Article for
Encyclopedia of Cryptography, Security and Privacy

Moving Target Defense

Hamed Okhravi
MIT Lincoln Laboratory
hamed.okhravi@ll.mit.edu

Synonyms

Moving target techniques, maneuvering, agility, dynamic defenses

Definition

Moving target defenses (MTD) are techniques that seek to enhance the resiliency (cross-ref: Cyber Resilience) of a computer system by making it more dynamic, random, and/or diverse.

Background

The static nature of computer systems has traditionally contributed to an 'economy of scale' for cyber attackers. Attackers can craft an exploit against a copy of a computer system (comprising hardware and various software layers) and because numerous instances of the system are internally alike, they can compromise hundreds to millions of machines at once. This problem is exemplified in modern malware that often impacts millions of machines and in recent vulnerabilities such as Log4Shell (Wortley et al. 2021) that impact millions of machines.

Techniques that fit within the scope of MTDs have long been studied in computer science. Some of the early examples of what are now called MTDs were used in the Apollo program for reliability reasons. The Apollo guidance computer system was called a Triple Modular Redundant (TMR) computer in which an equation was solved on three different circuits simultaneously and the results were compared to achieve fault tolerance (Lyons and Vanderkulk 1962) (cross-ref: Asynchronous Byzantine Fault Tolerance). This was an early example of diversity. Other data diversity approaches were studied in late 80s (Ammann and Knight 1988). Dynamic techniques for resiliency were also studied in early 2000s.

The term *moving target defenses* gained widespread adoption around 2010 and the Networking and Information Technology Research and Development (NITRD) offered one of its early definitions: increasing the complexity of cyber attacks by making systems less homogeneous, less deterministic, and less static (NITRD 2010).

The focus of this article is particularly on techniques against malicious cyber-attacks, and not those that target benign faults.

Theory

MTDs are defined in terms of three properties of making a system more dynamic, random, and/or diverse (Okhravi et al. 2013). Dynamism refers to changing the properties or internals of a system over time. Randomness refers to making an aspect of a system less predictable. Diversity refers to having different copies of a system with different internal properties.

An MTD may have one or more of these properties. For example, a technique that periodically replaces web servers running in a virtual machine (VM) with fresh copies of the VM to remove attacker's foothold

This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

Springer

is an example of dynamism. Randomizing the location of loaded libraries in the memory space of an application, on the other hand, is an example of randomization. Periodically replacing VMs with copies that are further scrambled internally provides both dynamism and randomness.

Based on their characteristics and implementation, MTDs can be categorized into five large domains: dynamic data, dynamic software, dynamic runtime environment, dynamic platform, and dynamic network. Dynamic data techniques change the syntax, representation, or format of data being processed by the system. Data randomization (Cadar et al. 2008) is an example of a dynamic data technique that masks all data values in an application using a random key to prevent their malicious modification. Dynamic software techniques randomize or diversify the application code. Multicompiler (Franz 2010) is an example of a dynamic software technique that inserts random number of no-operations (NOPs) into the application binary at compile time to enhance its resilience against control hijacking exploits. Dynamic runtime environment techniques change the system interface presented to the running application. Address space layout randomization (ASLR) is an example of a dynamic runtime environment technique that loads libraries at random offsets in memory at load time to resist code injection or code reuse attacks. Dynamic platform techniques change the properties of the underlying platform comprising the operating system, the hypervisor, and the hardware of the system. Self Cleansing Intrusion Tolerance (SCIT) (Bangalore and Sood 2009) is an example of a dynamic platform technique that periodically rotates server VMs with fresh copies to remove attackers' foothold on the server. Lastly, dynamic network techniques change the connectivity or addressing of machines in a network. PSI (Yu et al. 2017) is an example of a dynamic network technique that dynamically adjusts the connectivity of machines in a network based on security-related attributes and events.

Figure 1 illustrates the different domains of MTDs. Related surveys (Ward et al. 2018, Jajodia et al. 2012) provide a more extensive list of MTDs and their properties.

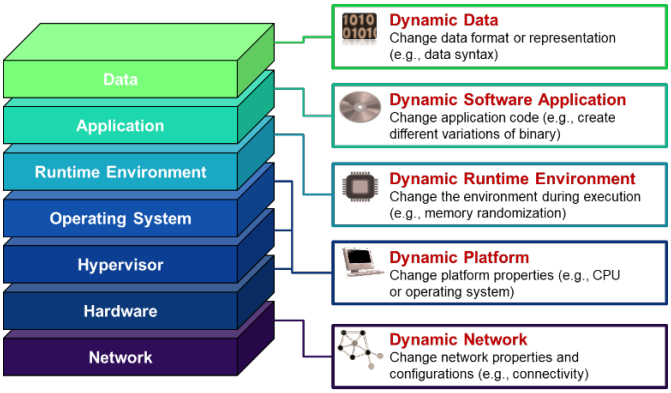

Figure 1: Moving target defense domains laid out against a typical compute stack.

The resiliency offered by MTDs can be evaluated using various approaches. The threat model is a crucial factor when evaluating the protection offered by an MTD. If an attacker has to enumerate all possible states in order to bypass an MTD, measures such as the amount of randomness (i.e., entropy) can offer some insight into the difficulty of an attack.

A more practical attack against MTDs, however, is information leakage attacks through which an attacker learns about how the system is randomized. For example, memory disclosure attacks (Strackx et al. 2009) or remote side-channels (Seibert et al. 2014) are used to learn how the system is randomized. They can be usedto bypass dynamic runtime environment techniques such as ASLR or dynamic software techniques such as the multicompiler.

This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

Leakage-resilient MTDs such as re-randomization techniques (Bigelow et al. 2015) and non-readable codes (Crane et al. 2015) have been studied as a way of mitigating the impact of information leakage attacks.

Application

MTDs have gained varying degrees of adoption in practice. Perhaps the most widely used MTD to date is ASLR that has been running on most major desktop and mobile operating systems (including Windows, Linux, Mac OS, iOS, and Android).

The Windows operating system also deploys heap metadata randomization by XORing it with a random value, a form of dynamic data technique.

Dynamic networks in the form of dynamic adjustment of connectivity has also been used in the form of network access control (cross-ref: Access Control Policies, Models, and Mechanisms) in various enterprises.

Other MTDs such as dynamic software techniques and dynamic platform technique have gained some adoption, albeit in more limited use cases.

Open problems and Future directions

Several open problems and practical challenges have impeded broader adoption of MTDs (Larsen and Franz 2020). These problems form the basis for future work in this area.

Despite the development of leakage-resilient techniques, information leakage attacks still pose a threat to many forms of MTDs (cross-ref: Leakage-resilience). Practical and comprehensive protections against information leakage attacks remain an open problem (Ward et al. 2019).

Some randomization/dynamism techniques impose a performance overhead, while diversification techniques often impose additional hardware overhead. Even when these overheads are small, they can be impractical/undesirable for resource-limited platforms such as embedded systems. This is particularly important in areas where MTDs compete with deterministic defenses such as control flow integrity (CFI) (Larsen and Franz 2020). Development of extremely low-overhead MTDs particularly for resource-limited systems remain an open problem.

Some MTDs also do not interoperate well with other defenses in a system or with some widely used system features. Among them, techniques that randomize the application binary on-disk pose a challenge to code signing defenses that are widely used in iOS devices and runtime randomization techniques also cannot seamlessly handle just-in-time compilation or dynamic loading of libraries. For MTDs to gain broader adoption these gaps need to be addressed.

Some forms of MTDs also pose a challenge to typical enterprise maintenance and operations tasks. For example, dynamic adjustment of connectivity in a network may impede network visibility for network operators, while code randomization techniques can complicate debugging.

Furthermore, additional research is needed to study and design proper MTDs for advanced microarchitectural and hardware vulnerabilities such as Spectre (cross-ref: Spectre), Meltdown (cross-ref: Meltdown), Foreshadow, Rowhammer (cross-ref: Rowhammer), and the like.

Cross-References

Access Control Policies, Models, and Mechanisms
Asynchronous Byzantine Fault Tolerance
Cyber Resilience

This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

Leakage-resilience
Meltdown
Rowhammer
Spectre

References

Ammann PE, Knight JC (1988) Data diversity: An approach to software fault tolerance, IEEE Transactions on Computers, Apr, 37(4):418-25

Bangalore AK, Sood AK (2009) Securing web servers using self cleansing intrusion tolerance (SCIT). In Second International Conference on Dependability, Jun 18, pp. 60-65

Bigelow D, Hobson T, Rudd R, Streilein W, Okhravi H (2015) Timely rerandomization for mitigating memory disclosures. In ACM CCS, Oct 12, pp. 268-279

Cadar C, Akritidis P, Costa M, Martin JP, Castro M (2008) Data randomization. Technical Report TR-2008-120, Microsoft Research

Crane S, Liebchen C, Homescu A, Davi L, Larsen P, Sadeghi AR, Brunthaler S, Franz M (2015) Readactor: Practical code randomization resilient to memory disclosure, In IEEE Symposium on Security and Privacy, May 17, pp. 763-780

Franz M. (2010) E unibus pluram: massive-scale software diversity as a defense mechanism. In Proceedings of the New Security Paradigms Workshop, Sep 21, pp. 7-16

Jajodia S, Ghosh AK, Subrahmanian VS, Swarup V, Wang C, Wang XS (2012) Moving Target Defense II: Application of Game Theory and Adversarial Modeling, Springer Science & Business Media, Sep 18

Larsen P, Franz M (2020) Adoption Challenges of Code Randomization. In ACM Moving Target Defense Workshop, Nov 9, pp. 45-49

Lyons RE, Vanderkulk W (1962) The use of triple-modular redundancy to improve computer reliability, IBM journal of research and development, Apr, 6(2):200-9

NITRD (2010) Cybersecurity Game-Change Research & Development Recommendations, https://www.nitrd.gov/pubs/CSIA_IWG_%20Cybersecurity_%20GameChange_RD_%20Recommendations_20100513.pdf

Okhravi H, Hobson T, Bigelow D, Streilein W. (2013) Finding focus in the blur of moving-target techniques, IEEE Security & Privacy, Nov 22, 12(2), pp. 16-26

Seibert J, Okhravi H, Söderström E. (2014) Information leaks without memory disclosures: Remote side channel attacks on diversified code. In ACM CCS, Nov 3, pp. 54-65

Strackx R, Younan Y, Philippaerts P, Piessens F, Lachmund S, Walter T (2009) Breaking the memory secrecy assumption, In Proceedings of the Second European Workshop on System Security, Mar 31, pp. 1-8

Ward B, Skowyra R, Spensky C, Martin J, Okhravi H (2019) The leakage-resilience dilemma, In ESORICS, Sep 23, pp. 87-106

Ward B, Gomez SR, Skowyra R, Bigelow D, Martin J, Landry J, Okhravi H (2018) Survey of cyber moving targets second edition, MIT Lincoln Laboratory Report TR-1228

Wortley F, Thrompson C, Allison F (2021) Log4Shell: RCE 0-day exploit found in log4j 2, a popular Java logging package, LunaSec, https://www.lunasec.io/docs/blog/log4j-zero-day/

Yu T, Fayaz SK, Collins MP, Sekar V, Seshan S. (2017) PSI: Precise Security Instrumentation for Enterprise Networks. In NDSS, Feb

This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.