Moving Target Defense: "Staying Ahead of the Enemy"

Don Maclean
ICIT Fellow and Chief Cyber Security Technologist, DLT

In cyberspace, the game is rigged; a tiny group of elite "commandos" can easily inflict major damage on big, heavily defended targets. Their advantage has two major causes: system standardization, and a defensive trap I call the Maginot Mentality. This paper will be primarily focused on the second, while touching briefly upon the first.

In general, standardization is good, both for cybersecurity and general system administration. It facilitates deployment, patching, incident response, and other aspects of system administration. However, when a bad actor finds a vulnerability in a standard system, the exploit works on *every instance* of that system, from applications to operating systems.

> There are several important lessons from this episode in history.
>
> 1. Trying to out-innovate the enemy is unlikely to work, especially over the long term.
> 2. A static approach allows the enemy time to study your defenses and find its weaknesses.
> 3. Expensive does not equal secure.
>
> \- Don Maclean

After WWI, the French were understandably worried about invasion from the east, so they built the Maginot Line: a huge, very expensive, set of defensive fortifications using state-of-the-art technology. (To this day, the Maginot Line still stands in many places, and tourists visit it regularly). The Line did little to stop the invaders, despite its ingenuity and enormous expense. When it was built in the 1930s, the Maginot Line was new and impressive. However, by the time it was attacked in 1940, Germany had developed more advanced technology, airplanes, which were able to easily bypass the fortifications. Additionally, the Maginot Line was static, allowing enemies to study it at their leisure. The defenders' efforts to update it were always a step behind the attackers' new ideas. Sound familiar?

Worse, the Maginot Line's impressive size and innovation created a false sense of security that left the French completely flat-footed when the Germans attacked. They had placed too much faith in technology and failed to appreciate that humans were an essential element of defense. Sound familiar?

There are several important lessons from this episode in history. First, trying to out-innovate the enemy is unlikely to work, especially over the long term. Second, a static approach allows the enemy time to study your defenses and find its weaknesses. Third, expensive does not equal secure. In fact, overspending on seemingly impregnable defenses can deplete resources unnecessarily, divert funds from more effective measures, and create an unwarranted sense of complacency. The Maginot Mentality did not work on the Germans during WWII, and it does not work in cybersecurity now.

Researchers and practitioners of cybersecurity have recognized this problem, leading to the development of moving target defense (MTD). The MTD mentality prizes agility over impregnability and seeks to avoid the security problems of standardization, a concept I would have considered an anathema not so long ago. MTD also seeks to eliminate the attacker's economic advantage by ensuring that if a bad actor compromises one system, the resources they have expended will not apply to the next. Instead, attackers must continually re-create the wheel to attack multiple systems and each exploit has no value on the black market since it is not generically usable.

"The MTD mentality prizes agility over impregnability and seeks to avoid the security problems of standardization."
- Don Maclean

This paper will look at the most recent developments in the arena of moving target defense, ranging from approaches that exist only in theory to commercially available products. The goals are to:

- Define key terms and provide a broad taxonomy of MTD
- Specify a set of evaluation criteria to assess current MTD technologies
- Enumerate and assess MTD technologies using these criteria

MTD and the Attack Surface

Since MTD focuses on the attack surface, it makes sense to delve into that concept. Key questions are how to define an attack surface, how to measure it, deciding *what* to move, *when* to move it, and *how* to move it.

Defining the Attack Surface

In academia, there is much discussion about the definition of an attack surface, resulting in numerous papers and theses on the topic. Most definitions focus on the defender's view of attack surface, namely, “what is my exposure?”, and “where am I vulnerable?” However, the attacker focuses on how many systems have the weakness and what the economy of scale is for creating an exploit. In other words, defenders see the attack surface on their own systems while attackers see an attack surface throughout the world. Thus, we can informally define an attack surface as the collection of systems on which an attacker’s exploit is effective.

Moving the Attack Surface

Sengupta et.al [1] outlines three broad criteria for MTD: *what* to move, *when* to move it, and *how* to move it.

1. What to Move
   - Data [3]
   - Memory (code, data, and flow-control mechanisms such as pointers) [3]
   - Applications [3]
   - Dynamic Runtime Environments [3]
   - Networks and Platforms [3]
   - Instruction sets [2]
   - Address space layouts [2]
   - IP addresses, port numbers, proxies [2]
   - Virtual machines [2]
   - Flow-control mechanisms such as pointers, stack and/or heap addresses
   - Keywords and tokens

2. How to Move
   A viable MTD technology must be [1]:
   - Highly agile
   - Readily automated
   - Easily disguised, so the move operation is no apparent to attackers or the system’s users.

3. When to Move
   Options for moving the attack surface should include [1]:
   - On demand
   - On a predetermined schedule
   - In response to an attack or attack predecessors, either automatically or manually

MTD Implementation

MTD technologies come in three basic flavors. The first is purely theoretical, with little or no code available to the public. The second are research projects with open-source code, while the last is commercial products. Between these categories, the performance impact of MTD technologies varies widely, from almost zero to very significant, especially for early-stage research projects yet to undergo optimization. Performance impact is a key criterion for any technology, and MTD is no exception.

In general, the cybersecurity field suffers from a lack of trained personnel, and the government is especially concerned about this issue. Consequently, it is essential to consider the level of expertise required for successful implementation of a given MTD technology.

MTD Technology Overview

MTD technologies and research falls into four broad categories: methods for moving data, software diversity, network technologies, and dynamic runtime environments. I will examine each of these in turn, moving progressively toward technologies with either theoretical promise or practical applicability.

Moving Data Using Data-Oriented Methods

Much research on data diversity is in progress, but little is available to the public. Unfortunately, most of the available research demonstrates methods with a significant performance hit, usable only in very specific environments, or are almost entirely theoretical in nature. Consequently, I will not address this area in depth. However, interested readers can delve into these technologies, via the references provided:

- Data Diversity Through Fault Tolerance [4]
- Redundant Data Diversity [5]
- Data Randomization [6]
- Diglossia [7]
- NOMAD [8]
- HERMES [9]
- Content Randomization of Microsoft Office Documents [10]

MTD Software Diversity

This area of research is much more promising, and closer to practical fruition, than efforts in data diversity. Readers interested in more depth would benefit from a close reading of Ward et al [3]. The following are some of the ideas currently being studied.

- *Moving Attack Surfaces (MAS) for Web Services* [11]
  The concept behind MAS is to create a group of servers with different configurations but identical or equivalent functions. Servers rotate in and out of service on a schedule, or in response to an event such as an intrusion or warning of an attack. Moreover, the choice of servers is random, creating as much confusion as possible for the attacker.

  MAS forces the attacker to expend more resources to achieve an intrusion and reduces the dwell time and consequent benefit of a successful breach. However, it also increases the defenders' burden, as they must determine and test multiple configurations, buy and maintain software from multiple vendors, and bear the risk of systems that might perform poorly or fail.

- *ChameleonSoft* [12]
  This interesting, but highly theoretical, approach aims to encrypt software *behavior*. It is far from practical at this point, but interested readers are referred to Azab and Eltoweissy.

- *Web Application Diversity* [13]
  Attackers frequently exploit weaknesses in high-level languages such as PHP and SQL. Researchers therefore try to complicate the attacker's task through automated translation of common languages, such as Python to PHP, or by using multiple dialects of SQL simultaneously. The concept may be sound, but implementation has thus far been impractical, given that this resource-intensive strategy requires two implementations of each web application.

- *Security Agility for Dynamic Execution Environments* [14]
  In this extremely interesting approach, software is used to synchronize security policies with applications, enabling dynamic responses to changes in policies and attempted intrusions. When implemented, security policies would automatically be modified in response to attempted intrusions. Unfortunately, this method requires a centralized system, called a policy controller, which also constitutes a central point of failure. Moreover, automated responses to intrusions requires accurate, real-time identification of attack behavior, which is an ongoing problem in cybersecurity.

MTD Dynamic Networking and Platforms

The software diversity approaches are promising, though currently less practical than the runtime environment approaches discussed below.

- *n*-variant systems [20]

Resembling MAS, *n*-variant systems automatically create and execute multiple versions of the same program. They validate equivalence by determining if a given input yields equal or equivalent output in all variants, protecting against malicious code injection and manipulation of flow-control mechanisms. The authors' performance tests [20, pp. 12] appear very promising, as they indicate a low performance penalty.

- Trusted Dynamic Logical Heterogeneity System (TALENT) [21]
  TALENT exploits the agility of containers and uses a portable checkpoint compiler to migrate applications across platforms in about one second, according to the authors. This approach literally moves an application from one platform to another, with the intent of keeping a step ahead of the adversary.

MTD Dynamic Runtime Environment Technologies

In my view, dynamic runtime environment technologies are the most promising and practical of the current crop of MTD solutions. The technologies below are available now as either open source projects or commercial products.

- *Address Space Layout Permutation* (ASLP) [15]
  To keep attackers guessing, ASLP randomly relocates code segments, data segments, the stack, the heap, and other memory regions, to keep the attacker guessing. The authors report minimal performance impact, confirmed by Ward et al. [3]. However, this approach entails an increase of about 20% in file size and memory footprint [3, pp. 73]. ALSP, along with similar techniques like polymorphic operating systems, could potentially mitigate entire classes of memory-based attacks.

- *Function-Pointer Encryption* [16]
  This method protects against function pointer overwrites by encrypting each function pointer with an almost trivial function: *fp XOR *random number*. Ward et al. [3, pp. 87] tested this system, finding a performance penalty of 4% and a "likely small" impact on memory consumption. However, this technology addresses buffer overflows, a chronic problem in cybersecurity.

- *In-Place Code Randomization* [17]
  This approach is designed to counter return-oriented programming (ROP) attacks, which use legitimate code for illegitimate purposes. This complex technique blends multiple methods at the machine code level, but the researchers claim that it thwarts all known

ROP techniques. Independent testing by Ward et al. [3, pp. 101] showed no memory overhead, and a “negligible” performance impact.

- *Morphisec*
  Morphisec is a commercially available product that randomizes memory and retains a copy of non-randomized memory for troubleshooting and exploit identification. Interested readers are referred to www.morphisec.com.

- *Dynaguard* [18]
  Dynaguard is a technique which hinders Blind Return-Oriented Programming (BROP) attacks. A BROP attack tries to evade detection by using the same a so-called “stack canary which is a random number, placed on the stack to warn of a potential buffer overflow of the stack that local parent and child processes use. In doing so, a BROP attack can reveal the canary to bypass various protections. Dynaguard alters the canary value of the child process [18, pp. 1], thereby fooling the attacker.

  Dynaguard has minimal impact on performance and uses very little memory. However, it is specific only to one type of attack, and simply continues the endless “cat-and-mouse” game, albeit quite brilliantly, between attackers and defenders.

- *ASLR-GUARD* [19]
  ASLR-GUARD tries to eliminate information leaks that reveal the memory locations of code gadgets, even when ASLR is in use. The creators claim it will “render leak of data pointer [sic] useless in deriving code address by separating code and data, provide a secure storage for code pointers.” Ward et al. [3, pp. 141] found a less than 1% performance impact, a 6.26% increase in executable file size, and a 31% increase in application load time.

- *Polyverse*
  In the interests of complete disclosure, Polyverse is a client of my company, DLT Solutions. Consequently, I will quote third-party evaluations and descriptions to avoid unintentional bias. Ward et al. describe Polyverse as follows:

  > Polyverse provides three different products. The first product is a compiler-based randomization technique. This provides install-time randomization that scrambles the program binary generated from the source code without affecting the semantics of the program. The scrambling can be performed by simply pointing the Linux package manager at the proper repository in a one-line command. The second

> product applies a similar randomization to closed source applications where the source code is unavailable, such as the Windows operating system. This technique employs binary rewriting to apply a boot-time randomization to the layout and instructions of close-source binaries. The third product is a rapid cycling technology that can be applied to continuously running services, like web servers, to periodically restore their environment to a pristine state [3 pp. 1502].

Ward et al. states that Polyverse has a “negligible” impact on performance and only a slight impact on load time. They evaluate Polyverse’s weaknesses as follows:

> Two of the Polyverse products implement one-time randomization. Such techniques are vulnerable to information leakage in which an attacker may be able to discover the location or content of relevant code to construct an attack. However, unlike traditional ASLR, in which the disclosure of one address gives sufficient information for an attacker to infer the entire program's address space, under Polyverse, an attacker would require far more information to be leaked.

Conclusion

MTD is still very new. There are only a few commercial products in play, with a large number of research projects and early-stage products in the space. The selection provided here is only a small sample. Ward et al. [3] provides an excellent and comprehensive survey of MTD technologies for anyone who is interested.

Although I see great promise in this branch of research and development, there is a countervailing view from other researchers. For example, the ideas of David Evans and Anh Nguyen-Tuong, who co-authored of “Effectiveness of Moving Target Defenses” [22], are worthy of serious consideration.

Though not covered here, hybrid MTD approaches could increase the defenders’ options, thus dramatically increasing the resources an attacker would need to successfully find an exploit. Hybrid MTD is an active area of research.

I believe it is foolish to seek an impregnable defense in cybersecurity. We have been trying that for decades now, with little success. We must learn from history and avoid the mistakes that led the to the Maginot Line. To win the war in cyberspace, we must believe we can win. To this end, we need credible weapons, and I believe MTD technology is one of them.

About the Author

As Chief Cybersecurity Technologist for DLT, Don Maclean formulates and executes cybersecurity portfolio strategy, speaks and writes on security topics, and socializes his company's cybersecurity portfolio. Don has nearly 30 years' experience working with U.S. Federal agencies. Before joining DLT in 2015, Don managed security programs for numerous U.S. Federal agencies, including DOJ, DOL, FAA, FBI, and the Treasury Department. This experience allowed him to observe the strengths and limitations of traditional cybersecurity defenses, leading to his interest in innovative technologies such as those featured in this article. In addition to his CISSP, PMP, CEH, and CCSK certificates, Don holds a B.A. in Music from Oberlin, an M.S. in Information Security from Brandeis Rabb School, and is a recipient of the FedScoop 50 award for industry leadership. An avid musician, Don organizes a concert for charity every year, and has been known to compete in chess and Shogi (Japanese chess) tournaments, both in person and online.

About ICIT

The Institute for Critical Infrastructure Technology (ICIT) is a 501c(3) cybersecurity Think Tank with a mission to cultivate a cybersecurity renaissance that will improve the resiliency of our Nation's 16 critical infrastructure sectors, defend our democratic institutions, and empower generations of cybersecurity leaders.

References

[1] Sengupta, Sailik & Chowdhary, Ankur & Sabur, Abdulhakim & Huang, Dijiang & Alshamrani, Adel & Kambhampati, Subbarao. (2019). “A Survey of Moving Target Defenses for Network Security”

[2] Cho, Jin-Hee & Sharma, Dilli & Alavizadeh, Hooman & Yoon, Seunghyun & Ben-Asher, Noam & Moore, Terrence & Kim, Dan & Lim, Hyuk & Nelson, Frederica. (2019). “Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense”

[3] B.C. Ward, S.R. Gomez, R.W. Skowyra, D. Bigelow, J.N. Martin, J.W. Landry, H. Okhravi, “Survey of Cyber Moving Targets, Second Edition” *Lincoln Laboratory, MIT*

[4] P.E. Ammann and J.C. Knight, “Data diversity: An approach to software fault tolerance," *IEEE Transactions on Computers* 37(4), 418-425 (1988).

[5] A. Nguyen-Tuong, D. Evans, J.C. Knight, B. Cox, and J.W. Davidson, “Security through redundant data diversity," in *IEEE International Conference on Dependable Systems and Networks*, IEEE (2008), pp. 187-196.

[6] C. Cadar, P. Akritidis, M. Costa, J.P. Martin, and M. Castro, “Data randomization," *Technical Report TR-2008-120, Microsoft Research*, 2008

[7] S. Son, K. McKinley, and V. Shmatikov, “Diglossia: Detecting code injection attacks with precision and efficiency," in *Proceedings of the 2013 ACM Conference on Computer and Communications Security*, ACM (2013), pp. 1181-1191.

[8] S. Vikram, C. Yang, and G. Gu, “Nomad: Towards non-intrusive moving-target defense against web bots," in *Proceedings of the 2013 IEEE Conference on Communications and Network Security*, *IEEE (2013)*, pp. 55-63.

[9] E. Pattuk, M. Kantarcioglu, Z. Lin, and H. Ulusoy, “Preventing cryptographic key leakage in cloud virtual machines," in *USENIX Security, USENIX Association (2014)*, pp. 703- 718

[10] C. Smutz and A. Stavrou, “Preventing exploits in Microsoft Office documents through content randomization," in *Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses* - Volume 9404, Springer-Verlag New York, Inc. (2015), pp. 225246.

[11] Yih Huang and Anup K. Ghosh, “Introducing Diversity and Uncertainty to Create Moving Target Attack Surfaces for Web Services”, *Springer, 2011*

[12] Azab, M., and Eltoweissy, M. (2012). "ChameleonSoft: Software Behavior Encryption for Moving Target Defense". *Mobile Networks and Applications*, 18(2), 271–292. doi: 10.1007/s11036-012-0392-0

[13] M. Taguinod, A. Doup´e, Z. Zhao, and G.-J. Ahn, "Toward a moving target defense for web applications," in *IEEE International Conference on Information Reuse and Integration (IRI)*, 2015, pp. 510–517

[14] T. Fraser, M. Petkac, and L. Badger, "Security agility for dynamic execution environments" in *AFRL-IF-RS-TR-2002-229 Final Technical Report, DARPA (2002)*, pp. 1-15

[15] Kil, C., Jun, J., Bookholt, C., Xu, J., & Ning, P. (2006). "Address Space Layout Permutation (ASLP): Towards Fine-Grained Randomization of Commodity Software". *2006 22nd Annual Computer Security Applications Conference (ACSAC06)*. doi: 10.1109/acsac.2006.9

[16] Zhu, G., & Tyagi, A. (n.d.). "Protection against indirect overflow attacks on pointers" *Second IEEE International Information Assurance Workshop, 2004. Proceedings*. doi: 10.1109/iwia.2004.1288041

[17] Pappas, V., Polychronakis, M., & Keromytis, A. D. (2012). "Smashing the Gadgets: Hindering Return-Oriented Programming Using In-place Code Randomization" *2012 IEEE Symposium on Security and Privacy*. doi: 10.1109/sp.2012.41

[18] Petsios, T., Kemerlis, V. P., Polychronakis, M., & Keromytis, A. D. (2015). "DynaGuard" *Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015.* doi: 10.1145/2818000.2818031

[19] K. Lu, C. Song, B. Lee, S.P. Chung, T. Kim, and W. Lee "Aslr-guard: Stopping address space leakage for code reuse attacks," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM (2015),* pp. 280- 291

[20] Benjamin Cox, David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack Davidson, John Knight, Anh Nguyen-Tuong, and Jason Hiser "N-Variant Systems: A Secretless Framework for Security
through Diversity" *15th USENIX Security Symposium*, Vancouver, BC, August 2006

[21] H. Okhravi, A. Comella, E. Robinson, and J. Haines, "Creating a cyber moving target for critical infrastructure applications using platform diversity," *International Journal of Critical Infrastructure Protection* 5(1), 30-39 (2012)

[22] David Evans, Anh Nguyen-Tuong "Effectiveness of Moving Target Defenses", John Knight University of Virginia e-mail:[evans,nguyen,knight]@virginia.edu S. Jajodia et al. (eds.), *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats, Advances in Information Security 54*, DOI 10.1007/978-1-4614-0977-92, © Springer Science+Business Media, LLC 2011