Poster: Bottleneck: A Generalized, Flexible, and Extensible Framework for Botnet Defense

Naurin Rasheed Ramay*, Sheharbano Khattak†, Affan A. Syed‡
Systems and Networks Lab (SysNet),
National University of Computer and Emerging Sciences,
Islamabad, Pakistan
Email: {naurin.ramay,sheharbano.k,affan.syed}@sysnet.org.pk

Syed Ali Khayam§
xFlow Research, 1392 Borregas Ave,
Sunnyvale CA, USA
Email: {ali}@xflowresearch.com
*Graduate Research Assistant †Graduate Student
‡Assistant Professor §Industrial Collaborator

I. Introduction

Botnets remain a persistent and evolving threat on the computer security landscape. Most existing botnet detection techniques rely on detection of specific botnet characteristics. Consequently, they can neither adapt to different types of botnets nor combine different detection techniques. Moreover, existing techniques do not have integrated defense mechanisms that can be triggered to curb the threat after detection.

Botnets exploit the rigidity of existing detection systems and evade them by using several infection, attack and control communication vectors [1]. Since the botnet threat landscape continues to evolve faster than the detection strategies, we advocate the need for a botnet detection framework which is: (1) General and flexible in detecting different classes of botnets while adapting to deployment requirements; (2) Extensible to the evolving threat model by providing seamless integration of new detection techniques; (3) Integrated with defense strategies which can be triggered after detection and classification of the threat.

In this poster we propose *Bottleneck*, a framework that meets all the design objectives set above. We realize an instance of this general framework using a Bayesian network which allows the system to make evidence-based predictive and diagnostic inferences for bot infections. We extend the Bayes net to an influence diagram which automates the defense strategy by optimizing a user-defined utility function over the detection and classification of the bot.

II. Bottleneck: A Generalized, Flexible, and Extensible Botnet Defense Framework

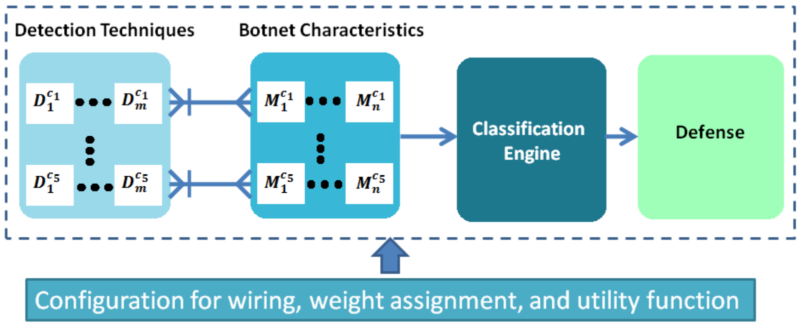
Detection Techniques
Botnet Characteristics
$D_1^{c_1}$ ••• $D_m^{c_1}$
$D_1^{c_5}$ ••• $D_m^{c_5}$
$M_1^{c_1}$ ••• $M_n^{c_1}$
$M_1^{c_5}$ ••• $M_n^{c_5}$
Classification Engine
Defense
Configuration for wiring, weight assignment, and utility function

Fig. 1. Bottleneck: An extensible framework for botnet detection.

Bottleneck as shown in Figure 1, leverages the observation that a set of five botnet characteristics — propagation, rallying, C&C, attack, and evasion — provide a complete and time-invariant high-level characterization of the botnet phenomena. Let $\{c_1, \cdots, c_5\}$ denote these characteristics. Botnet creators employ a number of mechanisms to implement each characteristic. For a given characteristic c_k, let $\{M_1^{c_k}, \cdots, M_n^{c_k}\}$ represent the set of existing and detectable mechanisms. For example, {scanning, client exploit, social engineering} represent a subset of all possible botnet propagation mechanisms, M^{c_1}. These mechanism implementations, for any given characteristic, can be detected by one or more of m detection techniques represented by, $\{D_1^{c_k}, \cdots, D_m^{c_k}\}$.

We make use of *Crow's Feet* notation to demonstrate the many-to-many relation between detection techniques and mechanisms for each botnet characteristic. The exact relationship between the detection techniques and these mechanisms, and the weight assigned to each relation, are configurable parameters in our framework. Thus, our framework is general but configurable for different types of botnets. This framework offers flexibility by adapting to different user requirements. For instance, location of deployment (host vs. network) or accuracy (precision vs. speed) can be implemented by the framework by simply changing configurations and weight assignments. Furthermore, the design is also extensible with the provision to easily add new detection techniques or mechanisms. The classification engine uses a learning mechanism (like Bayesian) to combine the detection of different mechanisms and can correlate different characteristics. Note that this classification can incorporate both vertical (or temporal) as well as horizontal (or spatial) correlation, depending on its deployment location. Furthermore, we can analyze detection results to give supporting evidence such as suspected malicious IPs/URLs, botnet types (IRC,HTTP,P2P), and C&C communication flows. This evidence is utilized to select the best strategy for defense.

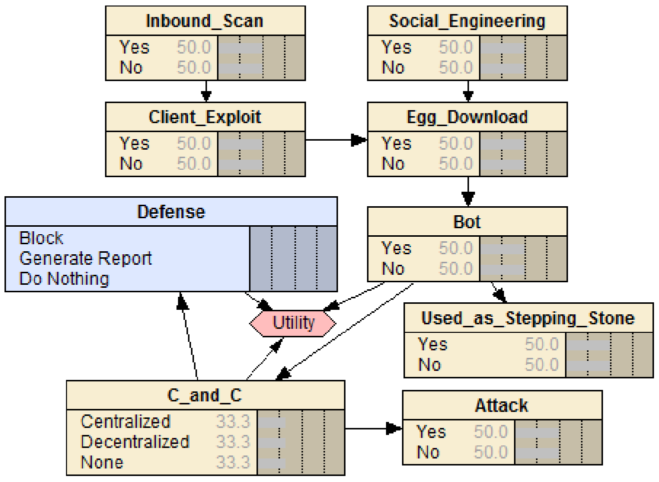

Fig. 2. An example implementation of Bottleneck using a Bayesian network.

III. An implementation of Bottleneck using Bayesian Networks

In this section, we describe a prototype Bayesian network implementation of the proposed Bottleneck framework. Bayesian network is a graphical model that encodes probabilistic relations among random variables using a Directed Acylic Graph (DAG). The nodes of the graph (uncertainty nodes) represent the random variables, and the edges represent causal relationships between them [3]. A directed edge $E(V_1, V_2)$ specifies that V_2 is directly dependent on V_1. In order for a DAG to be a Bayesian network, each node must be conditionally independent from all its non-descendants in the tree, given its parents. In addition to the graph structure, a conditional probability distribution is specified at each node.

A Bayesian network can be extended to an influence diagram, which can model inference as well as decision problems by adding decision and utility nodes. Parents of a decision node reflect information available at the time of a decision, while the utility function calculates the utility of a decision, given the values of uncertainty parent nodes of the utility node [4].

Figure 2 gives an influence diagram that instantiates the generalized Bottleneck framework. The uncertainty nodes in the diagram are either botnet characteristics or some specific mechanisms used to implement them. The causality of botnet life cycle is captured by edges between the nodes. The value of an uncertainty node is determined by aggregating one or more of its detection techniques. The results from these detection techniques can be combined using conjunction, disjunction or weighted sum. For example, three detection techniques for portscan are aggregated in [2] using Complex Event Processing, and their results are combined using weighted sum. The approach for achieving the highest accuracy using this combination is a topic for future work.

The decision and utility nodes are given by *Defense* and *Utility* respectively in Figure 2. Our choice of defense strategy is based on bot detection and type of C&C communication. We provide an example utility function for defense in Table I. In this example, the desired actions for centralized and decentralized C&C are to block the IP and generate a report, respectively. To achieve this, we scale utility values according to our preference with positive values ranking higher. A high negative value is assigned to blocking when no C&C is detected because it disrupts normal internet activity of user. Similarly, generating reports for centralized C&C creates unnecessary work for the network administrator. In contrast to these unfavorable decisions, we have certain situations where the second best decision is also acceptable. For instance, while blocking a decentralized C&C is not very effective, it does not yield any adverse results for the user. The last row aims to minimize false negatives of our defense system. Moreover, since we expect to have very low false positives in our ground truth data, the right most column summarizes the cases $(centralized, \overline{bot})$, $(decentralized, \overline{bot}$, $(none, \overline{bot})$ to $(-, \overline{bot})$. The utility values for each choice of action is given as well.

TABLE I
Utility Definition for Defense

	Centralized,bot	Decentralized,bot	None,bot	$-,\overline{bot}$
Block	100	10	-50	-100
G-Rep	-80	100	70	-150
Nothing	-200	-200	-100	100

The Expected Utility function is given by $EU(D|C) = \sum_B P(B|C) \times U(C, D, B)$ where C=C&C, D=Defense and B=Bot. The optimal policy for defense given a particular assignment for C will be the maximum of the Expected Utility values.

IV. Conclusion

In this poster we address the need for a generalized framework for botnet defense by proposing Bottleneck. Our framework combines several mechanism detection techniques and merges evidence from various botnet fronts. This approach promises a flexible and extensible system that can be configured by users to meet their specific needs. Furthermore, we use an influence diagram to realize one instance of the framework, which is also self improving and automates defense using a pre-defined utility function.

References

[1] Emerging Cyber Threats Report for 2011, Georgia Tech Information Security Center. http://www.gtisc.gatech.edu/pdf/cyberThreatReport2011.pdf, 2010. [Online; accessed 13-January-2012].
[2] Leonardo Aniello, Giorgia Lodi, and Roberto Baldoni. Inter-domain stealthy port scan detection through complex event processing. In *Proceedings of the 13th European Workshop on Dependable Computing*, EWDC '11, pages 67–72, New York, NY, USA, 2011. ACM.
[3] David Heckerman. Learning in graphical models. chapter A tutorial on learning with Bayesian networks, pages 301–354. MIT Press, Cambridge, MA, USA, 1999.
[4] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.