On the Effectiveness of the Metamorphic Shield

The effects of continuously changing the attack surface

Anh Nguyen-Tuong, Andrew Wang, Jason D. Hiser, John C. Knight and Jack W. Davidson
Department of Computer Science
University of Virginia
Charlottesville VA 22904
{nguyen | aaw6f | hiser | knight | jwd}@virginia.edu
http://helix.cs.virginia.edu/

***Abstract*—In this paper we analyze the effectiveness of dynamic artificial diversity, i.e., artificial diversity in which the subject of the diversity is re-randomized periodically. We refer to a mechanism that implements dynamic diversity as a *Metamorphic Shield* since this mechanism applies metamorphosis to the system's attack surface to try to shield the system from certain attacks. Contrary to intuition, our analysis reveals that dynamic diversity provides limited benefit except in special cases. In particular, it offers benefit for attacks that seek to leak information. We present a case study of the use of dynamic diversity applied to Instruction Set Randomization that is subject to an incremental attack on the key.**

Keywords: dynamic diversity, temporal diversity, virtual machine, attack surface

I. INTRODUCTION

Artificial diversity algorithmically sets the value of some attribute of a system to a random value where knowledge of the value is needed for a successful security attack. Having to determine the value of the attribute is an impediment to an adversary who has access to the system, and artificial diversity has proven to be an attractive security technology.

In principle, many characteristics of a system can be subject to artificial diversity. The effectiveness of artificial diversity rests upon the size of the space from which the instance of the attribute in use was selected. Unless the adversary can gain access to both the randomization function and the key that was used, defeating artificial diversity requires a state-space search.

To improve the strength of an artificially diverse system in which searching the state space can be accomplished relatively rapidly, an intuitive approach is to frequently re-randomize the system attribute subject to diversity, i.e., to effect *dynamic* artificial diversity. Applying dynamic artificial diversity effectively changes the attack surface seen by the adversary, and we have coined the phrase *Metamorphic Shield* (MMS) to describe the approach. The intuition of many people is that an MMS would provide considerable protection even in a context of low entropy because of the re-randomization. The idea of using an MMS in a general way by varying a variety of system characteristics over time is tempting.

In this paper we build upon a result by Shacham et al. [9] to show that this common intuition is misleading. The MMS varies the attack surface presented to the attacker such that the state space that the attacker has to search on average is only twice as large. However we find that the MMS provides significant benefits for other aspects of security, in particular, information leakage. We analyze the performance of a Metamorphic Shield on an arbitrary incremental attack against a system's key, and show the effect of re-randomization and of varying the rate at which re-randomization is effected. Finally, we present a case study in which we implemented a Metamorphic Shield for a particular instance of Instruction Set Randomization (ISR) of program binaries to demonstrate the feasibility of an MMS and to measure the MMS' overhead.

II. DYNAMIC DIVERSITY

We characterize diversity techniques by a function f that takes as input some aspects of a program P, a key K, and transforms P into a semantically equivalent program P' in normal use. Defenders should select a function f that maps to a large range so as to make brute-force attacks infeasible and should keep the key K a secret. The goal of attackers is to discover the value of $f(K, P)$ so as to mount a successful attack. This goal can be achieved by directly guessing either $f(K, P)$ or K (we assume that f and P are known to the attacker). Dynamic diversity adds a temporal component to the defender's arsenal. The hope is to vary $f(K, P)$ at a "fast enough" rate to prevent attackers from discovering $f(K, P)$.

In the following discussion we assume that dynamic diversity is applied to the key K (most automated diversity defenses fall under this category). The possibility of applying dynamic diversity to f itself is outside the scope of this paper.

Shacham et al. showed that re-randomization of the memory layout used in Address Space Randomization (ASR) only provides one extra bit of entropy, i.e., re-randomization of the address space in ASR at best only doubles the expected time to carry out a brute-force attack rather than the orders of magnitude that one might intuit [9]. Their analysis is based on modeling the state-space search in ASR as a sampling problem. Static diversity corresponds to sampling without replacement from the state space, whereas dynamic diversity corresponds to sampling with replacement. Even though the original analysis focused on the effect of re-randomization for ASR, the analysis generalizes to arbitrary $f(K, P)$.

Table I summarizes the effect of dynamic diversity in the general case. The columns of the table correspond to the amount of entropy for *f(K, P)*, low vs. high. The rows distinguish the cases of the key *K* being revealed or kept secret.

TABLE I. EFFECT OF THE METAMORPHIC SHIELD

Key	***Revealed***	Limits window of vulnerability	Limits Window of Vulnerability
	Secret	Increases attacker workload by 2x	Increases attacker workoad by 2x
		Low	***High***
		Entropy	

In the case where the key remains secret and an attacker is forced to carry out a brute-force attack against *f(K, P)*, the strict upper bound is a factor of two on the attacker's workload (bottom row in the table). This result is a direct consequence of the analysis of Shacham et al.

Under the secret key assumption, the factor of two upper bound on the benefit of an MMS indicates that dynamic diversity provides little benefit as a defense mechanism above and beyond the baseline diversity technique. When entropy is low, the expected time to mount a successful attack would be relatively short, and therefore a factor of two would be of little value. When entropy is high, the expected time to mount a successful attack would be long, and again a factor of two would not provide any significant benefit.

The factor of two might be of value if an attack required that an adversary determine details of *n* separate applications of diversity about a program. If dynamic diversity were applied to all *n* diversity randomizations, the maximum aggregate effect of dynamic diversity would be to increase the attacker's workload by 2^n. While a factor of 2^n (when *n* is small) is a marginal improvement on a relative basis, it may still be useful on an absolute basis. For example, with n=2, if it takes 12 minutes to carry out an attack instead of just three minutes, the additional nine minutes provides additional reaction time.

The effectiveness of artificial diversity relies on key secrecy. Information leakage attacks in which the key can be revealed all at once or incrementally as a series of parts reduces the effective entropy and obviates the need for a brute-force search against the entire state space of *f(K, P)* [3][11]. Examples of such attacks include those that exploit format string vulnerabilities, buffer overflows, and the predictable generation of random numbers. Assuming that attackers require knowledge of *K* to mount an attack, an MMS could limit the window of vulnerability in the case of an incremental attack by voiding partial information about *K* obtained by the adversary (top row of Table I). Provided *K* was changed before all of *K* was obtained, the adversary's efforts would be of no use.

An example of an incremental attack was reported by Sovarel et al. [10]. In that work, an instance of ISR that employed a long key was shown to be vulnerable to an attack in which small fractions of the key were determined sequentially. Each fraction required only a small amount of computation thereby bypassing the entropy of the long key upon which the owners of the system might rely.

In the next section, we present a model for analyzing the effectiveness of a Metamorphic Shield against an incremental attack.

III. MODELLING OF THE METAMORPHIC SHIELD FOR THE INCREMENTAL ATTACK MODEL

We model an incremental attack as a series of *b* state-space searches where the states are of the same size *s*. A state-space search is carried out as a series of probes, and each state-space search is designed to reveal a single key *fragment*. Hence the key length is *b* fragments. We define a successful attack as a sequence of successful state-space searches of the *b* spaces. We assume that:

- the adversary proceeds sequentially from space to space determining one fragment for each space,
- the adversary knows when a fragment has been revealed, and
- each probe of a space requires the same time.

In this case, the quantity of interest is the probability of a successful attack occurring in some specific number of probes, say *k*, or less. With that probability known, a Metamorphic Shield could re-randomize after the adversary had an opportunity to perform *k* probes and thereby limit the probability of a successful attack. Thus, our first goal in the analysis is to determine this probability. Clearly, we cannot know how many probes have occurred, but we can estimate the number of opportunities that the adversary had.

Searching each space will terminate with a successful probe, and each successful probe will be preceded by from zero to *s*-1 probes that fail. The initial step in the model is to determine the probability of a successful attack in exactly *k* probes. Such an attack will experience a total of *k-b* probe failures across all *b* spaces together with *b* successful probes. Thus, the total number of different sequences of probes that can lead to a successful attack in *k* probes is the number of ways that *k-b* failing probes can be distributed across *b* spaces with no more than *s*-1 occurring in any single space. This number is [1]:

$$N(k) = \sum_{t=0}^{b} (-1)^t \binom{b}{t} \binom{k - ts - 1}{b - 1}$$

In this expression, binomial coefficients are defined to be zero if the upper operand is smaller than the lower operand.

The probability of a successful attack occurring in exactly *k* probes for $b \le k \le sb$ is:

$$p(k) = \frac{N(k)}{2^{sb}}$$

The probability of a successful attack occurring in *k* probes or less for $b \le k \le sb$ is:

$$P(k) = \sum_{i=b}^{k} p(i)$$

This is the probability we sought, and with this probability we can determine the effect of a Metamorphic Shield operating against an incremental attack in which the MMS re-randomizes periodically.

We model the effect of a Metamorphic Shield by treating an attack as a series of independent trials by the adversary each of length *m* probes where the key is changed after each trial, i.e., after *m* probes. Thus, the effect of the Metamorphic Shield is to force the adversary to restart the attack after each series of *m* probes if the attack was not successful at that point.

The probability of a successful attack in *m* probes or fewer is *P*(*m*). With a Metamorphic Shield re-randomizing after each trial (each *m* probes), the probability of an attack succeeding in *jm* probes or fewer is:

$$M(jm) = 1 - (1 - P(m))^j$$

Note that this probability is defined only for every *m* probes. In order to derive the probability of a successful attack in *k* or fewer probes, we need to add the probability of a successful trial (determining a single fragment of the key) in *k-jm* probes where *k-jm* lies between 0 and *m*-1, i.e., between the points at which the key is changed. Adding this yields:

$$M(k) = 1 - (1 - P(m))^j (1 - P(k - jm))$$

M(*k*) is the probability of a successful attack in *k* or less probes with a Metamorphic Shield re-randomizing every *m* probes and *P*(*k*) is that probability without a Metamorphic Shield. With these two probabilities, we can determine the effectiveness of a Metamorphic Shield.

As an example, consider the case in which *b* = 4 and *s* = 256. This corresponds to a key that is four bytes long which would be expected to have a search space of size 2^{32}. However, the incremental attack proceeds one byte at a time so that there are four searches each of spaces of size 256. Obviously, the probability of success in 1024 probes or less is one.

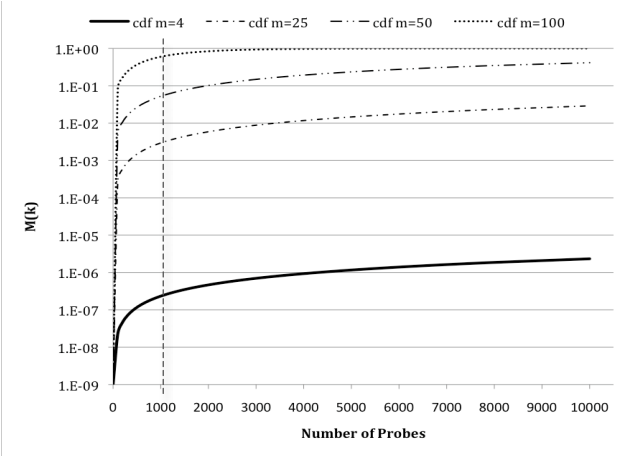

Figure 1. Cumulative distribution as a function of the rate of metamorphosis of the shield.

Figure 1 shows *M*(*k*) for this example for values of *m* = 4, 25, 50, and 100. Note that the Y axis is a logarithmic scale. The dashed vertical line is 1024 on the X axis. This is the point at which an attack is bound to succeed without an MMS, and the intersection of the dashed line with the four curves shows the relative advantage of the MMS. The case in which *m* is set to 4 is the limiting case in this example. Four is the least number of probes within which an attack might succeed since there are four bytes in the key and the adversary has to determine all four in sequence. Thus, the curve in Figure 1 for *m*=4 is the best that a Metamorphic Shield can do in this example.

As can be seen from the example, the effectiveness of a Metamorphic Shield against incremental attacks in this case depends critically on the rate of re-randomization. Varying this rate from every 100th probe to every 4th probe spans 6 orders of magnitude.

IV. Case Study: Instruction Set Randomization

Instruction Set Randomization (ISR) is a diversity technique introduced by Barrantes et al. and Kc et al. that randomizes the instruction set of a target machine [2][8]. Since an attacker does not know the randomized instruction set, attempts at code injections will fail. While the idea is applicable to instruction sets for a variety of machines and interpreters, including SQL, Perl, PHP and XML [5][6], we focus on the application of ISR to the X86 architecture.

A simple implementation of ISR is to encode at load time (or earlier) the native binary form of a program using an XOR key [2][8]. Just prior to execution, the program is decoded using the same XOR key to recover the original instruction stream. Injected code that is not encoded will likely result in the execution of random instructions that will lead to the target program crashing.

Sovarel et al. demonstrated an incremental attack against ISR using a deliberately crippled version of an XOR-based ISR implementation [2]. Their attack illustrates the potential pitfalls of relying on seemingly high entropy defense techniques or on the assumption of key secrecy. Sovarel et al. obtained a 4-byte key in approximately 20 seconds and a 64-byte key in approximately one minute.

We set out the following requirements for the design of the Metamorphic Shield for this instance of ISR:

- The shield should operate on arbitrary x86 binaries, similarly to the widely deployed ASR technique.
- The shield should not require re-randomization using a shutdown/restart sequence.
- The architecture of the Metamorphic Shield should be flexible and allow a wide range of possible diversity transformations.
- The Metamorphic Shield should be able to operate efficiently at the rates determined to be necessary to provide useful protection against incremental attacks.

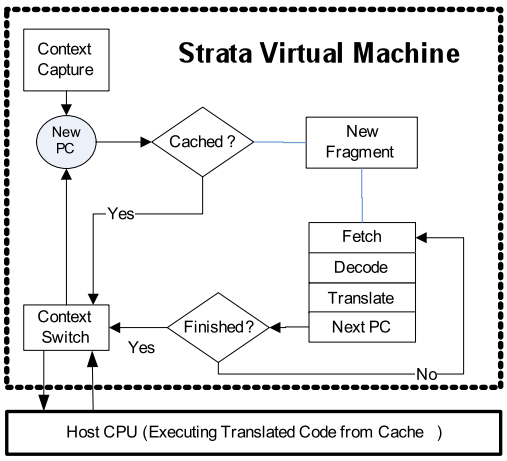

Figure 2. Strata Virtual Machine.

To our knowledge, the work by Bhatkar et al. on *self-randomizing programs* comes closest to fulfilling these requirements [4]. That approach requires access to source code and provides protection using fine-grained, address-space randomization with a reported overhead of 11%.

The Metamorphic Shield we developed to support ISR is implemented using the Strata software dynamic-translation system [7][12]. As shown in Figure 2, Strata is organized as a virtual machine that operates between an application and the host system. Strata loads a binary application dynamically and mediates application execution by examining and possibly translating the application's instructions before they execute. Blocks of translated application instructions are held in a Strata-managed code cache to improve efficiency.

We developed a tool to analyze ELF binary programs and to identify the ranges where executable instructions can exist. These instruction ranges are added to a new section in the ELF binary. When Strata starts up, it reads this new section at load-time and encrypts the sections using a simple XOR scheme with an n-byte key, where $n = 4$ by default:

$$P' = K \oplus P$$

To recover the original program's instruction stream, we add a decryption module between the fetch and decode modules of the Strata virtual machine, and apply the following transformation:

$$P = K \oplus P'$$

To rekey the text segment of the program during execution, we apply the old XOR key, followed by a new random XOR key:

$$P' = K_{new} \oplus K \oplus P'$$

$$K = K_{new}$$

Our current prototype implementation has the following limitations:

- The prototype handles statically and dynamically-linked libraries but not preloaded libraries such as libnss.
- The prototype does not support self-modifying code. This is the case with all ISR implementations of which we are aware.

V. Evaluation

We evaluated the performance of our XOR-based ISR implementation of a Metamorphic Shield using the SPEC2000 benchmark. We present performance results for a re-randomization rate of 100 milliseconds. All performance numbers were averaged over three runs for each of the program in SPEC2000. These numbers were obtained using version 8 of Fedora Core Linux, running in a VMWare image on a dedicated Mac Pro.

Figure 3 shows the performance of executing the benchmarks with and without the metamorphic shield. For a rekeying rate of 100 msec, the performance of the metamorphic shield is essentially the same as that of running the Strata virtual machine. This result is encouraging because it indicates that the metamorphic shield adds virtually no overhead beyond that of Strata itself. Despite measuring the performance on an unoptimized configuration of Strata, the overall average performance overhead of the metamorphic shield is only 14%.

Our concern with this work is to determine whether this Metamorphic Shield can compensate for a weak encoding mechanism and a short key length for ISR. If so, what is the appropriate re-randomization rate?

Answering these questions requires making real-time assumptions about the probing rate. For example, the average probe time in the attack by Sovarel et al. is approximately 20 msec. This time would translate to a re-randomization rate of every fifth probe in our analytical model (Figure 1). However, one could argue that the attack understates the capability of a motivated adversary. For example, an adversary could control a botnet and issue probes in parallel. The maximum scaleup factor would then depend on the number of concurrent requests that the targeted program could handle. If we assumed a scaleup factor of 100X, then a 100 msec re-randomization rate would correspond to the case where the shield was re-randomized every 100^{th} probe.

Instead of re-randomizing based on a real-time trigger, we plan on investigating the performance of re-randomizing the shield based on the number of probes. Since we cannot readily distinguish between normal traffic and attack probing traffic, we need to assume conservatively that every packet read over the network is potentially a probe. While re-randomizing programs on every four read system calls may seem excessive, whether this will turn out to be the case is unclear. Furthermore, we will investigate the use of anomaly detection techniques to distinguish between normal traffic and attack probes and thereby reduce the required rate of re-randomization.

VI. Conclusion

Artificial diversity is an effective security technology provided the randomization used yields a search space of sufficient size and the key cannot be recovered by an adversary. Dynamic artificial diversity adds the notion of

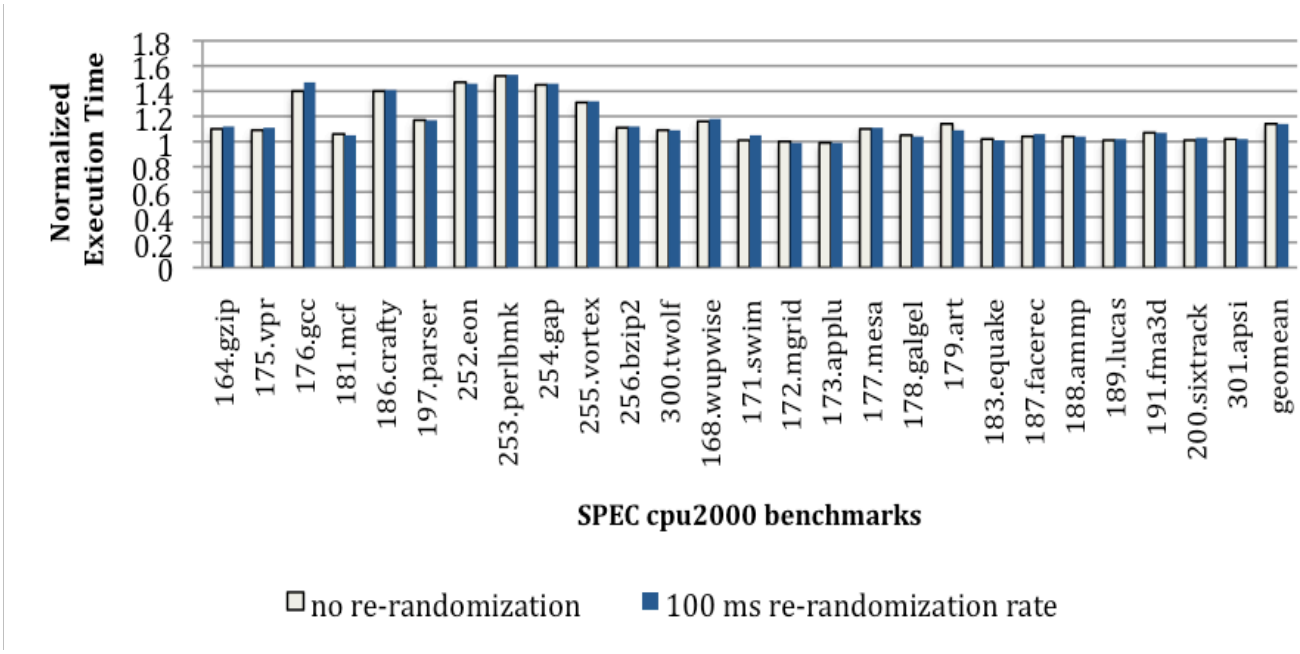

Figure 3. SPEC cpu2000 benchmark. Performance normalized to native execution.

periodic re-randomization to artificial diversity. We refer to the mechanism that implements dynamic diversity as a Metamorphic Shield because the mechanism applies metamorphosis to the attack surface and thereby offers the potential to shield the adversary from some forms of attack.

We have developed a general model of dynamic diversity and applied it to an incremental attack against instruction set randomization. In that case, re-randomization restores lost entropy provided re-randomization occurs at a rate that is fast enough. The model we have developed predicts the probability of a successful attack with a certain number of state space probes, and so the model allows the rate of re-randomization necessary for a predefined level of protection to be determined.

Finally, we note that any dynamic variation of a system's characteristics designed to vary the attack surface and thereby thwart an adversary is, in practice, an example of dynamic artificial diversity and is, therefore limited in effect to the extent predicted by our model.

ACKNOWLEDGMENT

This work was funded in part by the National Science Foundation under grant CNS-0524432 and in part by DoD AFOSR MURI grant FA9550-07-1-0532. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Department of Defense.

REFERENCES

[1] "Balls In Bins with Limited Capacity," http://www.mathpages.com/home/kmath337.htm

[2] G. Barrantes, D. Ackley, S. Forrest, and D. Stefanovic, "Randomized Instruction Set Emulation," ACM Transactions on Information Systems Security (TISSEC), Vol 8, No 1 pp. 3-40 (2005).

[3] D. J. Bernstein, "Cache-timing attacks on AES," http://cr.yp.to/antiforgery/cachetiming-20050414.pdf

[4] S. Bhatkar, R. Sekar and Daniel DuVarney, "Efficient Techniques for Comprehensive Protection from Memory Error Exploits," USENIX Security Symposium (USENIX Security) August, 2005.

[5] S. W. Boyd, G. S. Kc, M. E. Locasto, A. D. Keromytis and V. Prevelakis, "On the General Applicability of Instruction Set Randomization," IEEE Trans. On Dependable and Secure Computing, 07 Oct. 2008. IEEE computer Society Digital Library.

[6] M. Van Gundy and H. Chen, "Noncespaces: Using randomization to enforce information flowtracking and thwart cross-site scripting attacks," Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, February 8-11, 2009

[7] W. Hu, J. D. Hiser, D. Williams, A. Filipi, J. W. Davidson, D. Evans, J. C. Knight, A. Nguyen-Tuong and J. Rowanhill, "Secure and Practical Defense Against Code-injection Attacks Using Software Dynamic Translation," Second International Conference on Virtual Execution Environments. Ottawa, Canada, June 14-16, 2006.

[8] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 272–280.

[9] H. Shacham, M. Page, B. Pfaff, E. Goh, N. Modadugu and Dan Boneh, "On the Effectiveness of Address-Space Randomization," Proceedings of CCS 2004, pp. 298-307, ACM Press.

[10] N. Sovarel, N. Paul and D. Evans, "Where's the FEEB?: The Effectiveness of Instruction Set Randomization," USENIX Security 2005, August 2005.

[11] R. Strackx, Y. Younan, P. Philippaerts, F. Piessens, S. Lachmund, and Thomas Walter, "Breaking the memory secrecy assumption," Proceedings of the Second European Workshop on System Security, Nuremburg, Germany, 2009.

[12] D. Williams, W. Hu, J. W. Davidson, J. D. Hiser, J. C. Knight, A. Nguyen-Tuong, "Security through Diversity: Leveraging Virtual Machine Technology," IEEE Security and Privacy, vol. 7, no. 1, pp. 26-33, Jan./Feb. 2009.