AUTOMATED PLANNING FOR REMOTE PENETRATION TESTING

Lloyd Greenwald and Robert Shanley
LGS Innovations / Bell Labs
Florham Park, NJ USA
{lgreenwald,shanley}@lgsinnovations.com

ABSTRACT

In this work we consider the problem of automatically designing a penetration test plan that can be executed remotely, without prior knowledge of the target machine or network. We develop a methodology for generating and executing remote testing plans that takes into account the uncertainty of using remote tools both to gain knowledge of the system and to provide the penetration testing actions. Our solution provides automated generation of multi-step penetration test plans that are robust to uncertainty during execution. We tackle this problem by making use of modeling techniques from partially observable Markov decision processes (POMDPs). We automate this process by taking advantage of efficient solutions for solving POMDPs, and further, automatically derive these models through automated access to vulnerability databases such as the national vulnerabilities database (NVD). We demonstrate our implemented solution on a series of example problems.

I. INTRODUCTION

We consider the problem of automatically designing a penetration test to remotely evaluate the security of a target computer or network device. A penetration test is a method of evaluating the security of a target by simulating an attack. In black-box penetration testing, the tester has no prior knowledge of the system under test. This form of testing requires the use of probing tools and thus the tester must rely on uncertain information to identify targets and vulnerabilities.

Multi-step attacks are often the most effective way for an attacker to compromise a system (computer) or network. In such an attack, the attacker attempts to achieve an access goal through the exploitation of a series of vulnerabilities, often called an *attack chain*. An attacker can take advantage of vulnerabilities in the operating system of the target or vulnerabilities in any enabled applications or services. Each piece of software may be secure in isolation, but their combination and interaction can often provide a pathway for the opportunistic attacker. The ability to detect and analyze the interaction of multiple potential vulnerabilities on a system or network of systems leads to a better understanding of the overall vulnerability of the target.

Attack graph generation and analysis is a way to reach this understanding [1]. An attack graph contains all possible attack chains for a given system or network of systems. Attack graphs provide a visual means of analyzing the multiple pathways to a given goal. Attack graphs have been used in many domains of information security, from vulnerability assessment and network design to intrusion detection and forensics.

The typical approach to attack graph generation is to consider an attacker to be omniscient, with complete knowledge of vulnerabilities on a target system and the ability to observe a system from all sides of defensive devices. Our research innovation is to consider a realistic attacker that must deal with the uncertainty of using remote tools to gain knowledge of a system before planning an attack. In the traditional approach the attributes of the systems, operating system and applications are assumed to be known with certainty. A realistic adversary will not have perfect knowledge of their potential targets. Furthermore, even when attack graphs are used as a tool by a system administrator on a managed network, it is unlikely the administrator would have a completely accurate inventory of the systems, operating systems and services on the network. In both cases it is important to model uncertainty in order to understand the true security of a target system.

The model we chose to tackle this research problem is the known mathematical model of partially observable Markov decision processes (POMDP). Our solution provides automated generation of multi-step penetration test plans that are robust to uncertainty during execution. In related work, Wang et. al. [2] have proposed an attack-graph-based probabilistic security metric. That work represents security related conditions and the transitions between conditions that can be accomplished through vulnerability exploits. Our POMDP model captures similar information. While our work focuses on designing remote penetration test plans that are robust to uncertainty in both information gathering and exploit actions, that work focuses on developing probabilistic metrics to capture the intrinsic likelihood that an exploit can be executed and the overall likelihood that an attacker can successfully execute an exploit or satisfy a condition.

The rest of the paper is organized as follows. Section II defines the penetration test planning problem. Section III describes the proposed methodology and planning algorithms implemented to solve the problem. Section IV provides an empirical evaluation of the solution on a series of example scenarios. Finally, Section V concludes the paper.

II. PROBABILISTIC PENETRATION TEST PLANNING

Penetration testing provides an assessment of security as viewed externally, for example by a remote attacker. A penetration test includes information gathering, planning, and attempted exploitation of suspected vulnerabilities. In this section we focus on deriving a plan for attempted exploitation, given the results of information gathering.

We consider the problem in which we have a collection of exploitation tools and we want to test if a remote machine can be compromised using any combination of these tools. There are varying degrees of compromise, including denial of service, gaining user access to an application on the machine, and gaining root access to the machine. We typically consider the goal to be to gain root access to the machine, although other goals can be also accomplished with the set of methods discussed in this paper.

We assume we are given the IP address of the remote target machine as well as results from information gathering (e.g. probing, scanning, fingerprinting). Our methods recognize that information gathering using remote tools is an uncertain process. We assume that information gathering can only provide probabilistic information about the operating system and services running on the target machine. Tables 1 and 2 depict example information gathering input for the operating system and an application on port 25 of the target machine, respectively. Probabilistic information about operating systems and services can be derived from the output of existing tools.

Table 1 – Operating System Distribution

Operating System	Probability
Linux kernel 2.4.20	0.622
FreeBSD 6.1	0.102
Linux kernel 2.6.14	0.090
Linux kernel 2.6.13	0.090
Windows 98 Second Edition	0.036
Windows 2000 Adv Server	0.031
Windows NT 4.0	0.015
Windows XP SP 2	0.014

Table 2 – Application Distribution

Application	Probability
Exim	0.521
Sendmail	0.138
Postfix	0.115
Microsoft Mail	0.070
qmail	0.061
other	0.095

Given the probabilistic output of information gathering and the set of exploitation tools available for penetration testing, our methodology has two steps: (1) *vulnerability assessment*, and (2) *penetration test planning*. In vulnerability assessment we produce a metric that evaluates the potential success of each exploitation tool as well as the pre- and post- conditions of applying that tool. In penetration test planning we determine how to chain tools together in order to achieve the desired access goal.

A key step in vulnerability assessment is the process of finding potential vulnerabilities in the operating system and services on the target machine. This can be done in an automated way by accessing an electronic vulnerability database. We make use of the National Vulnerability Database (NVD) [3], a comprehensive cyber security vulnerability database sponsored by the Department of Homeland Security (DHS) and the National Institute of Standards and Technology (NIST). The NVD uses the Common Vulnerabilities and Exposures (CVE) naming standard and provides XML data feeds, making automatic retrieval straightforward. Each NVD entry is described by a unique CVE ID identifier.

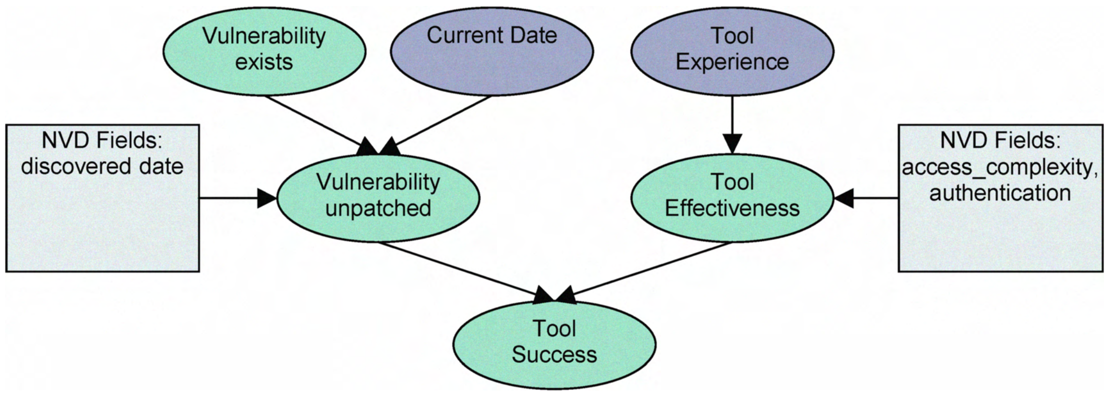

Figure 1 – Tool Success Probability Calculation

Each exploitation tool used in penetration testing is tagged with the CVE IDs of the vulnerabilities it exploits. We filter the potential vulnerabilities on a target machine based on CVE IDs in-common with the available exploitation tools. Only those potential vulnerabilities with corresponding penetration testing exploitation tools are considered further. Once an exploitation tool is matched to an NVD entry, we can extract fields of the entry to use in vulnerability assessment and planning. These fields include data about pre- and post- conditions for applying any exploit tool that corresponds to the CVE entry. We make use of additional fields to produce a *tool success probability* metric for evaluating each tool. Fig. 1 depicts the computation of tool success probability.

For each exploitation tool we first assess whether or not the vulnerability the tool exploits exists on the target machine by comparing the current date with the date on which the vulnerability was discovered. We apply an exponential decay computation based on the half-life of vulnerabilities [4] to assess the probability that the vulnerability exists and has not been patched. Second, we determine how effectively we can apply the exploitation tool by comparing the penetration tester's experience with the tool with the complexity of gaining access to the given vulnerability and whether or not authentication is required. The resulting calculation gives us a tool success probability for each combination of exploit tool and potential vulnerability.

Our approach to creating and evaluating multi-step penetration test plans is to formulate the problem as a probabilistic planning problem and adapt systematic planning methodologies toward solving the planning problem. In particular, we model the problem as a partially observable Markov decision process (POMDP). POMDPs are ideal for capturing planning problems in which there is uncertainty in the state of the world (both initially and after applying actions) and uncertainty in the effects of applying actions. As mentioned earlier, we assume information gathering can only provide probabilistic information about the operating system and services running on the target machine. Furthermore, by applying penetration testing tools remotely, we expect only noisy partial feedback (if any) on the success of the tools.

A POMDP is described by the following six components: state space, *S*, action space, *A*, reward function, *R(s,a),* state transition probability function, *P(s'|s,a)*, observation space, *O* and observation probability function, *P(o|s',a)*. The state space in our penetration test planning POMDP consists of the state variables: `Operating System`, `Operating System Privilege`; and for each port: `Application` and `Application Privilege`. Instead of assigning a deterministic value to each of these variables, we maintain a *belief state* that provides a probability distribution over combinations of values for each variable. For example, looking at a single variable, the belief state for the `Operating System` variable may be [("Windows XP SP2", 0.472) ("Windows XP SP1", 0.264) ("Windows XP", 0.264)].

The action space of our POMDP model consists of all available exploitation tools that have been identified as having common CVE IDs with potential vulnerabilities on the target machine. We also separately consider each potential vulnerability to which an exploitation tool may be applied. Our reward function provides a reward of 1 for reaching a terminal state that satisfies the user-specified goal and a reward of 0 otherwise. Rather than providing a reward function over non-terminal states, we specify a *discount factor* between 0 and 1 to favor shorter plans over longer plans. Plans involving multiple steps will be less valuable as the discount factor decreases.

The state transition probability function is derived using the tool success probability metric computed during vulnerability assessment. The probability of achieving a state with the post-conditions specified by an action (exploit tool) given that the action was applied in a state that provided the pre-conditions of the action is the tool success probability of that exploitation tool. We augment the state transition probability function by also considering potential side effects (both positive and negative) of applying an action.

Finally, we create a simple observation space and observation probability function. We provide for both silent operation and for penetration testing in which exploit tools attempt to report back successful exploitation. The observation probability function specifies how likely it is to receive a success signal if an action executes successfully.

Given a POMDP formulation, we can apply systematic planning methodologies toward generating and evaluating multi-step plans. In the next section we present one such solution.

III. EFFICIENT PLANNING SOLUTION

The problem of exactly solving POMDPs in general has been shown to be computationally intractable [5]. However, computationally efficient methods exist for exact solutions to problems with restricted structure as well as approximate solutions to general problems [6][7]. In this work, we make use of an early approximation method called Q-MDP [8]. Note that our formulation of the penetration test planning problem as a POMDP does not require the use of the Q-MDP algorithm. Other POMDP algorithms can also be applied.

Q-MDP is a hybrid solver that makes use of the relationship between POMDPs and fully observable Markov decision processes (MDPs). MDPs are more computationally tractable than POMDPs [9] and efficient algorithms are straightforward to implement. The basic solution technique of Q-MDP is to first construct an MDP from the penetration test planning POMDP by ignoring the uncertainty due to not knowing the true state of the target machine. This MDP can be solved with efficient algorithms. The solution to this MDP can then be used as part of a hybrid approach to solving the POMDP. In this approach the first action in a multi-step plan is selected by taking into account uncertainty in the state variables of the target machine (the belief state) but assuming that subsequent actions will be selected based on assuming the world is fully observable (i.e. using the solution to the MDP). If $Q^*_{MDP}(s,a)$ gives the optimal non-myopic value of taking action a in state s in a fully observable world, and $\boldsymbol{b}$ represents an initial belief state derived from given information, then the value of taking action a given the initial distribution is

$$Q(\boldsymbol{b},a) = \sum_{s \in S} b(s) Q^*_{MDP}(s,a)$$

While the Q-MDP algorithm does not provide good solutions to general POMDPs, it suffices for obtaining initial results in the domain of penetration test planning. Our formulation of the penetration test planning POMDP does not include information gathering actions within the POMDP. Therefore, assuming state uncertainty is resolved after the first action is not likely to cause us to choose suboptimal plans that ignore actions that can reduce the uncertainty in the belief state. To account for the actual uncertainty of executing actions, after each action, a, is executed and the feedback, o, received, a new belief state is calculated as follows:

$$b_{new}(s' \mid \boldsymbol{b},a,o) = \alpha * \Pr(o \mid s',a) * \sum_{s} \Pr(s' \mid s,a) * b(s)$$

α is a normalization factor

The complete algorithm for making use of the non-myopic MDP solution while myopically taking into account state uncertainty is then:

(1) solve the underlying MDP to obtain $Q^*_{MDP}(s,a)$
(2) start with an initial belief state **b** from information gathering
(3) execute the action with maximum value, $argmax_a\, Q(\boldsymbol{b},a)$
(4) receive observation (success signal)
(5) compute the next belief state, $\mathbf{b}=\mathbf{b}_{new}$
(6) repeat from step 3

In the next section we demonstrate our solution approach on several penetration test planning problems.

IV. EMPIRICAL EVALUATION

In this section we demonstrate our planning solution on a series of small example problems. In each example the goal of the penetration test is to obtain root privileges to the operating system of the target machine, starting without any privileges.

Consider a simple target host running Windows XP SP2 and no services. Our vulnerability assessment methodology retrieves all NVD CVE IDs that correspond to Windows XP SP2 and compares them against the available penetration testing exploit tools. Each tool that does not require any privileges as a pre-condition is evaluated by our tool success probability (TSP) metric. Table 3 depicts the result of this calculation. Column 2 of this table shows the TSP of each available tool.

Table 3 – Initial Action Ranking

Value	TSP(%)	Tool	CVE	Target Software	Pre	Post
0.435	16.61	22	2007-2218	MS, WinXP, SP2	None	Root
0.418	9.28	21	2007-1205	MS, WinXP, SP2	None	Root
0.403	5.53	20	2007-0214	MS, WinXP, SP2	None	Root
0.397	4.42	1	2007-0026	MS, WinXP, SP2	None	Root

Penetration test planning for this example does not require much non-myopic reasoning as all tools may achieve the goal in a single step. However, since tools may fail or cause side effects, the planner must still reason about multi-step plans. Column 1 shows the expected value of taking each action (i.e. executing the tool) as the first action in a multi-step plan to achieve root access. Actions are ranked according to their expected value, which can be interpreted as the probability of reaching a goal state if the action is taken. Since each tool may achieve the goal in a single step, the tools with the highest success probability are also the initial actions of the multi-step plans with the highest value.

Assume that the penetration tester executes Tool 22 and does not receive any success signal. Since the Q-MDP algorithm is myopic with respect to state uncertainty we only use the results for one action at a time. After each action the planner re-computes the belief state and re-plans. Table 4 shows the belief state after the first action is taken. While we were completely certain we had no privileges prior to applying the exploit tool, after the tool is applied and no success feedback is received we believe there is a 0.9845 probability we still do not have any privileges. However there is a slight probability that we achieved root privilege but the success signal was lost. Note that even if we receive a success signal we would not have complete certainty that the goal has been reached due to the possibility that the success signal was generated in error.

Table 4 – Belief State after Tool 22 without Feedback

Prob(%)	OS	OS.user	OS.root
98.45	MS, WinXP, SP2	False	False
1.55	MS, WinXP, SP2	True	True

Table 5 shows the results of re-planning in the new belief state that results from executing tool 22 and not receiving a success signal. An interesting point to note is that we have built into our POMDP model limited memory of previous actions. This allows the planner to realize that it is more valuable to take untried actions then to take an action that has recently been executed without a success signal. This is demonstrated by Tool 22 falling to the bottom of the ranking list, despite retaining the highest success probability (TSP).

Table 5 – Action Ranking after Tool 22 without Feedback

Value	TSP(%)	Tool	CVE	Target Software	Pre	Post
0.373	9.28	21	2007-1205	MS, WinXP, SP2	None	Root
0.365	5.53	20	2007-0214	MS, WinXP, SP2	None	Root
0.360	4.42	1	2007-0026	MS, WinXP, SP2	None	Root
0.345	16.61	22	2007-2218	MS, WinXP, SP2	None	Root

Consider an example in which we have initial uncertainty in what software is providing service on port 80 (due to uncertainty in information gathering). Table 6 depicts this initial belief state: 60% probability Microsoft IIS 5.0, 30% probability Apache Tomcat 3.3 and a 10% probability of an unknown application on port 80. Each combination of state variables is represented by one row in the table.

Table 6 – Initial Belief State

Prob(%)	OS	OS.user	OS.root	P80	P80.priv
60.00	MS, WinXP, SP2	False	False	MS, IIS, 5.0	False
30.00	MS, WinXP, SP2	False	False	Apache, Tomcat, 3.3	False
10.00	MS, WinXP, SP2	False	False	Unk, Unk, Unk	False

Assuming the penetration tester has tools that exploit potential vulnerabilities in Apache Tomcat 3.3 and Microsoft IIS 5.0, the planner produces a ranking of tools by expected plan value as shown in Table 7. Note that even though the tools for port 80 vulnerabilities are more likely to succeed (TSP) than many OS tools, the expected values of executing plans that start with those tools aren't as high as plans that start with tools that exploit the OS. This is because we are certain of the target OS but less sure about the application software.

Table 7 – Initial Action Ranking

Value	TSP(%)	Tool	CVE	Target Software	Pre	Post
0.514	16.61	22	2007-2218	MS, WinXP, SP2	None	Root
0.491	9.28	21	2007-1205	MS, WinXP, SP2	None	Root
0.474	5.53	20	2007-0214	MS, WinXP, SP2	None	Root
0.468	4.42	1	2007-0026	MS,WinXP,SP2	None	Root
0.290	13.68	9	2007-2815	MS, IIS, 5.0	None	Root
0.184	29.72	7	2007-3382	Apache, Tomcat, 3.3	None	Root

As in the previous example, after executing an action the planner re-computes the belief state based on the observation (success signal) and re-plans. Assume Tool 7 is executed against Apache. If the observation is success then the probability of Apache being present on port 80 increases from 30% to approximately 85%. If there is no signal then the probability of Apache being present decreases to approximately 7% and the probability of IIS being present increases to almost 80%.

The previous examples demonstrated vulnerability assessment, including tool success probability calculation, belief state updating and limited planning. Multi-step plans were generated that considered whether or not the first action would fail and require a new or repeat action. In the next example we consider the generation of penetration test plans that require multiple actions to succeed in order to achieve the user-specified goal.

Consider a penetration test planning problem in which there is initial uncertainty in both the applications and the operating system on the target machine. In this example problem we include a penetration test tool that only achieves user privilege (Tool 25) as well as a tool that requires user privilege and achieves root privilege if successful (Tool 3).

Table 8 – Initial Action Ranking

Value	TSP(%)	Tool	CVE	Target Software	Pre	Post
0.402	16.61	24	2007-2218	MS, Win2k, SP4	None	Root
0.393	45.47	25	2007-1912	MS, Win2k, SP4	None	User
0.290	13.68	9	2007-2815	MS, IIS, 5.0	None	Root
0.184	29.72	7	2007-3382	Apache, Tomcat, 3.3	None	Root
0.108	16.61	22	2007-2218	MS, WinXP, SP2	None	Root
0.000	47.62	3	2007-1212	MS, Win2k, SP4	User	Root

Table 8 shows the ranking of tools resulting from planning using the initial belief state. Note that Tool 3, the action that requires user privilege as a precondition, has an expected value of 0 as a first action. This is because any plan that starts with Tool 3 will be rejected because the initial belief state has zero probability of having user privilege.

The planner assigns the highest expected value to applying Tool 24, which tries to achieve root privilege in one step. The second highest expected value is a plan that starts with Tool 25. This must be a multi-step plan in order to achieve the goal of root privilege. This demonstrates that the planner has considered the possible actions to take after executing Tool 25 and has discovered that, if successful, Tool 3 will be available to achieve the goal of root privilege. To demonstrate this we simulate taking the action Tool 24 and not receiving a success signal followed by taking action Tool 25 and receiving a success signal. The resulting belief state is depicted in Table 9. Note that the three most likely states include successfully achieving user privilege.

Table 9 – Belief State after Tool 24 without Feedback and Tool 25 with Feedback

Prob(%)	OS	OS.user	OS.root	P80	P80.priv
50.75	MS, Win2k, SP4	True	False	MS,IIS,5.0	False
25.38	MS, Win2k, SP4	True	False	Apache, Tomcat, 3.3	False
8.46	MS, Win2k, SP4	True	False	Unk, Unk, Unk	False
3.46	MS, Win2k, SP4	False	False	MS, IIS, 5.0	False
2.67	MS, Win2k, SP4	True	False	MS, IIS, 5.0	True
1.85	MS, Win2k, SP4	True	True	MS, IIS, 5.0	True
1.73	MS, Win2k, SP4	False	False	Apache, Tomcat, 3.3	False
1.34	MS, Win2k, SP4	True	False	Apache, Tomcat, 3.3	True
1.27	MS, WinXP, SP2	False	False	MS, IIS, 5.0	False
0.92	MS, Win2k, SP4	True	True	Apache, Tomcat, 3.3	True

After re-planning in the subsequent belief state, the ranking of tools in Table 10 is achieved. Tool 3 is now the highest ranked tool, rising from 0 to 0.649 in expected value, indicating that the penetration test is most likely to achieve successful root privilege by applying this tool next. Initially it was not in the penetration tester's best interest to use a multi-step plan but after failing with the most valuable action, using the complementary tools was the best course of action. This example illustrates the planner's ability to look into the future, recommending actions that achieve non-goal states en route to goal states.

Table 10 – Action Ranking after Tool 24 without Feedback and Tool 25 with Feedback

Value	TSP(%)	Tool	CVE	Target Software	Pre	Post
0.649	47.62	3	2007-1212	MS, Win2k, SP4	User	Root
0.585	16.61	24	2007-2218	MS, Win2k, SP4	None	Root
0.375	13.68	9	2007-2815	MS, IIS, 5.0	None	Root
0.212	29.72	7	2007-3382	Apache, Tomcat, 3.3	None	Root
0.026	45.47	25	2007-1912	MS, Win2k, SP4	None	User
0.009	16.61	22	2007-2218	MS, WinXP, SP2	None	Root

This series of small example problems is relatively simple compared to the types of problems that are encountered during real-world penetration testing. However, the planning methodology discussed in this paper is designed to tackle more complex problems as well. Our experiments show that our planner can solve problems that expand ten thousand states in less than 10 seconds on a standard PC.

V. DISCUSSION

In this paper we have presented a methodology and implementation for automatically deriving multi-step penetration test plans. Our novel contributions include considering uncertainty in information gathering and uncertainty in the success of executing exploitation tools. We introduce a model for automated reasoning under uncertainty based on partially observable Markov decision processes (POMDPs) and demonstrate the use of an efficient approximation algorithm that satisfies the performance requirements of penetration test planning. Our methodology includes an automated assessment of vulnerabilities using an electronic vulnerabilities database (NVD) as well as a metric for evaluating the probability that a tool can successfully be applied to a vulnerability. Our automated planning methodology can find multi-step plans that may be difficult to discover manually and handles subtle interactions between information gathering uncertainty and tool execution.

REFERENCES

[1] O. Sheyner and J.Wing. Tools for generating and analyzing attack graphs. In Proc. of Formal Methods for Components and Objects, 2004.

[2] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric, Lecture Notes in Computer Science, Volume 5094, Data and Applications Security XXII, 2008.

[3] National Vulnerability Database, http://nvd.nist.gov/.

[4] The Laws of Vulnerabilities, G. Eschelbeck, Qualys, Inc., http://www.qualys.com/research/rnd/vulnlaws/.

[5] C. Papadimitriou and J. Tsisiklis, "The complexity of Markov decision processes," Mathematics of Operations Research, vol. 12, no. 3, pp. 441–450, 1987.

[6] K. P. Murphy. A survey of POMDP solution techniques. Technical report, Dept. of Computer Science, U.C.Berkeley, September 2000. http://www.cs.berkeley.edu/~murphyk/publ.html.

[7] D. Aberdeen, A (revised) survey of approximate methods for solving POMDPs, National ICT Australia, Tech. Rep., 2003. http://axiom.anu.edu.au/~daa/files/papers/pomdpreview.pdf

[8] M. L. Littman, A. R. Cassandra and L. P. Kaelbling. Learning Policies for Partially Observable Environments: Scaling Up. Proceedings of the Twelfth International Conference on Machine Learning. July 1995.

[9] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision process problems. Journal of the ACM, 47(4):681–720, 2000.