The Concept of Attack Surface Reasoning

Michael Atighetchi, Nathaniel Soule, Ron Watro, Joseph Loyall
Raytheon BBN Technologies
Cambridge, MA
{matighet, nsoule, rwatro, jloyall}@bbn.com

***Abstract*—Today's cyber defenses and cyber defenders face determined and diverse adversaries, who can study all aspects of deployed systems including networks, hosts, and the applications running on them, in order to find exploitable vulnerabilities and to devise attack vectors that exploit the detected vulnerabilities. The conflict between cyber attackers and cyber defenders is stacked against the defender. The defender must protect against all the ways that an adversary can cause potential loss of security, collectively called the *attack surface*, while the attacker needs to find only a single vulnerability and attack vector to be successful. This work-in-progress paper describes an AI-inspired approach for modeling and analyzing the attack surface of a distributed system. Once modeled, an attack surface can be quantified in terms of size and level of dynamism through four types of algorithms: path analysis, metric computation, path comparison, and path enumeration. Our approach supports relative comparison across multiple attack models for each combination of a system and a set of defenses, in order to select an appropriate set of defenses given a certain cost/benefit tradeoff.**

Keywords: security, semantic web, analytical models

I. INTRODUCTION

Today's cyber defenders face determined, diverse, and well-resourced adversaries who have a significant advantage over the defense. The adversaries need find only a single vulnerability and attack vector to be successful, while the defender must protect all vulnerabilities and defend against all attack vectors. Among the various choices of available defenses, a powerful class is the set of Moving Target Defenses (MTDs) that attempt to even the playing field for defenders by shifting the attack surface, i.e., the set of potential attack vectors an adversary can use to compromise security of a target system. MTDs attempt to change access paths before they can be exploited, thereby (1) making them more difficult to detect, (2) rendering attacks that are based on stale information ineffective, and (3) increasing detection by monitoring for the use of stale information.

However, as the number and complexity of defenses (including MTDs), system configurations, and potential attacks continually increase, cyber defenders face the problem of manually selecting and configuring defenses for a distributed mission-critical system without a clear understanding of the seams/integration points, residual risks, and costs (in terms of impact on performance and functionality). Integration of defenses performed in a non-structured way bears the risks of adding defenses with no value, inadvertently increasing the attack surface, or overly impacting critical functionality.

This paper describes work in progress for constructing a model-based environment for *Attack Surface Reasoning (ASR)*, i.e., integrating and evaluating systems and defenses and analyzing compositions of systems and dynamic defenses.

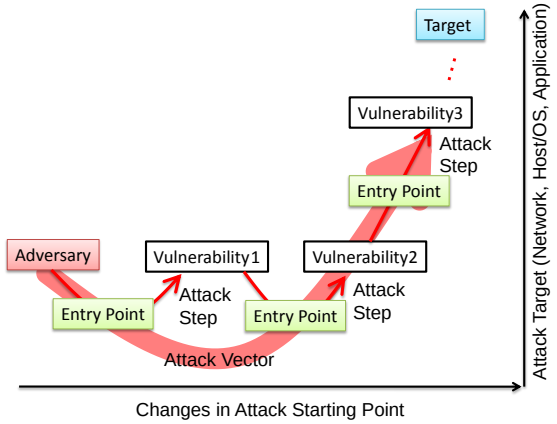

Fig. 1. Attack surface models include the starting points and the targets of attacks, which can exist at multiple horizontal and vertical layers, respectively.

The models, metrics, and algorithms used in measuring attack surfaces need to support the following key requirements:

The attack surface model needs to represent concepts and defenses situated at multiple layers. Attacks may target resources at the network layer (both network traffic observed on intermediary network components between the client and the server as well as the Transmission Control Protocol (TCP)/Internet Protocol (IP) stacks of servers and clients), the operating system and host layers (e.g., by running attacks against a Java Virtual Machine (JVM) compute platform), and the application layer (e.g., by corrupting Structured Query Language (SQL) tables). We refer to this requirement as *vertical layering*, as visualized by the attack vector in Fig. 1 as it moves from Vulnerability 2 to Vulnerability 3.

The attack surface model needs to capture ordering dependencies between control and data flows and defenses to be employed against attack vectors. Attacks consist of an orchestrated execution of individual attack steps, where the effectiveness of each attack step is contingent on the starting point (and level of privilege) in relation to the specific target. For instance, network attacks launched from a legitimate client have a significantly different starting point compared to the same attacks launched from an adversary-controlled device attached to the network. This knowledge should be part of the attack surface model. An example of ordering at a given vertical layer includes a firewall placed in front of a protected host, requiring all traffic to pass through the firewall first before reaching the host's TCP/IP stack. We refer to this requirement as horizontal layering, as visualized by moving from Vulnerability 1 to Vulnerability 2 in Fig. 1.

 ISBN: 978-1-61208-352-0

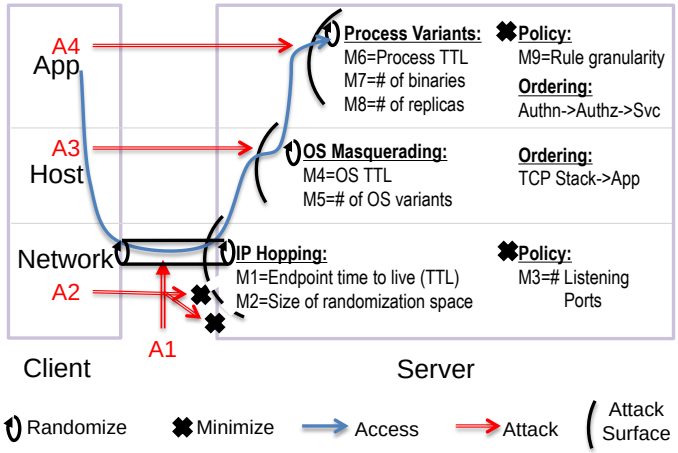

Fig. 2. Multi-layered attack surface randomization and minimization

The attack surface models need to capture complexities associated with dynamic defenses. Many of the attack effects described so far are difficult to detect automatically and cause observable events only when critical mission functionality starts failing, leaving little time for recovery and increasing the impact of the attack. While proactive shaping strategies, such as IP Hopping, Operating System (OS) masquerading, unpredictable replication, and Address Space Layout Randomization (ASLR) are clearly beneficial, the existence of multiple dynamic adaptations at different system layers makes management of the overall policy and configuration sets deployed on the target devices challenging.

Fig. 2 shows an example attack surface, consisting of a single Server composed with three MTDs: (1) IP Hopping that randomizes IP addresses between the client and server at the network layer, (2) OS masquerading that changes parameters in the OS stack to make one OS look like another, and (3) Process variants that generate different functionally equivalent versions of binaries and replicas. The overall goal of ASR is to quantify the attack surface along two main attributes – minimization and randomization – with the objective of minimizing access where possible and randomizing information about remaining access where affordable.

The algorithms to compute minimization rely on the system and mission models and metrics to determine how much of the attack surface really needs to be exposed to effectively function or, conversely, the parts of the attack surface that can be reduced or removed without adversely affecting critical functionality. For example, some applications with external facing interfaces (thereby providing entry into the system if penetrated) might not be needed during some phases of operation – or might not be as critical – and can be shut down, removing those entry points into the system and their associated vulnerabilities and attack vectors.

The algorithms to compute randomization execute over the models and metrics for a system deployment and a set of defenses, including MTDs, and compute how the defenses or combination of defenses change the attack surface.

The paper is structured as follows. Section 2 describes related work. Section 3 describes a high-level view of attack surface models. Section 4 describes algorithms for quantifying attack surface models. Section 5 concludes the paper.

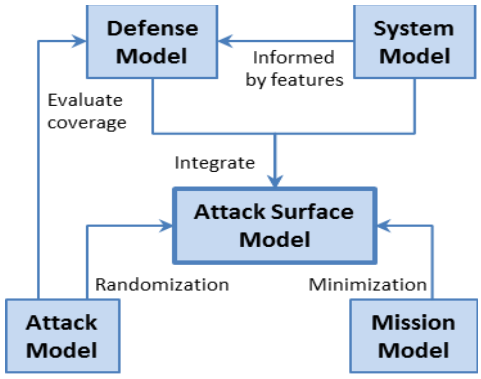

Fig. 3. Construction of the Attack Surface Model

II. Related Work

Our work relates to several efforts in the areas of cyber vulnerability modeling, light-weight formal methods, and mathematical attack surface definitions.

The Mitre Common Vulnerabilities and Exposures (CVE) [1] database maintains a comprehensive list of specific vulnerabilities that can be used to establish the attack model. Our definition of an attack surface extends the mathematical models presented in [2] by including vertical and horizontal layering as a central property of an attack surface. Decision support analysis systems for cyber defense such as [3] employ probabilistic techniques by annotating attack trees with defense information. This work focuses less on determining attack success probability over a given system, instead focusing on supporting comparative and recommendation based analytics. Performing quantitative reachability analysis has been studied extensively in the academic literature, using domain-specific languages such as Alloy [4], Lobster [5], or Cross Domain Entitlement Language (CDEL) [6]. The algorithms described in this paper operate on standards-based semantic web models rather than models described in proprietary languages. Our semantic web approach, using Web Ontology Language (OWL) [7], allows disparate sources of information to be automatically integrated [8] into a graph of interconnected information that is easy to understand and extend because it is based on real world concepts.

III. Attack Surface Model

Modeling an attack surface involves linking several different models, each representing aspects of the defended system. An attack model, describing goals of an attack, together with starting points, can be used to evaluate the attack surface for randomization attributes. The mission model, describing the set of required interactions in support of mission critical functionality, can be used to minimize access by pruning unnecessary access paths and highlighting those elements most important to mission success. Fig. 3 shows how a defense model and system model (shown in the top) together build the main ingredients of an attack surface model.

Fig. 4 displays a high-level ontological view of the defense model. It shows how a defense is described by a *setup* over a set of resources. For IP hopping, the setup contains the set of machines that have access to the hopping scheme's secret key and can therefore compute what the next IP address is going to

 ISBN: 978-1-61208-352-0

be. Attacks that start from the hosts that are part of the setup are not impeded in any way by the defense, since they all have access to the secret key of the hopping scheme and therefore perceive no randomization. Fig. 4 also shows that a defense provides protection for a set of resources, as expressed by the *Defense → provides → Protection* link. We model the type of protection provided in terms of overall security benefit, including confidentiality, integrity, availability, and discoverability. Furthermore, we express the mechanism of protection that the defense imposes. The *Protection → through → Filtering* link captures any security check that explicitly drops data, e.g., as it is being flagged by anomaly detection at various layers. The *Protection → through → Randomization* link captures aspects of dynamism associated with proactive shaping strategies. As shown in Fig. 4, the protections directly link to the scope of the defense, in terms of entry points, exit points, and data, which are key aspects of our definition of the attack surface. Finally, each defense has a cost associated with it (as shown to the right of Fig. 4). We include the fact that defenses can increase the attack surface via *Defense → adds → Data* and *Defense → adds → Point* links.

IV. QUANTIFICATION OF ATTACK SURFACES

The randomization, minimization, and other characterization metrics and analyses provided by ASR share an underlying common base of feasible-path analysis, but can be classified into distinct groups based on the operations that occur post-path-determination.

Metric Computation allows for calculation of metrics describing the randomization, minimization, or other characteristics of a point, path, or system, including the metrics in Tab. 1.

Path Comparison calculations execute path differencing and comparison algorithms to determine the set of elements in a system that are not required to support a given mission, and the disabling of which will help reduce the attack surface without degrading operation.

Path Enumeration analyses are undertaken to discover points, paths, and system configurations that exhibit certain properties, such as *all paths that contain defenses with dynamic frequencies less than 5 minutes*.

Fig. 5 depicts an example system model and defense model which, when integrated, form an attack surface model, along with an accompanying simplified attack model. The system model, shown in blue in the center, describes a single host with two network interface cards (NICs), through which a single service may be accessed. The service is also exposed internally on the host via an Application Programming Interface (API). A single defense has been modeled in the purple nodes at the top, in this case an IP hopping MTD named DYNAT [9]. The MTD has been configured to protect the IP address of *NIC 1* on *Host 1*. An attack class, shown in red at the bottom, has been modeled (in an abbreviated form) describing an attack category that is relevant to network endpoints. Given a model such as this, ASR's processing engine allows for the performance of many interesting analyses and metric computations. For example, in order to determine how many paths are protected by defenses that change more slowly than some known attack class's expected duration (and are thus not dynamic enough to provide robust protection), one may compute the metric *number of defenses with dynamic modulation frequencies greater than any applicable attack phase duration*. Calculation of this metric is accomplished by first performing a path analysis identifying all possible paths through the system – for this example binding the starting points to network entry points (thus ignoring insider threats) based on the given attack model domain, and binding the goal states to all "services." Given these starting points and goals, six possible attack paths exist. Fig. 5 shows three of these paths (paths 1, 2, 3), with the other three being symmetric mirrors, starting from *Network EP* 2. The ASR path analysis engine will identify the paths starting from the two network entry points (*Network EP 1*, and *Network EP 2*) and branching from there to go directly to Service 1 (via path 1 and its symmetric equivalent), indirectly to Service 1 through the API(via path 2 and its symmetric equivalent), or indirectly through the opposing NIC (via path 3 and its symmetric equivalent). At this point the identified paths are stored such that they may be reused for multiple queries in the second phase of analysis.

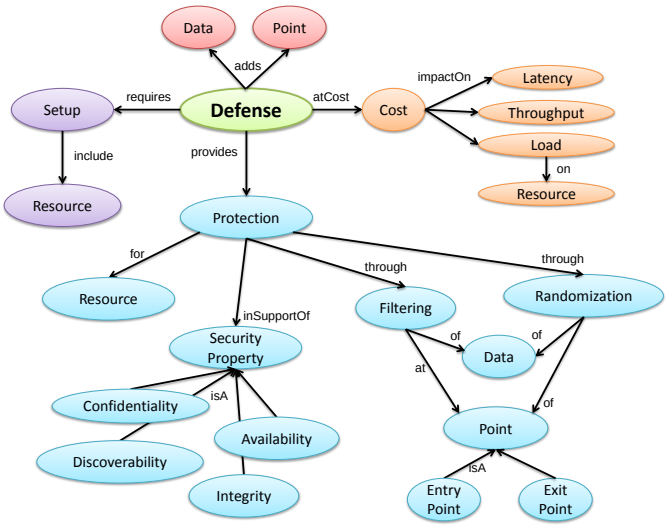

Fig. 4. Ontology representation of a generic dynamic defense

The second and third phases for the metric in question in-

TABLE I. HIGH-LEVEL METRICS

Randomization
Number of defenses with dynamic modulation frequencies greater than any applicable attack phase duration
Defense modulation frequency
Number of dynamic surfaces
Number of dynamic surfaces per protection type (confidentiality, integrity, availability, discoverability)
Minimization
Number of deployed defenses
Attack surface area change due to defenses
Number of defense boundaries crossed per path
Number of paths with less than N defenses
Rule granularity of defenses
Number of processes
Number of open ports
Number of users
Other Characterizations
Sum of path lengths from entry point to each defense
Is there at least 1 defense per known attack class
Number of paths with conflicting defense types
Number of entry points without a defense within 1 hop

 ISBN: 978-1-61208-352-0

volves executing SPARQL Protocol and RDF Query Language (SPARQL) queries (to identify all defenses meeting the prescribed criteria) and aggregation operations (here, a simple sum) over the set of paths. Three of the feasible paths (paths 1, 2, and 3) encounter an MTD whose randomization frequency is on the order of seconds. Since the attack in the simplified attack class model has an expected duration on the order of minutes, the MTD is determined to provide adequate protection (for the paths it covers) and thus this metric will have a value of 0, i.e., there are no defenses with dynamic modulation frequencies greater than the attack phase duration. However, determining the overall protection of this system should evaluate this metric in the context of other important metrics. For example, calculating the metric *number of paths with fewer than N defenses* would highlight that entry through *NIC 2* is unprotected. Further, had the metric been defined to include a larger set of starting points the resulting value would have been even larger, as insider attacks that start from a privileged base on *Host 1* will not pass through any defense.

In addition to the base path exploration, many other analyses may be performed. For example, the user may wish to elaborate on the numeric value calculated as part of the *number of paths with fewer than N defenses* metric by drawing from the enumeration category of analyses to ask ASR to ***identify*** *all paths that include less than one defense*. Again, a SPARQL query is defined to operate over the initial set of feasible paths, and select only those that include no defenses.

V. Conclusion and Next Steps

This paper describes work we are conducting on modeling and reasoning about attack surfaces of distributed systems, with the goal of selecting a properly configured set of defenses that together minimize access where possible and randomize observables where feasible. Our approach enables quantification of the attack surfaces for the purpose of performing relative comparisons between multiple surfaces, each one representing a set of defenses, system components, and attack classes.

In future work, we plan to develop and evaluate example scenarios, involving tradeoffs between multiple defenses at different layers, some providing great value in minimizing and randomizing the attack surface while others actually increase the attack surface or negatively impact availability by introducing excessive dynamism or cost.

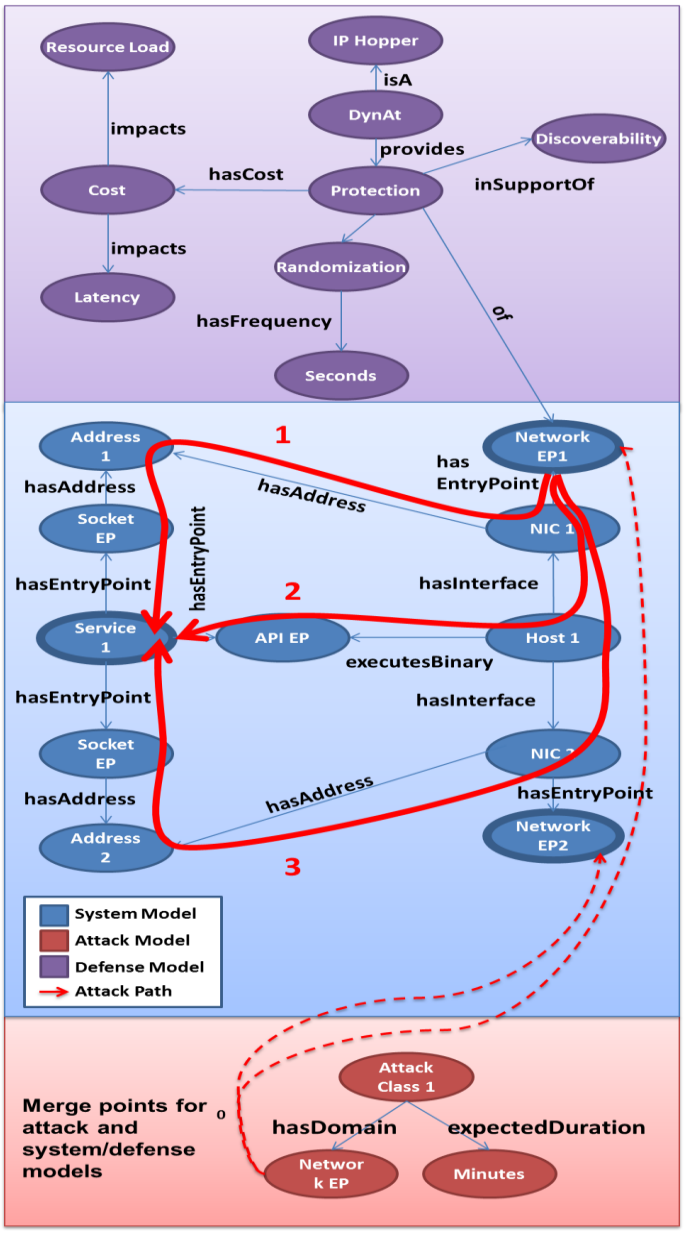

Fig. 5. Example attack surface model used for analysis

References

[1] Mitre, "Common Vulnerabilities and Exposures Home Page," 2014. [Online]. Available: http://cve.mitre.org/. [retrieved: 2014.03.11]

[2] P. K. Manadhata and J. M. Wing, "A Formal Model for a System's Attack Surface," in Moving Target Defense, Springer, 2011, pp. 1–28.

[3] T. Sommestad, M. Ekstedt, and P. Johnson, "Cyber Security Risks Assessment with Bayesian Defense Graphs and Architectural Models," presented at the Hawaii International Conference on System Sciences (HICSS '09), Hawaii, 2009, pp. 1–10.

[4] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song, "Towards a formal foundation of web security," in 23rd IEEE Computer Security Foundations Symposium (CSF), 2010, pp. 290–304.

[5] J. Hurd, M. Carlsson, B. Letner, and P. White, "Lobster: A domain specific language for SELinux policies," Galois Inc. internal report, 2008.

[6] J. Beal, J. Webb, and M. Atighetchi, "Adjustable autonomy for cross-domain entitlement decisions," in Proceedings of the 3rd ACM workshop on Artificial Intelligence and Security, 2010, pp. 65–71.

[7] D. L. McGuinness, et al., "OWL web ontology language overview," W3C Recomm., vol. 10, no. 2004–03, p. 10, 2004.

[8] M. Fisher and M. Dean, "Semantic Query: Solving the Needs of a Net-Centric Data Sharing." Semantic Technology Conference, 23-May-2007.

[9] D. Kewley, R. Fink, J. Lowry, and M. Dean, "Dynamic approaches to thwart adversary intelligence gathering," DARPA Information Survivability Conference & Exposition (DISCEX), 2001, vol. 1, pp. 176–185.

 ISBN: 978-1-61208-352-0