Improving Self Defense by Learning from Limited Experience

Karen Z. Haigh
BBN Technologies
10 Moulton St.
Cambridge, MA 02144
khaigh@bbn.com

Steven A. Harp
Adventium Labs
100 Mill Place, 111 Third Ave. S
Minneapolis, MN 55401
steven.harp@adventiumlabs.org

1 Introduction

Prevalence of new attacks or attack variants presents an interesting challenge for autonomic cyber-defense: how does the autonomic defense mechanism learn from previous failures, acquiring immunity with experience, and do so as rapidly as possible. In the limiting case, only a single a single observed failure may be available for learning.

In this paper, we describe an approach to the problem of learning rapidly from failures through a process of controlled experimentation. To sidestep the limited observations available to the learner, experimentation takes place using simulation and/or emulation of the defended systems.

We give two examples of this approach. The first, CORTEX, is a system that autonomously defends a critical service, in this case a MYSQL database. CORTEX operating without direct human intervention, can correctly generalize from a single novel attack, and immunize the database service from a wide range of similar attacks.

The second example, Cognitive Support for Intelligent Survivability Management, CSISM hereafter, is a reasoning system that can mount a knowledge-based defense of an entire network of defended machines. One of the learning components for CSISM, currently under development, also uses learning by experimentation to characterize the attack and evaluate alternative defensive tactics.

In both cases, machine learning techniques add a level of intelligence by deriving or identifying information that is key to effective defense.

2 Cortex: Learning at the Service Level

The CORTEX architecture consists of a defense planning module coupled with a rapid response module. The system uses a pool of internally identical "taste testers" that vet all service requests from clients. Once a service request has passed a taster with no ill effects, the service is replicated to the master and other tasters. Failures observed in a taste tester trigger defensive actions from the response module, and also stimulate a learning subsystem to analyze the attack and enhance defenses.

The goal of the CORTEX learning system is to autonomously formulate the most general defense from novel observed problems. While it is comparatively easy to circumvent a precisely identical attack or mishap, this protection is inadequate given the threat of an intelligent attacker who is ready to disguise an exploit in various ways. True generalization requires identifying the conditions that are necessary and sufficient for a problem to be reproduced, and translating these to the most general defense. To achieve this goal, the learning approach in CORTEX combines aspects of anomaly detection and learning by experimentation [2, 4, 7, 8].

The sequence of operations given in Figure 1. The first step is to differentiate between normal and abnormal traffic. Operationally, abnormal traffic is that which leads to an error condition in the service, as determined by the service itself or by an external watchdog. Normal traffic, i.e. not associated with errors, contributes to a model of service operation that is used to detect anomalies.

Abnormal traffic triggers an examination of the transaction that led to the error. Experiments with CORTEX used various versions of the MYSQL database as the defended service. The multi-

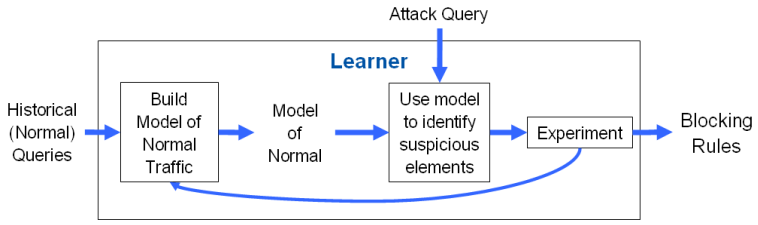

Figure 1: The learner builds a model of normal traffic to measure suspicion scores.

dimensional space of possible queries and commands in the MySQL protocol provides the context for assessing traffic. Potentially, any term in the query may hove induced the error, thus the learner assesses the suspiciousness of each term by comparing them to the model of normal traffic, and sorts these suspicious elements for experimentation.

The second step is to generalize from an observed instance of a failure to its most general form. To do this, the learner autonomously *experiments* with each suspicious element to determine which factors underly the problem, and what are the boundaries. The taster pool provides a laboratory for the learner to safely conduct experiments on the susceptible population. Each taster is a fully operational version of the service, but isolated from the rest of the on-line system. The learner is allowed to use one or more tasters for its experiments as conditions permit. The final step is to generate blocking rules for the newly mapped problem. Blocking rules are dynamically loaded into the proxy as soon as the rule has been validated, incrementally generalizing the attack so as to avoid blocking valid traffic.

3 CSISM: Learning Multistage Attacks

CSISM is an effort to build an autonomous defender that can take the place of a human network operations team against a sophisticated attacker. Its 45 host test bed is a simulation of a well-armored version of the Joint Battlespace Infosphere system (DPASA) that was originally built under the DARPA OASIS program. A two-week red team exercise in that program examined the survivability of this system in an operational environment, and demonstrated that it is possible to create a Information System that can survive against sustained attacks by a sophisticated adversary. However, that exercise included an expert team of live defenders. CSISM will attempt to defend against a similar range of attacks without human intervention.

Experimental learning is one approach being studied in CSISM. CSISM may be construed as a control system consisting of an inner rapid-response controller and an outer cognitive controller. The learning component ("learner") receives a set of observations from the control system and generates a set of detection and defense rules that the controller can use for future decisions. As sketched in Figure 2, the controller collects a set of sensor observations that identifiably contain an attack (observations 1-6). The *Attack Theory Experimenter* maps these observations to actions (A-D) whenever possible. Using the sandbox, the ATE then varies the initial conditions, reorders these actions, and varies the actions to identify which subsets, orders, and varitions are the key components of the attack. These viable theories are given to the *Defense Measures Experimenter* to explore possible defense strategies. The results are turned into detection rules and defense strategies.

The key challenge in learning generalized multi-step and distributed attacks is to recognize which observations indicate the necessary and sufficient elements of an attack (credit assignment). The *minimal subset* of actions is the smallest sequence, extracted from the complete list, that cause the attack to succeed. One sequence of observations may have more than one minimal subset, for example an attack with two high level goals. In practice, these key elements will always be surrounded by *incidental* observations that are either side effects of normal operations, and *probabilistic actions* that may succeed with some probability or improve the overall success of the

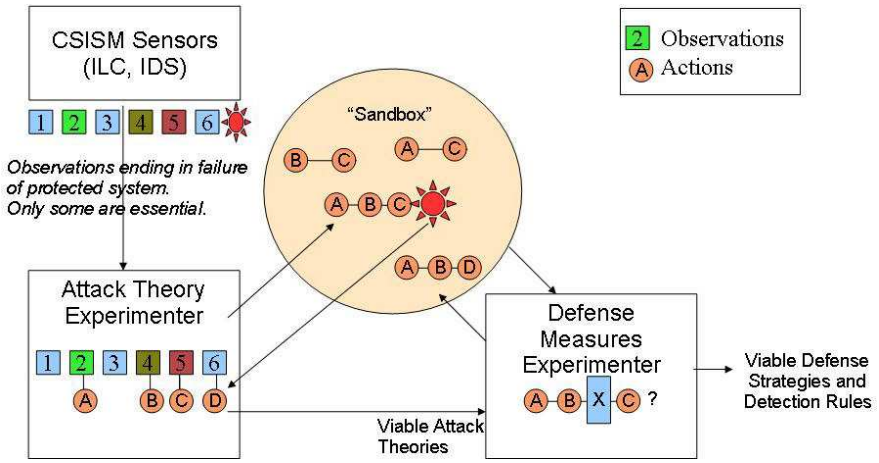

Figure 2: CSISM Learner Architecture

attack. *Chaff* actions may be explicitly added by an attacker to divert the defender.

In a practical system, the interesting issue is to generate the most efficient sequence of experiments. The complete set of theories is all permutations of all members of the powerset of the observations; for an observation sequence of n actions, there are $\sum_{i=1}^{n} {}_nP_r$ possible valid subsequences. For a short sequence of $n = 10$, that means approximately 10 million possible subsequences to test (not including variations on initial conditions). Thus an effective learner must have good heuristics that explore high quality theories early. The current prototype uses heuristics that favor shorter attacks and give credence to step ordering and interaction effects. Hypotheses for testing are generated incrementally as with CORTEX, and results can be harvested as needed.

4 Discussion

The learning approach we have described enhances an autonomic cyber-defense system by giving it the ability to autonomously learn for defenses problems not fully anticipated by its designers. The CORTEX learner can successfully learn to block attacks on a defended service from single observations. Moreover, by generalizing from one known attack, it can learn to block an entire class of closely related attacks. The CSISM learner attempts to translate this capability to multi-stage attacks over a larger system.

Several techniques of automatic learning are employed. It incorporates *expert knowledge* in that the abstract classes of axes of vulnerability are designed by experts. It uses *experiments* to determine culprits and boundary conditions, generalizing as much as possible from each attack instance. This can be seen as a species of *query learning* of concepts [1], in which a learner can pose membership queries to an "oracle," learning whether a proposed concept is actually a valid attack (or defense). The oracle in each case is a stand-in for the real system. In the case of CORTEX, a sacrificial taste tester is a full fidelity clone of the defended service. In the case of CSISM, the learner employs a sandbox simulation of the network.

Finally, these learners employ *anytime algorithms* [3, 5] that learn in the background while the rest of the system continues to operate. The learned defensive rules improve with experimentation time. The technique appears to be relevant to any domain where experimentation can help improve the models of the system. While designed as part of a larger system for on-line defense, this learning approach may have off-line applications as well, e.g. the analysis of attacks on honey pots, as a "red team" analysis toolkit, or an adjunct to systems for fuzzing [6] or vulnerability discovery.

References

[1] D. Angluin. Query and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[2] Y. Anzai and H. A. Simon. The theory of learning by doing. *Psychological Review*, 86(2):124–140, 1979.

[3] T. Dean and M. Boddy. An analysis of timedependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49–54, 1988.

[4] Y. Gil. *Acquiring domain knowledge for planning by experimentation.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1992.

[5] J. Grass and S. Zilberstein. Programming with anytime algorithms. In *IJCAI-95 Workshop on Anytime Algorithms and Deliberation Scheduling*, pages 22–27, 1995.

[6] J. Koziol, D. Litchfield, D. Aitel, C. Anley, S. Eren, N. Mehta, and R. Hassell. The art of fuzzing. In *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*, chapter 15. John Wiley & Sons, March 2004.

[7] D. Pearson. *Learning Procedural Planning Knowledge in Complex Environments.* PhD thesis, University of Michigan, Ann Arbor, MI, 1996.

[8] X. Wang. *Leaning Planning Operators by Observation and Practice.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996. Available as Technical Report CMU-CS-96-154.