Poster: Defining a Model for Defense-In-Depth

James Sullivan, Michael E. Locasto
University of Calgary
{jfsulliv, locasto}@ucalgary.ca

I. Introduction

Defense In Depth is a strategy for securing information systems that is characterized by the use of layered and redundant security measures. The principal concept behind Defense in Depth is fault-tolerance of a system- that any given attack path will be covered by multiple measures in case any of them fail.

Though the effectiveness of Defense In Depth is touted by many organizations (such as [1], [2]), there is a gap in the understanding of the limitations and benefits of Defense In Depth from a formal perspective. We present Defense Graphs, a formalized model for the properties of a layered security architecture, enabling a more principled analysis of the comparative effectiveness of security postures.

In our model we propose four properties arising in a layered security posture. *Independence* refers to the degree to which individual security measures in a system will not interact with one another. *Coverage* refers to the extent of the instrumentation provided by a given security measure, or the language that it instruments. *Redundancy* is the degree to which security measures overlap in their coverage. *Cost* is a measurement of the cost of a given system, including both financial and computational resource costs.

II. Motivation

It is known that in general, security policies are not secure under composition [3]- interference between security mechanisms may induce subtle failure in one or both. Technical discussion of Linux Loadable Security Modules (LSMs) in [4] shows that this problem is not restricted to formal domains, and is a concern in the practical domain as well.

Our own experiments have shown easily induced interactions between mechanisms. We performed a series of tests in which two commodity antiviruses were concurrently used on a system, which was subject to a set of standard EICAR [5] test files. Despite the simplicity of these tests, we observed that not all mechanisms were able to correctly operate once they were paired with another antivirus. Table 1 contains the results of our analysis, as proportional values of the tests in which failure was induced to the total number of tests.

	Trial 1	Trial 2	Trial 3	Average
IDF	0.056	0.056	0.048	0.054
IPF	0.234	0.266	0.242	0.247
IAPF	0.113	0.113	0.097	0.108

Fig. 1: Results from the Antivirus case study. "IDF, IPF" refer to Induced Detection/Protection Failure. "IAPF" refers to Induced Absolute Protection Failure.

Despite these well-established compositional effects, many organizations advocate the layering of security mechanisms as an effective strategy for information assurance. Though Defense-in-Depth is often characterized as a best-practice approach [1], the principle seems to stand at odds with other common recommendations for securing information systems, such as maintaining a simple, clean security architecture, or minimizing the interface that a system exposes and focusing efforts on controlling this interface [6].

The model that we present is a tool to make apparent the compositional structure of systems. Using Defense Graphs to model a system allow a system designer to identify regions where there may be undesirable interactions between mechanisms, and to inform better uses of their assets. By shifting the focus away from individual interfaces and mechanisms, patterns of interference and non-interference may be revealed that can be used to inform best practices for the layout of a set of security mechanisms. Rather than examining every mechanism and every interface individually, we view the system as a whole and search for key patterns that make a system more or less susceptible to exploitation.

III. Modelling Defense-In-Depth

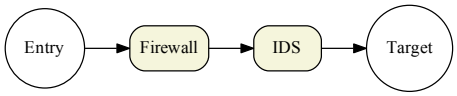
A Defense Graph is a directed, acyclic graph that is composed of:

1) Vertices V which are either security mechanisms (referred to as *policy enforcers*) or points where information is routed, multiplexed, or demultiplexed based on its semantics (referred to as *policy selectors*).
2) Edges E which indicate that there is an uninterrupted data stream between two vertices.
3) A unique entry point and a unique target in the system, with no in-bound and no out-bound edges respectively.

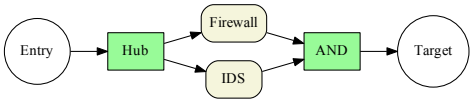
Each policy enforcer has associated with it an input language I and an output language O. The input language I of a mechanism is the set of inputs that the mechanism can accept as legal input. For example, syntactically correct network packets are the input language of a firewall. The output language O of the mechanism is the set of output which it will emit, given some input.

We say that two adjacent mechanisms *compose* when the input of one mechanism is a subset of the output of the other, and there is an edge between the two mechanisms. There are two key classes of composition, deterministic and non-deterministic.

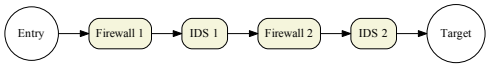
A deterministic composition of m_i, m_j is a policy enforcer m_{ij} that has an output language $O_{ij} \subseteq O_i \cap O_j$, and an input language $I_{ij} = I_i$. This is a composition where the order of application of mechanism is consistent and well-defined.



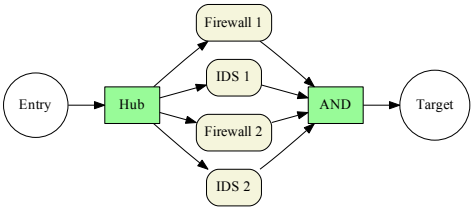
(a) D_1 - High Coverage, Low Redundancy, Low Independence



(b) D_2 - High Coverage, Low Redundancy, High Independence



(c) D_3 - High Coverage, High Redundancy, Low Independence



(d) D_4 - High Coverage, High Redundancy, High Independence

Fig. 2: The Four principal patterns in Defense Graphs (the empty pattern is omitted).

A non-deterministic composition of m_i, m_j is a policy enforcer n_{ij} that is formed when the two mechanisms compose in some unreliable order. For example, if two mechanisms operate on the same data, but their order depends on the scheduling by the operating system, then they are in non-deterministic composition. n_{ij} necessarily has that $I_{ij} = I_i \cup I_j$ and $O_{ij} \subseteq O_i \cup O_j$.

For any given enforcer m, we say that the coverage of m $C(m)$ is the set of all input that m instruments. In other words, $C(m) = I_m$. We emphasize that this is not a measure of the mechanism's effectiveness- this would require an effective method to decide if input is malicious or not, which is known [7] to be undecidable. The entire system's coverage is the union of its mechanism's coverage. See equation 1a.

Two mechanisms m_i, m_j have a certain degree of overlap, modelled by the redundancy property. The redundancy between two mechanisms m_i, m_j is the proportion to which their coverage overlaps, or more precisely the ratio of the size of their intersection to the size of their union.

Similarly, the redundancy of the entire system is the proportion of redundancy over all distinct mechanism pairs to the maximal possible redundancy (i.e. the number of distinct mechanism pairs). See equation 1b.

Given two mechanisms, it is either the case that the input of one mechanism depends on the output of another, or there is no dependence. We say that if there is a walk $m_i \dots m_j$ in D, then m_j depends on m_i, and set $I(i,j) = 1$. Otherwise, $I(i,j) = 0$. If $I(i,j) = 1$, there may be interference by m_i on the effectiveness of m_j. The independence of the system $I(D)$ is computed similarly to its redundancy; see equation 1c.

Cost is the last property in a system, which can model a number of different metrics such as financial or time cost of a mechanism. We use cost to model the time of execution, and define the cost of a system as the average difference in time between the unsecured system and the secured system. See equation 1d.

$$C(D) = \bigcup_{m \in M} C(M) \tag{1a}$$

$$R(D) = \frac{2 \sum_{i=0}^{|M|} \sum_{j \neq i}^{|M|} R(m_i, m_j)}{|M|^2 - |M|} \tag{1b}$$

$$I(D) = \frac{2 \sum_{i=0}^{|M|} \sum_{j \neq i}^{|M|} I(m_i, m_j)}{|M|^2 - |M|} \tag{1c}$$

$$P(D) = \sum_{i=0}^{|J|} \frac{T(D, j_i)}{T(D_0, j_i)} \tag{1d}$$

Figure 2 shows four principal patterns of Defense Graphs, which are idealized structures that demonstrate the relationship of these properties to the layout of a system.

Experiments with randomly generated network architectures being exposed to malicious input are planned. These experiments will be used to establish and support the accuracy of the model's predictions about a system, and reveal patterns and anti-patterns of composition in simple systems.

References

[1] "Defense in depth," http://www.nsa.gov/ia/_files/support/defenseindepth.pdf, National Security Agency.

[2] "Recommended practice: Improving industrial control systems cybersecurity with defense-in-depth strategies," https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/Defense_in_Depth_Oct09.pdf, USA Department of Homeland Security.

[3] M. Abadi and L. Lamport, "Composing specifications," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 15, no. 1, pp. 73–132, 1993.

[4] D. P. Quigley. (2007) Re: Linux security *module* framework (was: Lsm conversion tostatic interface). Linux Kernel Mailing List. [Online]. Available: http://lkml.iu.edu//hypermail/linux/kernel/0710.3/0546.html

[5] "AMTSO feature settings check," http://www.amtso.org/feature-settings-check.html, AMTSO.

[6] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[7] F. Cohen, "Computer viruses: theory and experiments," *Computers & security*, vol. 6, no. 1, pp. 22–35, 1987.