3-2021

Automated Network Exploitation Utilizing Bayesian Decision Networks

Graeme M. Roberts

AUTOMATED NETWORK EXPLOITATION UTILIZING BAYESIAN DECISION NETWORKS

THESIS

Graeme M. Roberts, Captain, USAF

AFIT-ENG-MS-21-M-076

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-MS-21-M-076

AUTOMATED NETWORK EXPLOITATION UTILIZING BAYESIAN DECISION NETWORKS

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science

Graeme M. Roberts, B.S.C.S.
Captain, USAF

March 25, 2021

AFIT-ENG-MS-21-M-076

AUTOMATED NETWORK EXPLOITATION UTILIZING BAYESIAN DECISION NETWORKS

THESIS

Graeme M. Roberts, B.S.C.S.
Captain, USAF

Committee Membership:

Gilbert L. Peterson, Ph.D.
Chair

Barry E. Mullins, Ph.D., P.E.
Member

David W. King, Ph.D.
Member

AFIT-ENG-MS-21-M-076

Abstract

Computer Network Exploitation (CNE) is the process of using tactics and techniques to penetrate computer systems and networks in order to achieve desired effects. It is currently a manual process requiring significant experience and time that are in limited supply. One way to supplement the shortage is through automation. This thesis presents the Automated Network Discovery and Exploitation System (ANDES) which demonstrates that it is feasible to automate the CNE process. The uniqueness of ANDES is the use of Bayesian decision networks to represent the CNE domain and subject matter expert knowledge. ANDES conducts multiple execution cycles, which build upon previous action results. This process simulates the iterative thinking process of human attackers. Cycles begin by modeling the current belief state using Bayesian decision networks. ANDES uses these networks to select and execute an expected best action. Observed results are used to update the systems current belief state before the next cycle begins. ANDES was tested in a live-execution event, taking place within a virtual network environment. The target network mimics a small business' internal network. ANDES successfully performed a series of information gathering and remote exploit actions, across multiple network hosts to gain access to the target.

Table of Contents

List of Figures

Figure Page

List of Tables

AUTOMATED NETWORK EXPLOITATION UTILIZING BAYESIAN DECISION NETWORKS

I. Introduction

1.1 Background

The *2018 National Cyber Strategy* identified four pillars the federal government deemed critical to "advance an open, secure, inoperable, and reliable cyberspace [1]". Pillar three establishes "preserving peace and security by strengthening the ability of the United States, its partners, and allies to deter and punish those who use cyber maliciously" as one of the critical steps the United States must take. Speaking at the Department of Homeland Security Cybersecurity and Infrastructure Security Agency's second annual Cybersecurity Summit in 2019, Former Defense Secretary Mark Esper espoused on how the Department of Defense (DoD) fits into this strategy.

> "But winning in cyberspace requires an offensive strategy. We need to do more than just play goal line defense. As such, the department's 2018 Cyber Strategy articulates a proactive and assertive approach to defend forward of our own virtual boundaries." [2]

Two out of the five DoD lines of effort identified to meet this national directive focus on cultivating and retaining talent. The country as a whole has a talent shortage that continues to grow in highly technical fields [3], Computer Network Exploitation (CNE) is no different. One way to help alleviate some of the demand for highly skilled technical labor is to utilize advances in the field of computer automation.

Researchers have consistently explored Automated CNE over the past two decades, however it has never moved towards mainstream adoption [4]. One cause for the lack of adoption is the complexity of the decision space and dynamic nature of CNE scenarios. Previous automation attempts have relied upon modeling entire networks and completely solving an attack plan prior to execution [4]. This approach is not representative of the iterative approach employed by human attackers. Attackers leverage tools and techniques to gain valuable information throughout their attack, which informs future decision making. The development of Automated Network Discovery and Exploitation System (ANDES) is an attempt to create an automated CNE system that aligns with the human attacker process.

Given the closely related nature of the CNE field and Vulnerability-Assessment and Penetration Testing (VA-PT) both problem domains are considered throughout this work. Unless otherwise specified the reader may assume references to either problem domain is applicable to both.

1.1.1 Existing Systems

With an average of only 2 research papers per year applying artificial intelligence to automated VA-PT being published between 2002-2017, and with only 32.26% being peer reviewed journal papers [4] it is easy to see how there is room for innovation. Of those papers 84% fall into three general AI disciplines:

- Contingent planning using Planning Domain Definition Language (PDDL)
- Markov Decision Process (MDP) / Partially Observable Markov Decision Process (POMDP) Reinforcement Learning
- Attack Graph / Trees.

These approaches and their associated strengths and weaknesses are examined in Chapter II.

Within the published work only a single group of researchers have focused on producing a system capable of live-network execution [5][6]. Their work focused on developing systems using POMDP capable of capturing the non-deterministic nature of the VA-PT domain to enable live-execution. Sarraute et al have integrated portions of their research into the commercial product *Core Insight Enterprise* distributed by *Core Security*, allowing for limited automation of VA-PT events [5]. Unfortunately, several of the assumptions made by [5][6] while operating in the VA-PT domain make their systems non-transferable to the CNE domain. ANDES attempts to build upon these previous works to produce an automated CNE system.

1.2 Research Objectives

The hypothesis is that using Bayesian decision networks to capture the CNE domain and Subject Matter Expert (SME) knowledge, combined with an iterative execution cycle, allows for functional automated CNE systems capable of augmenting human operators. This hypothesis was broken into several sub-goals:

- Develop and evaluate a Bayesian decision network which represents the CNE domain and incorporates SME knowledge and preferences.
- Develop and evaluate a decision making system which utilizes developed Bayesian decision networks to select system actions which correspond to SME action selection.
- Develop and evaluate a software system capable of conducting information gathering and exploitation actions in a live network environment.
- Develop and evaluate a software system capable of conducting iterative execution cycles, capturing previous action's output as input for future decision making.

- Test the developed system in a virtual environment, designed to enable live-execution test events.

Evaluation of the hypothesis included the development and thorough testing of ANDES. ANDES was tested within a virtual network range developed for this purpose. The virtual network range represents the uncertain, non-deterministic environment encountered by CNE operators.

1.3 Methodology

To accomplish the proposed objectives the author developed the Automated Network Discovery and Exploitation System (ANDES). The first innovation is ANDES's usage of Bayesian decision networks to effectively capture and model the CNE domain. The second innovation is the iterative approach ANDES utilizes during execution. Each execution cycle ANDES undertakes is an isolated event whose observed outcome is utilized to inform all future decisions. In this way ANDES is able to react to an uncertain environment and mimic the human decision making process of first gathering information, deciding what is the best course of action, and finally taking that action and observing the outcome. ANDES does not operate without making several assumptions regarding its domain, which come with associated limitations.

1.4 Overview of System Assumptions

It is critical to remember that the goal of ANDES is to demonstrate the feasibility of utilizing Bayesian decision networks along with iterative execution cycles to enable automated Computer Network Exploitation (CNE). Results presented in this work must be evaluated with the knowledge of these goals. ANDES was able to successfully operate in an unknown network, chaining together numerous information gathering and various exploitation actions, ultimately achieving access to an objective host.

The environment itself was simplified to include four target hosts, and three different remote exploit actions to be chosen. It is the author's belief these limitations do not detract from ANDES capability to demonstrate the goals it was developed to fulfill. The author acknowledges that ANDES is incapable of operating in a realistic target environment a CNE operator would face today. Future development and research is required to achieve this critical milestone, but the author believes ANDES is a step in the right direction that other contributors can build upon.

1.5 Results

Testing of ANDES shows it was able to achieve all of the desired research objectives. ANDES successfully and autonomously executed in a live-execution test network, gaining access to a target host. Throughout execution ANDES made sensible decisions that matched expected output of the Subject Matter Expert (SME) who designed the Bayesian decision network. Additionally it accounted for, and reacted appropriately to action failures, chaining together actions to compromise multiple hosts and reach the target host which was hidden within the target network, inaccessible from the external attack network. These results show the feasibility of the ideas introduced by ANDES and form the foundation for future work in the field.

1.6 Overview

The rest of this work walks the reader through the process of developing and testing ANDES, including the results obtained. Chapter II expands upon the relevant previous work in the field of automated CNE and VA-PT, before identifying gaps in that research. It also includes an introduction to Bayesian decision networks and how they function. In Chapter III the methods ANDES uses to accomplish the defined objectives are explored. The functionality of ANDES is broken down into three distinct systems, the **Control Component**, the **Decision Component** and the **Execution Component**. Each Component's architecture and responsibilities are explored. Testing of the system's functionality is reported in Chapter IV. ANDES underwent several rounds of testing throughout development, the most relevant of which are reported in this work. Reported results primarily focus on a culminating test, which saw ANDES operating within a live-execution target environment seeking to gain access to a hidden host via a series of information gathering and remote exploit actions. This work concludes in Chapter V with a summary of ANDES's contributions, results, and avenues of future research that would build upon this work.

II. Concepts and Background

Understanding the current state of applying Artificial Intelligence (AI) to the Vulnerability-Assessment and Penetration Testing (VA-PT) domain requires examining previous attempts at creating autonomous systems. This thesis work builds upon some of the works presented here, as well as avoiding many of the limitations previous systems had by consciously choosing to take different approaches. This chapter also serves as an introduction to critical concepts employed within Automated Network Discovery and Exploitation System (ANDES) .

This chapter begins with an overview of the Vulnerability-Assessment and Penetration Testing (VA-PT) process and the various methods of automation previous researchers have applied to the problem. Previous systems are broken down by the AI discipline applied within their systems, as well as the systems functionality. Special attention is given to systems capable of live-execution events as this most closely aligns with this works hypothesis. A formal definition of Bayesian decision networks is presented, to include an example network. This chapter concludes by identifying research trends which highlight gaps in the existing body of research, which this thesis attempts to fill.

2.1 Vulnerability Assessment and Penetration Testing (VA-PT)

As the complexity and criticality of computer systems continues to increase across all facets of our lives, the importance of securing these systems correspondingly increases [7]. One process to measure a computer system's security is through Vulnerability-Assessment and Penetration Testing (VA-PT). VA-PT is the process of analyzing systems, looking for vulnerabilities or other weaknesses that allow the system to be exploited or misused in some way, ultimately enabling some type of attack [7].

The VA-PT processes can be extremely resource intensive as it typically requires highly trained and experienced personnel dedicating time and resources to the problem. These limitations led researchers to explore to the idea of automating VA-PT actions, using computer based systems to either augment or replace trained security professionals. The major difficulties facing automated VA-PT systems is the ability to capture the Subject Matter Expert (SME) knowledge of security professionals, as well as their ability to synthesize observed information into optimal decision policies. This last capability of adapting to the observed state of the domain proves particularly difficult and is a major focus addressed in this work.

It must be remembered that ANDES has been designed with automated Computer Network Exploitation (CNE) as the primary objective. The similarities between VA-PT and CNE are a closely-related research problems, allowing for sharing knowledge between the two applications of the technology. CNE problems, if anything, are more difficult than corresponding VA-PT problems due to having additional constraints on the knowledge domain and additional risks of detection. Given the nature of non-permissive CNE, in contrast to VA-PT, there is a general lack of widely available research related to the specific field. This lack of available research led to the focus on automated VA-PT research examined in this work instead of works detailing automated CNE research.

2.2 Attempts at Automation

There has been a variety of methods applied to the problem of automating VA-PT [4]. This work focuses on those applying Artificial Intelligence (AI) and Machine Learning (ML) to the problem. For the purposes of this work, AI is defined as the employment of a rational agent within the problem domain. For an agent to be rational it "acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome." [8]. It should be noted, most of the previously presented systems do not behave as an 'actor', instead they largely model a static environment with no mechanism for interacting with their environment or observing action and event outcomes. This fact alone disqualifies most of systems from feasibly enabling automated CNE events.

Defining how the CNE / VA-PT process can be broken down into individual steps, which can subsequently be automated provides insight into the research problems that past researchers have been trying to solve. Those steps are defined by [9] and consist of:

- Pre-engagement Interactions
- Intelligence Gathering
- Threat Modeling
- Vulnerability Analysis
- Exploitation
- Post Exploitation
- Reporting

Of those steps, the first and last steps are considered outside the scope of current automation and correspond to system development and analysis.

Previously explored solutions can be divided into a number of categories related to foundational AI/ML principles. The categories are classical planning (constraint satisfaction), probabilistic planning, adversarial planning (min-max), live-execution, and adaptive planning (learning-models). The work conducted by McKinnel et al. in [4] presents a recent review of contributions to the research field. The following section highlights prominent works across these various categories as well as evaluates their relative contributions to the advancement of the field.

2.2.1 Classical Planning

The earliest attempts to apply a level of automation to VA-PT came from work into developing what is known as “Attack Graphs” or “Attack Models”. In [10] the authors describe Attack Graphs as a succinct way to represent the various paths through a system available to an attacker that allows them to achieve their desired goals. Various methods have been used to create these Attack Graph models, but they all generally model the network in a manner that allows the application of Constraint Satisfaction Problem (CSP) algorithms to find viable attack paths through the network [10][11][12][13][14][15][16]. The CPS algorithms are able to match exploits to vulnerabilities, configurations to misuse, etc., until an attack chain is found. These systems generate all possible attack chains that could theoretically compromise a critical host or achieve a designated objective.

These early models were an application of classical planning to the problem of VA-PT. The models contained a description of the network as state variables, a starting condition as the initial state, critical hosts / data as the goal condition, and available exploits as the set of available actions. Each action (exploit) required an existing vulnerability or pre-condition to be met and generated some effect or post-condition [16]. As is the case with classical planning, the solutions generated by these algorithms were directed graphs showing paths from an uncompromised network (initial condition) and ending at a compromised network. A simple example of one round of this type of action selection / execution can be seen in Figure 1.

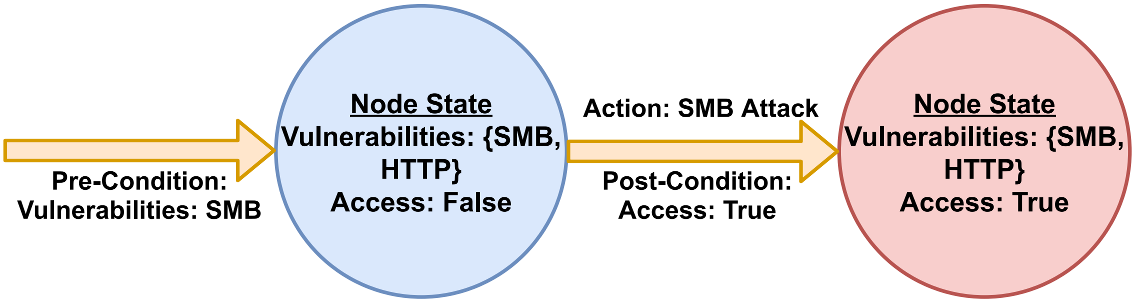

Node State
Vulnerabilities: {SMB, HTTP}
Access: False
Action: SMB Attack
Node State
Vulnerabilities: {SMB, HTTP}
Access: True
Pre-Condition: Vulnerabilities: SMB
Post-Condition: Access: True

Figure 1: Shown here is an example of how the planning algorithms would match a pre-condition for an action to a node's state, conduct that action, and apply the post-condition to produce the node's new state. In this case the algorithm is performing the SMB attack algorithm to gain access to the host.

2.2.2 Probabilistic Planning

These early classical planning models were extended to allow for probabilistic planning. Probabilistic planning adds the capability to account for uncertainty within the VA-PT domain to existing classical planning models. Probabilistic planning for VA-PT led to several new avenues of research.

One avenue explored was the introduction of models attempting to solve not only the binary question of whether a given network is secure, but to quantify how secure a target network is [17][18][19][20]. This is most exemplified within [17] which chose to represent the networks as a standard Bayesian network. Given this adaptation for how to represent the problem domain, common solvers for Bayesian networks could be applied to the problem. Additionally, the generated solutions are no longer merely binary states of whether a system is vulnerable or not, but show a measure of how vulnerable the network is. This is accomplished by solving for the likelihood certain state variables would exist within Bayesian networks, and considering that as the likelihood existing vulnerabilities might lead to successful exploitation in the VA-PT problem domain.

Expanding these earlier models of probabilistic planning allowed for the introduction of state uncertainty. To accommodate the desire for state uncertainty researchers looked towards the introduction of Partially Observable Markov Decision Process (POMDP) as the domain model used to represent the VA-PT problem. This application allows the system to represent both imperfectly known states as well as account for uncertain dynamics (the domain changing outside of agent interactions.) Within [5] and [6] we see the introduction not only of POMDPs applied to the problem of automated VA-PT, but also initial attempts to deal with state uncertainty.

A severely limiting assumption of all previously referenced models has been their assumption of perfect knowledge of their target networks and their corresponding domain states. While this assumption simplifies the problem space it does not accurately represent the state of most real-world computer networks. The system developed by Sarraute et al. [5] introduced information gaining as possible actions choices for the system. These actions allow the system to attempt to turn uncertainty into known states. These actions are necessarily associated with a cost which is reflected in the POMDP's reward calculations.

Despite the introduction of information gaining actions, the developed system is still limited in capability to model unknown environments. The main limitation in the model is their reliance upon a known initial starting state for the domain [5]. The developed system begins with a complete initial knowledge of the problem domain and proceeds to artificially introduce state uncertainty into the system. This serves as an attempt to mimic a changing / dynamic network environment. While this proposed solution could be considered adequate in the usage of conducting recurring VA-PT events in a known customer's network, it is a limiting assumption when dealing with our desired case of automated non-permissive CNE. It should be noted, that given the systems information gathering capabilities it could theoretically be adopted to work in unknown environments, however the method by which the POMDP is constructed and solved would need to be fundamentally altered.

Another serious limiting factor of the system proposed by [5] is the computational complexity associated with solving POMDPs. The authors' attempted to mitigate this issue by developing a clever way of decomposing the network into subgraphs of connected systems within [6]. This allows for optimal solutions of the decomposed sub-problems to be combined into an approximate fully-optimal solution for the entire system. This helps mitigate the complexity problem, but for very large domains it is still a limiting factor.

2.2.3 Machine Learning

Previously discussed models rely upon expert knowledge being imparted during domain model construction and are static during employment. The work by [21] proposes and evaluates an AI-based VA-PT system which makes use of machine learning techniques to allow the automated VA-PT system to learn and update characteristics about the domain as execution progresses. Specifically the authors recommend using Reinforcement Learning (RL) to learn and reproduce both average and complex penetration testing activities. Just as past research has done, [21] utilize a POMDP to model the complex and dynamic nature of VA-PT. Acknowledging the limitations automated systems have had in the past, these authors proposed a system capable of being trained by capturing human expertise throughout numerous learning events. The system learns by observing human experts' actions, as well as asking human experts for decision choices during its own execution and remember those choices. The aim of this learning model is to produce a system that can not only reproduce the human levels of effectiveness but build upon them. The aim is to allow the system to cleverly and quickly take learned knowledge and applying it in novel ways to future penetration testing events leading to a more efficient, effective and scalable system eventually able to operate without relying upon human expertise. The combination of live execution, reinforcement learning, POMDP model solving and development of ‘policies’ (chains of individual actions to achieve some desirable effect) are all novel in their approach and represent the culmination of many years and previous threads of research. One aspect the authors have still assumed is nearly complete knowledge of the target network and a pre-built world model upon which to operate. While their POMDP model allows for some level of uncertainty regarding absolute states and probabilistic outcomes (live execution leads to a level of stochasticity that has to be accounted for) the model is solved with the assumption of perfect knowledge and probabilistic actions. The system generates an entire attack plan prior to execution. Execution feedback is fed into the solver after each action execution and the attack plan is updated according to observed results. This serves to merely create a new complete plan from the current state instead of truly dynamic and iterative model solving.

Most critically however is the fact the work presented within [22] and [21] is aspirational. Many of the learning features are still being developed and implemented. The authors have proposed a long-term project with many developmental goals still to be met in the future, with nearly non-existent results currently presented. In this sense the work should be thought of as a guiding document for what features a robust system should contain, and roughly how that development and learning process might look.

2.3 Real-World Execution

Even with the capabilities provided by POMDP models allowing systems to more accurately model the real world, very few systems are capable of functioning within non-simulated environments. Those that do interact with real-world environments, nearly universally operate off domain snapshots as opposed to interacting with the dynamic nature of the environment. The working assumption is that an accurate representation of the target network will be provided as input into the system. Often the requirements of the system represent an unrealistic perfect knowledge of the entire network domain. Given the domain in which ANDES is intended to operate within, it can be assumed this level of knowledge is functionally unattainable.

The first real attempt at incorporating these research concepts into a usable product was seen in [23]. This work used the Metric-FF planner [24], an extension of classical PDDL planning systems, and applied it to automated VA-PT. To accomplish this goal the authors integrated the planning system with an existing penetration testing tool, *Core Impact*. The objectives accomplished was the ability to convert the internal state of *Core Impact* into the language of the PDDL planning tool and to convert the attack plan output back into a format usable by *Core Impact*. As seen previously though this system once again considered an entire target network as a whole and generated an attack plan designed to traverse the entire target network, assuming deterministic outcomes. The key contributions were the ability to take the state of a real-world VA-PT tool and utilize it as the input to a planning system, and the capability to take the planning systems output and execute it in a real network. Realizing a main weakness of the system came from the inability for the planning system to account for uncertainly and these limitations ultimately led the authors to pursue the work shown in [5] and [6]. Unfortunately the authors never adapted the POMDP systems of [5] and [6] into a working system capable of interacting with a live network. This is perhaps due to the fact that several of the assumptions the authors made while developing the system are incompatible with live execution and would need to be redesigned.

Regarding the current state of implementation, the current version of the *Core Impact* tool offers what they call Rapid Penetration Test (RPT)s which automate portions of the VA-PT process [25]. Specifically, you are able to provide the tool a computer system as a target and it will automatically conduct information gathering actions, develop an attack profile, conduct those attack steps and ultimately report the results. The system is incapable of using gathered information or access to pursue additional network targets and simply provides a summary of the results as the output. The system does not attempt to identify a 'best' point of access to the system, but merely attempt all exploits meeting certain system thresholds determined by the user's chosen rule-set (level of risk, type of exploit, etc.). *Core Security*, the developers of *Core Impact*, do not elaborate upon what underlying planning system, if any they are utilizing for their commercial product, presumably to protect their commercial interests. However, based upon their descriptions and the systems capabilities it can be assumed that at the very most a PDDL planning system is being utilized as shown in [23], but very likely it is a simple constraint satisfaction type system due to the behavior of attempting all 'valid' exploits instead of reasoning the 'best' option.

Another difficulty towards allowing systems to operate within real-world environments is the creation of the system's domain representation. The starting state of the network, required for most of the presented systems, needs to be externally produced and provided to the automated systems prior to execution. One solution to this limitation as presented in [13] relied upon commercial network vulnerability scanning products. These products rely upon having administrative credentials and knowing the layout of the target network. The results of the commercial scanning products were imported into a model building system. This solution severely limits the applicability to unknown environments or non-cooperative VA-PT scenarios.

2.4 Bayesian Decision Networks

From the outset Bayesian decision networks, also known as influence diagrams, have been explored as a solution to deal with uncertainty and incomplete knowledge within the AI community. A foundational description of Bayesian decision networks is presented by [26]. The authors describe an easy way to conceptualize these Bayesian decision networks are as Directed Acyclic Graph (DAG)s in which "(i) nodes represent individual variables, (ii) arcs demonstrate influence among the nodes, and (iii) functions associated with the arcs indicate the nature of that influence." As evidenced by the name, Bayesian decision networks are an extension to the standard Bayesian networks. The introduction of additional variable types allows these networks to capture domain knowledge and decision maker preferences.

A valuable example highlighting the capacity for these Bayesian decision networks to capture the preferences of decision makers and automate the decision making process exists within the medical community. Here Bayesian decision networks are actively employed to help produce consistent and mathematically sound decision making. The work of [27] shows how Bayesian decision networks are implemented within Elvira, a tool for building and evaluating graphical probabilistic models. Elvira is used to help teach medical decision makers probabilistic and repeatable decision making skills [27]. ANDES follows the same mathematical principles as Elvira during network construction, the details of which are presented in the following section.

2.4.1 Formal Description of Bayesian Decision Networks

For this work it is important to understand the mathematical principles that underline Bayesian decision networks as well how to formally define them. These principles are used in Chapter III when describing the ANDES decision network and are critical to understanding how ANDES functions.

2.4.1.1 Notation and Definitions

Notation within Bayesian decision networks is derived from DAG notation. Each node within the DAG represents a variable within the Bayesian decision network. Variables have several ways to reference them. A capital V represents a variable, while a lowercase v represents a value of V. A bold capital **V** represents a set of variables while a bold lowercase **v** represents a configuration of **V**.

Useful definitions include a *finding* f, which corresponds to a value a chance variable can take on. In the case of ANDES, a finding corresponds to a belief ANDES maintains about a specific target. For example, ANDES might 'find' a target host is running the Windows 7 Operating System. The set of all available findings results in *evidence* e, which is a certain configuration of observed variables E [27].

Bayesian decision networks contain three types of variables: chance variables $\mathbf{V}_C$, decision variables $\mathbf{V}_D$, and utility variables $\mathbf{V}_U$. Chance variables are those which represent domain states the decision maker has no direct control over. Decision variables are those which represent actions directly controlled by the decision maker. Lastly utility variables represent the decision maker's preferences. Since variables are represented as nodes within the graphical representation, the terms variable and node are used interchangeably [27].

2.4.1.2 Network Structure

To produce the graph structure of the Bayesian decision networks, nodes are connected via directional arcs. Every node must be connected to at least one other node and no cycles may exist, fulfilling the requirements of the DAG designation. An incoming arc's meaning is determined by the type of connected node:

- Arcs into a decision node represent information known at the time of the decision

- Arcs into a chance node represent probabilistic dependence

- Arcs into a utility node represent functional dependence, i.e., which node values are used to calculate utility

Another characteristic of Bayesian decision networks is the requirement a directed path exists which includes all decision nodes, indicating the order of the decision nodes. This ordering produces a partition of $\mathbf{V}_C$ such that for a network having n decisions $\{D_0, ..., D_{n-1}\}$, there are $n+1$ subsets $\{C_0, ..., C_n\}$. Each subset $\mathbf{C}_i$ is the set of chance variables C such that there is a link $C \rightarrow D_i$ and no link $C \rightarrow D_j$ with $j < i$ [27]. This means $\mathbf{C}_i$ represents the set of chance variables which are known for D_i but are unknown for any previous decisions. The final set $\mathbf{C}_n$ is the set of variables which have no link to any decision. These sets allow the informational predecessors of D_i to be determined. Denoted as $IPred(D_i)$ it represents the variables known to the decision maker when deciding on D_i. Assuming the decision maker remembers all previous observations and decisions this results in the following property:

$$IPred(D_i) = IPred(D_{i-1}) \cup \{D_{i-1}\} \cup \mathbf{C}_i. \tag{1}$$

Similarly every utility node U has functional predecessors $FPred(U)$ which define the domain of the utility function. For a standard Bayesian decision networks this simply means the $FPred(U) = Pa(U)$.

Once the Bayesian decision network has been defined, the quantitative values associated with the variables must be set. Each chance node C requires a probability distribution $P(c|pa(C))$ for each configuration of its parents. This is most commonly represented as a Conditional Probability Table (CBT) listing the probability the event will occur given every possible configuration of the node's parents. Each utility node U requires a function $\psi_U(pa(U))$ which maps every possible configuration of the utility node's parents onto a real number. This utility function is also commonly displayed as a table in graphical form.

2.4.2 Bayesian Decision Network Example

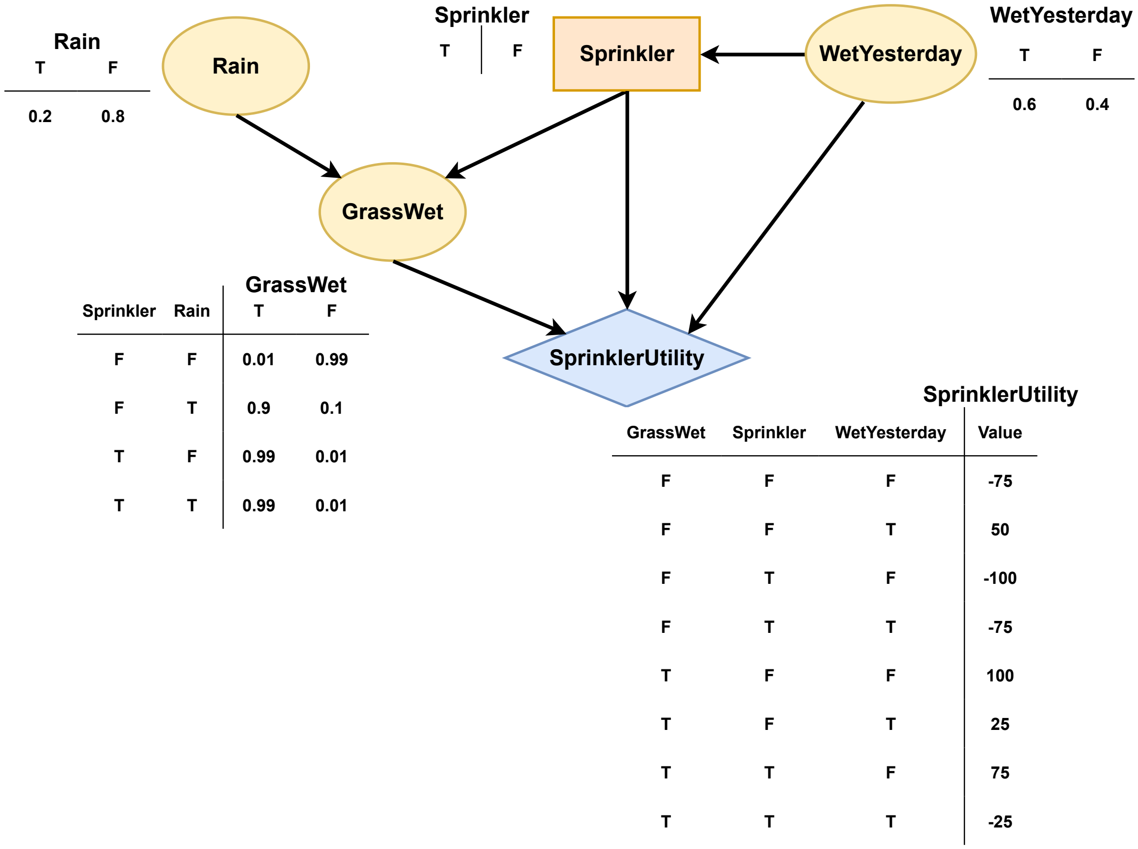
Using the properties of Bayesian networks, along with the described extensions provided by decision networks, one can calculate utilities for all combinations of possible decisions. The decision network is populated with available evidence, the evidence is propagated, and the network is repeatedly solved for every combination of decisions. This principle is shown in the following equation

$$P(\mathbf{v}_C : \mathbf{v}_D) = \prod_{C \in \mathbf{V}_C} P(c|pa(C)). \tag{2}$$

In theory, the optimal decisions are those that lead to the maximum utility value, a measure which captures the decision network creator's preferences via the utility value function. The exact method of how decision networks have been applied to the VA-PT problem within ANDES will be covered in Chapter III.

2.4.3 Example Network

The example network shown in Figure 2 demonstrates how a Bayesian decision network can be used to make decisions in the face of imperfect knowledge, while accounting for operator preferences. This network is an extension of the common Bayesian network example showing the probability of the grass being wet, given the given the presence of a sprinkler or rain. In this case, the state of the sprinkler is no longer a chance variable and instead is a decision variable, representing whether the operator of the sprinkler system chooses to turn on the system in the morning or not. The operator must make the decision without knowledge of whether it is going to rain, but they are aware of whether the grass was properly watered the previous day. The network has been designed to balance the cost of running the sprinkler system with the grass's requirement of being properly watered. The utility function



Rain

T	F
0.2	0.8

Sprinkler

T	F

WetYesterday

T	F
0.6	0.4

		GrassWet	
Sprinkler	Rain	T	F
F	F	0.01	0.99
F	T	0.9	0.1
T	F	0.99	0.01
T	T	0.99	0.01

			SprinklerUtility
GrassWet	Sprinkler	WetYesterday	Value
F	F	F	-75
F	F	T	50
F	T	F	-100
F	T	T	-75
T	F	F	100
T	F	T	25
T	T	F	75
T	T	T	-25

Figure 2: Sprinkler Utility: This Bayesian decision network has been constructed to aid the decision maker in choosing whether to turn on their sprinkler system or not. It takes into account decision maker preferences regarding how often they wish the grass to be watered. Shown are the nodes, connections and associated CBTs.

is constructed in such a manner as to favor the grass being watered every other day, and will penalize both too frequent or too infrequent watering. The network consists of three chance variables shown as nodes on the graphical representation. Each of these variables is either **true** or **false**.

$$\mathbf{V}_C = \{V_{Rain}, V_{GrassWet}, V_{WetYesterday}\} \tag{3}$$

Of these variables, two of them are directly observed as evidence.

$$\mathbf{E} = \{Rain, WetYesterday\} \tag{4}$$

There is a single decision variable, the decision to turn on the sprinkler system or not (encoded as **true** or **false**).

$$\mathbf{V}_D = \{Sprinkler\} \tag{5}$$

Finally there is a single utility variable which calculates the networks overall utility given the observed states.

$$\mathbf{V}_U = \{SprinklerUtility\} \tag{6}$$

Associated probabilities and values for all variables can be found in Figure 2 in the CBTs and utility function. As mentioned previously incoming arcs to chance nodes represent conditional relationships, incoming arcs to decision nodes represent information predecessors, and incoming arcs to utility nodes represent functional predecessors. Within this network only $V_{GrassWet}$ is conditionally dependant upon other variables.

$$Pr(V_{GrassWet}) = Pr(V_{Rain})Pr(V_{Sprinkler}) \tag{7}$$

The only information the operator has access to when deciding whether to turn on the sprinkler system is whether the grass was properly watered the previous day.

$$\mathit{IPred}(V_{Sprinkler}) = \{V_{GrassWet}\} \tag{8}$$

When determining the value of the network's configuration, the utility function will consider whether the grass will be sufficiently watered, whether the sprinkler system is running, and whether the grass was properly watered the previous day.

$$\mathit{FPred}(V_{SprinklerUtility}) = \{V_{GrassWet}, V_{Sprinkler}, V_{WetYesterday}\} \tag{9}$$

The process of using this Bayesian decision network to determine whether the operator of the sprinkler system should turn on the sprinklers demonstrates the ease of use. Given an observed evidence, the network is solved all possible decision choices (in this case there are only two, **true** or **false**) and the decision choice yielding the greatest expected utility is selected. For instance assume our evidence is

$$e = \{v_{Rain} = \mathbf{T}, v_{WetYesterday} = \mathbf{F}\}. \tag{10}$$

Propagating the evidence throughout the network and solving for each decision choice yields the Expected Utility (EU) value of each decision. Evidence propagation in the case of Bayesian decision networks is synonymous with computing the posterior probability of each variable given the available evidence [27]. Starting with the choice to run the sprinkler system, the EU can be calculated by examining the given probability tables and combing all possible outcomes. In this case that results in:

$$\begin{aligned}
&(Pr(GrassWet = T|Rain = T, Sprinkler = T)\\
&\times \psi_{U\,SprinklerUtility}(GrassWet = T, Sprinkler = T, WetYesterday = F))\\
&+ (Pr(GrassWet = F|Rain = T, Sprinkler = T)\\
&\times \psi_{U\,SprinklerUtility}(GrassWet = F, Sprinkler = T, WetYesterday = F)) = EU
\end{aligned} \tag{11}$$

which evaluates to

$$(0.99 \times 75) + (0.01 \times -100) = 73.25. \tag{12}$$

In the case the sprinkler system is turned off the EU is calculated as

$$
\begin{aligned}
&(Pr(GrassWet = T|Rain = T, Sprinkler = F)\\
&\times U(U_{SprinklerUtility}|GrassWet = T, Sprinkler = F, WetYesterday = F))\\
&+ (Pr(GrassWet = F|Rain = T, Sprinkler = F)\\
&\times U(U_SprinklerUtility|GrassWet = F, Sprinkler = F, WetYesterday = F)) = EU
\end{aligned}
\tag{13}
$$

which evaluates to

$$(0.9 \times 100) + (0.1 \times -75) = 82.5. \tag{14}$$

Given the observed evidence of V_{Rain} being **true** and $V_{WetYesterday}$ being **false**, the decision with the Maximum Expected Utility (MEU), with a value of 82.5 is to leave the system off. This process can be repeated for any combination of observed evidence and unknown evidence, yielding an expected optimal decision for any network configuration. This is what allows Bayesian decision networks to make decisions in the face of unknown information and probabilistic outcomes.

2.5 Limitations of Previous Systems

Out of the previously proposed systems, the ones most similar in the approach presented by this thesis are those utilizing POMDP models to account for uncertainty. The works of [5] and [6] developed an initial system utilizing a POMDP to enable automated VA-PT. Unfortunately their approach had several main limiting factors which prevent the direct application to a live execution system. First their system still assumes knowledge of network topology, system configurations and a system's inherent value (required for the reward functionality.) The system assumes the target network is static during execution, outside of the agent's interactions, and they assume the system's actions are deterministic given a known configuration. Combine these assumptions with the well documented complexity issues of POMDP models [6] and you realize the system was not designed to operate in a live network environment. These POMDP model approaches could potentially be modified to eliminate many of the assumptions made, however this thesis proposes that instead utilizing Bayesian decision networks provides many of the same advantages, without the accompanying issues.

Another trend that presents itself within the previous work is their systems of performance and evaluation. The authors have focused on comparing their systems' computational performance to previous approaches, showing how their new approach can handle more complex networks, or solve the problems quicker. The work of McKinnel et al. in [4] identifies the need for researchers to move towards a more qualitative performance metric. If an automated VA-PT system is to be valuable to real world operators it should be a given it is able to operate in the real-world environment, with all of its complexities. Instead the performance metric of newer systems should be attempting to capture the measure of value they provide to the operator, or attempt to capture how 'well' they perform the automated VA-PT tasks, and not simply whether they 'can' theoretically solve the mathematical problem of the tasks. The work presented in this thesis attempts to follow the advice provided by McKinnel et al. and move towards filling the gap of a system that is useful to a real world VA-PT operator.

2.6 Summary

Across all of these approaches, general trends appear regarding which portions of the automation problem they attempt to solve and how they attempt to solve them. With very few exceptions the systems could all be classified as planning agents attempting to produce some type of plan output, capable of leading to an end-state achieving some VA-PT objective, usually a compromised target system. While this work does not change that general dynamic, it approaches the problem instead not as a single planning problem, but instead as a series of decision points. The results of executing the chosen actions can then be observed by the agent and used to inform future decision making.

This work presents ANDES to fill this gap in existing systems by developing an automated system capable of network enumeration, model generation, sequential decision making, and plan execution. Previous research has focused on simulating automated VA-PT, this means the systems were designed to operate under different conditions and assumptions than ANDES. This focus on simulation means these systems are fundamentally different in their objectives and operations than a system focused on enabling live execution required for automated CNE events. This solution allows for real-time, live exploitation events, as well as being capable of operating within previously unknown environments. Since ANDES begins with only a knowledge of its own initial state and desired outcome, the previous examples which have attempted to develop complex network models prior to beginning execution are not feasible or even necessary for this solution. Instead ANDES relies upon Bayesian decision networks to model its environment and iterative execution cycles as will be seen in Chapter III.

III. Methodology

This chapter provides an in-depth look into how Automated Network Discovery and Exploitation System (ANDES) accomplishes the different objectives that make up the Computer Network Exploitation (CNE) process. Understanding the methods ANDES uses to accomplish the CNE process is critical to understanding how it differs from previous approaches and how its contributions are valuable to the field of automated CNE research.

ANDES can be logically separated into three major components as seen in Figure 3, each enabling a critical element of the CNE process. For the purposes of this work the process of CNE consists of examining the current understanding of the environment, viewing all possible decision choices, comparing those decisions with the context of prior knowledge and current beliefs, before finally choosing the action with the greatest chance of leading to objective completion. The first component is the Control Component which guides the flow of execution between the various other components and manages ANDES knowledge of the world and ANDES own internal state. Next is the Decision Component which receives the Control Component's current beliefs about the domain and decides the best next action for ANDES to take. The final component, the Execution Component, is the portion of ANDES that directly interacts with the target domain. The inner workings of those components, and how they differ from previous approaches is presented here. Appendix C contains technical information regarding the systems installation and dependencies.

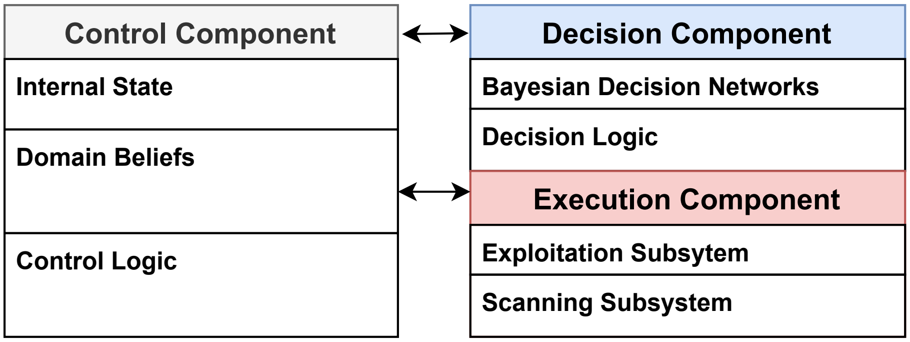


Figure 3: ANDES three components work together to accomplish required objectives. The Control Component provides current beliefs to the Decision Component which determines the next action for the system to take. The Control Component processes the selected action and tasks the Execution Component accordingly providing requisite targeting information. The Execution Component provides action results which are captured by the Control Component before beginning the cycle anew.

3.1 Purpose and Goals of ANDES

The primary purpose of ANDES is to evaluate the hypothesis that using Bayesian decision networks to capture the CNE domain and Subject Matter Expert (SME) knowledge, combined with an iterative execution cycle, allows for functional automated CNE systems capable of augmenting human operators. Given ANDES is a research project, meant to solely demonstrate feasibility of the employed concepts, it is critical to view ANDES capabilities through this lens. All methodologies presented here focus on achieving the sub-goals of this research:

- Develop and evaluate a Bayesian decision network which represents the CNE domain and incorporates SME knowledge and preferences.

- Develop and evaluate a decision making system which utilizes developed Bayesian decision networks to select system actions which correspond to SME action selection.

- Develop and evaluate a software system capable of conducting information gathering and exploitation actions in a live network environment.

- Develop and evaluate a software system capable of conducting iterative execution cycles, capturing previous action's output as input for future decision making.
- Test the developed system in a virtual environment, designed to enable live-execution test events.

The nature of ANDES means it makes several critical assumptions accompanied by various limitations that impact its ability to operate within the complexities of a real world network.

3.1.1 Assumptions and Limitations

The single largest assumption ANDES makes is that it only needs to successfully operate within a controlled test environment. ANDES is able to execute within any standard TCP/IP network, but if no profitable actions are available for ANDES it will terminate execution prior to objective completion. Currently ANDES terminates execution once the objective host has been compromised, it does not support delivering CNE post-exploitation effects to the objective host. These assumptions introduce the following limitations:

- Target network must be designed to have existing vulnerabilities ANDES is able to exploit.
- ANDES assumes all hosts in the target network share a subnet.
- ANDES only conducts remote service exploit actions, no host-based exploits are employed.
- If ANDES is to achieve the desired objective, an attack path must exist between the initial target host and the objective host.

- ANDES does not employ any Deny, Degrade, Disrupt, Destroy, or Manipulate (D4M) capabilities.

3.2 Control Component

The Control Component of ANDES guides system execution and coordinates information flow between the other two components. The Control Component also maintains and updates the systems internal belief state and domain knowledge. The process of capturing and utilizing information gained from previous execution cycles allows ANDES to conduct multiple, isolated actions to accomplish incremental goals. As long as ANDES Decision Component has properly defined utility functions, the culmination of individual goals should come together to the accomplish ANDES's user defined objective. Prior to any of this happening however ANDES must be able to capture an internal representation of the domain its operating within.

3.2.1 Domain Representation

One of the first tasks in developing an autonomous agent is determining the state representation, or how to best capture the domain in which it operates. The agent must be able to capture relevant information about the world in some kind of internal representation. ANDES is designed to operate within a standard IP based computer network. The critical components of a computer network that ANDES captures are the details of the computer systems themselves and the connections between them. ANDES represents both computer systems and their connections within ANDES host objects. Host objects are a custom data structure which capture not only various possible configurations of computer systems such as their operating system, hostname, IP address, etc. but also captures ANDES relationship to the system. Possible relationships are whether ANDES has performed a scan of the host, results from

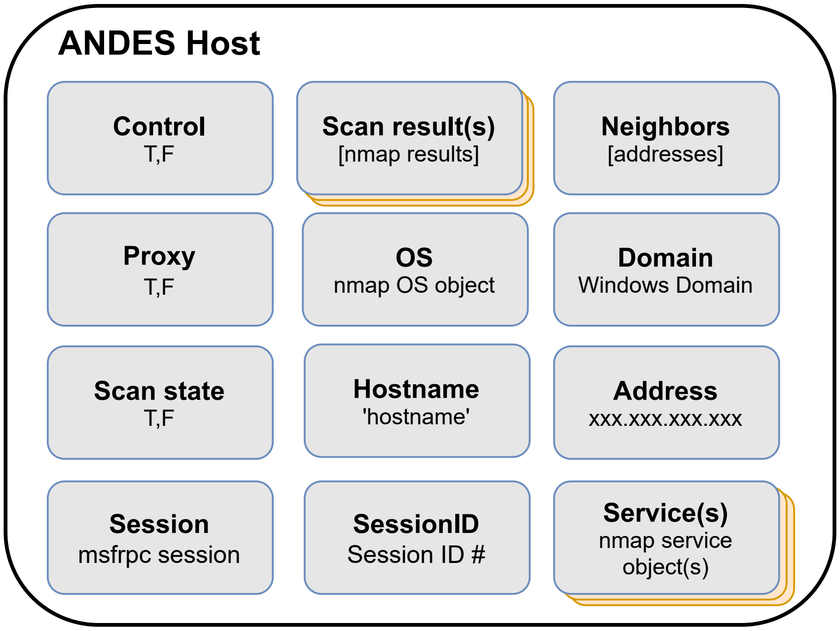


Figure 4: The ANDES host object captures applicable information about ANDES target systems. Host attributes are populated as information is acquired during information gain actions, such as conducting a remote scan. Attribute values represent ANDES current beliefs regarding the hosts state.

any previous scans, what type of access ANDES has to the host, if any, and any known neighbors of the host. Currently ANDES does not capture any additional metadata about the target network, and all observations / sensed information about the environment is captured only within ANDES host objects. A complete view of an ANDES host is provided in Figure 4.

Additional information regarding ANDES own internal state and the status of execution is captured, but is stored separate from the domain representation. The format of ANDES internal state is described later in this chapter.

3.2.2 Populating and Updating Target Network

During initialization of ANDES, the user must provide initial targeting information (entry point into target network) and a terminating objective. The minimum information required for both target systems is simply their IP addresses, however any additional information the user knows about the network could also theoretically be provided. If ANDES is unable to access the entry point provided by the user, execution will be terminated. When ANDES learns about a new target system (whether via initial target information provided by the user, or as the result of an information gathering action) it creates an associated Host object as described above. Host objects are stored in an unsorted list representing the entirety of the target network as currently observed by ANDES. Host objects are accessed via their associated IP addresses, which means any observed systems with matching IP addresses are assumed to be the same target host. In the case where newly observed information is associated with a previously observed host, the existing belief states are updated with the newer information. Any previous belief states that are not in conflict with new information are retained. When new information conflicts with previous belief states, the old information is assumed to be outdated and is replaced with the updated belief states. This is a minor limitation as separate systems could theoretically be configured to share external IP addresses, such as in virtual environments.

3.2.3 Execution Cycle

Once ANDES has a populated target network, containing at least one host object, it can begin conducting execution cycles. Figure 5 shows an overview of this process and the seven individual steps. Each execution cycle begins with a decision selection process and ends with checking for objective completion. More detailed explanations of each individual step are included in the responsible Component section.

3.2.3.1 Decision Selection

Step 1 begins when the Decision Component receives the system state from the Control Component. As discussed, ANDES utilizes Bayesian decision networks as its

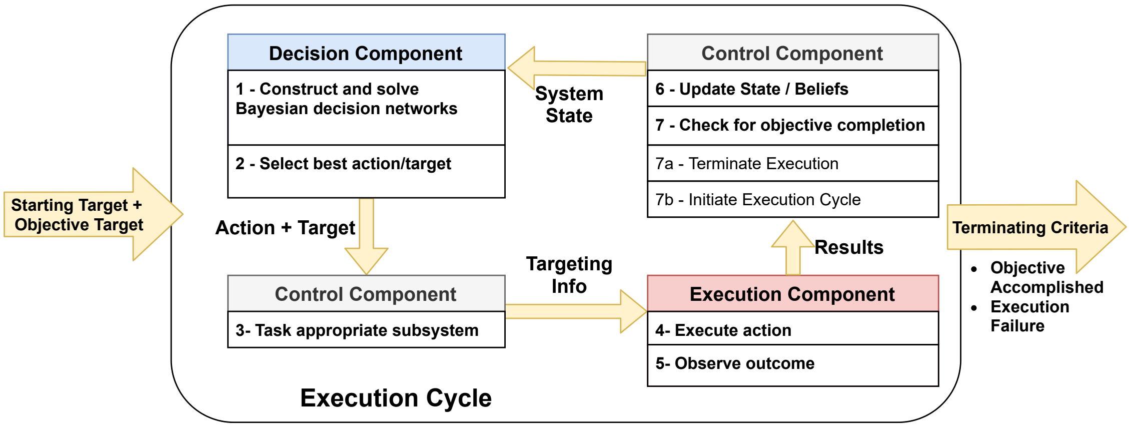


Figure 5: Execution Cycle: This figure shows the flow of execution during ANDES's execution cycle. Prior to initial execution ANDES performs initiation actions to include setting the user provided objective and initializing network starting targets. Execution is terminated when either no productive actions are available or ANDES has reached its objective state.

decision mechanism. The Control Component provides the Decision Component access to every ANDES host object currently in the target network as well as ANDES's internal state. In turn the Decision Component creates a Bayesian decision network for each host object. The Decision Component calculates the Maximum Expected Utility (MEU) as well as the associated decision(s) for each target system (Host object). Step 2 utilizes selection logic to determine which target and action combination is expected to yield the greatest utility. These results are returned to the Control Component. In Step 3 the Control Component will first determine what targeting information is required to carry out the selected action. Once this information is compiled it will be passed to the appropriate Execution Component subsystem.

3.2.3.2 Action Execution

Step 4 begins when the Execution Component receives this targeting information and begins preparing for the appropriate action. Currently the Execution Component consists of two subsystems, each responsible for various types of actions. Information gathering actions are split across both subsystems, the Scanning Subsystem conducts network scans and the Exploitation Subsystem conducts local information gathering actions on compromised targets (post initial compromise). In the case of an exploitation action, the Exploitation Subsystem is always utilized. Details on these subsystems are found in the Execution Component section. Once the action is prepared the Execution Component launches the action and prepares to observe the results. Step 5 consists of observing and processing the results in an attempt to determine the actual action outcome. Step 5 concludes when the results are recorded and passed back to the Control Component.

3.2.3.3 Results Analysis

In Step 6 the Control Component takes the results from Step 5 and updates any applicable Host objects or internal belief states. This step is also provides a chance to perform internal status checks and updates. One such example that ANDES currently employs is ‘pinging’ all target hosts it believes it has active connections with. If the connections respond appropriately no action is required, however if the hosts do not respond appropriately ANDES can assume the connection has somehow been interrupted and should update the target’s host object’s belief state accordingly.

In Step 7, using this updated system state the Control Component will check to determine whether ANDES has reached the user defined objective. If the objective state has not been reached, Step 7b begins the cycle anew passing the system state to the Decision Component. If the objective state has been reached, Step 7a will begin the process of terminating execution, alerting the user to objective completion and conducting the required shutdown actions.

3.2.4 Execution Cycle Advantages

It is this iterative process which enables ANDES to take advantage of information gained during execution, as well as to account for unexpected results. An added benefit of this method is the decision space (and corresponding solution space) is kept very small allowing for efficient solving. The size of any specific Bayesian decision network never grows, only the number of networks generated grows (as number of target hosts increases.) This means the decision making time grows linearly as the target network grows. As highlighted in the related-works section the downfall of similar Partially Observable Markov Decision Process (POMDP) approaches is that as network complexity grows, associated POMDP models can quickly grow intractable [4].

3.3 Decision Component

The Decision Component represents the thinking portion of ANDES. When combined with the supporting infrastructure provided by the Control Component, the methods employed within are the true contributions of ANDES to this field of research. ANDES utilizes Bayesian decision networks to model each individual decision point. This is in contrast to previous solutions which have attempted to represent the entire decision space as a single problem [4]. In previous systems the solution yields an entire attack chain, starting from target information and terminating at objective completion. ANDES instead selects a single action at each decision point, which it then executes, observes the outcome, updates the belief state and repeats the process. ANDES still starts with target information, and terminates at objective completion, but does so iteratively. This method allows for easy handling of non-deterministic actions as well as account for any domain uncertainty. Bayesian decision networks represent a way to account for uncertainty of knowledge and probabilistic outcomes, and also as a way to capture and utilize expert knowledge within the system. The core principle ANDES attempts to accomplish is automating the decision making process in a manner as closely as possible to how human operators approach their decisions process.

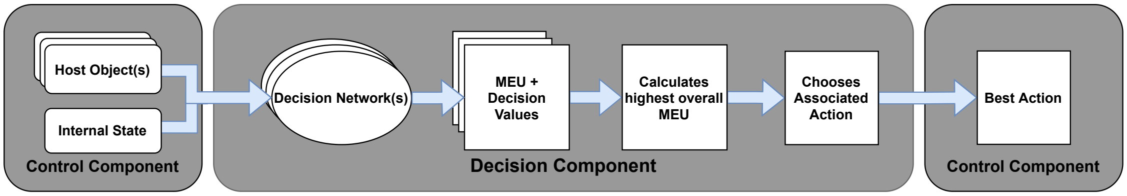


Figure 6: The Decision Component receives the Control Components current beliefs about the domain in the form of ANDES host objects. These host objects are used to construct and populate Bayesian decision networks, one for each host. The Decision Component will individually solve each network, compare maximum values across the networks, and ultimately return the next best action for the Control Component to take.

3.3.1 Bayesian Decision Network Overview

Employing a Bayesian decision network requires developing a model that accurately represents the agent's problem domain. Once the environment is represented, injecting critical decision points, developing a usable utility function, and attempting to capture relevant expert knowledge are key to producing an effective decision network. ANDES represents each available target host as its own Bayesian decision network, populating each network with observed evidence from ANDES's current belief state of that target.

The Bayesian decision network ANDES utilizes contains the three general node types as described in Chapter II, a refresher of each node type and how they are utilized within ANDES follows.

- Chance nodes: Capture the probability of a variable and are present in two distinct types. Observed chance nodes directly capture observable attributes of the host and are directly populated by ANDES during network initiation. The second type, contingent chance nodes, attempts to capture domain knowledge regarding expected outcomes, given observed evidence and potential decisions. The 'compatible exploit' chance node is an example of a contingent chance node as it combines the observed evidence for the operating system and vulnerabilities to calculate the probability of whether ANDES contains a compatible exploit given observed evidence. In this way it introduces outside knowledge regarding exploit and operating system compatibility, as well as potential chances of success given various operating systems and vulnerability combinations.

- Decision nodes: Represent potential decisions ANDES could make. ANDES's networks contain two decision nodes corresponding to whether ANDES should acquire information (scan) or attempt to acquire new access (exploit).

- Utility nodes: Evaluate the expected utility of a given network configuration. This utility node is what allows ANDES to make 'rational' selections for potential actions. By evaluating the expected utility for each combination of potential decisions, given the current evidence state, ANDES can select the decision(s) which corresponds to the highest value of the utility function. In this way the utility node serves as the heuristic for ANDES. This utility function can and should be modified by ANDES operators to account for preferences and desired behavior. For instance the utility node's values could be adjusted to value information more highly in an attempt to minimize taking risky uninformed actions. Figure 7 shows a visual representation the primary network ANDES produces, without any evidence population.

This next section contains the details of the Bayesian decision networks designed by the subject matter expert SME and which ANDES implements during execution.

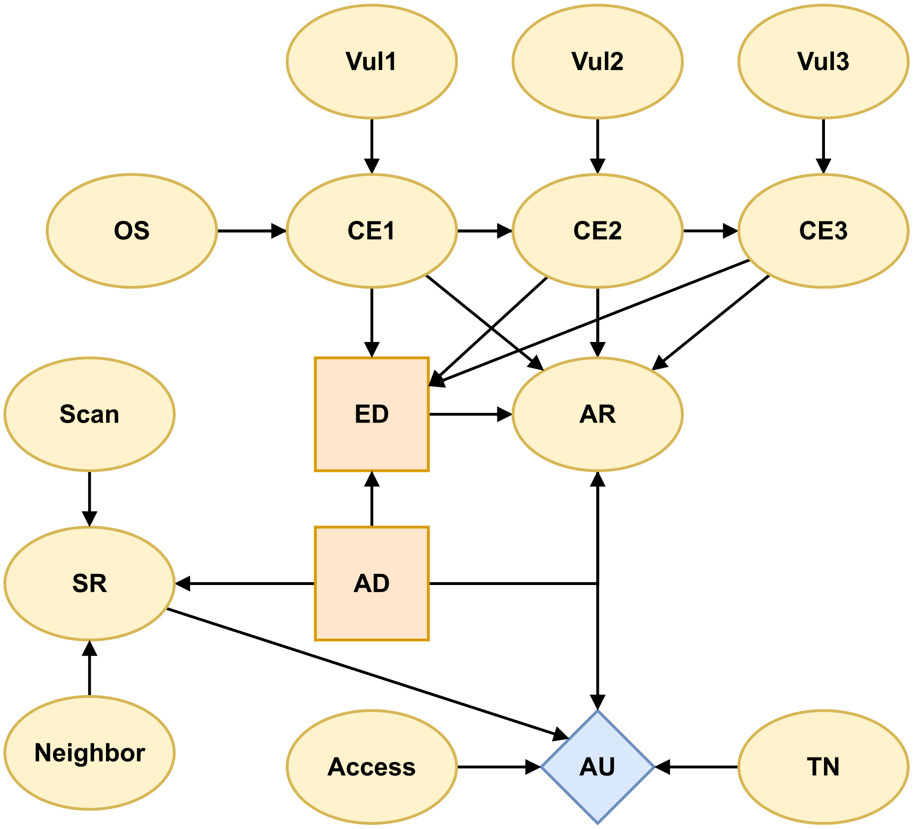


Figure 7: Depicted is the visual representation of the Bayesian decision network. Oval nodes represent chance nodes, square nodes represent decision nodes and the diamond node represents the utility node. Chance nodes with no incoming arcs are observed nodes which can be directly observed and populated by ANDES, all other nodes values are calculated during solving.

3.3.2 Bayesian Decision Network Construction

To aid in the prototyping of a Bayesian decision network representing a target host, the author used the Java application UnBBayes [28]. UnBBayes contains a Graphical User Interface (GUI) for constructing Bayesian decision networks. Using the GUI the author was able to develop the initial network configuration, produce initial probability tables, and tune utility values. Additionally, the GUI allows for real time network solving and a visual representation of evidence propagation. These capabilities were employed to conduct initial testing of network performance and ensure desired behavior prior to implementation within the ANDES framework. The visual representation also aided in ease of troubleshooting and seeing how network changes impacted utility outcomes. Once the author finalized the network design in UnBBayes, it was converted into Python3 code leveraging the pyAgrum [29] software library. pyAgrum itself is a Python3 wrapper for the aGrUM C++ library which implements various graphical models, including Bayesian decision networks.

UnBBayes saves its networks in a text-based format, enabling a direct conversion between the prototype network file into applicable Python3 code. This allowed the author to import the prototype network into ANDES as well as enabling other users of ANDES to more easily modify the Bayesian decision network as required using the UnBBayes GUI as opposed to directly editing the pyAgrum code. Currently ANDES is only capable of using this predefined network, instead of being able to dynamically generate new networks and associated probabilities. This limitation manifests itself as ANDES only being able to represent whether specific attributes have been observed, rather than dynamically generating nodes for every attribute of a scanned system. For instance, if the scan module detected ten running services on a target host, if ANDES only recognizes three of those services the information gain from those other seven services is currently wasted. Dynamic network generation was explored, but was abandoned due to the complexity of not only dynamically generating a proper network, but also the supporting code required. The network would need to be dynamically populated with initial probabilities, observed evidence would need to be dynamically populated, decision and utility functions would need to account for network changes, etc. The author determined this line of development unfeasible for Bayesian decision networks. This represents one of the disadvantages of using Bayesian decision networks, while easy to understand and solve they do not allow for as much flexibility as more complicated domain models.

3.3.3 Bayesian Decision Network Formal Description

Examining the Bayesian decision network ANDES utilizes, as depicted in Figure 7, provides insight into how ANDES captures its domain. Utilizing the formal notation as described in Chapter II, the influence diagram employed by ANDES can be represented in the following manner.

The network contains 13 chance variables represented as 13 nodes,

$$\mathbf{V}_C = \{OS, Vul1, Vul2, Vul3, Scan, Neighbor, Access, TN, SR, AR, CE1, CE2, CE3\}. \tag{15}$$

Of those 13 variables, 8 are observed variables representing some state ANDES could potentially observe during execution. These variables are represented by the set

$$\mathbf{E} = \{OS, Vul1, Vul2, Vul3, Scan, Neighbor, Access, TN\}. \tag{16}$$

An explanation of what those variables represent within ANDES follows:

- $v_{OS} = \{Win7, Win2k19, Linux, Unknown\}$: Observed Operating System
- $v_{Vul1/2/3} = \{T, F\}$: Does the corresponding vulnerability potentially exist?
- $v_{Scan} = \{None, Unfiltered, Filtered\}$: Has the target been scanned and were any results likely to have been filtered?
- $v_{Neighbor} = \{T, F\}$: Does the target have a neighbor within the network ANDES has access to?
- $v_{Access} = \{T, F\}$: Does ANDES have access to the host via a Metasploit session?
- $v_{TN} = \{T, F\}$: Is the target within ANDES's target's subnet?

The remaining five chance variables, which are not observed variables, represent contingent chance variables. Contingent chance variables provide a method to introduce domain knowledge into the network. By observing the incoming arcs it can be determined which other variables influence the calculated probabilities. An explanation of these variables follows:

- $v_{SR} = \{None, Unfiltered, Filtered\}$: If the target is scanned, what is the likelihood additional information will be gained?
- $v_{AR} = \{T, F\}$: Given the chosen actions and current state, what is the likelihood ANDES will have access to the host post execution?
- $v_{CE1/2/3} = \{T, F\}$: Given observed Operating System and potential vulnerability, what is the likelihood ANDES has a compatible exploit?

The final three variables represent ANDES's decision points and its utility function. The two decision variables:

$$\mathbf{V}_D = \{AD, ED\} \tag{17}$$

and the single utility variable:

$$\mathbf{V}_U = \{AU\}. \tag{18}$$

Observing the incoming arcs to decision nodes represents what information is available at the time the decision is made, while observing the incoming arcs to the utility node represent what state information is used to calculate the final utility of that network configuration. These arcs show the utility variable's *functional predecessors*,

$$FPRED(AU) = \{SR, AR, Access, TN\}. \tag{19}$$

In the case of the decision variables these arcs represent *informational predecessors*. In this network the decision variables are ordered such that $D_0 = v_A D$ and $D_1 = v_E D$.

From this the information the chance variables can be divided into three sets (number of decisions + 1) such that:

$$\mathbf{C}_0 = \emptyset \tag{20}$$

which represents the variables known before AD is determined. The next set:

$$\mathbf{C}_1 = \{CE1, CE2, CE3, AD\} \tag{21}$$

represents the information known before ED is determined. The final nodes are part of the set:

$$\mathbf{C}_2 = \{OS, Vul1, Vul2, Vul3, AR, TN, Scan, SR, Neighbor, Access\} \tag{22}$$

which represent the variables who have no direct connections to a decision variable which means their true values are never fully known at any decision point [27]. An explanation of what these three non-chance variables represent within ANDES follows:

- $v_{AD} = \{Exploit, Scan\}$: Should ANDES conduct an information gathering action or attempt to exploit the target?
- $v_{ED} = \{Exploit1, Exploit2, Exploit3\}$: If ANDES conducts an exploit action, which potential exploit should be chosen?
- $v_{AU} = \psi_{AU}(SR, AR, Access, TN)$: What is the expected utility of the network's resulting state? Considers the current combination of known states, chosen actions, and calculated probabilities.

Due to a software limitation in the pyAgrum *getMEU* function, ANDES sometimes employs two separate networks for each host. During the course of testing the author discovered an error in the pyAgrum library which leads to incorrectly reported decision preferences when two sequential decisions are present. This manifested itself when ANDES would correctly identify whether a scanning action or an exploit action would be more advantageous, but would always select the first exploit from the available list. Further investigation led the author to discover the network was being solved properly and no errors existed in the solving algorithm, however pyAgrum would incorrectly report the exploit decision which led to its calculated MEU. To accommodate this error, the SME designed a second network whose sole purpose is to determine which of the available exploits led to the reported MEU, once it had already been determined an exploit is the most valuable course of action. Using this network, in combination with the previously shown network, allows ANDES to answer both decision points properly. This secondary network is shown in Figure 8, but will not be discussed at length since it merely is a minimization of the already explored network.

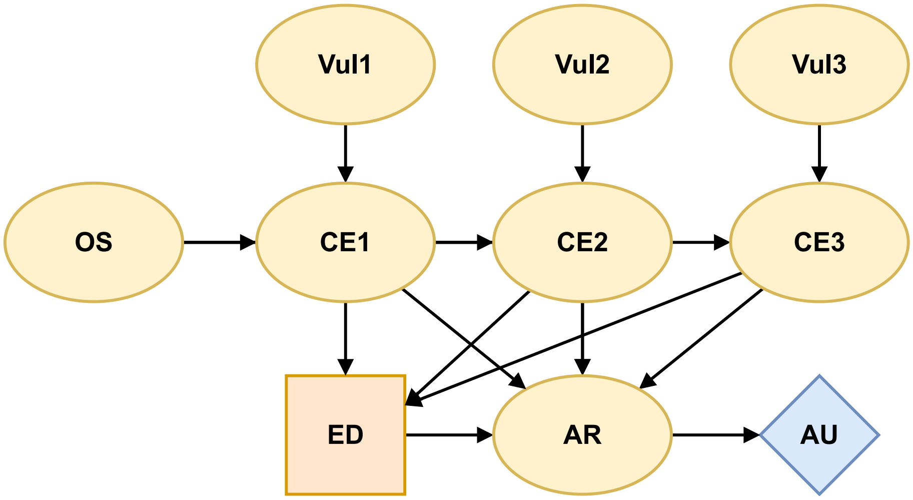


Figure 8: Visual representation of the minimized network used to determine which exploit action leads to the greatest MEU. Only employed once it has been determined an exploit action leads to the greatest MEU. Nodes represent same values as in the original network.

3.3.4 Solving the Bayesian Decision Network

Once the construction of the network has been completed as described above, the solving of each network is fairly trivial, as it is enabled by the pyAgrum library. Utilizing the ANDES host objects maintained by the Control Component, the Decision Component generates a new instance of a Bayesian decision network for each host object. To solve each network ANDES encodes the current belief state of the host object into the network. This is accomplished by populating associated evidence e into the corresponding network variable v_c. For example, if a previous scan action resulted in the finding the host was running Windows Server 2019

$$f = Win2k19, \tag{23}$$

the corresponding Operating System (OS) variable would be set to Windows Server 2019

$$\mathbf{v_{OS}} = Win2k19. \tag{24}$$

Populated evidence propagates throughout the network, producing a Bayesian decision network representing ANDES current belief state, fused with the outside knowledge encoded into the network structure itself. In this manner the Bayesian decision network is able to account for both uncertainty of state (missing evidence) and uncertainty of outcome (actions are not guaranteed to succeed.) The solution algorithm repeatedly solves the network for every possible combination of the decision variables until the combination resulting in the greatest MEU is determined.

Once all of the target hosts' Bayesian decision networks have been evaluated for their MEUs and associated actions, an additional layer of meta-analysis is conducted. This additional meta-analysis is a mechanism ANDES employs to account for internal state conditions outside of the Bayesian decision networks themselves. Utilizing the internal state of ANDES, as well as predetermined preferences the Decision Component selectively adds or subtracts from the raw MEUs returned by the Bayesian decision networks. ANDES keeps a record of previously performed action/target pairs, subtracting from a potential action's MEU if it represents a repeated action. The current discount is tuned in such a manner that if a valid exploit fails due to chance, it should still be selected again. The main goal of this discount is to prevent lines of execution in which ANDES repeats the same unproductive action ad nauseam. This process also includes an associated check that should all available actions fall below a set utility threshold, ANDES will terminate execution prior to objective accomplishment, returning a 'No Productive Actions Available' message as well as the execution chain that led to ANDES current state. Another mechanism used to modify the raw MEUs is more advanced targeting information and system user preferences. In the current implementation of ANDES the only positive MEU modification is one applied to the target's MEU when the target matches ANDES defined terminating objective. That is, only when the target is ANDES objective does it receive a bonus.

These modifications to the raw MEUs returned by the Decision Component represent a trade-off between Bayesian decision network complexity and post-processing complexity. The data stored within ANDES internal state could theoretically also be captured within the ANDES host objects and utilized as inputs into the Bayesian decision networks. In the case of the MEU bonus given to the target associated with ANDES's objective, it can easily be seen how an ANDES host object could capture whether it was associated with ANDES's objective. The state of whether the target was the objective could be represented as a chance variable with binary state options, easily represented within the Bayesian decision network. If that were the case the utility function associated with the utility node within the Bayesian decision network would capture the value of the state variable and return an MEU relatively higher than a similar system which is not directly associated with ANDES objective. In this case the raw MEU would not need to be adjusted by the Decision Component to account for the objective preference. Without comprehensive testing regarding computation times it is difficult to know whether increasing complexity within the Bayesian decision networks is better, or whether having a more complex decision selection process within the Decision Component is more efficient. In the current version of ANDES the author chose to reduce Bayesian decision network complexity and instead increase the complexity of the Decision Component.

Once all post-processing as been accomplished and final MEUs for every target host have been determined the Decision Component can return the best next action for ANDES to undertake. From here the Decision Component waits for the next round of execution to begin, at which point the process will begin anew. Every time the Decision Component acts it is an isolated event and it maintains no knowledge about previous decision making cycles. Only the Control Component keeps track of any type of past or current events. Armed with the decision from the Decision Component it is now up to the Control Component to determine which subsystem of the Execution Component will accomplish the task at hand.

3.4 Execution Component

The current version of ANDES contains two separate subsystems that combine to make up the Execution Component. Each subsystem performs disparate tasks but combine to enable all the actions ANDES has access to. At a high level the Scanning Subsystem accomplishes network scanning actions and the Exploitation Subsystem accomplishes both remote exploitation as well as local information gathering actions. Both subsystems are initialized during ANDES initialization process and both rely upon external software libraries to accomplish their tasks. The following subsections explore how each subsystem functions as well as their capabilities.

3.4.1 Scanning Subsystem

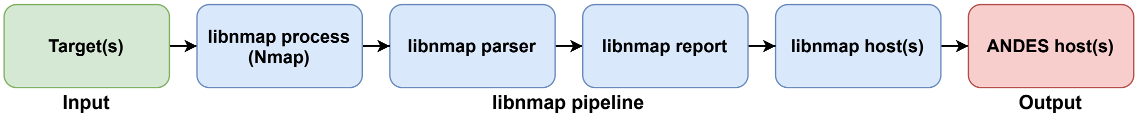


Figure 9: The pipeline utilized by the network scanner subsystem to automatically produce ANDES host objects from target information.

A critical component of all Artificial Intelligence (AI) agents is their ability to sense their environment [8]. Traditionally in computer networks this corresponds to enumerating the computer systems within that network. ANDES leverages the libnmap Python3 library, which implements a Python3 API for the open-source Network Mapper (Nmap) project [30]. Nmap is a trusted industry standard network scanning tool, employed throughout the CNE and VA-PT communities. Nmap was selected for the scanning capability of ANDES due to wide usage, ease of use, and demonstrated capabilities.

When the Decision Component decides the optimal action is to initiate a scan of a remote host, the Scanning Subsystem is tasked. The type of scan and the target's IP address are given to the subsystem as the sole inputs. Taking the target information, the subsystem employs the libnmap library to initiate an Nmap scan, storing and parsing the results as they are returned. The parsed results ultimately produce a libnmap.host object which contains all the desired information. Information currently utilized by ANDES is the host's operating system, open ports and associated services, and the system's hostname. Relevant information is captured in a custom ANDES Host object returned to the Control Component to enable easy access to the rest of the ANDES system. Figure 9 shows how this process works in practice. ANDES typically executes a scan action as its first action when encountering a new target, as this provides the system with baseline information which informs future decision making.

ANDES is also able to leverage the capabilities of Nmap and those contained within the exploit subsystem to enable scanning remote targets via a remote proxy. When tasked to conduct a scan of a remote target, ANDES will first determine whether the target has any neighbors. If any neighbors have been identified, those neighbors are examined to see if any of them have an active connection with an established proxy implant module. All of these attributes are stored as state variables within the ANDES host object. If a neighbor exists, which also has an active proxy connection available, Nmap will conduct the network scan through the proxy tunnel. The rest of the scanning process is handled as normal and the fact the traffic is proxied is transparent as far as Nmap is concerned once the scan is initiated. This enables ANDES to leverage access inside of a target network for improved information gathering and to bypass certain security measures.

3.4.2 Exploitation Subsystem

To enable exploitation actions ANDES interfaces with a remotely running Metasploit-4.19.0 RPC server [31] via the Pymetasploit3 Python3 library [32] (the server can be co-located with the ANDES host or on a networked system reachable by the ANDES host). During ANDES's initialization actions it checks to ensure the Metasploit RPC server is running, if it is not ANDES will start a local instance of the server. Once it has confirmed the remote RPC server is running, ANDES creates a persistent RPC client utilizing the Pymetasploit3 library, and establishes a connection to the Metasploit RPC server. ANDES uses this RPC client to conduct its exploit actions, utilizing the capabilities inherent to the Metasploit framework. Metasploit is a publicly available exploitation framework developed and maintained by Rapid7 that is an industry standard exploitation tool within the VA-PT community. Metasploit contains both an exploitation and post-exploitation framework enabling numerous remote exploitation attacks, payload delivery, and interactions with compromised systems. Metasploit was chosen as the exploitation capability for these reasons, as well as ease of use and integration with the Python3 codebase.

ANDES is able to conduct remote exploitation actions, interact with returned sessions, conduct limited host enumeration activities, and establish proxy pivots on compromised hosts. The remote exploits selected for inclusion were meant to be representative of the various types of remote exploit options. One of the available exploits targets a vulnerable third-party service (Dup Scout Enterprise), one targets a vulnerable operating system service (SMB) and another targets a weak security policy (remote root SSH logon) enabling remote brute-force logon attempts to an SSH server.

When the Decision Component has determined exploitation to be the best action, the Control Component compiles required targeting information and passes control to the Exploitation Subsystem. Using the targeting information from the Control Component, the subsystem begins populating the exploit module. Once the exploit module is configured with the proper targeting information, in conjunction with the desired exploit payload, the Exploit Subsystem utilizes the RPC client to request the listening Metasploit server launch the exploit. The RPC server responds with a status message, letting the client know if the exploit was successfully launched. This return message only serves to inform the client whether the exploit was initiated properly, not whether the exploit itself was successful. Determining whether the exploit was successful requires a slightly convoluted process. To determine whether an exploit was successful, the subsystem must capture the state of active Metasploit sessions prior to the exploit, then, once the exploit has completed, determine whether a new active session is present. The subsystem assumes that if a new session is identified, the new session is the result of the exploit. This highlights one of the main issues with the Pymetasploit3 library and the Metasploit RPC server.

The Metasploit RPC server is mostly silent when it comes to sending the RPC client information regarding server status and action results. The RPC server remains silent when an exploit is unsuccessful, a session is lost, or when numerous other events occur. The main contributing factor to this non-optimal behavior is a lack of official support for this usage case by Metasploit developers. A more feature rich RPC server & client is available to Metasploit Pro users, a paid version of the software. Additionally the Pymetasploit3 library is a community developed open-source implementation of a Metasploit RPC client and does not fully implement all of the functionality available even in the free version of the API. For ANDES purposes these limitations can be worked around, but show how a more closely integrated system would hold advantages when developing a production systems.

In the event a new session has successfully been obtained, ANDES automatically conducts a host survey to gather additional information about the compromised target. Currently the survey focuses on querying the host's ARP table in an effort to locate potential new targets within the network. As a filtering process any ARP table entries corresponding to IP addresses outside the target objective's local network are ignored.

When a new session is obtained on a remote host, the Metasploit RPC client creates a session object which contains a handle to the active session which resides within the RPC server. A reference to this session object is stored in the target's ANDES host object, tying the target and session together. This enables the Exploit Subsystem to interact with active sessions in the future by accessing the targets via their associated ANDES host object. The only other post-exploit action currently available within the Exploit Subsystem is the ability to establish a proxy pivot on the compromised host. This enables future scanning actions from the Scanning Subsystem, as well as future remote exploits, to be routed through the target's session in an attempt to avoid external security measures.

The capabilities described above represent only a portion of the capabilities Metasploit provides. Given the research purposes of ANDES, the author determined this feature set was robust enough to demonstrate feasibility of the concepts presented without developing a full feature-rich system that would be required for a production system. A critical component that has been partially implemented is a robust exploit execution feedback module. This module is responsible for updating the systems belief state to reflect the real-world result of Metasploit frameworks execution. This is what allows ANDES to deal with the many possible outcomes associated with conducting a live execution event (exploits can fail, implants can disconnect, services be crashed,etc.). While ANDES is not an attempt to develop a production system capable of automated CNE, it should demonstrate how such a system could navigate more difficult network environments corresponding to a more complex decision landscape. The capabilities present in ANDES demonstrate the potential success of production systems employing the same underlying principles.

3.5 Summary

This chapter explored how the three components of ANDES function, as well as how they come together to accomplish the required portions of CNE events. The differences between the system presented here, compared to those presented in Chapter II should be apparent. The next chapter contains the results of testing the systems performance in a live-execution event, and how the novel techniques presented here combine to produce a working system.

IV. Results and Analysis

The testing of Automated Network Discovery and Exploitation System (ANDES) was performed with the sole objective of proving the viability behind the principles and concepts employed. This qualitative approach of evaluation is in contrast to performance metrics presented in previous works. This is in line with the avenues for potential research directions within [4] who advocates that systems should both move towards live-execution testing and a more qualitative approach of assessing the systems performance regarding the benefits of the system.

Evaluating a system based upon its capability to quickly process large target networks is only valuable as long as systems exist that can not meet the threshold of being usable in the real world due to computational constraints. Once it can be assumed that presented systems are capable of handling real world environments, it is then much more valuable to assess how useful is the automation system at accomplishing its objectives in the Vulnerability-Assessment and Penetration Testing (VA-PT) / Computer Network Exploitation (CNE) domains. To this end ANDES is evaluated not on whether it meets any quantitative performance metrics such as execution time or computational complexity, but instead is evaluated on whether it could feasibly provide value to an operator wishing to automate CNE tasks to help alleviate resource shortages.

Results provided here are not meant to prove the commercial viability of ANDES as the system currently stands or even that ANDES itself represents an optimal solution. To accomplish this goal two stages of testing were performed, the first stage of decision network testing was conducted to show the correctness of the underlying Bayesian decision network. The second stage of testing, System Performance Testing demonstrates the ability of ANDES to accomplish Computer Network Exploitation (CNE) goals. As a final note regarding all testing performed, because of stated objectives, the *correctness* of performance results is entirely subjective to the author's intents for the system. ANDES is designed to capture a Subject Matter Expert's SME knowledge and utilize it in unknown environments to reproduce their decision making process and skills. Given these factors the metric ANDES is held to is whether, when presented with the same domain knowledge, ANDES makes the same choices as the SME whose knowledge was encoded into the system. By meeting this criteria, the system demonstrates its potential capability to augment the SME operator during normal VA-PT / CNE events, providing a force multiplying effect.

4.1 Decision Network Testing

The first test evaluates the Bayesian decision network's ability to capture the CNE domain, as well as incorporate SME knowledge and preferences. UnBBayes was used to compile the developed network and introduce various iterations of evidence into the network. Network performance validation consists of ensuring the network makes the same decisions as a SME when given the same evidence. Once this validation of the Bayesian decision network was confirmed, testing could be moved within the ANDES framework to ensure the network was reproduced properly. This stage of testing corresponds to the research objective of developing and evaluating a Bayesian decision network which represents the CNE domain and incorporates SME knowledge and preferences.

4.1.1 Decision Network Testing Methodology

Once the Bayesian decision network design was finalized in UnBBayes, testing was performed to ensure the network's output matched expected decisions when provided selected inputs.The process of testing each network begins by populating observed chance variables with select findings, propagating observed evidence throughout the rest of the network, and solving for the network's maximum expected utility. Figure 10 shows the Bayesian decision network with no populated findings. To interpret these images, the chance variables with populated evidence are shown in grey, while unobserved variables are shown in yellow. The utility values, shown within the bold black rectangles, represent the expected utility associated with the corresponding decision choice. The highest expected utility value for each network represents the action with the greatest expected utility for the populated evidence. These results were used to confirm the network was selecting the expected decision actions for the input evidence. This process was repeated for numerous combinations of findings representing potential configurations of hosts ANDES might encounter during execution. If the network returned an unexpected result the network's utility function and contingent chance variables conditional probability tables could be adjusted as necessary by the subject matter expert. Additional screen captures showing the testing of various combinations of Bayesian decision network configurations are contained in Appendix A.

Once the network was confirmed to be behaving as expected, testing was moved into the ANDES framework. Within ANDES this same process was repeated utilizing the pyAgrum library and the *getMEU* function. Evidence was manually generated and populated into a test network, the network was then compiled and solved using the *getMEU* function which returns the network's Maximum Expected Utility (MEU) and associated decisions, given a provided evidence state. Generated values were compared to those received from the UnBBayes software to confirm the designed network was properly reproduced. The expected utility values associated within decision choices are irrelevant in the absolute sense, these values have no set scale or outside values to compare them to. The values of the expected utilities will vary greatly depending on the utility function produced by the network's author.

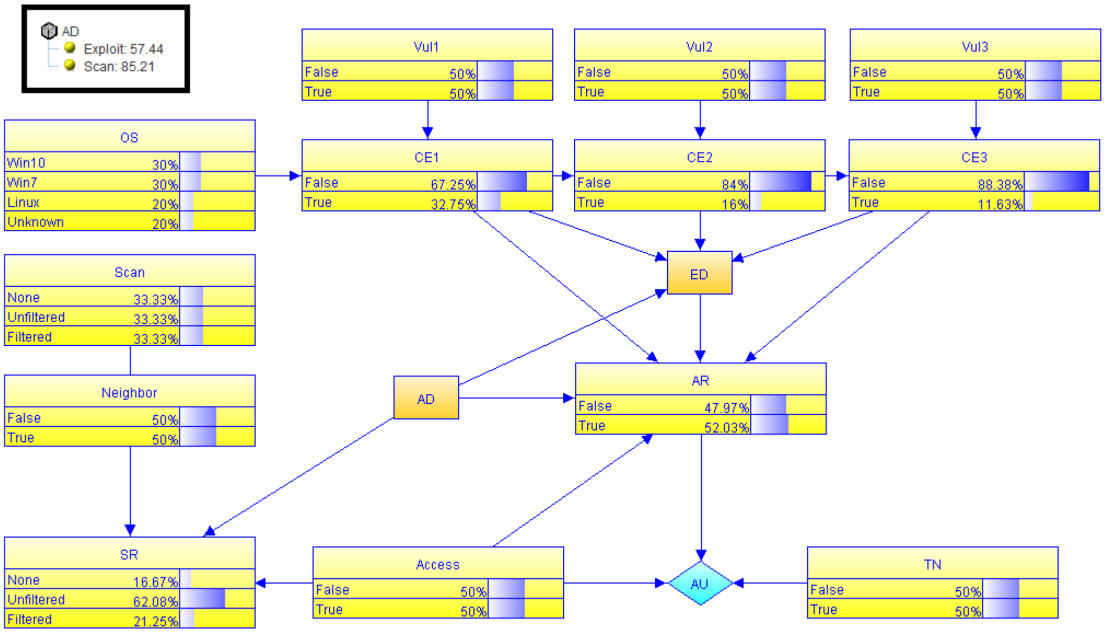


Figure 10: This network shows the Bayesian decision network with no populated findings. This test confirmed the network was structurally sound and could be compiled. All required components of network construction were completed and variable Conditional Probability Table (CBT)s were filled in.

The usefulness of the expected utility function is that it allows the network to determine what decisions lead to the maximum value among all decision choices, given the current evidence.

4.1.2 Decision Network Testing Results

The final results of the decision network testing confirmed the networks functioned as expected. It should be noted decision network testing was performed throughout the Bayesian decision network development process, the results of which were used to adjust chance variable probability tables and utility function values. Certain unexpected results were observed during this initial development stage, but values were adjusted accordingly until the network output actions that aligned with the SME selections.

One critical thing was discovered during the pyAgrum portion of functionality testing. This testing discovered the reporting error present in the *getMEU* function of pyAgrum referenced in Chapter III. This error causes the incorrect decision choice to be reported as the optimal choice associated with the network's maximum expected utility (MEU). This error led to the development of a secondary Bayesian decision network. This second network is utilized for determining which exploitation action was associated with the network's MEU. This current method, while non-optimal, allowed the pyAgrum model to achieve the correct results as had been seen in the UnBBayes model.

4.2 System Performance Testing

Once validation of the Bayesian decision network's capability to produce proper action choices was completed, testing was performed on the entirety of the ANDES system. This stage of testing corresponds fulfills the research objectives to test the entire developed system within a live-execution environment. Testing consisted of repeatedly running ANDES through increasingly complex CNE scenarios and observing the outcomes. Tests were considered successful if ANDES completed the scenario successfully multiple times(reached objective state), while simultaneously making decisions deemed rational by a SME. Measured criteria are qualitative in nature and leave future quantitative measures to be performed on a more production representative system. All performance testing was completed in a virtual environment designed to emulate a real-world target network. Details of the testing environment are shown in the following subsection.

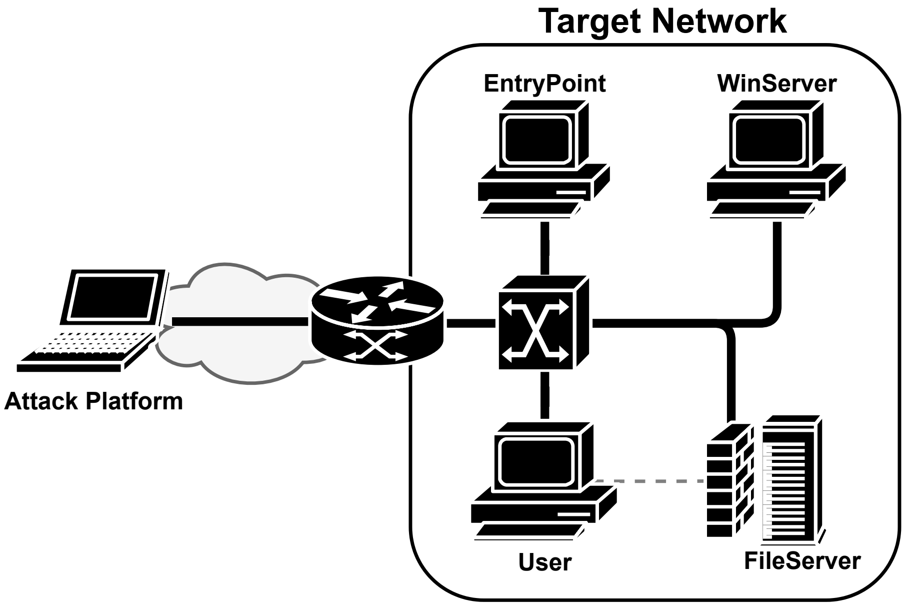


Figure 11: A visual representation of the target network developed for ANDES testing. The dotted line represents the Firewall rule which only allows access to the **FileServer** system from the **User** system.

4.2.1 System Performance Testing Environment

Figure 11 shows the virtual testing environment used to safely conduct live-execution tests of ANDES. The network is a simplified network meant to mimic an environment an attacker might encounter in a small business network. The network consists of four possible targets, each with a unique configuration. The network contains:

- Two user machines running Windows 7 corresponding to employee workstations.
- Machine running Windows Server 2019 and acting as the Windows Domain Controller and internal DNS Server.
- Ubuntu host serving as a file server, utilized by the business to store company data.

Every system has been configured to be vulnerable to at least one of the exploits available to ANDES with the exception of the **WinServer**. The **WinServer** system is running a vulnerable service, however the advanced protections included in the Windows Server 2019 Operating System prevent the service from being successfully exploited. This system serves to show how ANDES reacts to non-vulnerable hosts.

Table 1 provides an overview of each system's configuration and potentially exploitable software. One important thing to note is that the **FileServer** system is running a host-based firewall which drops all traffic that does not originate from the **User** system. This means the **User** system is the only host capable of reaching the **FileServer**. Additionally only the **User** system knows about the existence of the **FileServer** (disregarding the internal virtual networking switch). The entire virtual environment was hosted on an instance of VMware Workstation 16 Pro.

Table 1: Target Network Host Configurations

Name	**Operating System**	**OS Version**	**Relevant Software**
EntryPoint	Windows 7	SP 1 - Build 7601	Dup Scout Enterprise v10.0.18
Attack Platform	Kali Linux	Kali-Rolling 2020.3	Pymetasploit3 3-1.0.3 libnmap 0.7.0 pyAgrum 0.18.2
WinServer	Win Server 2019	Build 17763	Dup Scout Enterprise v10.0.18
User	Windows 7	SP 1 - Build 7601	
FileServer	Ubuntu	20.10	openssh server 1.8.3p1-1

4.2.2 System Performance Testing Methodology

A series of tests were performed on ANDES in an attempt to validate the developed capabilities, each test was repeated multiple times to confirm system consistency. Each test focused on validating capabilities critical towards enabling automated Computer Network Exploitation (CNE). The culminating test combined all previous objectives to create a scenario designed to mimic a simplified real-world problem a CNE operator might face. ANDES was tasked to traverse the target network in search of the **FileServer** system, enabling future data exfiltration from the target. The results from one execution of that final scenario are presented here, as this test covered all five test objectives. Individual testing was performed to validate objectives one through four prior to the culminating event, however those results are subsumed by the results presented here. Results of individual tests can be found in Appendix B.

- Objective 1 - Exploit Functionality Testing
  - Objective 1a: Validate functionality of Dup Scout Enterprise - 'Login' Buffer Overflow.
  - Objective 1b: Validate functionality of MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption.
  - Objective 1c: Validate functionality of SSH Brute-Force Login Scanner.
- Objective 2 - Host / Information Discovery: Validate ANDES is able to acquire new targets during execution from information gained during execution within the network.
- Objective 3 - Dealing with Failure: Validate ANDES is able to account for unsuccessful action outcomes and adapt accordingly.
- Objective 4 - Bypassing Security Measures (pivoting): Validate ANDES is able to bypass restricted network connectivity, in this case a host-based firewall.
- Objective 5 - Full Attack Chain: Validate ANDES is able to combine all of the previous objectives to traverse through the target network and compromise the objective system.

4.2.3 System Performance Testing Results

Table 2 shows the entire series of actions ANDES performed during an instance of the culminating test. The final column includes references to when each objective, as defined previously, was achieved during execution. During initiation of the test event ANDES was provided with only two pieces of information, the IP Address of the **EntryPoint** host and the IP Address of the **FileServer** as the objective target.

Table 2: System Performance Test Results

#	Action	Target	Results	Comments & Objective Demonstrated
1	Scan	EntryPoint	Success	Initial Target
2	Exploit1	EntryPoint	Success	Dup Scout BOF - Obj 1a
2.1	HostSurvey	EntryPoint	Success	Added WinServer, User to targets- Obj 2
3	Scan	WinServer	Success	
4	Exploit1	WinServer	Failure	Dup Scout BOF - Obj 3
5	Scan	User	Success	
6	Exploit2	User	Success	EternalBlue - Obj 1b
6.1	HostSurvey	User	Success	Added FileServer to targets- Obj 2 Found Target Objective
7	Pivot	User	Success	Added pivot to internal network through 'User'
8	Scan	FileServer	Success	Scan conducted through pivot - Obj 4
9	Exploit3	FileServer	Failure	Attack conducted through pivot - Obj 4, Obj 3
10	Exploit3	FileServer	Failure	Attack conducted through pivot - Obj 4, Obj 3
11	Exploit3	FileServer	Success	SSH Brute Force - Obj 1c. Target Objective Reached
12	Shutdown	N/A	Success	Performed clean-up actions

As expected ANDES chose to begin execution by conducting a remote network scan of the initial target (**1**). From here ANDES detected the **EntryPoint** host was running the vulnerable *Dup Scout Enterprise* service, as well as being potentially vulnerable to the *EternalBlue* exploit. The SME knowledge captured within ANDES's Bayesian decision network led ANDES to chose to attempt to remotely exploit the *Dup Scout Enterprise* service as it had a higher chance of success (**2**). The action was successful and ANDES gained access to the **EntryPoint** host. When ANDES gains access to a host it conducts a host survey in an attempt to discover additional targets, ultimately hoping to find a route to the objective host (**2.1**). This host survey resulted in ANDES discovering the **EntryPoint** host had connectivity to the **WinServer** host and the **User** host, both of which were then added into ANDES target network. The next action selected was to perform a remote network scan of the **WinServer** host (**3**). Given the fact ANDES did not have any additional information besides the hosts' IP Addresses at the time, and neither host was an objective, selecting to scan either of the two new targets represents a reasonable decision. From the scan ANDES determined the **WinServer** host was also running the vulnerable *Dup Scout Enterprise* service. Armed with this information, and still having no additional information about the **User** host, ANDES decision to attempt to exploit the **WinServer** is rational (**4**). As expected given the **WinServer** host's operating system protections the exploit was unsuccessful, which resulted in ANDES detecting action failure. Following this unsuccessful exploit attempt, and lacking any additional vulnerabilities on the **WinServer** host, ANDES decided to scan the **User** host (**5**). Without knowledge of any differentiating value between the two targets ANDES does not have access to, the decision to scan the alternative after unsuccessfully exploiting the **WinServer** host is rational. This scan revealed to ANDES the **User** host may be susceptible to the *EternalBlue* exploit. Armed with that information ANDES then decided to attempt to exploit the **User** host (**6**). The *EternalBlue* exploit was successful and ANDES gained access to the **User** host, after which it conducted the standard host survey (**6.1**). This host survey reveals that the **User** host has connectivity to the **WinServer** host, ANDES's objective host. Additionally, the access afforded by the *EternalBlue* exploit is sufficient to allow ANDES to establish a Pivot on the host. Armed with the information from the host survey, ANDES proceeds to create a pivot which will forward traffic to the **User** host's neighbors (**7**), in this case the **FileServer** host. Having located the objective host ANDES selects to perform a remote network scan of the **FileServer** target (**8**), which will be sent through the pivot established on the **User** host. This scan revealed the fact that the SSH service is reachable on the **FileServer** from the **User** host. Armed with this information ANDES then proceeded to choose to launch an *SSH Brute Force* attack against the **FileServer**, which will once again be forwarded through the **User** host pivot. The first two attempts at this exploit fail (**9+10**). ANDES continues to attempt this exploit until it is eventually successful on the third attempt (**11**). ANDES has now reached the desired objective host and without further objectives will proceed to gracefully shutdown, including terminating all active connections with compromised hosts (**12**).

An item to note is ANDES's willingness to repeat attempts at exploiting the **FileServer** host following previous failed attempts. This is in contrast to the previously failed exploit against the **WinServer** host, after which ANDES proceeded to pursue a different action. This version of ANDES highly favors taking actions against the objective host, which led to ANDES attempting the exploit multiple times before it would eventually discount the action enough to pursue alternative options. It should be noted that a brute force logon attempt is not normally an exploit that is commonly repeated, but in this virtual environment, coupled with the behavior of the *Metasploit SSH Login Scanner* through a proxy, this attack routinely will eventually work after previously failed attempts.

4.3 Analysis

From a test perspective the results are extremely positive. ANDES was able to demonstrate all five test criteria being accomplished, which in turn shows ANDES's functional capabilities. Examining more closely the decisions ANDES made reveals no obviously incorrect or irrational decisions. When accomplishing the same test during target range validation, the human SME took nearly an identical route through the network only deviating when ANDES chose to scan the **WinServer** host prior to the **User** host. In this case the human SME knew the User host was the proper path to the **FileServer**, knowledge unknown to ANDES during execution. With the functional capabilities of ANDES confirmed it is important to consider whether this test is sufficient to accomplish ANDES designated purpose.

Remembering the the hypothesis of this thesis is to determine the feasibility of utilizing Bayesian decision networks along with iterative execution cycles to enable automated Computer Network Exploitation (CNE), results must need to be viewed through this lens. ANDES was able to successfully operate in an unknown network, chaining together numerous information gathering and various exploitation actions, ultimately achieving access to an objective host. The environment itself was obviously simplified with only four target hosts, and three different remote exploit actions to be chosen, but it still demonstrated the goals ANDES was developed to fulfill. First, it demonstrated ANDES was able to successfully represent the CNE domain, and leverage captured SME knowledge to make reasonable decisions. Second, it demonstrated ANDES capability to operate within a live-execution environment, to include accounting for domain uncertainty and non-deterministic actions. Finally, it demonstrated how multiple rounds of iterative execution, guided by proper a utility function, can accomplish a multi-step objective. These facts demonstrate how the usage of the principles introduced by ANDES combine to create a system with the capacity to successfully augment human CNE operators.

V. Conclusions

The Automated Network Discovery and Exploitation System (ANDES) represents a shift in the fundamental design philosophy towards automated Vulnerability-Assessment and Penetration Testing (VA-PT) and Computer Network Exploitation (CNE) systems. Examining the previous body of work reveals only one true attempt at creating a system capable of conducting live execution events in [12].

The introduction of representing each host within a target network as Bayesian decision networks addresses previous issues with computational complexity as the target network size grows. This choice, along with chaining together isolated decision points, allows ANDES to mimic the decision making process employed by human Subject Matter Experts Subject Matter Expert (SME) when conducting VA-PT / CNE events. The five steps of the VA-PT process [9] ANDES automates are:

- Intelligence Gathering
- Threat Modeling
- Vulnerability Analysis
- Exploitation
- Post Exploitation

Based on the results demonstrated in Chapter IV it is the author's belief ANDES has successfully proved the hypothesis that using Bayesian decision networks to capture the CNE domain and Subject Matter Expert (SME) knowledge, combined with an iterative execution cycle, allows for functional automated CNE systems capable of augmenting human operators. The author also believes the proposed method of performing multiple rounds of execution, each of which builds upon the knowledge gained from previous actions and observations of the agents domain, is critical in producing systems capable of autonomously operating within the CNE domain.

The current body of research exploring automated CNE / VA-PT is sparse compared to most other areas of computer security research. It is the author's hope this work encourages additional research in the field, ultimately helping to bring about production systems capable of augmenting the highly skilled workforce that is in such demand today. The final section presents several such areas of potential future research.

5.1 Future Work

In ANDES's current state, it contains the minimum functionality required to conduct automated CNE events. It serves the purpose of demonstrating the concepts introduced in this research but should not be taken as the foundation for any production system. Presented here are several additional avenues of research that build upon the ideas presented here and which the author believes would be valuable additions to the body of research.

5.1.1 Unified Automated CNE System

While ANDES as a system is quite limited in its potential applications, the development of a more robust system utilizing the same underlying principles seems reasonable and valuable. The most important area of additional development would be a concerted effort to produce a robust and complex Bayesian decision network which captures vast amounts of SME and domain knowledge. Development of such a network however is limited by information availability. The Bayesian decision network can only consider domain elements that are at least partially observable to the CNE system and chosen actions must be reasonably executed by the system. ANDES reliance upon external libraries and tools greatly aided in the ease of development, but also severely limited options available to ANDES. The development of a unified system with robust information gathering capabilities allows for a more complicated Bayesian decision network with a greater number of observed chance variables. This translates into allowing SMEs a greater capability to introduce more knowledge into the system, which ultimately results in a much more refined and nuanced decision making process.

5.1.2 Probabilistic Evidence

Another valuable extension to the existing Bayesian decision network models would be the utilization of more of the probabilistic capabilities of observed chance variables. Currently when ANDES populates decision networks with observed evidence it does so in a binary fashion. Either ANDES observed a chance variable state or it did not, it currently will not utilize probabilistic evidence (i.e., 60% chance of being a Windows 7 host.) It should be noted however when evidence is not populated for a given chance node it maintains the originally populated probability, derived from it's Conditional Probability Table (CBT).

5.1.3 Machine Learning

Another potential area of future research would be to combine the field of learning models with ANDES's method of Bayesian decision networks. Currently, chance variable probabilities are determined during network construction by the SMEs and remain consistent throughout execution. In this case if environmental factors change, which impact previously observed probabilities, the network would need to be manually adjusted by the SME. One such example is a software developer releasing a patch for a vulnerable version of their software. Previously the associated exploit could have had an 80% chance of success, but after the patch the probability of success would dwindle as the patch is distributed across the software user-base. In this case the Bayesian decision network developer would have to manually adjust the associated chance variables' probabilities. Instead of this manual process, a system which observed and stored action outcomes corresponding to domain beliefs could automatically update chance variable probabilities. An additional benefit of this type of system would be to eliminate much of the burden of developing an initial Bayesian decision network from the SME. Instead of the SME populating all chance variables' associated CBTs, they could be generated through an initial learning process conducted in a training environment. The system could be exposed to a training environment and run through scenarios again and again while observing action outcomes, updating applicable CBTs.

5.1.4 Post-Exploitation Capabilities

Currently ANDES only is capable of very limited post-exploitation actions, greatly limiting ANDES capacity to achieve most real-world desirable objectives. The additional of a comprehensive post-exploitation component would allow ANDES to accomplish objectives associated with normal CNE campaigns, as well as gather additional information to help inform future decision making processes. This area of exploration is less-so an area for future research and much more a development that must take place before any automated system would be useful for real-world applications.

Appendix A. Bayesian Decision Network Test Results

This appendix includes screen captures of the Bayesian decision network testing that was performed via UnBBayes. Each figure represents a network presented with different findings and shows the maximum expected utility for each decision choice. The significance of each example is included in the figures caption.

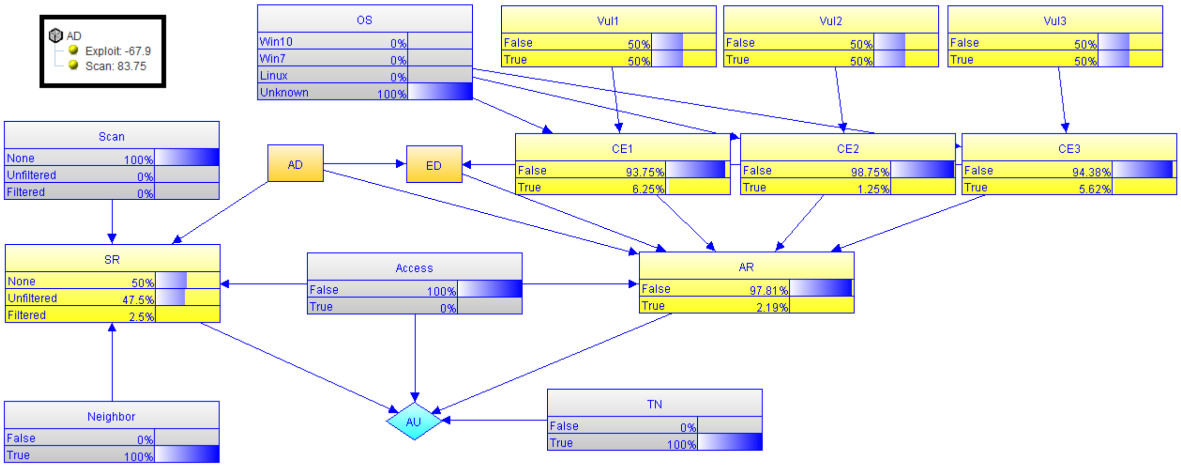


Figure 12: This network shows the results of testing a network configuration which represents an ANDES host which has just been discovered by conduction a host survey of a compromised target. The evidence $e = \{OS = Unknown, Scan = None, Neighbor = True, Access = False, TN = True\}$ is populated and propagated throughout the network. Given ANDES lack of knowledge about the system, you can see the expected utility (EU) for an exploit action against the host is -67.9, while a scanning action has an EU of 83.75. If this host is selected as ANDES next target it would conduct a scanning action which aligns with subject matter expert (SME) behavior.

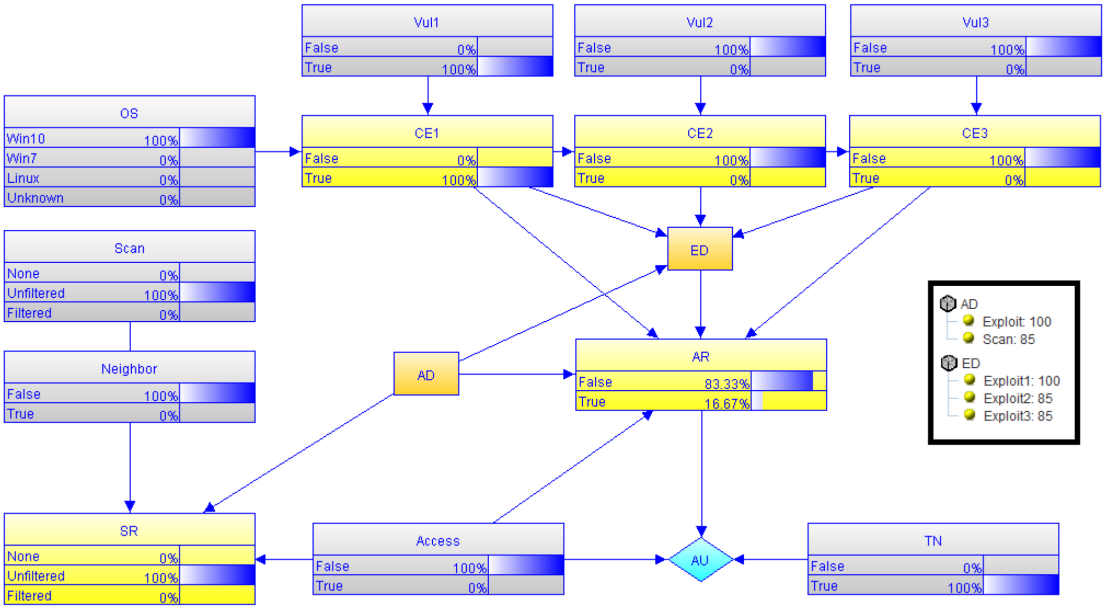


Figure 13: This network shows the results of testing the network configuration corresponding to ANDES scanning a target host and discovering a vulnerability which the system has the potential capability to exploit. The evidence $e = \{OS = Win10, Scan = Unfiltered, Neighbor = False, Access = False, TN = True, Vul1 = True, Vul2 = False, Vul3 = False\}$ is populated and propagated throughout the network. Given the level of information ANDES has captured about the host, and the host resides in the same target network as the objective host, a SME would choose to conduct the Exploit 1 action against. The decision network came to this same conclusion, the MEU for the AD decision node (100) corresponds to Exploit and the ED decision node MEU (100) corresponds with Exploit1.

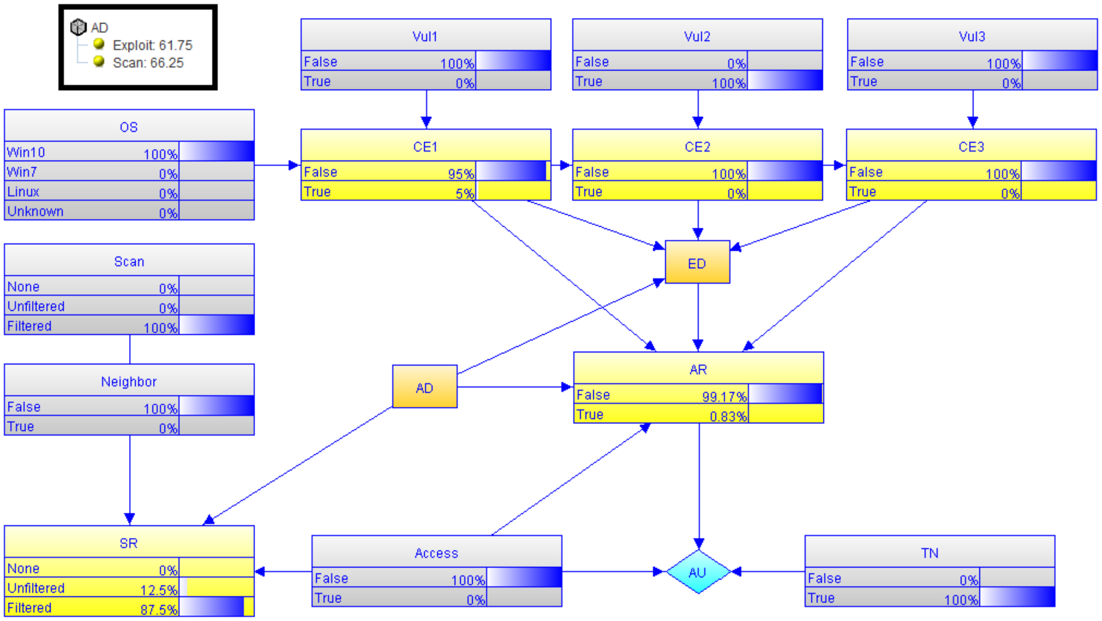


Figure 14: This network shows the results of testing the network configuration corresponding to ANDES scanning a target host and discovering a potential vulnerability, but due to captured SME knowledge it recognizes the potential exploit is incompatible with the observed Operating System (OS). ANDES also has reason to believe the scan results were filtered by a security device. The evidence $e = \{OS = Win10, Scan = Filtered, Neighbor = False, Access = False, TN = True, Vul1 = False, Vul2 = True, Vul3 = False\}$ is populated and propagated throughout the network. The SME evaluation says given the available information, taking an exploit action against the system seems unwise. A scanning action has the potential to discover additional information that ANDES could use for additional targeting, however pursing a different target might be more fruitful. The decision network came to this same conclusion, as can be seen by the relatively smaller EU values of the exploit (61.75) and scan (66.25) actions compared to previous network tests.

Appendix B. Detailed System Performance Test Results

Prior to the culminating testing event presented in Chapter IV, smaller individual tests were performed on the ANDES system within the virtual range environment. The results of these tests served to validate the functionality of the ANDES system as well as help tune the Bayesian decision network behavior. These tests helped identify any Component failures prior to the culminating test. The results of the final individual tests are presented here in the same table format used in Chapter IV.

Table 3: Exploit 1 Validation Test Results

#	**Action**	**Target**	**Outcome**
1	Scan	User	Nmap Scan Report
2	Exploit1	User	Meterpreter Session
3	Shutdown	N/A	ANDES Shutdown

Table 4: Exploit 2 Validation Test Results

#	**Action**	**Target**	**Outcome**
1	Scan	FileServer	Nmap Scan Report
2	Exploit3	FileServer	SSH Session
3	Shutdown	N/A	ANDES Shutdown

Table 5: Exploit 3 Validation Test Results

#	**Action**	**Target**	**Outcome**
1	Scan	EntryPoint	Nmap Scan Report
2	Exploit1	EntryPoint	TCP Shell Session
3	Shutdown	N/A	ANDES Shutdown

Table 6: Host Discovery (Information Gain) Test Results

#	**Action**	**Target**	**Results**	**Comments**
1	Scan	User	Success	
2	Exploit2	User	Success	EternalBlue
2.1	HostSurvey	User	Success	Added FileServer to targets
3	Pivot	User	Success	Added pivot to internal network through ‘User’
4	Scan	FileServer	Success	Scan conducted through pivot

Table 7: Dealing With Failure Test Results

#	Action	Target	Results	Comments
1	Scan	EntryPoint	Success	Initial Target
2	Exploit1	EntryPoint	Success	Dup Scout BOF
2.1	HostSurvey	EntryPoint	Success	Added WinServer, User to targets
3	Scan	WinServer	Success	
4	Exploit1	WinServer	Failure	Dup Scout BOF
5	Scan	User	Success	

Table 8: Pivoting Test Results

#	Action	Target	Results	Comments
1	Scan	User	Success	
2	Exploit2	User	Success	EternalBlue
2.1	HostSurvey	User	Success	Added FileServer to targets
3	Pivot	User	Success	Added pivot to internal network through ‘User’
4	Scan	FileServer	Success	Scan conducted through pivot
5	Exploit3	FileServer	Success	Attack conducted through pivot
6	Shutdown	N/A	Success	Performed clean-up actions

Appendix C. Software Dependencies and Install Instructions

Due to pre-installation of software packages and configuration of services the author recommends running ANDES on a Kali Linux system.

3.1 Software Dependencies

- Python3
- Metasploit Framework
- Nmap Framework

3.2 Python3 Library Dependencies

- libnmap 0.7.0 or newer
- pyAgrum 0.18.2 or newer
- pysploit3 3-1.0.3 or newer

3.3 Installation Instructions

ANDES consists of various Python3 files which must all be available to the Python3 environment when running the system. On the first execution of ANDES run the following command:

```
sudo -E python3 ANDES.py -i -s <Metasploit RPC Server IP Address>
```

This will attempt to install any missing required Python3 libraries as well as attempt to create a local DNS record utilized by the ANDES system.

3.4 Execution Instructions

ANDES requires root privileges to execute properly, because of this it is recommended to run ANDES via the *sudo* command. To initiate execution of ANDES the user must provide a starting target *-t* and an objective host *-o*. The command execution will look as follows:

```
sudo -E python3 ANDES.py -t <Target Host IP Address> \\
-o <Objective Host IP Address>
```

Bibliography

1. Defense primer: Cyberspace operations, 12 2020.

2. Jim Garamone. Esper describes dod's increased cyber offensive strategy, 09 2019.

3. Ryan Craig. Closing the cybersecurity skills gap, 11 2019.

4. Dean Richard McKinnel, Tooska Dargahi, Ali Dehghantanha, and Kim Kwang Raymond Choo. A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. *Computers and Electrical Engineering*, 75, 2019.

5. Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. Pomdps make better hackers: Accounting for uncertainty in penetration testing. *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 2012.

6. Carlos Sarraute, Olivier Buffet, and Jorg Hoffmann. Penetration testing == pomdp solving? *Proceedings of the 3rd Workshop on Intelligent Security (SecArt'11)*, 2011.

7. Yugansh Khera, Deepansh Kumar, S. Sujay, and Nidhi Garg. Analysis and impact of vulnerability assessment and penetration testing. pages 525–530. Institute of Electrical and Electronics Engineers Inc., 2 2019.

8. Peter Norvig and Stuart Russel. *Artifical Intelligence: A Modern Approach*. Prentice Hall, third edition edition, 2010.

9. Ptes technical guidelines, 2014.

10. S Jha, O Sheyner, and J Wing. Two formal analyses of attack graphs *. *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, 2002.

11. Luis Muñoz-González, Daniele Sgandurra, Andrea Paudice, and Emil C. Lupu. Efficient attack graph analysis through approximate inference. *ACM Transactions on Privacy and Security*, 20, 8 2017.

12. Carlos Sarraute, Gerardo Richarte, and Jorge Lucángeli Obes. *An Algorithm to Find Optimal Attack Paths in Nondeterministic Scenarios General Terms Security*. 2011.

13. Alaa T. Al Ghazo, Mariam Ibrahim, Hao Ren, and Ratnesh Kumar. A2g2v: Automatic attack graph generation and visualization and its applications to computer and scada networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–11, 5 2019.

14. L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. volume 2, pages 307–321. Institute of Electrical and Electronics Engineers Inc., 2001.

15. Sudip LP Saha Bloomberg, Anil S Kumar Vullikanti, Mahantesh Halappanavar, Sudip Saha, and Samrat Chatterjee. Identifying vulnerabilities and hardening attack graphs for networked systems. *Conference: 2016 IEEE Symposium on Technologies for Homeland Security (HST)*, 2016.

16. Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerabilty analysis. *ACM Proceedings*, pages 1–79, 1998.

17. Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An attack graph-based probabilistic security metric. *IFIP Annual Conference on Data and Applications Security and Privacy*, 2008.

18. Anoop Singhal and Xinming Ou. *Security risk analysis of enterprise networks using probabilistic attack graphs*. 2017.

19. Ni Gao, Yiyue He, and Beilei Ling. Exploring attack graphs for security risk assessment: A probabilistic approach. *Wuhan University Journal of Natural Sciences*, 23:171–177, 4 2018.

20. Ben Thorne. Using attack graphs to understand vulnerabilities. 2018.

21. Mohamed C. Ghanem and Thomas M. Chen. Reinforcement learning for efficient network penetration testing. *Information (Switzerland)*, 11, 1 2020.

22. Mohamed C. Ghanem and Thomas M. Chen. Reinforcement learning for intelligent penetration testing. 2019.

23. Jorge Lucangeli Obes, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. *SecArt'2010 at AAAI 2010*, 2010.

24. Jorg Hoffman. Metric-ff.

25. Core Security. Core impact's rapid penetration tests.

26. Izhar Matzkevich and Bruce Abramson. Decision analytic networks in artificial intelligence, 1995.

27. Carmen Lacave, Manuel Luque, and Francisco Javier Díez. Explanation of bayesian networks and influence diagrams in elvira. *IEEE Transactions on Systems, Man, and Cybernetics*, 2007.

28. Unbbayes, 2020.

29. agrum/pyagrum.

30. Gordon Lyon. Nmap.

31. Metasploit.

32. Dan McInerney. Pymetasploit3, 2020.

Acronyms

AI Artificial Intelligence. 7, 8, 47

ANDES Automated Network Discovery and Exploitation System. 2, 4, 7, 27, 53, 65

API Application Programming Interface. 47

ARP Address Resolution Protocol. 50

BOF Buffer Overflow. 61

CBT Conditional Probability Table. viii, 19, 56, 67

CNE Computer Network Exploitation. 1, 4, 8, 27, 53, 65

CSP Constraint Satisfaction Problem. 10

D4M Deny, Degrade, Disrupt, Destroy, or Manipulate. 30

DAG Directed Acyclic Graph. 17

DNS Domain Name System. 58

DoD Department of Defense. 1

EU Expected Utility. 23

GUI Graphical User Interface. 38

MDP Markov Decision Process. 2

MEU Maximum Expected Utility. 24, 33, 55

ML Machine Learning. 8

Nmap Network Mapper. 47

OS Operating System. 44

PDDL Planning Domain Definition Language. 2

POMDP Partially Observable Markov Decision Process. 2, 12, 35

RL Reinforcement Learning. 13

RPC Remote Procedure Call. 48

RPT Rapid Penetration Test. 15

SME Subject Matter Expert. 3, 5, 8, 28, 54, 65

TCP/IP Transmission Control Protocol / Internet Protocol. 29

VA-PT Vulnerability-Assessment and Penetration Testing. 2, 7, 53, 65

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 03–02–2021	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2019 — Mar 2021

4. TITLE AND SUBTITLE

Automated Network Exploitation
Utilizing Bayesian Decision Networks

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Graeme M. Roberts

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-21-M-076

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Information Exploitation and Operations Division
525 Brooks Rd Rome Lab
AFB NY 13441
COMM 315-330-2575
Email: Chad.Heitzenrater@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RIG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Computer Network Exploitation (CNE) is the process of using tactics and techniques to penetrate computer systems and networks in order to achieve desired effects. It is currently a manual process requiring significant experience and time that are in limited supply. This thesis presents the Automated Network Discovery and Exploitation System (ANDES) which demonstrates that it is feasible to automate the CNE process. The uniqueness of ANDES is the use of Bayesian decision networks to represent the CNE domain and subject matter expert knowledge. ANDES conducts multiple execution cycles, which build upon previous action results. Cycles begin by modeling the current belief state using Bayesian decision networks. ANDES uses these networks to select and execute an expected best action. Observed results are used to update the systems current belief state before the next cycle begins. ANDES was tested in a live-execution event, taking place within a virtual network environment. ANDES successfully performed a series of information gathering and remote exploit actions, across multiple network hosts to gain access to the target.

15. SUBJECT TERMS

Cyberspace Network Exploitation, Vulnerability Assessment, Penetration Testing, Artificial Intelligence, Bayesian Decision Networks, Influence Diagrams, Automation, Autonomous Agent, Automated Decision Making

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Gilbert Peterson, AFIT/ENG
U	U	U	UU	93	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-6565, ext 4281; Gilbert.Peterson@afit.edu