can be done on the response after an attack is detected and verified. Looking deeper into the what inspection tools to run in case CacheLight flags potential malicious code and the appropriate response mechanisms to employ if the tool does find malicious code in memory.

10 CONCLUSION

In this paper we present CacheLight, a lightweight approach to preventing malicious use of cache locking mechanisms while allowing time-critical applications to legitimately utilize them to ensure execution times in embedded and real-time systems. CacheLight allows the Normal World to perform cache locking through requesting it as a service from the Secure World. All that is needed is a minimal increase in the Trusted Code Base to handle a new SMC, which the OS running in the TEE can then validate and verify to prevent any malicious code from being hidden in the cache.

Upon world switch, the Secure World can now handle and verify the validity of any cache lock request to ensure that any data that will persist in the cache not only maps to a valid address in memory but is also consistent with what is present in main memory; effectively bringing the contents of the cache to light. Additionally, because the Secure World does not hand control back to Normal World after verifying the address, but rather performs the loading and locking on behalf of the Normal World, the attacker cannot bypass the security checks by passing different addresses in the arguments. Should CacheLight find that the attempt to lock the cache is malicious, it can then flush the caches and run memory introspection tools to determine the nature of the attack and retrieve any relevant data for forensic analysis. On the other hand, if the request is determined to be legitimate, CacheLight can service it by taking advantage of World-Shared Memory.

Therefore, CacheLight can successfully prevent malicious code from hiding from SW introspection tools in the NW cache for any significant amount of time. Additionally, while we present a solution for the ARM architecture, the approach can be generalized to any architecture that employs the same execution separation idea. If the attack can be modified to a new architecture, then so can the defense. Moreover, CacheLight incurs the overhead of a world-switch for the set-up of the time-critical data. However, the initial setup of locking data in the cache is already expected to be expensive so that the performance and timing requirements can be met once the setup is done and the application running. CacheLight makes additional overhead to the setup process but not the execution of the time-critical process that requested the lock. Given that it provides security against an otherwise undetectable attack, the trade-off in setup time is extremely worthwhile.

REFERENCES

[1] ARM. 2006-2010. ARM Cortex-A8 Technical Reference Manual. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0344k/DDI0344K_cortex_a8_r3p2_trm.pdf. (2006-2010).

[2] ARM. 2009. ARM Security Technology Building a Secure System using TrustZone Technology. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html. (2009).

[3] ARM. 2012. ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0406c/index.html. (2012).

[4] ARM. 2015. ARM Cortex-A Series Programmerś Guide for ARMv8-A. http://infocenter.arm.com/help/topic/com.arm.doc.den0024a/index.html. (2015).

[5] ARM. 2016. SMC CALLING CONVENTION System Software on ARM Platforms. http://infocenter.arm.com/help/topic/com.arm.doc.den0028b/ARM_DEN0028B_SMC_Calling_Convention.pdf. (2016).

[6] Ahmed M. Azab, Peng Ning, and Emre C. Sezer. 2009. A hypervisorbased integrity measurement agent. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, Honolulu, Hawaii, 461–470.

[7] Ahmed M Azab, Peng Ning, Jitesh Shah, Quan Chen, Rohan Bhutkar, Guruprasad Ganesh, Jia Ma, and Wenbo Shen. 2014. Hypervision Across Worlds: Real-time Kernel Protection from the ARM TrustZone Secure World. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*. ACM, Scottsdale, Arizona, 90–102.

[8] Ellick Chan, Shivaram Venkataraman, Francis David, Amey Chaugule, and Roy Campbell. 2010. Forenscope: A framework for live forensics. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, Austin, Texas, 307–316.

[9] Francis M. David, Ellick M. Chan, Jefferty C. Carlyle, and Roy H. Campbell. 2008. Cloaker: Hardware supported rootkit concealment. In *Proceedings of the 29th IEEE Symposium on Security and Privacy (Oakland)*. IEEE, Oakland, CA, 296–310.

[10] Advanced Micro Devices. 2013. *Amd64 Architecture Programmer's Manual*. AMD.

[11] Shawn Embleton, Sherri Sparks, and Cliff Zou. 2013. Smm rootkit: a new breed of os independent malware. Security and Communication Networks. (2013).

[12] Norman Fenske. 2017. Genode Operating System Framework Foundations. (2017).

[13] John Heasman. 2007. Implementing and detecting a pci rootkit. BLackhat DC. (2007).

[14] Gene H. Kim and Eugene H. Spafford. 1994. The design and implementation of tripwire: A file system integrity checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security (CCS)*. ACM, Fairfax, VA, 18–29.

[15] Genode Labs. 2017. An Exploration of ARM TrustZone Technology. Genode OS Documentation and Articles. (2017). https://genode.org/documentation/articles/trustzone

[16] Genode Labs. 2017. Genode: Operating System Framework. https://github.com/genodelabs/genode. (2017).

[17] Genode Labs. 2017. An in-depth look into the ARM virtualization extensions. Genode OS Documentation and Articles. (2017). https://genode.org/documentation/articles/arm_virtualization

[18] Y. Liang, T. Mitra, and L. Ju. 2015. Instruction Cache Locking Using Temporal Reuse Profile. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 9 (Sept 2015), 1387–1400. https://doi.org/10.1109/TCAD.2015.2418320

[19] Tiantian Liu, Minming Li, and Chun Jason Xue. 2012. Instruction Cache Locking for Embedded Systems using Probability Profile. *Journal of Signal Processing Systems* 69, 2 (01 Nov 2012), 173–188.

[20] Y. Lu, L. Lo, G. R. Watson, and R. G. Minnich. 2006. CAR: Using Cache as RAM in LinuxBIOS. http://rere.qmqm.pl/âĹijmirq/cacheasramlb09142006.pdf. (2006).

[21] Nick L. Petroni, Timothy Fraser, Jesus Molina, and William A. Arbaugh. 2004. Copilot - a coprocessor-based kernel runtime integrity monitor. In *Proceedings of the 13th USENIX Security Symposium (Security)*. USENIX, San Diego, CA, 179–194.

[22] Joanna Rutkowska. 2006. Subverting vistatm kernel for fun and profit. Black Hat Briefings. (2006).

[23] Sherri Sparks and Jamie Butler. 2005. Shadow walker: Raising the bar for rootkit detection. *Black Hat Japan* (01 2005), 504–533.

[24] H. Sun, K. Sun, Y. Wang, J. Jing, and S. Jajodia. 2014. Trustdump: Reliable memory acquisition on smartphones. In *Proceedings of the 19th European Symposium on Research in Computer Security (ESORICS)*. Springer, Wroclaw, Poland, 202–218.

[25] Intel Trusted Execution Technology. 2016. *Intel Software Development Guide*. Intel.

[26] F. Zhang, J. Wang, K. Sun, and A. Stavrou. 2014. HyperCheck: A Hardware-AssistedIntegrity Monitor. *IEEE Transactions on Dependable and Secure Computing* 11, 4 (July 2014), 332–344.

[27] Ning Zhang, He Sun, Kun Sun, Wenjing Lou, and Y Thomas Hou. 2016. CacheKit: Evading memory introspection using cache incoherence. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy*. IEEE, Saarbrücken, GERMANY, 337–352.

[28] V. J. Zimmer, M. A. Rothman, and S. M. Datta. 2004. Using a processor cache as ram during platform initialization. US Patent 20,040,103,272.. (2004).