID 2010, Ottawa, Ontario, Canada, September 15-17, 2010. Proceedings*, pages 158–177, 2010.

[76] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *Proceedings of the 20th IEEE Symposium on Security and Privacy (Oakland)*, pages 133–145, Oakland, CA, May 1999.

[77] R. Wartell, V. Mohan, K. Hamlen, and Z. Lin. Securing untrusted code via compiler-agnostic binary rewriting. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC'12)*, Orlando, FL, December 2012.

[78] R. Wartell, V. Mohan, K. Hamlen, and Z. Lin. Binary stirring: Self-randomizing instruction addresses of legacy x86 binary code. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12)*, Raleigh, NC, October 2012.

[79] Wikipedia. C dynamic memory allocation — wikipedia, the free encyclopedia, 2015. URL http://en.wikipedia.org/w/index.php?title=C_dynamic_memory_allocation&oldid=658580417. [Online; accessed 13-May-2015].

[80] Y. Xu, W. Cui, and M. Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland)*, San Jose, CA, May 2015.

[81] B. Yee, D. Sehr, G. Dardyk, J. B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, and N. Fullagar. Native Client: A sandbox for portable, untrusted x86 native code. In *Proc. IEEE Sym. Security and Privacy*, pages 79–93, 2009.

[82] J. Zeng, Y. Fu, and Z. Lin. Pemu: A pin highly compatible out-of-vm dynamic binary instrumentation framework. In *Proceedings of the 11th Annual International Conference on Virtual Execution Environments*, Istanbul, Turkey, March 2015.

[83] C. Zhang, T. Wei, Z. Chen, L. Duan, L. Szekeres, S. McCamant, D. Song, and W. Zou. Practical control flow integrity and randomization for binary executables. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (Oakland)*, pages 559–573, San Francisco, CA, May 2013.

[84] F. Zhang, J. Chen, H. Chen, and B. Zang. Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 203–216, Cascais, Portugal, 2011. ACM. ISBN 978-1-4503-0977-6.

[85] M. Zhang and R. Sekar. Control flow integrity for COTS binaries. In *Proceedings of the 22th Usenix Security Symposium (Security)*, pages 337–352, Washington, DC, Aug. 2013.