Artificial Diversity as Maneuvers in a Control Theoretic Moving Target Defense

Jeff Rowe[1], Karl N. Levitt[1], Tufan Demir[1], Robert Erbacher[2]
[1]Department of Computer Science, University of California at Davis, CA, USA
[2]U.S. Army Research Laboratory (ARL), Adelphi, MD, USA

Abstract

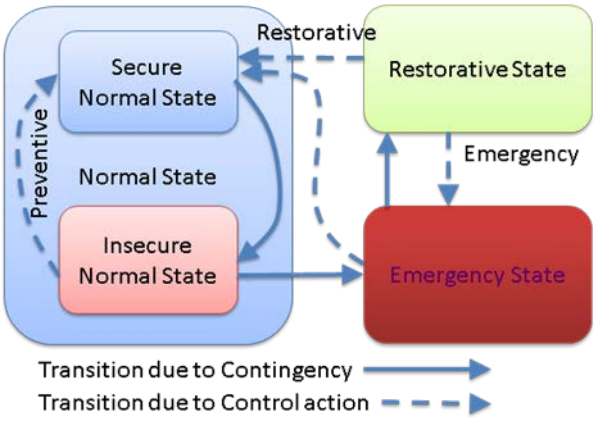
Moving target cyber-defense systems encompass a wide variety of techniques in multiple areas of cyber-security. The dynamic system reconfiguration aspect of moving target cyber-defense can be used as a basis for providing an adaptive attack surface. The goal of this research is to develop novel control theoretic mechanisms by which a range of cyber maneuver techniques are provided such that when an attack is detected the environment can select the most appropriate maneuver to ensure a sufficient shift in the attack surface to render the identified attack ineffective. Effective design of this control theoretic cyber maneuver approach requires the development of two additional theories. First, algorithms are required for the estimation of security state. This will identify when a maneuver is required. Second, a theory for the estimation of the cost of performing a maneuver is required. This is critical for selecting the most cost-effective maneuver while ensuring that the attack is rendered fully ineffective. Finally, we present our moving target control loop as well as a detailed case study examining the impact of our proposed cyber maneuver paradigm on DHCP attacks.

1 Introduction

Moving target cyber-defense systems encompass a wide variety of techniques in multiple areas of cyber-security. The dynamic system reconfiguration aspect of moving target cyber-defense can be used as a basis for providing an adaptive attack surface. Making this approach difficult is the large software monoculture in common use that provides a stable, widespread attack surface that is difficult to reconfigure in proprietary off-the-shelf systems. This dynamic system reconfiguration is informed by intrusion detection systems as to the current overall attack state. Intrusion detection systems also identify which system reconfiguration is appropriate for reduction of the attack surface against the currently appearing threats. The problem here is that intrusion detection is imperfect and can lead to costly overreactions by a moving target system that will have minimal effect on the security of the system. We need a moving-target defense system with a variety of potential reconfiguration techniques, which can use all available information about the system's security to estimate the current state, and the ability to take actions that enhance system security overall.

We view moving target cyber-defenses as a problem in optimal secure reconfiguration. How can networks and devices be adaptively controlled, using all available cyber-threat information, to maintain a secure state? Our approach is to model the cyber-defense as a control system using sound control-theoretic principles. To implement this vision, we need a closed moving target control loop. In particular, we need:

- *Maneuvers* that form a constantly changing attack surface and can provide transitions to future system states, with associated estimated *costs* for any resulting loss in service
- *Security state estimation* based upon any available security information, which would be used to select appropriate maneuvers, and is associated with an estimated *cost* for the associated loss in security
- *Minimal cost maneuver selection* procedures that move the entire system towards the most secure state possible with minimum cost overall



In traditional cyber-security analysis, the system is modeled using two states, i.e., secure and compromised.

When modeling the defenses as a control system, the cyber-maneuver system would have *Normal*, *Emergency* and *Restorative* states. Normal states are further subdivided into *Secure Normal* and *Insecure Normal*. When the system is operating normally, with no danger of compromise, it is Secure Normal. The system operating normally but with indications of attack behavior, perhaps from an intrusion detection system or other runtime monitors, is in the Insecure Normal state and a preventative control action must be taken to return it to Secure Normal. Otherwise, it will eventually transition to an Emergency state. Emergency states occur when the system's security is at least partially compromised but the system still provides service. Once the Emergency state is entered, control actions must be taken to move the system into the Restorative state, where services are interrupted in order to block the malicious activity. Once the Restorative state is entered, new secure resources are brought to bear to transition back into full Secure Normal operations. This type of model is widely used in power systems as the main control loop [1] and is designed to handle a wide variety of unanticipated disruptions. We believe this is a promising model to serve as the basis for a moving target defense control loop to classify and select relevant actions to take in anticipation of a variety of uncertain security conditions.

In this paper, we describe a method for injecting artificial diversity into systems for use as cyber maneuvers in a moving target defense. The moving target defense we employ uses control theoretic principles. We describe our secure system state estimation procedure performed with all available, albeit imperfect, intrusion detection information. Finally, we discuss maneuver selection as a minimal cost closed loop optimized control procedure.

2 Artificially Diverse Cyber-Maneuver Techniques

Optimization of the moving target control loop depends upon having a collection of cyber-maneuver techniques from which to choose, each with its own specific relevance and cost. In order to provide a principled way to select from among these maneuvers, we classify specific techniques according to assumptions (a.k.a, preconditions) and effects. Maneuvers are effective in protecting and restoring the system by violating assumptions made by the attacker, hence requiring some modeling of the attacker's capabilities. For example, buffer overflow attacks against a particular vulnerable application assume that the target application has the same memory layout as the application for which the exploit was developed. For effective deployment, many maneuvers depend on a *shared secret* that is intended to be unavailable to the attacker. If it is believed this secret is known (even partially) by the attacker, it must be changed, similar to the way a cryptographic key is revoked and changed. In this way, a moving target defense is achieved since the information on pre-conditions that an attacker needs to compromise the system is continually changing.

What is needed is a mechanism similar to that used to encrypt information for confidentiality but instead used to change the configuration of systems such that the assumptions (perhaps not explicitly known) built into the exploit used by the attacker are invalidated, rendering the attack ineffective. Of the many possible mechanisms for achieving a moving target defense, one mechanism we have investigated is the creation and injection of artificial system diversity. In vulnerability monoculture environments, where the system is composed of large numbers of nodes running identical software and communicating using identical protocols, the potential for widespread compromise and disruption is enormous. One of the main assumptions of an attacker is that specific technical details of system operation remain the same across all targeted machines and networks. If each machine and network could generate a custom configuration, using a secret not available to the attacker, when an insecure normal, emergency, or restorative state are entered, then a constantly moving vulnerability profile could be achieved to stay ahead of adaptive adversaries. Several moving target defenses developed previously can be classified as artificial diversity. This includes binary transformations and network diversity transformation.

2.1 Binary Transformations

In our previous work [3], we modified MS Windows binary images to randomize memory locations, preventing execution of attacker-injected code. Modifications to the Windows kernel binary produced randomized system call numbers, DLL memory layout and kernel data structures to prevent any code injection attack from making system calls. The loader is also modified to provide correct memory mappings for locally run applications. For most code injection attacks, a malicious binary is crafted assuming that critical OS resources are in the same location across most systems. Randomization of these locations breaks the attacker's assumptions, causing the attack to fail. The defender assumes that although the attacker can inject code using some unknown vulnerability, the injected exploit is incapable of static analysis of memory to uncover the system binary locations. The shared secret in this case is the deployed memory randomization.

2.2 Network Diversity Transformation

As described in [2], network address space randomization (NASR) has been proposed as a way to counter malicious worm attacks. Network hosts can randomly hop to different IP address assignments to thwart hit list worm propagation. The assumption made by the attacker is that the host found to be vulnerable during hit list generation will reside at the same IP address at attack time. NASR violates this assumption and the attack fails. When selecting the NASR technique, the defender assumes that the attacker has performed reconnaissance to construct the hit list and little information is available regarding this hypothesis. In the absence of information regarding specific reconnaissance attack instances, this assumption leads to a decision to perform continuous NASR, provided it is not too costly. Another key assumption when using NASR is that the attacker is unable to access the defender's shared secret that maps fixed host identities to mutable IP address assignments. If information is available indicating that the shared secret may be compromised, NASR will not be effective in maintaining a secure state, so the randomization must be modified.

3 Generating Diverse Network Protocols

Local code obfuscation and diversification protects single hosts from external attack (and even internal attack, where a local user attempts to gain greater privilege). Many attacks, however, exploit weaknesses in the network protocols that regulate services between multiple hosts on a network. The end hosts will most likely be configured very differently and will be running different services, which themselves will be based upon different versions of software and operating systems. By exploiting the vulnerabilities in a network protocol, an adversary may be successful even in an environment with a large degree of host system diversity. We present a novel technique for introducing heterogeneity into common network protocols in ways that will thwart an outside attacker, while leaving the normal network operations of the system unchanged.

Our approach is motivated by our previous work in specification-based intrusion detection [6]. For this work, we developed specifications for several common network protocols that describe their correct operations under normal conditions. Network protocols were specified as state transition diagrams. For simple protocols, such as the Address Resolution Protocol (ARP), a single state diagram could be used to specify the behavior of both client and server on the network [7]. Complex protocols, such as the Dynamic Host Configuration Protocol (DHCP), require a separate specification be created for the client and the server. For intrusion detection, deviations from this expected behavior is cause for alarm and provides a way to detect attacks without reliance on known signatures or statistical anomalies. In this paper, however, we describe a method that uses these protocol specifications, not for intrusion detection, but generating a diverse set of network protocols that:

- are identical in functionality to the common network standards,
- are diverse in network message sequencing as well as message content,
- can be implemented in a separate network proxy so that no modification of existing operating system code would be required, and
- could be generated and changed automatically, even dynamically during run time to thwart an attack.

We base our approach on the state-machine specification for the correct protocol behavior. The states of a protocol and an existing transition path between them would become the invariants that must be preserved in any transformation. If these invariants hold in any new protocol, then we assert that the new protocol will be identical in functionality to the commonly implemented network standard. Using this technique, an existing transition between protocol states would be replaced by new states connected by new transitioning messages, all of which eventually lead to the final invariant state. This effectively creates a new network protocol that is different from the standard but with the same functionality. Performing this operation with different transitions and between different states allows us to transform existing, commonly implemented network protocols, into a diverse set of network protocols, each with a different set of vulnerabilities but identical functionality.

Changing the source code (or executable if source is unavailable) of the various networked systems to implement a new protocol would be a daunting task, if it were even possible. With this in mind, we are developing a protocol proxy whose external network communications conform to the transformed non-standard protocol, but passes the standard invariant portion to existing client or server implementations. All of the standard applications, configuration files and tools would still be used to configure the standard protocol handler processes and daemons. The proxy would handle all of the variations in protocols, passing only the invariant states and their immediate

transitions from the specification to the local client or server processes. In this way, no modification of any existing code running on the host would be required. For coordination of client and server proxies, we are investigating the use of existing off-the shelf security synchronization systems. In government systems, for example, connecting remotely requires either a SecureID card or a CAC card; both provide authentication mechanisms. SecureID is interesting in that it provides a unique synchronized key between the client and the server. We envision incorporation of this as part of a cyber-maneuver. When connecting to a system with an artificially diverse configuration, the SecureID card algorithm is used to identify the shift of the diversity mechanism to the next shared secret state. This ensures the main externally visible attack surface will be almost completely obfuscated and inaccessible.

4 An Example with the DHCP Protocol

The Dynamic Host Configuration Protocol is used by services that automatically provide clients with the basic parameters needed to configure their network interfaces. In the most common configuration, hosts joining a local network have DHCP clients that obtain an IP address, DNS server address and Internet Gateway address from a local DHCP server. This alleviates the need for users to manually add or modify these parameters when joining a network for the first time, or when changing locations in a mobile environment. The procedure used to automatically obtain these parameters is as follows:

1. The client broadcasts a DHCPDISCOVER message on its local network. There may be more than one DHCP server on the network. For this broadcast message to be passed over all subnets of this network, a relay agent will forward the DHCP message.
2. The server that received the DHCPDISCOVER message responds with a DHCPOFFER message that includes an available network address. At this stage, the server has to broadcast its response to the whole network because the client does not yet have an IP address.
3. After receiving one or more DHCPOFFER messages from DHCP server(s), the client chooses one based on the parameters in the DHCPOFFER messages. The client broadcasts a DHCPREQUEST message that includes the server identifier to indicate which server has been selected.
4. The servers receive the DHCPREQUEST message from the client. The IP address of the selected server is put in the Server Identifier option field of the DHCPREQUEST message so that the servers are able to distinguish whether they have been selected or not. The selected server commits the binding for the client and responds with a DHCPACK message containing the requested IP address and configuration parameters for the requesting client. The server also includes a lease with the DHCPACK message. The client must renew the lease before the lease period expires.
5. Upon receiving the DHCPACK message, the client checks if the IP address offered by the server is already in use. If the IP address is occupied, the client sends a DHCPDECLINE message to the server. In this case, the client starts the whole process again. Otherwise, the client will configure itself using the IP address and associated parameters.
6. The client may choose to relinquish its binding on a network by sending a DHCPRELEASE message to the server; for instance, when the IP address is no long needed by the client. The server will then put the released IP address into the available IP address pool.

4.1 Attacks on DHCP

There are weaknesses in the DHCP protocol that make it susceptible to many attacks, especially denial of service attacks. Although it is possible to use firewalls to decrease the possibility of the denial of service attacks coming from outside of the network, it is nearly impossible to prevent attacks launched by internal attackers masquerading as either DHCP clients or servers. Since there is virtually no reliable client identification, malicious hosts can easily masquerade as valid DHCP.

4.2 Malicious DHCP Clients

In order to attack a DHCP server, a malicious client can request all available addresses from the server. This can be done simply by continuously sending DHCPDISCOVER messages and repeat until the DHCP server exhausts all its available IP addresses. Any client machine wanting to join the network after the attack could not be allocated an IP address and would be denied network service. In another type of attack, malicious hosts release valid clients'

network address binding on the DHCP server, by adjusting the "chaddr", "xid" and "ciaddr" fields in forged DHCP messages. This way, the attacker dupes the DHCP server into relinquishing another clients' address binding causing a denial of service. Finally, the malicious agent can masquerade as a legitimate client and obtain confidential information.

4.3 Malicious DHCP Servers

Since DHCP is an unauthenticated protocol, DHCP clients have no way of distinguishing between an attacker's malicious DHCP server and a real one. An attacker can easily set up a new DHCP server using a single compromised machine, resulting in significant damage to the remaining uncompromised hosts on the network. One of the most dangerous server-side attacks used by a rogue DHCP server is a "man in the middle attack".

A rogue server might give clients non-functional or fake network configurations so that they are unable to use the network. A smarter attacker can even provide the clients with real network configurations except for some critical information such as the gateway and DNS server. By providing the fake critical information, the rogue server could divert all clients' traffic through themselves, thus enabling them to eavesdrop on every packet sent by the clients to the Internet. In order to perform this man-in-the-middle attack, a rogue server first masquerades as a DHCP client and obtains a valid IP address from the real DHCP server. Then, the rogue DHCP server starts to listen to the traffic on the network and looks for a client that broadcasts a DHCPDISCOVER message to the network. After receiving the DHCPDISCOVER message, both the rogue and the real DHCP server send a DHCPOFFER message. At this stage, the rogue server offers the IP address obtained from the real DHCP server so that no IP address conflicts occur. Usually the client selects the DHCPOFFER that arrives first. If the client chooses the DHCPOFFER from the rogue server, the man in the middle attack will succeed. Once successful, the attacker may perform many malicious activities such as stealing credential information.

4.4 A DHCP State-Transition Specification

We use a finite state machine (FSM) to represent the specification of normal DHCP protocol behavior. We assume that the FSM are deterministic and complete. This is because for every state and input, there is a unique next state and action. Also, for every state and input there always exists a next state and action. Based on the deterministic and complete properties of FSMs, we can always unambiguously define and determine the actions of our systems.

4.5 DHCP Server FSM Specification

We build the DHCP servers' specification according to its major tasks. According to the description above, a DHCP server has the following responsibilities:

- Answer the DHCP request from clients (DHCPOFFER)
- Assign IP addresses to clients (DHCPACK)
- Reject DHCP requests from clients (DHCPNAK)
- Renew the address binding (DHCPACK)

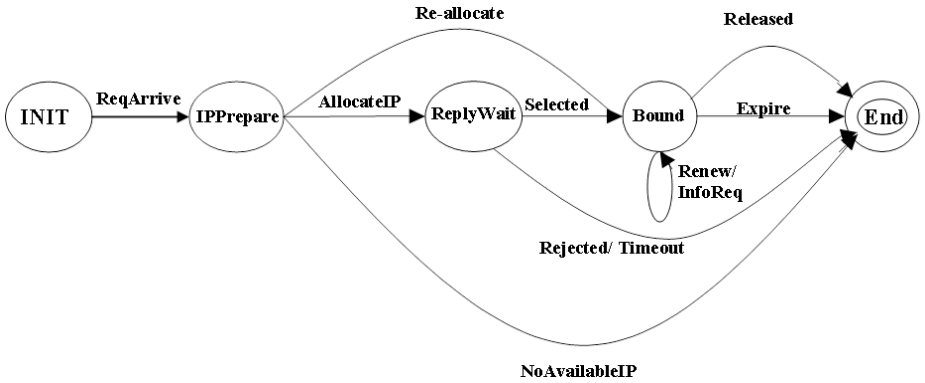


Figure 1: A state machine specification for correct DHCP server behavior. Transitions between server states occur upon receipt or transmission of client messages, or upon timeouts.

- Assign a new address binding for a client (DHCPACK)
- Expire address binding (DHCPNAK)

Using these tasks and other detailed properties of the DHCP protocol, we construct a DHCP server FSM illustrated in Figure 1. In this diagram, we have five states and several transitions/events between the states. The normal behaviors of the protocol are transformed into transitions. This state machine specification of a DHCP server clarifies the relationships between each action and makes it easier to locate security related activities. For example, before the server can move from state IPPrepare to state ReplyWait, the server has to check if there is an available IP address in its address pool. This implies that the server could check the IP address availability and usability, creating an opportunity to detect abusive usage from a malicious client.

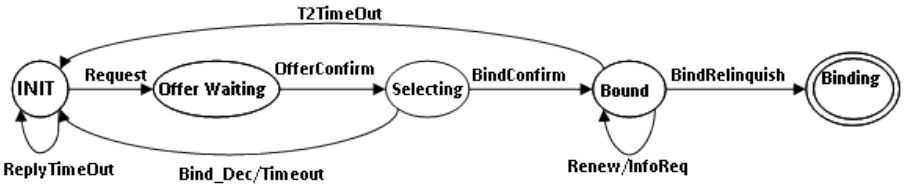


Figure 2: State machine specification for correct DHCP Client behavior. Transitions between states occur based upon messages to or from the DHCP server.

4.6 DHCP Client FSM Specification

We can apply the approach in the previous section to build the DHCP client specification. According to RFC 2131, a DHCP client has the following responsibilities

- Discover DHCP servers (DHCPDISCOVER)
- Inform the DHCP servers of its needs (DHCPREQUEST)
- Select a DHCP offer from one of the servers and confirm it (DHCPREQUEST)
- Inform the server that the network address is being used (DHCPDECLINE)
- Relinquish the network address (DHCPRELEASE)
- Ask for local configuration parameters (DHCPINFORM)

Figure 2 shows the protocol behavior of a DHCP client. In this diagram, we also have five states and several transitions/events between the states. Again, the normal behaviors of the protocol were transformed into transitions.

4.7 Specification Invariant Protocol Diversity

An example of our approach to automatic network protocol diversification will now be described for the case of DHCP. Specification invariant protocol diversity is based upon the observation that, in DHCP attacks, the attacker is supplying false state transitions, forcing the victim client or servers into compromised states that benefit the attacker. Suppose that instead of using the standard DHCP protocol specification, we generate new intermediate states that are inserted into an existing transition. Forged DHCP messages, then, will move the victim client or server into a state not known by the attacker at the outset, and not into the state necessary for his attack to succeed. Naturally, valid clients and servers will still need to be able to transition to the desired state in the protocol, so they would be provided with a specification of the new messages to use to complete the transition. This involves adding new states that are coordinated between client and server FSMs.

For example, notice that the AllocateIP transition in the server, and the OfferConfirm transition in the client correspond to an available IP address transferred from server to client. Suppose that we now introduce a new state and a new transition message into the existing ones. Now, valid clients and servers will require an additional step to complete the IP address transaction; this could even be the exchange of secret identifiers. Attack processes exploiting the standard DHCP vulnerabilities will no longer work, yet clients and servers still share a method for

arriving at the desired state as specified by the invariants. The resulting FSMs for the new protocol are shown in figure 3.

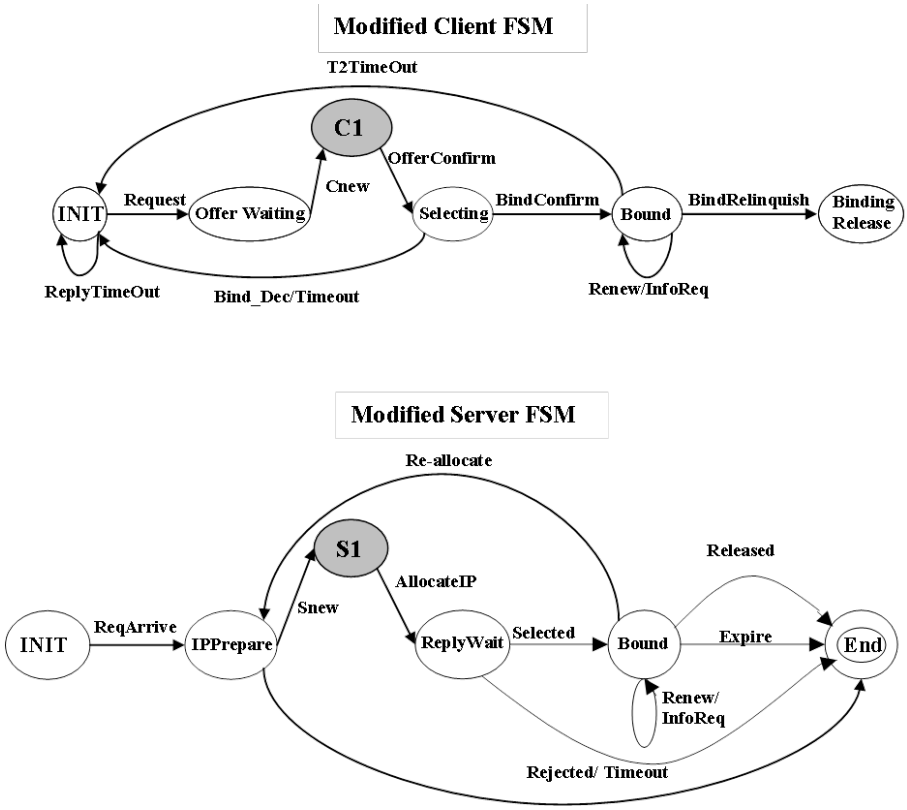


Figure 3: Modified DHCP server and client FSMs showing a single additional new state unknown to an attacker. Notice that the invariant requirement is represented by the existence of all previous normal DHCP states and transition paths to them.

Note that an attacker who knows the exact procedure will still be able to complete their attack. However, by modifying different transitions in different ways, we can produce a diverse collection of protocols, each slightly different, that replicates the exact behavior of the original protocol specification. No single attack will work against all members. An infinite number of alternate FSMs can be generated by adding several intermediate states and alternative transition paths and messages.

5 Security State Estimation

Given a set of cyber-maneuver techniques, the question then is how to select the appropriate maneuver for the current security state. In the control model, one needs a principled method for estimating the state in which the system currently resides. This must be performed in the presence of imperfect or missing information. Our approach is to use sequential hypothesis testing, as in our previous work [4], to infer the correct state with known bounds of uncertainty.

In this formulation, let H_1 and H_0 be the hypotheses that the system is and is not in the specified security state respectively. Let Y_i be the random variable indicating an attack reported by sensor i, which could be from cyber-maneuver instrumentation, an independent IDS, or any other relevant information source. Each sensor has a false positive f_p and false negative f_n performance.

$$Y_i = \begin{cases} 1 \text{ if there is an attack (or a false positive } f_p) \\ 0 \text{ if there is no attack (or a false negative } f_n) \end{cases} \quad (1)$$

By definition,

$$P[Y_i = 0|H_1] = f_n; \quad P[Y_i = 1|H_1] = (1 - f_n)$$
$$P[Y_i = 1|H_0] = f_p; \quad P[Y_i = 0|H_0] = (1 - f_p) \qquad (2)$$

The observation vector $\boldsymbol{Y} = \{Y_1, Y_2, ..., Y_n\}$ then is the set of measurements obtained by *n* conditionally independent sensors. We then define the Likelihood Ratio from the observation as:

$$L(\vec{Y}) = \frac{f_n}{f_p} = \frac{P[\vec{Y}|H_1]}{P[\vec{Y}|H_0]} \quad or \quad L(\vec{Y}) = \frac{P[Y_1|H_1]P[Y_2|H_1]...P[Y_n|H_1]}{P[Y_1|H_0]P[Y_2|H_0]...P[Y_n|H_0]} \qquad (3)$$

This assumes that all Y_i sensors provide independent measurements in a given specific security state. For a sequence of sensor inputs, $L(\vec{Y})$ is the ratio of products shown in equation 2. These equations are used in conjunction with many random walks through the collection of sensors to compute a table of the likelihood of specific outcomes. The strength of the desired state estimator, then, is specified by two quantities: desired correct estimation rate, *DD*, and tolerable false estimation rate, *DF*. Using these, one can calculate two thresholds in the table of outcome likelihoods: $T_0 = (1\text{-}DD)/(1\text{-}DF)$ and $T_1 = (DD/DF)$. The global state estimator then makes decisions as follows: if, after including the next sensor input, the calculated likelihood ratio, $L(Y) < T0$, accept the hypothesis H_0, that the system hasn't changed its security state and reinitialize the observation vector. If $L(Y) > T1$, accept the hypothesis H_1, that the system has entered an Insecure Normal, Emergency or Restorative state and initiate a new cyber-maneuver selection procedure. Otherwise no decision is reached, so maintain the current maneuver and continue collecting sensor reports. The thresholds define upper and lower blocks in a table of sensor sequences as a region likely to have produced the sequence if the system had entered a new state, and a region likely to come from the current state. By independently sampling a variety of weak or strong information sources with given f_p and f_n, one can achieve a strong state estimator if enough sensors are sampled. Data collection continues until the uncertainty in the state estimation falls below the preconfigured limit.

6 Minimal Cost Maneuver Selection

Once the current control state and the set of effective maneuvers are estimated, the control action must be selected to minimize the overall cost to the system. We use infinite horizon dynamic programming to select actions that minimize expected long-term costs. Making this decision is hard in the presence of uncertain information and random processes. Suspending a service component is oftentimes desirable if it protects the larger system, but it is harmful in response to a false alarm. Deliberate triggering by a malicious adversary might also cause self-inflicted denial-of-service. Intuitively, it seems desirable to shutdown services until the state is more accurately determined. Balancing the consequences of maintaining a suspect service and risking its malicious faults against denying the service for protection is a critical question.

In the control-theoretic model, the system consists of two features: (1) a discrete-time dynamic system and (2) a cost function that is additive over time. The cost function is additive in the sense that the cost incurred accumulates over time. However, because of the presence of uncertainty in the state, the cost is generally a random variable and cannot be meaningfully optimized. We therefore formulate the problem as an optimization of the *expected cost* where the expectation is with respect to the joint distribution of the random state variable. The optimization is over the controls, where each control, is chosen based on the current observation of the system. This is *closed loop* optimization as opposed to *open loop* optimization when all controls have to be decided at once at time zero without knowledge of the system state at runtime. Mathematically, in closed loop optimization, we want to find a sequence of functions, mapping the system state into a control which when applied to the system minimizes the total expected cost. This sequence is referred to as a *policy* or *control law*. Details of how we applied this to automatically block global attacks in a distributed collaborative IDS system can be found in [5].

To complete the moving target defense loop, the maneuvers at our disposal must be categorized according to the attacker assumptions that they violate and the cost to the system for a set of assumptions about the attacker's capability. It may be the case that, even in the absence of sensor information about the attacker, we assume that the attacker has a set of minimal capabilities at all times. This allows for the selection of a set of maneuvers to be implemented asynchronously with sensor reports. Continuous randomization of a system diversity technique, for example, might be performed if the method incurs minimal cost to the system. The assumption here is that the attacker has moved the system into an insecure normal state with network probing, and changing to a different diversified copy as a preventative action keeps the system from moving into a true emergency state. Operating with sensor evidence of a compromise on a host with relatively low importance might be sufficient to indicate an emergency state, assuming that the attacker can now launch attacks on more important hosts. In this case, a network

transformation cutting off the affected machine is low cost, and moves the system to a restorative state violating the assumption that the attacker has access to important machines.

One of the major difficulties with cost based moving target defenses comes from assigning useful cost parameters to specific maneuvers. In our previous work, cost was reduced to a single parameter; the cost ratio of the damage from a successful attack to the cost of lost service due to reconfiguration. We are investigating methods for dynamic, moving cost assignments to complement maneuver selection. We envision the incorporation of an aging mechanism into the cost of reconfiguration. For example, if we detect a persistent attacker repeatedly initiating the same or similar threats, the type of maneuver could change over time, both in type and in frequency. If the moving target defense mechanism itself is known, sophisticated attackers will anticipate a maneuver and will attempt to deduce its nature. Incorporating time dependent cost properties will allow the maneuvers to change at different levels to reduce any the potential to deduce characteristics of the maneuver. For the diverse DHCP protocol described above, the cost would be associated with the loss in service when switching to a new, shared secret state; perhaps leaving behind clients who are assumed to be already compromised and subsequently cut off from the service.

7 Discussion and Future Work

We believe approaching the moving target defense problem as a closed loop control system has advantages in that it focuses the problem upon the three specific related tasks. First, what low-level techiques, we call cyber-maneuvers, are available to move the protected system between security states? We present an approach for injecting artificial diversity in a monoculture system that can be used as a maneuver, but many other types of maneuvers could be considered and will be a continued focus of our ongoing work. Usually such a move will involve loss or degradation of some service and so will involve some cost. Care must be taken that the process of estimating these costs doesn't become more difficult than simply hard-coding manual responses based upon expert knowledge. The costs of maneuvers must be balanced with the damage or loss of service that a suspected attack may cause. So the second moving target defense task is to estimate the attack state of the current system. Note that even a lack of any attack state may be due to imperfect sensors and a preventative maneuver may still be warranted if the cost is sufficiently low. We present a model for security state estimation based upon intrusion sensors. In our ongoing work, we are extending this work and developing models of attacks based upon post and pre-conditions that can chain to provide attackers with sets of additional capabilities. By augmenting maneuver models with specific capabilities denied to the attacker, relavant maneuvers can be classified according to their ability to remove attacker capabilies to move to a less costly security state. Finally, we present a closed-loop control approach that generates specific maneuver strategies for a given security state that will minimize the overall system cost in time. Further work is need here to define recovery transitions that will allow the control procedure to transition to a very costly *Restorative* state if it is followed by a cost effective transition to the *Secure Normal* state once again. Development of *Restorative-to-Secure Normal* state transition maneuvers is one of the primary research challenges of our ongoing work.

References

[1] Maharana, M.K.; Swarup, K.S.;, "Identification of Operating States of Power System Using Transient Stability Analysis," *Power System Technology and IEEE Power India Conference, 2008. POWERCON 2008. Joint International Conference on*, pp.1-6, October 2008.

[2] Antonatos, S.; Akritidis, P.; Markatos, E. P.;, Anagnostakis. K. G.;, "Defending against hitlist worms using network address space randomization," *Computer Networks*, Vol. 51, No. 12, pp. 3471-3490, August 2007.

[3] Nguyen, L. Q.; Demir, T.; Rowe, J.; Hsu, F.; Levitt. K. N.;, "A framework for diversifying windows native APIs to tolerate code injection attacks," *In Proceedings ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pp. 392-394, March 2007.

[4] Cheetancheri, S. G.; Agosta, J. M.; Dash, D. H.; Levitt, K. N.; Rowe, J.; Schooler E. M.;, "A distributed host-based worm detection system," *In Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense (LSAD)*, pp. 107-113, September 2006.

[5] Cheetancheri, S. G.; Agosta, J. M.; Levitt, K. N.; Wu, F.; Rowe, J.;, "Optimal Cost, Collaborative, and Distributed Response to Zero-Day Worms-- A Control Theoretic Approach," *in Recent Advances in Intrusion Detection Symposium (RAID)*, pp. 231-250, Heidelberg 2008.

[6] Ko, C.; Ruschitzka, M.; Levitt., K.;, "Execution monitoring of security-critical programs in distributed systems: a specification-based approach," In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 175–187, Washington, DC 1997.

[7] Ko, C.; Brutch, P.; Rowe, J.; Tsafnat, G.; Levitt, K;, "System Health and Intrusion Monitoring Using a Hierarchy of Constraints," *Proceedings of 4th International Symposium, Recent Advances in Intrusion Detection*, pp. 190-204, October 2001.