Research Frontiers for Moving Target Defenses

Nathan Burow
MIT Lincoln Laboratory
nathan.burow@ll.mit.edu

ABSTRACT

New software security threats are constantly arising, including new classes of attacks such as the recent spate of micro-architectural vulnerabilities, from side-channels and speculative execution to attacks like Rowhammer that alter the physical state of memory. At the same time, new defensive technologies are proposed and adopted, including advancements in programming languages and novel hardware architectures with a focus on security.

Moving target defenses were developed to provide performant, incremental security to programs written in unsafe languages running on processors designed solely for performance. Here we examine the challenges and opportunities for moving target defenses in the evolving security landscape, and in new applications domains such as cloud computing and real-time systems.

ACM Reference Format:
Nathan Burow. 2021. Research Frontiers for Moving Target Defenses. In *Proceedings of the 8th ACM Workshop on Moving Target Defense (MTD '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3474370.3485658

1 INTRODUCTION

The contours of moving target defenses, including their strengths and weaknesses, are well understood by the research community. Randomization prevents attackers from finding their target, but provides no security if an attacker can locate their target. Attacks on randomization include information leaks and even partial pointer overwrites that bypass randomization without requiring and information leak [3]. Moving target defenses are attractive because they raise the bar for attackers without imposing significant overhead on programs, compared to deterministic defenses that either only protect a subset of sensitive targets [1], or else come with prohibitive overhead [8].

New security threats [5, 6, 11] and counter measures [7, 14] are changing this well known calculus, however. The new micro-architectural attacks, targeting cache side channels, speculative execution, physical properties of memory, and other CPU specific features highlight the need for software to protect itself from the underlying hardware. This is a significant new attack vector, but one where moving target defenses have a significant role to play, if designed correctly [9, 12].

While moving target defenses have shown great promise for addressing micro-architectural attacks, recent progress on efficient memory safety for both new and legacy applications may render moving target techniques unnecessary for application security. The Rust programming language [10], for example, offers memory safety without an extensive language runtime, and is effectively as performant as C. The Cheri architecture [14], while still under development, offers the promise of efficient memory safety for legacy C applications. Whether randomization based defenses still offer value as a defense in depth for memory safe applications has not yet been studied in detail. As we discuss, the new languages and hardware are not perfect and do leave some research opportunities, but to our knowledge memory safety addresses all threats mitigated by randomization.

In our opinion, the most promising approach for future moving target research is to look beyond traditional enterprise computing applications to new domains, and new layers of the overall system architecture. Cloud applications and real-time / industrial control systems will both continue to increase in importance, and attractiveness to attackers.

2 RESEARCH FRONTIERS

The emergence of new threats, such as micro-architectural attacks, naturally spurs the development of novel defenses against them. For micro-architectural attacks, moving target defenses are promising until hardware level defenses can be developed and deployed. On the defensive side, new technologies with the promise to end the decades old "eternal" wars in memory [13], are finally seeing deployment. The impact of such technologies on moving target defenses is an important question for the community to answer to ensure that research efforts are correctly allocated.

Micro-Architectural Security. Micro-architectural vulnerabilities target several features of the underlying architecture, including speculative execution [5, 6], and hardware level vulnerabilities in memory design [11]. The speculative execution, and similar, vulnerabilities all rely on cache side channels. Brute force defense against them, such as disabling speculation, or inserting memory fences around sensitive operations, have proven to be impractical. One alternative is a moving target defense for caches – randomizing their layout to deny attackers the ability to dictate cache contents at specific locations to leak information. However, correctly designing such a defense is difficult as recent work as shown [9, 12]. Randomizing process use of hardware resources, including caches, to mitigate micro-architectural attacks will remain an active research area, particularly as we are witnessing rapid developments in new attack techniques.

Challenges. New software security techniques, such as memory safe systems programming languages that have C level performance, notably Rust [10], and hardware support for memory safety [14] present a challenge for moving target defenses. The great appeal of moving target defenses for applications is in raising the bar for

MTD '21, November 15, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8658-6/21/11.
https://doi.org/10.1145/3474370.3485658

attackers, while incurring minimal performance penalties. However – Rust makes systems programming in a memory safe language possible with good performance, addressing all threats mitigated by moving target defenses for applications to our knowledge. Rust is not a panacea, however, and does restrict programmer flexibility, and requires the `unsafe` keyword for some common constructs. `unsafe` Rust code requires the programmer to verify its memory safety, the compiler / runtime no longer do so. The Rust community is actively working to encapsulate as much `unsafe` code as possible in formally verified libraries [4], but this will not always be possible. Identifying the remaining gaps in the security model of Rust and other memory safe applications, and analyzing them to determine if moving target defenses are appropriate to fill them, is a key research task for the moving target community.

In addition to safe languages like Rust, recent advances in hardware security extensions have the promise of significantly improving application security. Intel's MPK, ARM PAC and MTE, and the Cheri [14] research architecture are all examples. Such tools are particularly important for securing legacy C applications, a key focus of the Cheri project. Similar to memory-safe programming languages, hardware extensions offer the promise of memory-safety without performance penalties, removing the use case for moving target defenses.

Given the twin developments in programming language and hardware security, what is the next "killer application" for moving target defenses? While there is certainly research to be done on safe languages like Rust, and micro-architectural attacks illustrate that hardware is never perfect, it is time to look for the next set of challenges for moving target defense work.

3 NEW DOMAINS

One promising way to find new research challenges is to look at applying high level concepts, *e.g.,* moving target defenses, to new domains where different design constraints and use cases necessity different trade-offs. Here we highlight two such domains: cloud computing and real-time systems.

Cloud. The importance of cloud services only continues to grow, and the cloud architecture presents new opportunities for moving target defenses. What VMs are co-resident with yours in the cloud effects your security. In the extreme case, it is possible to rent an entire node – though this defeats the flexible scaling of the cloud. In general, the cloud architecture of many VMs running on shared hardware, and co-resident with other, potentially untrusted or malicious, VMs creates new isolation challenges. Hypervisors attempt to address these, but do not always succeed. Consequently, there is a need for novel defenses generally in this space, including moving target defenses.

Real-Time. Real-time systems are increasingly networked, and so exposed to attack. Last year, we showed that moving target defenses are promising for real-time systems, *if their overhead can be made deterministic* [2]. Doing so will require a better understanding of the performance impacts of different code layouts, which is a significant research challenge in its own right. Questions include how architecture dependent the results are, and best practices for dynamically measuring worst case performance.

4 CONCLUSION

Moving target defenses remain an important tool for hardening software. They have proven well adapted to addressing novel micro-architectural attacks. However, the emergence of memory safe programming languages and increasing hardware support for memory safety has addressed the primary challenges for moving target defenses on individual applications, while creating comparatively few new opportunities. Consequently, it is time for the moving target defense community to look further afield to areas such as cloud computing and real-time systems for new challenges.

REFERENCES

[1] Martín Abadi, Mihai Budiu, Ulfar Erlingsson, and Jay Ligatti. 2009. 4Control-Flow Integrity Principles, Implementations, and Applications. *ACM Transactions on Information and System Security (TISSEC)* (2009).

[2] Nathan Burow, Ryan Burrow, Roger Khazan, Howard Shrobe, and Bryan C Ward. 2020. Moving Target Defense Considerations in Real-Time Safety-and Mission-Critical Systems. In *Proceedings of the 7th ACM Workshop on Moving Target Defense*. 81–89.

[3] Enes Göktaş, Benjamin Kollenda, Philipp Koppe, Erik Bosman, Georgios Portokalidis, Thorsten Holz, Herbert Bos, and Cristiano Giuffrida. 2018. Position-independent Code Reuse: On the Effectiveness of ASLR in the Absence of Information Disclosure. In *EuroS&P*. Paper=https://download.vusec.net/papers/pirop_eurosp18.pdfWeb=https://www.vusec.net/projects/pirop

[4] Ralf Jung, Jacques-Henri Jourdan, Robbert Krebbers, and Derek Dreyer. 2017. RustBelt: Securing the Foundations of the Rust Programming Language. *Proceedings of the ACM on Programming Languages (POPL)* (2017).

[5] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2019. Spectre Attacks: Exploiting Speculative Execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*.

[6] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium (USENIX Security 18)*.

[7] Nicholas D Matsakis and Felix S Klock. 2014. The rust language. *ACM SIGAda Ada Letters* 34, 3 (2014), 103–104.

[8] Santosh Nagarakatte, Jianzhou Zhao, Milo MK Martin, and Steve Zdancewic. 2009. SoftBound: Highly Compatible and Complete Spatial Memory Safety for C. In *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*.

[9] Antoon Purnal, Lukas Giner, Daniel Gruss, and Ingrid Verbauwhede. 2021. Systematic analysis of randomization-based protected cache architectures. In *42th IEEE Symposium on Security and Privacy*, Vol. 5.

[10] Rust Foundation. [n.d.]. What is Ownership? - The Rust Programming Language. https://doc.rust-lang.org/book/ch04-01-what-is-ownership.html. Accessed on 2021-05-14.

[11] Mark Seaborn and Thomas Dullien. 2015. Exploiting the DRAM rowhammer bug to gain kernel privileges. *Black Hat* 15 (2015), 71.

[12] Wei Song, Boya Li, Zihan Xue, Zhenzhen Li, Wenhao Wang, and Peng Liu. 2021. Randomized last-level caches are still vulnerable to cache side-channel attacks! But we can fix it. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 955–969.

[13] Laszlo Szekeres, Mathias Payer, Tao Wei, and Dawn Song. 2013. SoK: Eternal War in Memory. In *2013 IEEE Symposium on Security and Privacy*.

[14] Jonathan Woodruff, Robert NM Watson, David Chisnall, Simon W Moore, Jonathan Anderson, Brooks Davis, Ben Laurie, Peter G Neumann, Robert Norton, and Michael Roe. 2014. The CHERI capability model: Revisiting RISC in an age of risk. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*.