Poster: Automatic Identification and Protection of Memory-resident Sensitive Data to Defend Against Data-Oriented Attacks

Salman Ahmed
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
ahmedms@vt.edu

Hans Liljestrand, N. Asokan
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
hans@liljestrand.dev, asokan@acm.org

Danfeng (Daphne) Yao
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
danfeng@vt.edu

***Abstract*—Software and hardware-based countermeasures for protecting memory-resident data and its pointers to prevent data-oriented attacks suffer from high performance overhead due to a large number of memory data objects and their pointers. In this ongoing work, we propose a framework utilizing rule-based heuristics to identify sensitive memory data and its pointers automatically from an application and protect those sensitive data and pointers utilizing existing countermeasures. Our evaluation suggests that an application needs to protect less than 30% of its total data and pointers. Besides, our preliminary result shows that this prioritized protection reduces the performance overhead of existing countermeasures by 50%.**

With the advances toward practical code pointer protection countermeasures and practical Control-Flow Integrity (CFI), we anticipate a shift towards the manipulation of memory-resident sensitive data or pointers as the attack vector as this manipulation works in the presence of Code Pointer Integrity (CPI) protections and CFI countermeasures. This is why in recent years, we observed a momentum in data-oriented attacks (DOAs), also known as non-control attacks [11]–[13], [15], [21], [22], [26] even though DOAs were introduced more than a decade ago [7].

The manipulation of memory-resident sensitive data or their pointers has become an appealing attack technique for data-oriented attacks as they conform to CFG and do not violate CFI. Ideally, DOAs [7], [11], [13] can modify all kinds of memory data to change program behavior for leaking sensitive information [4] or performing privilege escalations [9]. But the corruption of data pointers [8] is often desirable. For example, the manipulation of data pointers can lead to the leak of critical information about an application's address space layout [10], [23], gadget stitching in DOP-based attacks [12], stack-based exploitations [7], and heap-based exploitations [24].

Researchers have proposed both software and hardware-based countermeasures to stop attackers from manipulating memory-resident data or their pointers. However, software-based countermeasures such as Data-Flow-Integrity (DFI) [6], Data Space Randomization (DSR) [2], [5], [20], and memory tagging [16], [17] usually suffer from performance overhead (48-116% [16], [17]) due to using static analysis, inter-procedural DFI, encryption, and masking. On the other hand, hardware-based countermeasures (e.g., HDFI [25], Intel's CET, ARM Pointer Authentication (PA), and MPX) are efficient, but in general, limited to one or a few platforms. Furthermore, the overhead is non-negligible. For example, ARM pointer authentication and Intel's MPX cost on average around 19.5% [14] and 50% overhead, respectively, for protecting data pointers.

The key reason for this overhead to protect the memory-resident data and their pointers is that the number of such data or their pointers is huge compared to the code pointers. One solution for reducing this overhead is the identification of memory-resident sensitive data and their pointers and protecting the sensitive data and their pointers, rather than protecting all data/pointers.

The idea of protecting sensitive or critical data for preventing data-oriented attacks is not new. Researchers have designed mechanisms [11], [18], [19] to protect sensitive memory data by manually-labeled or predefined data. In this ongoing work, we aim to complement existing work by identifying sensitive data and its pointers automatically and protecting them to prevent data-oriented attacks. A few existing automated techniques [13], [15] also can determine the critical data. For example, Jia et al. [13] determined the decision-making data by recording the execution of two traces with normal execution and violated execution and observing the data that get modified and changed executions. Access-driven trace data [15] are also useful to determine and understand the critical data and their structures. However, these works are not scalable as we need huge and relevant execution and access traces.

In this ongoing work, we aim to develop a generic framework for the automatic identification of memory-resident sensitive data and their pointers for protection using existing defense mechanisms. Figure 1 shows the high-level overview of the framework. The identification process starts with attack entry points usually the external inputs from the network, file system, and keyboard. A precise taint analysis with the help of points-to analysis helps track the data flow of sensitive memory data and pointers. The generic nature of the framework enables the protection of the identified sensitive data and pointers with existing defenses such as ARM Pointer Authentication, Intel MPX, AddressSantizer, and memory tagging solutions [16], [17].

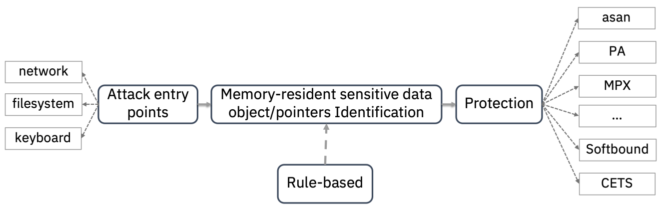


Fig. 1. High-level overview of the data object prioritization technique.

This framework can prevent attackers from modifying the integrity of sensitive memory-resident data or its pointers even with arbitrary memory read/write capabilities. The capability of identifying sensitive memory data prevents data-oriented attacks by making the existing defense mechanisms practical. Our framework utilizes rule-based heuristics to detect sensitive data objects/pointers using knowledge from advanced data-oriented exploits [7], [10], [11], [23] and vulnerabilities (CVE-2001-0820, CVE-2006-5815, CVE-2006-5815, CVE-2017-9430, CVE-2018-6151, CVE-2018-10111, CVE-2021-23017, etc.).

Our preliminary evaluation using manually constructed ground truths of vulnerable data objects/pointers by identifying vulnerable data objects/pointers from vulnerable datasets [1], [3] including 5 real-world applications shows that less than 30% of the data objects and their pointers are sensitive. Thus, in our testing environment, protecting less than 30% of total memory-resident sensitive data or their pointers is sufficient to protect the tested applications from data-orient attacks. Besides, the rule-based identification of sensitive memory data and pointers can lead to almost 50% performance improvements in existing defenses in our tested environments.

References

[1] The defense advanced research projects agency (darpa). https://github.com/CyberGrandChallenge/samples. Accessed Feb 12, 2020.

[2] Sandeep Bhatkar and R. Sekar. Data space randomization. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 1–22. Springer, 2008.

[3] Paul E. Black. A software assurance reference dataset: Thousands of programs with known bugs. *Journal of Research of the National Institute of Standards and Technology*, 123:123005, April 2018.

[4] Heartbleed Bug. http://heartbleed.com, 2020. Accessed April 03, 2020.

[5] Cristian Cadar, Periklis Akritidis, Manuel Costa, Jean-Phillipe Martin, and Miguel Castro. Data randomization. Technical report, Technical Report TR-2008-120, Microsoft Research, 2008, 2008.

[6] Miguel Castro, Manuel Costa, and Tim Harris. Securing software by enforcing data-flow integrity. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 147–160. USENIX Association, 2006.

[7] Shuo Chen, Jun Xu, Emre Can Sezer, Prachi Gauriar, and Ravishankar K Iyer. Non-control-data attacks are realistic threats. In *USENIX Security Symposium*, volume 5, 2005.

[8] Crispin Cowan, Steve Beattie, John Johansen, and Perry Wagle. PointGuardTM: Protecting Pointers From Buffer Overflow Vulnerabilities. In *Proceedings of the 12th conference on USENIX Security Symposium*, volume 12, pages 91–104, 2003.

[9] Daniel Moghimi. Subverting without EIP. https://moghimi.org/blog/subverting-without-eip.html, 2014. Last accessed 6 January 2021.

[10] Isaac Evans, Sam Fingeret, Julian Gonzalez, Ulziibayar Otgonbaatar, Tiffany Tang, Howard Shrobe, Stelios Sidiroglou-Douskos, Martin Rinard, and Hamed Okhravi. Missing the point (er): On the effectiveness of code pointer integrity. In *2015 IEEE Symposium on Security and Privacy*, pages 781–796. IEEE, 2015.

[11] Hong Hu, Zheng Leong Chua, Sendroiu Adrian, Prateek Saxena, and Zhenkai Liang. Automatic generation of data-oriented exploits. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 177–192, 2015.

[12] Hong Hu, Shweta Shinde, Sendroiu Adrian, Zheng Leong Chua, Prateek Saxena, and Zhenkai Liang. Data-oriented programming: On the expressiveness of non-control data attacks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 969–986. IEEE, 2016.

[13] Yaoqi Jia, Zheng Leong Chua, Hong Hu, Shuo Chen, Prateek Saxena, and Zhenkai Liang. "the web/local" boundary is fuzzy: A security study of chrome's process-based sandboxing. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 791–804, 2016.

[14] Hans Liljestrand, Thomas Nyman, Kui Wang, Carlos Chinea Perez, Jan-Erik Ekberg, and N Asokan. {PAC} it up: Towards pointer integrity using {ARM} pointer authentication. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 177–194, 2019.

[15] Micah Morton, Jan Werner, Panagiotis Kintis, Kevin Snow, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose. Security risks in asynchronous web servers: When performance optimizations amplify the impact of data-oriented attacks. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 167–182. IEEE, 2018.

[16] Santosh Nagarakatte, Jianzhou Zhao, Milo MK Martin, and Steve Zdancewic. Softbound: Highly compatible and complete spatial memory safety for c. In *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 245–258, 2009.

[17] Santosh Nagarakatte, Jianzhou Zhao, Milo MK Martin, and Steve Zdancewic. Cets: compiler enforced temporal safety for c. In *Proceedings of the 2010 international symposium on Memory management*, pages 31–40, 2010.

[18] Tapti Palit, Jarin Firose Moon, Fabian Monrose, and Michalis Polychronakis. DynPTA: Combining static and dynamic analysis for practical selective data protection. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1919–1937, May 2021.

[19] Tapti Palit, Fabian Monrose, and Michalis Polychronakis. Mitigating data leakage by protecting memory-resident sensitive data. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 598–611, 2019.

[20] Prabhu Rajasekaran, Stephen Crane, David Gens, Yeoul Na, Stijn Volckaert, and Michael Franz. Codarr: Continuous data space randomization against data-only attacks. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 494–505, 2020.

[21] Roman Rogowski, Micah Morton, Forrest Li, Fabian Monrose, Kevin Z. Snow, and Michalis Polychronakis. Revisiting browser security in the modern era: New data-only attacks and defenses. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 366–381. IEEE, 2017.

[22] Cole Schlesinger, Karthik Pattabiraman, Nikhil Swamy, David Walker, and Benjamin Zorn. Modular protections against non-control data attacks. *Journal of Computer Security*, 22(5):699–742, 2014.

[23] Jeff Seibert, Hamed Okhravi, and Eric Söderström. Information leaks without memory disclosures: Remote side channel attacks on diversified code. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 54–65, 2014.

[24] Shellphish. Educational Heap Exploitation: how2heap . https://github.com/shellphish/how2heap, 2019. Last accessed 6 January 2021.

[25] Chengyu Song, Hyungon Moon, Monjur Alam, Insu Yun, Byoungyoung Lee, Taesoo Kim, Wenke Lee, and Yunheung Paek. Hdfi: Hardware-assisted data-flow isolation. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 1–17. IEEE, 2016.

[26] Laszlo Szekeres, Mathias Payer, Tao Wei, and Dawn Song. Sok: Eternal war in memory. In *2013 IEEE Symposium on Security and Privacy*, pages 48–62. IEEE, 2013.

POSTER: Automatic Identification and Protection of Memory-resident Sensitive Data to Defend Against Data-Oriented Attacks

Salman Ahmed[1], Hans Liljestrand[2], N. Asokan[2], Danfeng (Daphne) Yao[1]
[1]Compute Science, Virginia Tech, [2]David R. Cheriton School of Computer Science, University of Waterloo
ahmedms@vt.edu, hans@liljestrand.dev, asokan@acm.org, danfeng@vt.edu

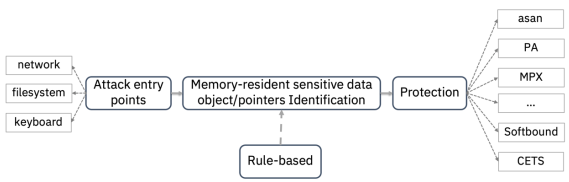
1. Motivation

Memory-resident data and its pointers need protection to defend against data-oriented attacks.

However, the protection incurs huge **performance overhead** due to a large number of memory data and pointers.

2. Approach

- ❑ **Rule-based heuristics** to detect sensitive data objects/pointers.
- ❑ Rules are based on knowledge of existing data-oriented exploits and vulnerabilities.
- ❑ **Generic** nature of the framework enables integration with existing defenses.



Overview of rule-based sensitive data and pointer detection framework

3. Challenges

How to confirm the **coverage of rules**

How to construct **ground truths** for evaluation

4. Experimental Design

- ❑ We **solve the coverage issue** by breaking down the knowledge from exploits and vulnerabilities into smaller and generic rules.
- ❑ The generic rules are applicable for wide-range of exploits and vulnerabilities, both known and unknown.
- ❑ We **manually construct ground truths** by analyzing memory-resident data and pointers from **27** programs.

DARPA Cyber Grand Challenge [1] and Software Assurance Reference [2] datasets with five real-world programs

Less than 30% of the data and their pointers are sensitive. Thus, **only this 30% need protection.**

Performance overhead is reduced by almost **50%.**

5. Conclusion

We presented a framework for identifying memory-resident sensitive data and their pointers automatically. Our experiments show that on average 30% of memory-resident data and pointers are sensitive, hence requiring protection. We can improve the performance overhead of existing defenses by 50% when protecting these sensitive data and pointers.

[1]. The Defense Advanced Research Projects Agency (DARPA) dataset. https://github.com/CyberGrandChallenge/samples. Accessed Feb 12, 2020.
[2]. Paul E. Black. A software assurance reference dataset: Thousands of programs with known bugs. Journal of Research of the National Institute of Standards and Technology, 123:123005, April 2018.