*Journal of Advances in Science and Technology*

*Vol. III, No. VI, August-2012, ISSN 2230-9659*

REVIEW ARTICLE

# COMPOSITION OF DIVERSITY TECHNIQUES

www.ignited.in

*Journal of Advances in Science and Technology*
*Vol. III, No. VI, August-2012, ISSN 2230-9659*

# Composition of Diversity Techniques

**Rishi Pal Bangarh[1] Dr. K.K. Jain[2]**

[1]Research Scholar of Singhania University

[2]Asst. Prof., P.G.V. College Gwalior (M.P.)

----------♦----------

## GENERALIZING AND ASSIGNING SOFTWARE DIVERSITY

Monocultures, like a field of corn, are susceptible to infections, but genetically diverse cultures, like a prairie, are extremely robust.

## INTRODUCTION

We create a generalized framework for classifying and analyzing diversified software that is driven not by the diversity schemes themselves but by how diversified software appears to an attacker.A single system can be diversified by running an operating system variant, such as Linux or OpenBSD. Any single implementation inside the diversified set can then be diversified again by running the system on a different base hardware platform, namely either x86 hardware or SPARC hardware. If an attacker has a working exploit against software running on a Linux x86 system that they wanted to use against an OpenBSD/SPARC system, the attacker would have to mutate the attack so that it is effective against both a different operating system and a different platform.

In general, each diversity technique applied to a single system creates a pool of diverse systems from the originating system. Each system inside that pool can then be diversified by a separate technique to create additional instances of diverse software. This concept forms the basis of our diversity model. We consider every possible instance of software that can be generated by the application of diversity techniques, then place the software instances into the same set if they appear to an attacker as if they are separated by a single diversity technique. A single piece of software can be in multiple sets, as it can be used as a seed for multiple different diversity techniques. The software instances are the vertices of a hypergraph, with the sets of diversified.

## HARDWARE

A single Apache installation is diversified by the introduction of different operating systems. One of the instances is further diversified by the introduction of multiple hardware platforms. An abstraction of the model is presented . Variants generated by a single diversity technique forming the hyperedges. Since the hyperedges naturally overlap at points where a software package is diversifiable using more than one technique, we are able to reason about the use of multiple diversity techniques on a single software package. We can abstract the behavior of combining multiple diversity techniques as being a walk across intersecting hyperedges on the hypergraph. To an attacker, the amount of work that he or she must undertake in order to modify an effective exploit against one system so that it can compromise another is a function of the number of hyperedges, or diversity techniques, which separate the two instances of the software.We describe how metrics which derive from attack and defense modeling can be applied to the hyperedges for purposes of determining an optimal balance of attack tolerance and implementation cost. We examine what is effectively a trivial application of the model by examining the application of diversity techniques to a single system , and in more depth in Appendix A. The remainder of the dissertation is spent examining the problem of assigning diverse software packages to networks of systems.

## RELATEDWORK

Our model is generated by examining how diversity appears to an attacker, and can easily be extended to encompass new diversity techniques and new forms of attacks. The generation of a diversity hypergraph is not dependent upon taxonomies of previously developed diversity techniques. The generation of a diversity hypergraph for a real system may in fact lead to new forms of diversity taxonomies, ones where the effect of diversity on an attacker is central to the taxonomy. Taxonomies of attack techniques and methodologies would potentially be useful for modeling the abilities of an adversary who is confronted with diversity techniques.

## DEFINING THE DIVERSIFICATION HYPERGRAPH

**Definition 1** Let d ∈ D be a single diversity technique in the set D of all diversity techniques. Let u ∈ U be a

single binary in the set U of all possible software binaries. The application of a single diversity technique in D takes a single instance of software in U and generates a set of software packages. The software generated by a single diversity technique is viewed to be interchangeable with one another as defined by the bounds of the diversity technique. Elements of the set of diversified software packages can be grouped together into equivalence classes, where any element in the class can be mutated to become another element in the class using a single diversity technique. For example, if the diversity technique requires systems to run separate operating systems, the set of diverse systems are equivalent under the bounds of operating system diversification.

**Definition 2** The elements d ∈ D form equivalence relations on the set U. Two software packages u1, u2 ∈ U are said to be equivalent under diversification scheme d if the only difference between the two elements in U results from the application of the diversity technique described by d.

**Definition 3** Each equivalence relation generated by the elements of D creates equivalence classes over the elements contained in U. We denote the equivalence created by technique d between the two elements u1 and u2 in U as u1 ≡d u2.

We can loosely classify the equivalence classes into several categories. Two software packages lie in a binary equivalence class if the lowest cost modification required to transform one software package into the next can be done at the binary level. If a vulnerable software package is diversified via a binary technique, the original attack target will still exist; the exact memory location of the attack target becomes far harder to find, however, due to the increased space over which the memory location of the attack target may exist. An example of a binary-level modification would be the randomization of the layout of a program and its linked libraries in memory.

While it is possible to convert one program to another through bitwise adjustments, the process of doing so may be extremely time consuming. It could be far easier do make the modifications at the source code level and allow the compiler to generate the different binary. Likewise, if it becomes less costly to convert one binary package to another via source code modification than recompilation, then the two software packages lie in a source equivalence class created by the diversity technique. Attacks against software packages which have undergone a source modification technique must be modified themselves to be made effective against the newly diversified software packages. The modifications for the attack code may be as simple as a single modification in the attack binary, but given the stage of the development cycle at which the diversity technique is introduced, it is likely that a more advanced algorithm or manipulation scheme would have to be utilized for the attacker to successfully attack the diversified software package.

As stated in the introduction, several diversification strategies exist for the algorithmic modification of source code, such as adding or deleting nonfunctional code, code reordering, and randomizing memory layouts . More invasive techniques which modify data and control flow are also feasible . Incidentally, these code reordering and reforming techniques can also be effective against reverse engineering attacks . Two software packages may have extremely differing lineage or development histories but serve the exact same purpose in a system. If two software packages provide effectively the same functionality, such as two distinct flavors of UNIX, then the software lies in a functional equivalence class. Probably the most studied method of generating functionally equivalent software packages uses the N-Version Programming technique .

The equivalence class generated by any given diversity technique may not be directly tied to the stage in the development cycle at which the diversity technique was applied. For example, consider a source code modification technique that works by repositioning variable declarations. The effect on the final diversified binaries that result from the technique's application can also be generated by directly modifying a compiled binary's memory structure. If a system designer uses both the source and binary-level modifications, all the binaries generated using the diversity techniques would reside in the same equivalence class. The use of multiple diversity techniques on the same piece of software is described further in the following section.

## COMPOSITION OF DIVERSITY TECHNIQUES

**Definition 4** A compositions of diversity techniques is the serial application of the techniques one by one in order of temporal precedence. Composition increases the amount of work necessary to convert an attack which is effective against one software package to be effective against another one generated from 23 the first via a set of diversity techniques. For example, a binary can be diversified using a compile-time memory space randomization scheme , then executed on a system which utilizes an encrypted instruction set . Any attacker who wishes to take an attack which is effective against a single binary and mutate it so that it is effective against a binary which has undergone diversification using both techniques discussed would have to simultaneously de-randomize the memory space and decrypt the instruction set utilized by the diversified binary. An attacker may not need to manipulate an attack to solve two diversity techniques at the same time; if N-version programming is employed in the selection of the base operating system employed to run the binaries, an attacker can first solve all the mutations necessary to combat the introduction of the foreign operating system and then solve the

issues associated with the instruction set and memory space manipulations.

**Definition 5** The set of all equivalence classes created by the diversity techniques in D form the hyperedges E,which along with the elements of U define the diversification hypergraph H = (U, E).

In order to define properties about interactions between hyperedges, being able to identify individual hyperedges becomes a necessity. It is easy to see that every hyperedge in E can be identified by a diversity technique in D and a single binary in U which lies in the hyperedge. Consider two hyperedges which are created by the same diversity technique and containing the same binary. The diversity techniques from each hyperedge would create two equivalence classes that cover all binaries separated by the single diversity technique. Since both diversity techniques are identical, they would create equivalence classes which contained the same set of elements, and thus create the same hyperedge.

**Definition 6** The composition of diversity techniques can be formally expressed as a path of hyperedges P on the diversification hypergraph, where two edges are adjacent in the path if and only if their intersection contains at least one element of U.

The act of composing multiple diversity techniques can be thought of in terms of the diversity hypergraph H as moving from one binary in U to another by walking from one adjacent hyperedge to the next. OS diversification and hardware diversification is applied to a single Apache web server installation. By composing these two diversity techniques, the system engineer would force an attacker who is able to directly exploit Apache on a Windows machine to mutate his attack for both a different operating system, namely OpenBSD, and a different hardware platform.

**Definition 7** Temporal Precedence is an ordering on all diversity techniques necessitated by the stage in the design process where the techniques must be applied.

The application of one diversification technique may undo the work of a previously applied technique. Therefore, two diversity techniques can be composed if and only if they respect temporal precedence. A simple but illustrative example of temporal precedence can be found in the use of both source code modification and compile time automatic variable location randomization diversity techniques. Both techniques can be utilized to make a single software package more diverse than its standard, reference compilation. The temporal hierarchy places any source code modification before the address space randomization since it is necessary for any source code modification techniques to be applied before any address space randomization techniques are considered. We deconstruct the temporal hierarchy into the diversification stages, namely Requirements,

## ARCHITECTURE, IMPLEMENTATION, AND REALIZATION

Both composition and precedence requirements can be visualized. In the example, software package u1 belongs to two equivalence classes generated by diversity techniques d1 and d2. The diversity technique d1 encompasses a large number of diversified software packages, including u2, which is in turn further diversified by technique d2. Similarly, package u3 is related to u1 by diversity technique d2, and is then further diversified by d1. The diversity techniques d1 and d2 create hyperedges which form a path from u1.The generalized view of software diversity described where diverse software instances are set elements, diversification techniques are equivalence classes, and the composition of multiple diversity techniques forms a path across equivalence classes. We represent the a simplified view of the diversification hypergraph H in where the edges represent individual hyperedges and the vertices represent software packages in U which lie at the intersection of two hyperedges. u4 through both u2 and u3. Since d1 and d2 can be applied in any order without violating temporal precedence, the application of d2 after d1 to u1 reaches the same software instance as the application of d1 after d2. Finally, we show a single instance of the application of d3 to u4, which is diversification techniques away from u1. While d3 can be applied to u1, u2, and u3, the equivalence classes created by such an application are omitted for the sake of clarity.

## ATTACK AND DEFENSE MODELING

The utility of dividing diversified software packages into equivalence classes is more clear when examined through the lens of attack modeling. The deployment of a wide variety of commercial-off-the-shelf operating systems to a network may be an effective method of combating a worm which is designed to attack a single exploit, but is ineffective against an attacker who is willing to purchase each of the operating systems and invest the necessary time required to develop a set of custom exploits against each OS. Conversely, a compile time randomization which alters the structure of a binary for each system would be an effective method of combating a human being who develops their exploits using a debugger and a local copy of the software under attack, but would only delay a worm which uses a search algorithm to determine the memory locations of the previously used attack targets.

The diversity schemes discussed are also not equally effective against all forms of attack. Diversifying the

instruction sets utilized by different binaries can combat buffer overflow attacks, but the technique is ineffective against a resource exhaustion attack. Producing several versions of the software to utilize different network protocols may evade a denial of service attack yet produce binaries which are vulnerable to a buffer overflow attack.

**Definition 8** Let the types of attacks that would take place be denoted by the set T. We denote the set of software attacks as A. The mapping of attacks on software packages to the attack techniques used is defined as _ :A 7→ T.

For every diversity technique in D there exists a series of hyperedges in which the vulnerable software package resides. The attack technique _ (a) employed by an attack a ∈ A can be mutated to attack another software package which resides in one of the vulnerable software package's equivalence classes. The system designer can then choose which attacks are the most threatening to system survivability by weighting the range of _ to the most critical attack types. Let M be the set of all implementation metrics which are of interest to the system designer. The implementation metrics can be exhibited in several forms, such as a slowdown associated with the execution of a binary which underwent modification by a diversity technique. In a similar fashion, the increase in runtime memory consumption and program storage size of the diversified binary are also accounted for this way. M is not limited to system performance metrics, however, as the total economic cost incurred by the implementation of diversification techniques can be included in this set.

**Definition 9** The diversification cost _ is a function which maps each diversification technique in D and performance metric in M along with the type of binary which is being diversified in U to a positive and real multiplicative factor corresponding to the implementation cost: _:D ×M ?U 7→ R+

**Definition 10** The effectiveness probability _ is a function which maps each diversification technique in D and attack technique in _ (A) to a quantity which reflects the ability of a diversification technique to resist the specified form of attack. _:D ?_ (A) ?U 7→ [0, 1] Each diversity technique has twometrics with which it is associated. The diversification cost _ is a function which quantifies the cost to each system performance metric associated with implementing each diversity technique, be it memory consumption, loss of execution speed, or economic cost of implementation. The effectiveness probability _ is a function which quantifies the probability that an attack technique can be modified to compensate for the diversity introduced by a given technique. The effectiveness probability reflects an attacker's ability to mutate an attack against any one binary in the equivalence class to be effective against any other binary in the equivalence class, and is a function of the attacker's skill and the type of attack that is being combated. Metrics of this type have been employed for describing code obfuscation techniques to combat reverse engineering.

The effectiveness probability need not be defined for the entire set of attack techniques, as indicated by the choice of _ (A) rather than T(A). The system designer can choose a subset of attack techniques which he or she considers to be of the greatest threat and model the effectiveness of each diversity techniques against only the attack techniques of interest. Additionally, it is possible to use an element in u rather than the chain of all diversity techniques to define the cost and effectiveness of applying a single diversity technique even though the effectiveness of a diversity technique may be a function of previously applied techniques. Each element in u, by its nature, encodes the set of all diversity techniques which have been previously applied in order to reach that point. The concept of a diversity technique's cost and effectiveness being a function of previously applied diversity techniques is discussed in the following definition.

**Definition 11** The property of diversity non-linearity dictates that the cost and effectiveness of a diversity technique is a function of the previously applied diversity techniques. The cost and effectiveness of the currently applied diversity technique can be amplified or attenuated, which we term a non-linear composition. If the cost and effectiveness of a diversity technique is unaffected by previous diversity techniques, we define the interaction as being a linear composition. The effectiveness and cost of applying a diversity technique to a binary is not con29 stant for all systems. A diversification technique that depends upon linking to functionally equivalent but different standard libraries may cost more to apply for a closed-source operating system than for an open-source operating system. Address space randomization techniques become more effective as the address space available on the hardware platform increases. This property of non-linear composition holds implications for the development of algorithms for the optimization of diversity.

**Definition 12** We define an attack relevance function w_: _ (A) 7→ [0..1] which sets the relative importance of individual attack threats to the system designer. A similar weighting function, or the cost relevance w_:M 7→ [0..1], is provided to balance out the diversification cost.

A system engineer can then form an attack model in which diversity is involved by choosing an appropriate attacker profile and use historical data to generate the effectiveness probability expected for the diversity techniques against the attacker. Furthermore, the system engineer can create a defense model consisting of the set of attacks which become critical for system defense. A first attempt at generating a survey of diversity techniques which examines their effectiveness against various classes

www.ignited.in

of attacks is presented . Both of these functions are utilized in the optimization of diversity assignments, as demonstrated in the following section.

## HYPERPATHS AND CHOOSING DIVERSITY

The hypergraph framework presented provides a method for determining when and how to apply diversity techniques to a piece of software on a single server and for an entire network of systems. For a single piece of software, the system designer is faced with determining a walk on the hypergraph which provides the greatest distance

## FUNCTIONS

_ - Mapping from attacks to attack techniques

_ - Cost of implementation of a diversity technique as a function of the diversity technique, cost metric, and the software being diversified.

_ - Probability of an attack against a diversified software package as a function of the diversity technique, the type of attack, and the software being diversified. between two diverse software packages while keeping the project under pre-specified cost bounds. When faced with a network of systems, the designer must determine a set of diverse software packages which, when assigned to systems on the network, span the largest distance in the diversification hypergraph if they are neighbors of each other on the network.

In both general cases, we show that determining optimal solutions to both of these problems is NP Hard. For all current practical instances of the host-based diversity assignment problem which can be currently envisioned, however, heuristic methods can be used to determine the optimal choice of diversity techniques. The same is not true for the network diversity assignment problem.