

# An Architecture for Network Security using Feedback Control

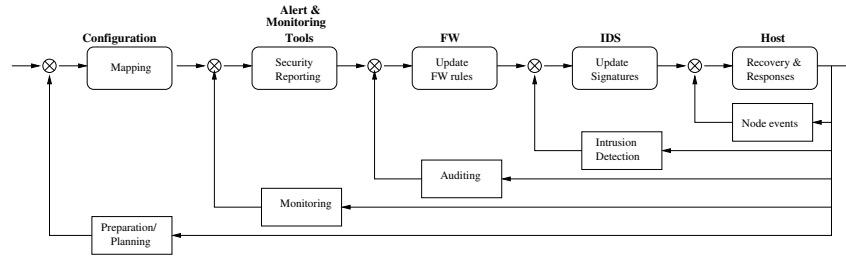
Ram Dantu<sup>1</sup> and João W. Cangussu<sup>2</sup>

<sup>1</sup> Department of Computer Science  
University of North Texas  
`rdantu@unt.edu`

<sup>2</sup> Department of Computer Science  
University of Texas at Dallas  
`cangussu@utdallas.edu`

In the past active worms have taken hours if not days to spread effectively. This gives sufficient time for humans to recognize the threat and limit the potential damage. This is not the case anymore. Modern viruses spread very quickly. Damage caused by modern computer viruses (example - Code red, sapphire and Nimda) is greatly enhanced by the rate at which they spread. Most of these viruses have an exponential spreading pattern. Future worms will exploit vulnerabilities in software systems that are not known prior to the attack. Neither the worm nor the vulnerabilities they exploit will be known before the attack and thus we cannot prevent the spread of these viruses by software patches or antiviral signatures. Hence there is a need to control fast spreading viruses automatically since they cannot be curtailed only by human initiated control. Some of the automatic approaches like quarantining the systems and shutting down the systems reduce the performance of the network. False positives are one more area of concern. Feedback control strategy is desirable in such systems because well-established techniques exist to handle and control such systems. Our technique is based on the fact that an infected machine tries to make connections at a faster rate than the machine that is not infected. The idea is to implement a filter, which restricts the rate at which a computer makes connection to other machines. The delay introduced by such an approach for normal traffic is very low (0.5-1 Hz). This rate can severely restrict the spread of high-speed worm spreading at rates of at least 200 Hz. As a first step, we apply feedback control to the first level of hierarchy (i.e., host). We will then expand the model to further levels (e.g., firewalls, IDS) as shown next in the description of the system architecture.

**Architecture:** It is assumed a secured network consists of firewalls, sensors, analyzers, honey pots, and various scanners and probes. These components are either separate elements or collocated with hosts, servers, routers and gateways. In this architecture, a (centralized or distributed) controller is responsible for collection and monitoring of all the events in the network. This controller is knowledgeable about the network topology, firewall configurations, security policies, intrusion detections and individual events in the network elements. This controller is logical function and can be deployed anywhere in the network. As shown in Figure 1, the controller communicates with clients located in different network elements.



**Fig.1.** Controller architecture for end-to-end security engineering (FW: Fire Wall, IDS: Intrusion Detection System).

Clients are responsible for detection and collection of the events in the node and communicate to the controller. Subsequently, controller will run through the algorithms, rules, policies and mathematical formulas (transfer functions) for next course of action. These actions are communicated to the clients. As described in Figure 1, the architecture evolves from a concept of closed loop control. Changes regarding the security behavior are captured and mixed with the incoming network signals. This piece of information is used to formulate the next course of action. The final result is outcome from multiple loops and integration of multiple actions. The response times within each loop (we call them defender-loops) varies from few milliseconds to several tens of minutes. For example, nodal events like buffer overflows, performance degradation can be detected in matter of milliseconds. On the other hand, it may take several seconds to detect failed logins, changes to system privileges and improper file access.

The anomalies and the measurements from the hosts and servers are fed back to the intrusion detection systems and firewalls. We have represented velocity of newly arriving requests using a state space model. In particular, we have used velocity of arrival of new connections for detecting abnormalities. These new connections are categorized as either safe or suspect connections and enqueued in two different queues. Suspected connections are delayed for further processing. These suspected connections are dropped either because of higher rate of arrival or application layer protocol timeouts.

The size of both the queues follows an S-shaped behavior similar to the input function. However, safe queue saturates far earlier than the delay queue. This is because of the entries in the delay queue are dropped due to application timeouts. The velocity increases initially until an inflection point is reached and it starts to decrease until the saturation level in the safe queue is achieved and finally velocity goes to zero. To regulate the number of connection requests, a PI (proportional and integral) controller has been used. Due to PI control the detection of abnormality at different points on the S-curve will reach the same saturation point. This behavior is due to the fact that the system is stable, i.e., it will converge to the same equilibrium point even when starting with different initial conditions. We found that these results are similar when the controller is applied at the firewall or at the host in the feedback loop.