Reasoning about Timeliness for Computer Security Reactions: CIRCA and AIA Experiment 001

David J. Musliner and John M. Maloney*
Automated Reasoning Group
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
musliner@htc.honeywell.com

1. Introduction

DARPA's Autonomic Information Assurance (AIA) program[1] is exploring the use of automatic systems to detect and respond, at computer speeds, to high-speed computer security attacks. The first formal experiment of the AIA program, termed AIA Experiment 001, explored the relationship between the effectiveness of responses to scripted security attacks and the speed of those responses [5]. This paper discusses how the CIRCA system for automatic controller synthesis can reason about the problem explored in AIA Experiment 001, can automatically predict the results of the experiment, and can exploit those predictions itself. By modeling the individual steps of the attack and the potential response actions, CIRCA can explicitly compute the response-time threshold distinguishing effective responses from ineffective responses. In fact, CIRCA can use this knowledge to build a reactive security controller that *guarantees* to respond quickly enough to prevent the attacker from succeeding.

To show how CIRCA does this reasoning, we begin with a brief review of Experiment 001 and its results, then provide a short review of how CIRCA works. We then illustrate how CIRCA models the experiment and builds a controller that will always defeat the attack. The intent is to clearly illustrate CIRCA's reasoning processes that build guaranteed controllers, and how they relate to information assurance.

This paper is not meant to be an introduction to CIRCA; instead, our goal is to describe how CIRCA can address the type of information security challenges explored in Experiment 001. Accordingly, we refer readers to other publications [3, 4, 1] for more comprehensive information on CIRCA, its planning algorithms, and related work.

*Now with Parametric Technology Corporation, Arden Hills MN 55112, jmaloney@ptc.com.

[1] Now merged into the DARPA/ATO Cyber Panel program.

2. AIA Experiment 001

AIA Experiment 001 investigated the effectiveness of automated defenses against attacks that proceed at machine speeds. In particular, the experiment explored the relationship between the latency with which responses are applied and the effectiveness of those responses.

The major conclusions of the experiment were that [5]:

- Longer latencies make simple responses less effective.
- Removing an attacker's uploaded scripts and programs in addition to terminating an attacker's login session is more effective than termination alone.
- Even if an attack is ultimately successful, simple responses can increase an attacker's risk and exposure by increasing the amount of time required to complete the attack.

The experimental setup used a host protected with an automated response system. During the experiment, this host was repeatedly attacked *while varying the response system's latency in a controlled fashion.* The experiment was repeated using several different well-known exploits including the `imwheel` (CVE CAN-2000-0230) attack. In this paper our analysis focuses on this attack, but the results are equally applicable to the other attacks studied in the experiment. The experiment was also repeated using two different responses:

1. Terminate the attacker's login session.
2. Terminate the attacker's login session and also remove the files the attacker uploaded to the victim host.

Figure 1 illustrates the different states of the `imwheel` attack as implemented in the experiment. Note that the attack involves uploading two files and then decoding, compiling, and executing the resulting exploit code. The attack script is moderately adaptive in that it can detect when some

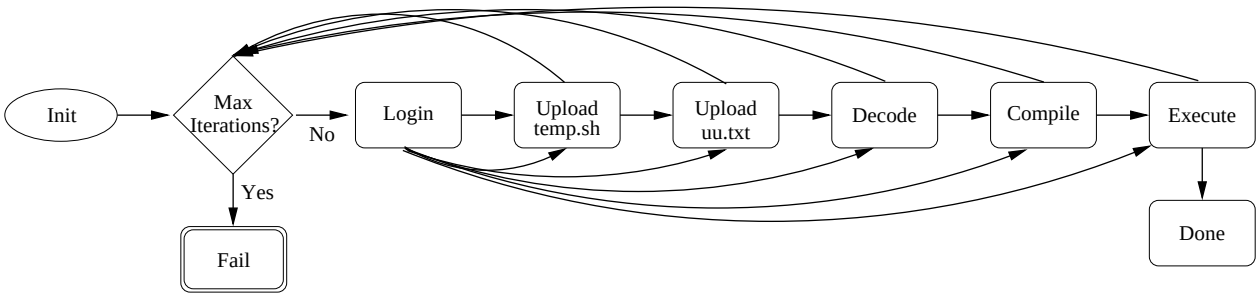

Figure 1. The state machine used to implement the `imwheel` attack (from [5]).

response	average time
kill login	100
remove files	380

Figure 3. Time required for responses [5].

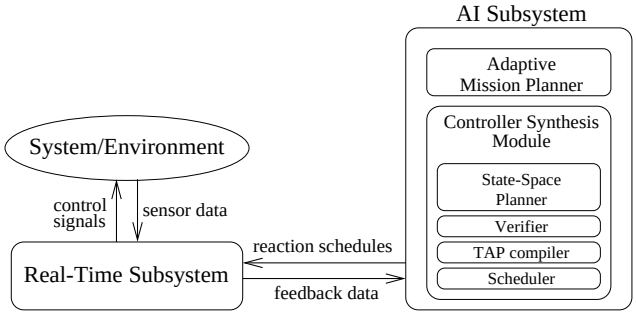

Figure 4. The Cooperative Intelligent Real-Time Control Architecture.

of the steps have already been accomplished, and move immediately past them. For example, if the attacker's script is able to upload the first file and then his login process is killed, then after logging back in the script will proceed immediately to uploading the second file. Of course, if the victim/defender removes the first uploaded file, then the script will have to start at the beginning.

Attacks were detected by an idealized intrusion detection system (IDS) that had complete knowledge of all relevant attacker actions and state. Using this ideal IDS removed any experimental variability that would otherwise be introduced due to missed detections and false alarms.

Key states, events and network traffic were logged during the experiment. This log data was processed to extract relevant metrics such as attack success and duration. We use this data in our CIRCA models of the experiment scenarios, so we have repeated the relevant information here to show the alignment of our CIRCA models (described later) to the actual experiment. Figure 2 presents the empirically-derived durations in microseconds for each step in the `imwheel` attack. Figure 3 shows the amount of time, in microseconds, required to perform each of the response actions *after* the variable response latency had expired. Note that less-precise data was supplied for the response actions, and this may have affected the alignment of our analytic CIRCA models with the experimental results, as we shall see.

3. CIRCA: Planning Timely Actions

The Cooperative Intelligent Real-Time Control Architecture (CIRCA) is an architecture for intelligent real-time control systems that has been applied to several domains including robots and simulated autonomous aircraft. CIRCA is designed to support both hard real-time response guarantees and unrestricted AI methods that can guide those real-time responses. Figure 4 illustrates the architecture, in which an AI subsystem (AIS) reasons about high-level problems that require its powerful but potentially unbounded planning methods, while a separate real-time subsystem (RTS) reactively executes the AIS-generated plans and enforces guaranteed response times. Within the AIS, several modules cooperate to develop executable reaction plans that will assure system safety and attempt to achieve system goals when interpreted by the RTS.

In this paper, we are primarily concerned with CIRCA's ability to automatically build plans (or controllers) for mission-critical domains. We are currently developing an enhanced version of CIRCA for computer security applications, called CIRCA for Dynamic Information Assurance (CIRCADIA). In this domain, CIRCA uses a suite of computer security tools to provide timely responses to intrusions in order to maintain the integrity and availability of network resources for legitimate users. In this paper, we focus on CIRCA's planning system (the State Space Planner (SSP) within the Controller Synthesis Module) to show how it can reason about the computer security domain and the recent

description	average time	standard deviation	minimum time	maximum time
telnet login	112189	1755	110107	122265
upload temp.sh	40089	103	39978	41102
upload uu.txt	1344833	48309	1059074	1389070
decode uu.txt	20332	2017	16565	39812
compile imexp.c	133476	22370	130803	355830
execute imexp	144099	10358	139356	218196
telnet logout	307	3	302	324

Figure 2. Time required to perform the steps in the `imwheel` attack [5].

AIA experiments.

CIRCA's planning system builds reaction plans based on a world model and a set of formally-defined safety conditions that must be satisfied by feasible plans [4]. To describe a domain to CIRCA, the user inputs a set of transition descriptions that implicitly define the set of reachable states. Each transition description specifies the set of preconditions that must be true in a state for that transition to be applicable, and the postconditions representing changes in system state if the transition occurs. The transitions are of four types:

Action transitions represent actions that can be performed by the RTS. Actions are characterized by their worst-case execution time.

Event transitions represent external world events as instantaneous state changes. Events are considered *non-volitional* and uncertain, meaning that CIRCA cannot determine whether or not they will occur (if their preconditions are satisfied).

Temporal transitions represent the progression of time and external processes. Temporals are also non-volitional and uncertain, but are characterized by a lower bound ("min-delay") time. The transition may only occur if its preconditions have held true for at least the min-delay.

Reliable temporal transitions represent processes that are guaranteed to occur, and they are characterized by both a lower and an upper bound time.

The SSP plans by projecting over these transitions, generating a nondeterministic finite automaton (NFA) from a description of the system's initial state(s). The SSP assigns to each reachable state either an action transition or `no-op`. Using the temporal information associated with the different transitions, CIRCA is able to derive time bounds on how quickly the system must detect and react to different states, in order to preserve system safety. Actions are selected to *preempt* transitions that lead to failure states and to drive the system towards states that satisfy as many goal propositions as possible. Figure 5 illustrates a simple preemption, where an action has been planned that will definitely occur before the system could possible transition to failure, and thus the failure state has been made unreachable.

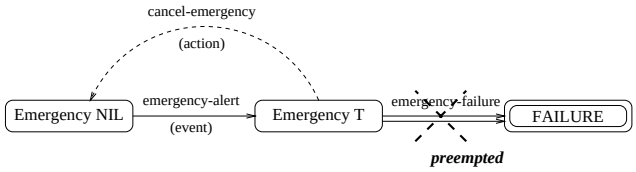

Figure 5. A simple preemption example showing an action that prevents failure.

The assignment of actions determine the topology of the NFA (and so the set of reachable states): preemption of temporal transitions removes edges and assignment of actions adds them. System safety is guaranteed by planning action transitions that preempt *all* transitions to failure, making the failure state unreachable [4]. It is this ability to build plans that guarantee the correctness and timeliness of safety-preserving reactions that makes CIRCA suited to mission-critical applications in hard real-time domains.

In the context of computer security, we are developing CIRCADIA to automatically model computer security attack elements and primitive defense mechanisms, and build reactive controllers that detect and respond to attacks in a timely fashion. CIRCADIA's planning capabilities will be used in both an offline fashion, to build security controllers before a system is commissioned, and online, to build customized security controllers on-the-fly as policies, priorities, defensive systems, attack models, and other environmental parameters change.

4. CIRCA Stopping the `imwheel` Attack

Figure 6 shows the CIRCA domain encoding of the `imwheel` attack steps. The precondition and postcondi-

```
(def-event login
  :preconds  ((logged-in F))
  :postconds ((logged-in T)(attack-detected T)))

(def-temporal upload-temp
  :preconds  ((logged-in T)(uploaded-sh F))
  :postconds ((uploaded-sh T)(attack-detected T))
  :min-delay *upload-temp-delay*)

(def-temporal upload-uu
  :preconds  ((logged-in T)(uploaded-sh T) (uploaded-uu F))
  :postconds ((uploaded-uu T)(attack-detected T))
  :min-delay *upload-uu-delay*)

(def-temporal decode-uu
  :preconds  ((logged-in T)(uploaded-sh T) (uploaded-uu T)(decoded F))
  :postconds ((decoded T)(attack-detected T))
  :min-delay *decode-delay*)

(def-temporal compile-imexp
  :preconds  ((logged-in T)(decoded T)(compiled F))
  :postconds ((compiled T)(attack-detected T))
  :min-delay *compile-delay*)

(def-temporal execute-imexp
  :preconds  ((logged-in T)(compiled T))
  :postconds ((failure T))
  :min-delay *execute-delay*)

(def-machine imwheel-attack
  (login upload-temp upload-uu decode-uu compile-imexp execute-imexp))
```

Figure 6. CIRCA definition of the `imwheel` attack.

tion expressions for each transition control the logic of the attack state machine. The login step is modeled as an event since the login can occur any time its preconditions hold true. The two upload steps, the decode step, the compile step and the execute step are modeled as temporal transitions. Each of these temporal transitions has a lower bound on the amount of time that the system must reside in an enabling state (where the transition's preconditions hold true) before the transition may occur. The lower bound, or minimum delay, for each temporal in the `imwheel` encoding corresponds to the minimum time reported in the experimental results for the step to be completed (see Figure 2). This model thus captures the worst-case behavior of the attacker: we assume he can begin the attack with the login step any time he is not already attacking, and after that the subsequent steps of the attack may occur as quickly as possible.

CIRCA's modeling language uses the reserved state feature `(failure T)` to indicate that some state is considered a catastrophic failure. CIRCA always endeavors to synthesize a controller that makes those failure states unreachable. For the `imwheel` attack, our initial inclination was to model the final `execute` step of the attack as leading to failure, as shown in Figure 6. However, further discussions with the Dr. Van Hook, who conducted the experiment, revealed that the exploit was considered successful before the execute state completed; the execution of the exploit would affect a data file at some point during its temporal extent, and this point was considered the failure time.

Since the exact duration from the start of the execute action to the time at which the attack succeeded was not measured, we were forced to make a worst-case assumption and model the `compile-imexp` transition as leading directly to failure. The modified portion of the domain encoding is shown in Figure 7.

In addition to modeling the steps in the attack we also modeled responses to the attack. As in the original experiment, we used two different responses: killing the attacker's processes and removing attacker's files in addition to killing the attacker's processes. Figure 8 shows the encoding of these two responses. The system's response latency is modeled as a reliable temporal and the two responses as actions. Each of the actions has an associated worst-case execution time (`wcet`). The `wcet` we used for each action corresponds to the action duration reported in the experimental results (see Figure 3).

We defined two different problems for CIRCA to solve, corresponding to the two different experiments that were conducted. One problem uses the kill attacker only response while the other problem uses the kill attacker and remove files response. Figure 9 shows the encoding of the two problems and the system's initial state.

When we ask CIRCA's planner to solve a problem, it either returns a plan or `NIL`, indicating that it could not find a plan. To determine the threshold at which the system's response latency prevents it from defeating the `imwheel` attack, we can manually vary the upper bound on the reliable temporal that was used to model latency (`*response-latency*` in Figure 8). At the threshold, increasing the latency by one time unit causes CIRCA to fail to find a plan. In this way, CIRCA can be used to derive the upper bound on the system's response latency.

In a more typical application of CIRCA, the domain timing constraints would remain fixed and the system would automatically determine how quickly it needs to respond to different safety threats using its available actions. CIRCA's controller synthesis module can derive time-constrained reactive plans that are guaranteed to avoid failure, without the sort of manual assistance described above. We used this iterative manual technique to yield results that are most easily compared to the experimental data.

5. Comparison to Experimental Results

The experiment results show that the relationship between attack success and response latency is a step function. For the `imwheel` attack there exists a point in time before which all attacks fail and after which attacks begin to succeed. When using the kill only response against the `imwheel` attack this threshold occurs between 1050 and 1100 ms.

The experiment results also support the hypothesis that removing the attacker's files in addition to terminating the attacker's login session increases the time at which the threshold occurs. For the `imwheel` exploit, using the kill and remove response pushes the threshold at which attacks start to be successful out to between 1200 and 1250 ms.

As described above, the CIRCA planner can be used in an iterative fashion to derive the threshold at which the latency is too large for attacks to be successfully defeated. Figure 9 shows the definition of the two cases for which CIRCA derived thresholds: `imwheel-kill` and `imwheel-kill-and-remove`. Figure 12 summarizes the results derived by CIRCA, and shows that they align perfectly with the experimental data. The experimental data for each test case is listed as a range, with the lower number representing the highest latency at which no attacks succeeded, and the upper number representing the lowest tested latency at which one or more attacks succeeded.

Careful inspection of the plan shows that, given the experimental setup, the results derived by the CIRCA planner and achieved in empirical studies are expected. As the steps of the attack proceed, the response latency continues to expire until finally the responding agent acts. The steps of the attack form a dependent chain of temporal transitions [2, 4]. That is, the delay until the responder acts from a particu-

```
(def-temporal compile-imexp
  :preconds  ((logged-in T)(decoded T)(compiled F))
  :postconds ((compiled T)(attack-detected T)(failure T))
  :min-delay *compile-delay*)
```

Figure 7. Modifications to the `imwheel` attack model to account for failure immediately after the compile step.

```
(def-reliable response-latency
  :preconds  ((attack-detected T)(response-enabled F))
  :postconds ((response-enabled T))
  :delay (make-range 0 *response-latency*))

(def-action kill
  :preconds  ((attack-detected T)(response-enabled T))
  :postconds ((logged-in F) (attack-detected F) (response-enabled F))
  :wcet *kill-delay*)

(def-action kill-and-remove
  :preconds  ((attack-detected T)(response-enabled T))
  :postconds ((logged-in F) (uploaded-sh F)(uploaded-uu F) (decoded F)
              (compiled F) (attack-detected F) (response-enabled F))
  :wcet (+ *kill-delay* *remove-delay*))
```

Figure 8. CIRCA definition of the system's responses.

```
(def-state initial-state
  :features ((failure F) (attack-detected F) (response-enabled F)
             (logged-in F) (uploaded-sh F) (uploaded-uu F)
             (decoded F) (compiled F)))

(def-problem imwheel-kill
  :machines (imwheel-attack)
  :transitions (response-latency kill)
  :initial-states (initial-state))

(def-problem imwheel-kill-and-remove
  :machines (imwheel-attack)
  :transitions (response-latency kill-and-remove)
  :initial-states (initial-state))
```

Figure 9. CIRCA definitions of the problems to be solved.

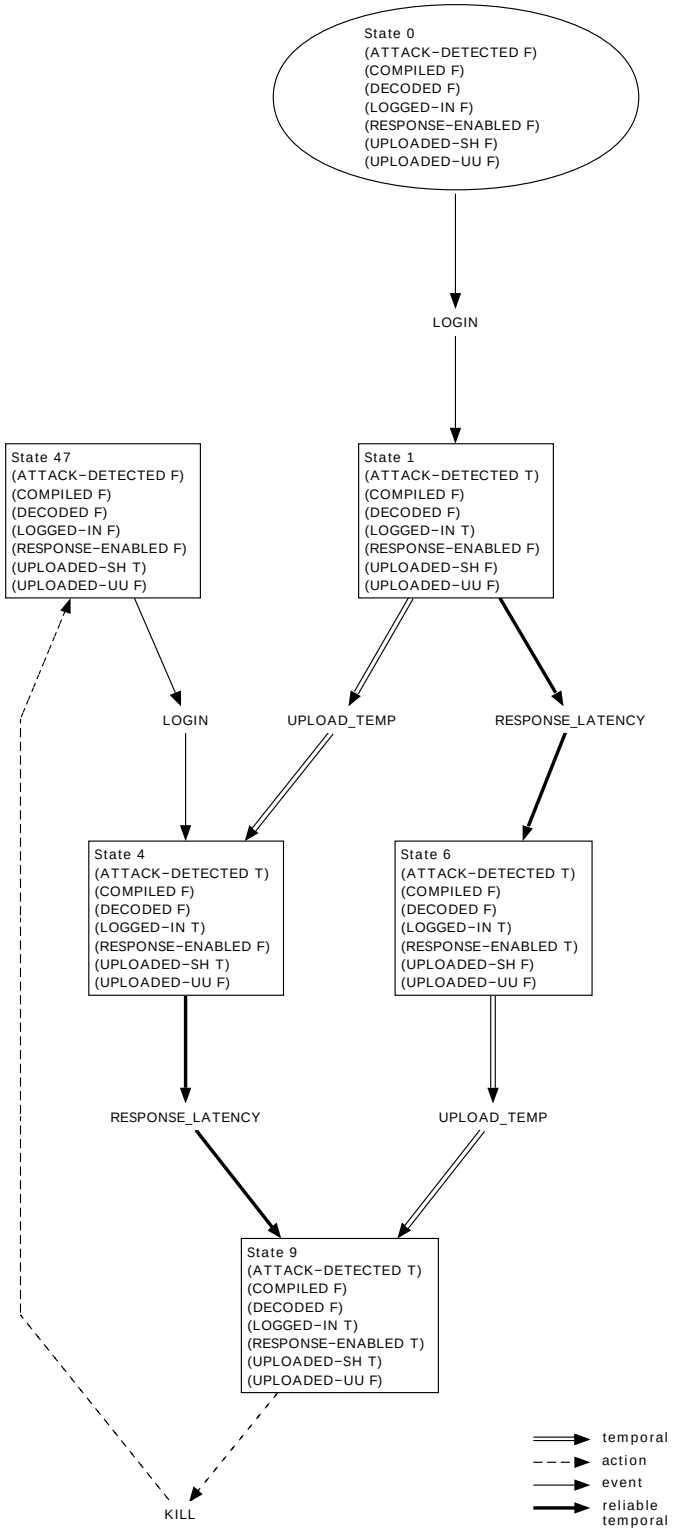

Figure 10. Reachable state graph for the solved `imwheel-kill` problem. CIRCA has guaranteed to kill the attacker after the first upload, and failure is unreachable.

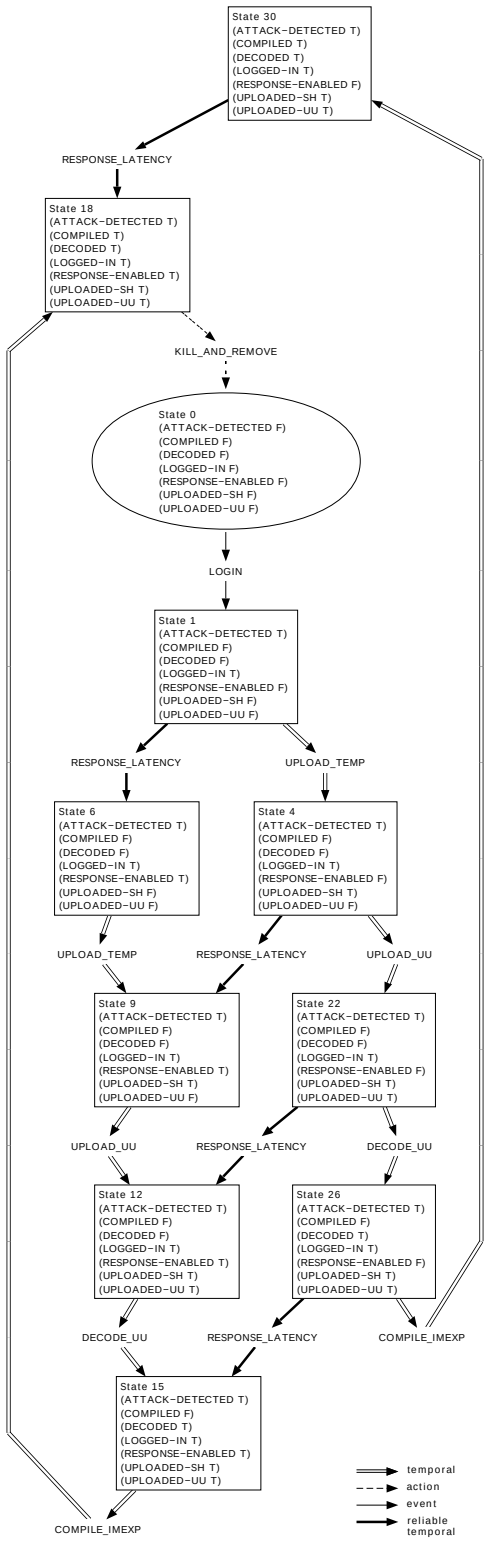

Figure 11. Reachable state graph for the solved `imwheel-kill-and-remove` problem. CIRCA lets the attack proceed nearly to completion before killing the attacker and removing his files.

response	experimental	CIRCA
kill attacker	[1050000 1100000]	1058973
kill and remove	[1200000 1250000]	1245939

Figure 12. Response-latency thresholds derived by experiment and by CIRCA.

lar state is dependent on how long the attacker allows the system to dwell in prior states. In the worst case, we assume the attacker works as quickly as possible. In the case of the `imwheel-kill` problem, if any one of the attack steps is interrupted, the attacker will need to repeat that step before moving on to the next step. Since we do not remove the attacker's files, when the attacker logs back in he can skip all the steps that have already succeeded and just start the attack at the point where it was interrupted. If we can always stop at least one of the steps before it completes, then we can prevent the attack from being successful. Therefore, the largest response latency we can tolerate and still prevent failure must be strictly less than the duration of the longest attack step minus the time required to complete the response. The longest step is the `upload-uu` step with a (minimum) duration of 1059074 microseconds. The duration of the kill-attacker response is 100 microseconds. Therefore, the longest latency that can be tolerated by the system before the attacker begins to succeed is 1058973 microseconds, which agrees with the results derived by the CIRCA planner.

The response in the `imwheel-kill-and-remove` domain not only kills the attacker's process, it removes the attacker's files as well. In this case, the attacker needs to start the attack over from scratch each time it is defeated. If we can guarantee to always complete our response before the last step in the attack is completed, then we can prevent the attack from being successful. Therefore, under the `imwheel-kill-and-remove` scenario, the largest latency we can tolerate and still prevent failure must be strictly less than the duration of the attack minus the time required to complete the response. The attack duration is 1246420 microseconds and the kill and remove response requires 480 microseconds. Thus, the longest latency that can be tolerated in this case is 1245939 microseconds, which agrees with the results derived by the CIRCA planner. As predicted by the experimental hypothesis, this threshold is also greater than the threshold derived for the case where only the kill attacker response is employed.

6. Conclusion

In this paper we have shown how CIRCA can be used to create a timely response plan that defends a computer system from a security threat. We showed how the plan constructed by CIRCA is consistent with the experimental results obtained from AIA experiment 001. CIRCA was able to recognize the different consequences of the two alternative defensive responses it was allowed to use (kill-only and kill-and-remove), and build correspondingly different plans. This brief demonstrations illustrates both how CIRCA reasons about the timeliness of its activities, and how it can adapt dynamically to varying threats, goals, and system capabilities.

It should be noted that CIRCA's response planning does not depend on a rigid attack script: the individual attack transitions can be given weaker preconditions that allow them to be combined in different ways. This provides a more realistic representation of the flexibility and alternative approaches available to a real attacker (or attack script). CIRCA's planning is specifically designed to deal with this type of unconstrained adversarial environment, and will build controllers that deal with *all* attack behaviors, if possible.

Of course, it may not be possible to build guaranteed-safe controllers for a system as the breadth and speed of modeled attacks grows. We are currently developing methods to model and trade off security level against quality of service (to the computing system's intended mission). Our overall goal is to develop CIRCADIA to intelligently adapt its robust, reliable security responses as necessary to changes in the environment and mission, maximizing the expected utility of the computing system in the face of a broad spectrum of possible attacks.

Acknowledgments

This material is based upon work supported by DARPA/ISO and the Space and Naval Warfare Systems Center – San Diego under Contract No. N66001–00–C–8039. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the U.S. Government, or the Space and Naval Warfare Systems Center – San Diego. Approved for public release; distribution is unlimited.

References

[1] R. P. Goldman, D. J. Musliner, K. D. Krebsbach, and M. S. Boddy. Dynamic abstraction planning. In *Proc. National Conf. on Artificial Intelligence*, pages 680–686, 1997.

[2] D. J. Musliner. *CIRCA: The Cooperative Intelligent Real-Time Control Architecture*. PhD thesis, University of Michigan, Ann Arbor, 1993. Available as University of Maryland Computer Science Technical Report CS-TR-3157.

[3] D. J. Musliner, E. H. Durfee, and K. G. Shin. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Trans. Systems, Man, and Cybernetics*, 23(6):1561–1574, 1993.

[4] D. J. Musliner, E. H. Durfee, and K. G. Shin. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence*, 74(1):83–127, March 1995.

[5] D. Van Hook. Aia experiment 001 final report. In *Unpublished MIT Lincoln Labs technical report*, September 2000.