Policy Evolution with Genetic Programming: a Comparison of Three Approaches

Yow Tzu Lim, Pau Chen Cheng, John Andrew Clark and Pankaj Rohatgi

Abstract— **In the early days a policy was a set of simple rules with a clear intuitive motivation that could be formalised to good effect. However the world is now much more complex. Subtle risk decisions may often need to be made and people are not always adept at expressing rationale for what they do. Previous research has demonstrated that Genetic Programming can be used to infer statements of policies from examples of decisions made [1]. This allows a policy that may not formally have been documented to be discovered automatically, or an underlying set of requirements to be extracted by interpreting user decisions to posed "what if" scenarios. This study compares the performance of three different approaches in using Genetic Programming to infer security policies from decision examples made, namely symbolic regression, IF–THEN rules inference and fuzzy membership functions inference. The fuzzy membership functions inference approach is found to have the best performance in terms of accuracy. Also, the fuzzification and de-fuzzification methods are found to be strongly correlated; incompatibility between them can have strong negative impact to the performance.**

I. Introduction

In computer systems, a security policy is essentially a set of rules specifying the way to secure a system for the present and the *future*. Forming a security policy is a challenging task: the system may be inherently complex with many potentially conflicting factors. Traditionally security policies have had a strong tendency to encode a static view of risk and how it should be managed [2], typically in a pessimistic or conservative way. Such an approach will not suffice for many dynamic systems which operate in highly uncertain, inherently risky environments. In many military operations for example we cannot expect to predict all possible situations.

Much security work is couched in terms of risk but in the real world there are benefits to be had. In military operations you may be prepared to risk a compromise of confidentiality if not doing so could cost lives. There is a need for operational flexibility in decision making, yet we cannot allow recklessness. Decisions need to be defensible and so must be made on some principled basis. People are typically better at making specific decisions than in providing abstract justification for their decisions. It is very useful to be able to codify in what a "principled basis" consists of since this serves to document "good practice" and facilitates its propagation.

The above discussion has been couched in terms of human decision making. In some environments the required speed of system response may force an automated decision. Such automated decisions must also be made on a "principled basis" and some of these decisions may be very tricky. Automated support must be provided with decision strategies or rules to apply.

In this paper we investigate how security policy rules can be extracted automatically from examples of decisions made in specified circumstances. This is an exercise in *policy inference*. The automation aspect of the inference is doubly useful: automated inference techniques can discover rules that humans would miss; and policies can be dynamically inferred as new examples of tricky decisions become available. Thus the current policy can evolve to reflect the experience of the system. For example, if a human determines what the proper response should be based upon the information available, either in real-time or post facto, a conclusion is drawn that similar responses should be given under similar circumstances. Essentially, we attempt to partition the decision space such that each partition is associated with a response that is commensurate with the risk vs. benefit trade-off for that partition.

In practice, different decision makers may come to different decisions in the same circumstances, particularly if the decisions are tricky. Decision makers may use data that are not available to the inference engine to reach a decision, or else one decision-maker may simply have a different appetite for risk. Any inference technique must be able to handle sets of decision examples that do not seem entirely consistent. Genetic Programming (GP) is used for the experiments presented in this paper. Previous research has demonstrated that this approach is promising; security rules can be extracted automatically from examples of decisions made [1].

In this paper we present a comparison of three approaches in using Genetic Programming to infer statements of policies from decision examples made. In the first experiment, we take the view that policy is a function that maps decision making factors to a decision. Each individual is a candidate representation of this function. Essentially, this is an exercise

Yow Tzu Lim and John Andrew Clark are with the Department of Computer Science, University of York, UK. (Email: {yowtzu, jac}@cs.york.ac.uk)

Pau Chen Cheng and Pankaj Rohatgi are with the Department of Security and Privacy, IBM Hawthorne Watson Research Labs, USA. (Email: {pau,rohatgi}@us.ibm.com).

Research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

978-1-4244-1823-7/08/$25.00©2008 IEEE

of symbolic regression; determining the mapping function using previous decision examples as the guiding points. In the second experiment, we assume that policy can be represented with a set of IF <condition> THEN <action> rules. The condition is a list of logical predicates in terms of decision making factors, comparators and values (e.g. $TrustValue > 5$ AND $RiskValue \leq 4$). Each GP run is used to determine the condition for 1 decision; N runs of GP are required to search for a policy with N possible decisions. In the third experiment, we incorporate the fuzzy membership concept by considering each of the N decisions to be a fuzzy set. GP is used to search for the fuzzy membership functions for each fuzzy set. Again, N runs of GP are required to search for a policy. Then, an additional de-fuzzification step is employed to determine from the N membership functions the most appropriate decision. The work presented here significantly extends the work presented in [1].

The results show that the fuzzy membership functions inference approach has the best performance in terms of accuracy. Also, the fuzzification and de-fuzzification methods are found to be strongly correlated; incompatibility between them can have strong negative impact to the performance.

The organisation of this paper is as follows: Section II presents the related work. Section III presents a brief introduction about the chosen policy – Fuzzy MLS policy model, which serves as the target policy to be learnt. Sections IV to VI presents the experiment designs and results of three different approaches, namely, the symbolic regression, IF–THEN rules inference, and fuzzy membership inference. In Section VII, we compare and discuss the experimental results. Lastly, Section VIII concludes the report with a summary of the experiment results and possible future researches.

II. Related Work

Security policies are mostly written in high level forms and later followed by a series of transformations and refinements. Improvement in easing this process include automated transformation of high level human understandable rules to low level machine executable rules, automated policy conflicts and coverage checking and resolution. Despite many improvements achieved in these techniques, there are no known attempts in generating the policy automatically from previous decision examples using machine learning techniques. Our present study is a radical attempt to address an important problem (similar concept is mentioned in [3] but no published work have been released so far).

On the other hand, rule inferencing techniques have been around for many years in the machine learning domain. There are various approaches proposed, e.g. decision tree induction, Genetic Algorithm (GA), Genetic Programming (GP), Artificial Immune Systems (AIS), etc. In this paper, emphasis is placed on GP. In [4], a grammar-based genetic programming system called LOGENPRO (The LOGic grammar based GENetic PROgramming system) is proposed. In the performance test conducted, it is found that LOGENPRO outperforms some Induction Logic Programming (ILP) systems. In [5], a GP experiment on co-evolution between rules and fuzzy membership of variables is designed. The result shows that the output set of rules and variables are well adapted to one another. In [6], an attempt is made to invent a generic rule induction algorithm using grammar based GP. The result is shown to be competitive with well known manually designed rule induction algorithms. However, in all the above mentioned cases, there have been no previous known research on rule inferencing technique in the our domain of interest – security policy.

III. Fuzzy MLS Model

The Fuzzy MLS policy model [7] is used throughout the experiments to illustrate the concept of policy evolution.

Fuzzy MLS model is an adaptive extension to traditional MLS (multi-level security) Bell-LaPadula policy model [8]. In the traditional model, every subject or object is assigned a security label ($\langle$ sensitivity level, categories set $\rangle$). For a read access, r, the policy can be summarised as follows:

$$\begin{aligned} &\text{IF } sl \geq ol \text{ AND } sc \supseteq oc \text{ THEN } r \text{ is } \textit{allowed} \\ &\text{IF } sl < ol \text{ OR } sc \not\supseteq oc \text{ THEN } r \text{ is } \textit{denied} \end{aligned} \quad (1)$$

where sl and ol are subject and object sensitivity levels and sc and oc are subject and object category sets. In other words, a subject can access an object iff the subject is trustworthy enough ($sl \geq ol$) and has the legitimate "need-to-know" ($sc \supseteq oc$) to access the object. In terms of risk, the traditional MLS policy can be viewed as setting a fixed trade-off between risk of information disclosure versus the benefit an organisation can gain from it. As indicated in [7], it is a non-adaptive, binary access control decision model where accesses have been pre-classified as having either acceptable or unacceptable risk and only accesses with acceptable risk are allowed.

The Fuzzy MLS model uses this risk based rationale, but extends the MLS model to be based on risk management rather than risk avoidance inherent in the binary decision process [2]. The Fuzzy MLS model takes a more flexible and sophisticated view by computing *quantified estimates* of the risk from *unauthorised disclosure of information* and using these estimates to build a risk scale shown in Figure 1. The risk scale is divided into multiple bands. Each band is associated with a decision. The risk in the bottom band is considered low enough so the decision is simply *allow* whereas the risk in the top band is considered too high so the decision is *deny*. Each band between the top and bottom is associated with a decision *allow with band-specific risk mitigation measures*.

The Fuzzy MLS model defines risk as the *expected value of damage* caused by unauthorised disclosure of information:

$$\begin{aligned} risk \;=\; &(expected\ value\ of\ damage) \times \\ &(probability\ of\ unauthorised\ disclosure) \end{aligned} \quad (2)$$

The value of the damage is estimated from the object's sensitivity level. The probability of unauthorised disclosure

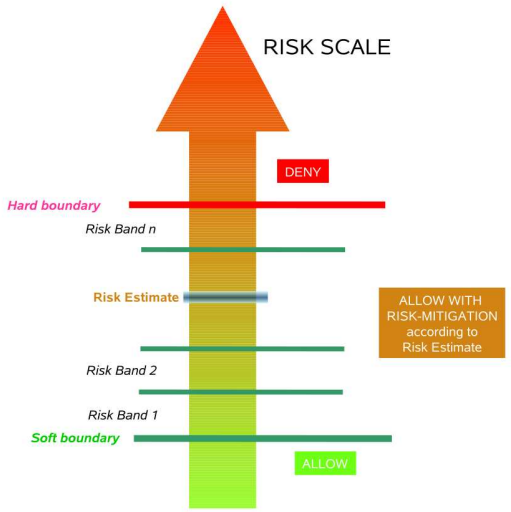

Fig. 1. Risk adaptive access control on a risk scale [7]

is estimated by quantifying two "gaps": one between the subject's and the object's sensitivity levels and the other between the subject's and the object's category sets. For simplicity, this experiment looks only at the sensitivity levels and assumes the categories sets are the same[1], thus risk becomes a function of subject and object sensitivity levels only. For more detail on risk quantification, see [7], [9].

In [1], a way to partition the risk bands (which is in turn associated with a decision) is defined to map an estimated risk to a risk band:

$$band(risk(sl, ol)) = min(\lfloor log_{10}(risk(sl, ol)) \rfloor,\ N - 1) \quad (3)$$

The band numbers start from 0, N is the number of bands desired on the scale and the function $risk(sl, ol)$ is defined in Appendix according to [7]. The intuition of using base-10 logarithm in (3) is to use a risk band number to represent *the order of magnitude* of risk. Since each band is associated with a decision, a risk band number computed using formula 3 represents a possible decision in the policy.

Based upon this model, experiments have been conducted to assess the performance of three different Genetic Programming based approaches in inferencing policies from decision examples. The following sections describe experiments done based upon each of these approaches in detail.

IV. Experiment 1: Symbolic Regression Approach

In this experiment, we view a policy as a function that maps decision making factors to a decision itself. For example, the "no read up" part of traditional MLS Bell-LaPadula model can be viewed as a boolean function $access(sl, ol, sc, oc)$ which maps to $True$ iff $sl \geq ol$ and $sc \supseteq oc$. In the Fuzzy MLS model, this mapping function is the composition band and risk function in (3). GP is used to search for an equivalent function of this composition function. This is an exercise of symbolic regression based upon the decision examples.

[1]Therefore the gap between categories sets is 0.

A. Individual representation

Each individual represents a candidate function that corresponds to a policy. Since only the sensitivity levels are considered, the terminal set T has only two variables, namely sl and ol and a set of real constant numbers in the range of $(-1, 1)$ is added to T. In the function set, 12 basic arithmetic operators are defined. These are $+$, $-$, $*$, $/$, exp, log, sin, cos, max, min, $ceil$ and $floor$. The $/$ and exp operators defined are the conventional protected operators; division by 0 returns 1 and the exponential function returns the maximum value defined in the IEEE-standard double precision (64-bits) floating point number if overflow happens.

B. Training set and testing set generation

The training sets and testing sets described here are used throughout all 3 experiments described in this paper to allow consistent comparison. Each example x in the training and testing sets is a $(sl_x, ol_x, band_x)$ triple, where $band_x$ is calculated using (3). In other words, all the examples used are assumed to be correct.

3 different training sets are used in each experiment. The first training set consists of all possible 100 (sl, ol) pairs where sl and ol are integers in $[0, 9]$. This optimal setting lends itself to act as the control. The second and third sets consist of 100 and 500 randomly generated (sl, ol) pairs where sl and ol are also integers in $[0, 9]$. Unlike in the control, these two sets would have incomplete coverage and uneven distribution of examples over risk bands.

After going through the evolution process, the best individual in the population is selected to test against two sets of examples. The first set is same as the first 100-example training set. This testing set provides a good indication on how much "knowledge" has been acquired by the approach employed in a fixed number of generations. The second testing set consists of 100 randomly generated (sl, ol) pairs where sl and ol are *real numbers* in $[0.0, 9.0]$. Therefore, most of these examples are unseen yet similar to training examples. This set provides a good measure on how much the acquired knowledge can be applied for unseen cases.

C. Fitness evaluation

Two principles are used to determine the score for a decision made by an individual. For an example x and an individual i, if i evaluates x to be in band j (j is $i(x)$ rounded to the nearest integer), then:

- For a correct decision, *reward more* the higher the risk band; i.e., reward higher j more than lower j. (We care more about higher risk bands)
- For an incorrect decision, *punish more* the more the decision deviates from the target; i.e., punish more

as $|j - band_x|$ becomes larger. Also, punish *under-estimation* of the risk band more than over-estimation of it; i.e., punish more if $band_x > j$.

Based upon these principles, the fitness of an individual i, $fitness(i)$ is defined as follow:

$$fitness(i) = \sum_{\forall\ example\ x} score(x) \tag{4}$$

where

$$score(x) = \begin{cases} band_x + 1 & \text{if } j \equiv band_x, \\ -(band_x - j) & \text{if } band_x > j, \\ -(j - band_x)/2 & \text{if } band_x < j \end{cases}$$

D. Policy resolution

The learnt function might not be perfect; sometimes the function might map a particular (sl, ol) pair to a value that is out of band range. When this happens, we assume that all out-of-range values represent the highest band, i.e., in our case, any value not in the range $[0, 9]$ is assumed to be 9. This is consistent with the usual attitude to security which favours overestimation of risk rather than underestimation. In a future run-time deployment of our inference approach it would be possible to involve human interaction. An alert would be given to the security administrator and the administrator would decide which band an input should be mapped to. Then, this decision can be used as a new training example.

E. Summary

The experiment setup is summarised in Table I. This experiment is carried out using the ECJ Framework v16 [10] and the default values are used for unmentioned parameters.

Objective	Search for the nearest equivalent function of $band(risk(sl, ol))$ in (3)
Terminal set T	$\{sl, ol\} \cup \{r \mid -1.0 < r < 1.0\}$
Function set F	$\{+, -, *, /, exp, log, sin, cos\ max, min, ceil, floor\}$
Fitness function $f_j(i)$	$fitness(i)$ in (4)
Number of generation	500
Population size	1024 (default)
Population initialisation	Ramp half and half method with the minimum and maximum height of the tree set to be 2 and 6 respectively (default)
Genetic Operators	crossover, mutation, reproduction (default)
Maximum height of tree	17

TABLE I
SUMMARY OF EXPERIMENT 1 SETUP

V. EXPERIMENT 2: IF–THEN RULES INFERENCE APPROACH

We take an alternative view here on policy as a set of IF <condition> THEN <action> rules. Each decision action is considered as a set, we shall use GP to search for the condition corresponding to some particular decision action (e.g. "allow read"). Essentially, the security policy inference problem is transformed into a N-classes classification problem, in which N is the number of rules in the policy. GP is used to search for the condition part of each rule. Therefore, the number of GP runs increases linearly with the policy size. This is not as daunting as it seems to be. As all these searches are independent from one another, this design approach can benefit from the multi-core processor revolution and execute searches in parallel. With only 1 processor, the binary decomposition method[2] can be employed to solve this problem in $N - 1$ GP runs.

In the Fuzzy MLS model, the risk scale is divided into 10 bands numbered from 0 to 9. Therefore, 10 GP runs are required to search for conditions for all the bands. The target condition for band j, TC_j to be learnt is:

$$TC_j(sl, ol) \ = \ (band(risk(sl, ol)) \ \equiv \ j) \tag{5}$$

A. Individual representation

As each individual reassembles the condition expression in the IF–THEN rules, the root node in a tree must evaluate to a Boolean. At the highest (nearest to the root) layer, there are composition operators, AND, OR and NOT. The second layer consists of logic relational operators such as $<$ or $=$. The next layer consists of arithmetic operators such as $+$ or sin and the leaf nodes at the bottom layer are elements of the terminal set T. Thus, no boolean nodes can have real numbers as their ancestors. Strongly Typed Genetic Programming (STGP) [11] is used to ensure this structure. Figure 2 shows an example of well-typed individual.

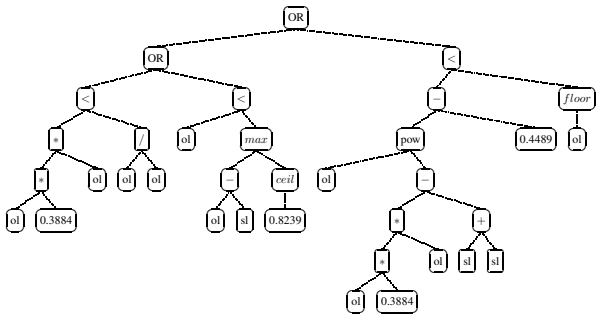

Fig. 2. An example of the well-typed individual

The terminal set T has only two variables, namely sl and ol and a set of real constant numbers in the range of $(-1, 1)$ is added to T. These elements are real numbers. There is an additional set of boolean constants; $True$ and $False$ added to T. For real numbers, 15 operators are defined: 3 relational operators ($=$, $<$ and $>$), each returns a boolean value and 12 arithmetic operators $(+, -, *, /, exp, log, sin, cos, max, min, ceil, floor)$,

[2]Binary decomposition method decomposes the N classes classification problem into $N - 1$ binary classification problems. The first classification problem is $(c_1, {c_1}' \equiv P - c_1)$, second problem is $(c_2, {c_2}' \equiv {c_1}' - c_2)$, $N-1$ problem is $(c_{N-1}, {c_{N-1}}' \equiv {c_{N-2}}' - c_{N-1} \equiv c_N)$. The n^{th} binary classification problem can only be solved after the n previous problems are solved. The algorithm is inherently sequential.

each returns a real number. Again, the / and exp operators defined are the conventional protected operators as described in Experiment 1. For the boolean values, there are 3 operators, AND, OR and NOT defined, each returns a boolean value.

B. Fitness evaluation

As in Experiment 1, different scores are given to different kinds of decisions made by an individual according to the following four principles; two of them are the same as the principles in Experiment 1, the other two are added following the fact that this design allows us to distinguish between positive and negative decisions. In the search for the condition of band j, TC_j, let $d_{i,x}$ be the decision made by i for an example x, then:

- For a correct decision, *reward more* if
  - the risk is higher for security concerns; i.e., reward more for a larger j.
  - the decision is *true positive* (hits the target); i.e., reward more when $band_x \equiv j$. This is to overcome the effect of having relatively few examples in band j when $j \neq 0$ (less than 10% on average).
- For an incorrect decision, *punish more* if
  - the decision is *false positive* ($d_{i,x} \equiv True$ and $band_x \not\equiv j$) and is *more off the target*; i.e., punish more as $\mid j - band_x \mid$ becomes larger. Also, for security concerns, punish more if this false positive decision *underestimates the risk*; i.e., punish more if $band_x > j$.
  - the decision is *false negative* ($d_{i,x} \equiv False$ and $band_x \equiv j$) when the risk is higher for security concerns; i.e., punish more for a larger j.

Based upon these four principles, the fitness score for an individual in the search for the condition of band j, $fitness_j(i)$ is defined as:

$$\begin{aligned} fitness_j(i) \;=\; & \sum_{\{x|band_x \equiv j\}} w_{tp}\{d_{i,x} \equiv True\} \;+ \\ & \sum_{\{x|band_x \not\equiv j\}} w_{tn}\{d_{i,x} \equiv False\} \;- \\ & \sum_{\{x|band_x \not\equiv j\}} w_{fp}\{d_{i,x} \equiv True\} \;- \\ & \sum_{\{x|band_x \equiv j\}} w_{fn}\{d_{i,x} \equiv False\} \end{aligned} \quad (6)$$

where

$$\begin{aligned} w_{tp} &= j+1, \\ w_{tn} &= (j+1)/10, \\ w_{fp} &= \begin{cases} band_x - j & \text{if } band_x > j, \\ (j - band_x)/2 & \text{if } band_x < j, \end{cases} \\ w_{fn} &= j+1 \end{aligned}$$

C. Policy resolution

When two or more TC evaluated to $True$, the highest band with $TC \equiv True$ is selected for security concerns. Formally, if $TC_j \equiv True$ and $TC_k \equiv True$ and $j > k$, then band j instead of band k is used. Also, if there is no TC evaluated to $True$, the highest band is used again for the same reason. An alternative implementation which involve human decision making as described in Section IV-D can also be used instead. Then, this decision can be used as a new training example.

D. Summary

The experiment setup is summarised in Table II. As in Experiment 1, this experiment is carried out using the ECJ Framework v16 [10] and the default values are used for unmentioned parameters.

Objective	Search for the equivalent functions of TC for all bands, $\forall j \in [0,9], TC_j$ in (5)
Terminal set T	$\{sl, ol\} \cup \{r \mid -1.0 < r < 1.0\}$ $\cup \{TRUE, FALSE\}$
Function set F	$\{+, -, *, /, exp, log, sin, cos$ $max, min, ceil, floor\}$ $\cup \{AND, OR, =, >, <\}$
Fitness function $f_j(i)$	$fitness(i)$ in (6)
Number of generation	500
Population size	1024 (default)
Population initialisation	Ramp half and half method with the minimum and maximum height of the tree set to be 2 and 6 respectively (default)
Genetic Operators	crossover, mutation, reproduction (default)
Maximum height of tree	17

TABLE II
SUMMARY OF EXPERIMENT 2 SETUP

VI. EXPERIMENT 3: FUZZY MEMBERSHIP APPROACH

In this section we aim to provide some degree of smoothing to our search space by adopting a fuzzy-inspired approach. We shall seek for each target band j a fuzzy membership function $M_j(sl, ol)$, whose response reflects the likelihood of a given risk should be assigned that band. Later we shall use these band membership functions to determine the most appropriate band for given input (sl, ol).

A. Fuzzification

To guide the learning of the fuzzy set membership function for band j, $M_j(sl, ol)$, the *target membership* of each of the 10 bands in the band j fuzzy set is first defined. Essentially, these 10 pre-defined points characterise the shape and location of the M_j curve. The learning process becomes a *curve fitting* exercise to search for a curve that best fits the 10 points, using all the examples in the training set. Curve fitting is naturally more tolerant of incomplete coverage in the training set because it uses *interpolation* and *extrapolation* to compensate for the "missing points". It is also more resilient to a few out-liars in the training set.

Furthermore, the fuzzy membership range is changed to $[-1.0, 1.0]$ with 0.0 represents full membership. This is different from the traditional fuzzy membership range, $[0.0, 1.0]$ with 1.0 representing full membership. The expansion on the negative range allows the information about the direction (left or right to the target band) to be encoded. For example, for $M_5(sl, ol)$, the target membership of each band, starting from band 0, can be defined as:

$$[-0.5, -0.4, -0.3, -0.2, -0.1,\ 0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4]$$

Band 5 has membership 0. The target memberships for other $M_{j\neq 5}$ can be defined similarly. With all 10 membership functions learnt, one can determine the band of an input by feeding the input to all 10 functions and computing the band number by examining the 10 membership values returned by these functions. For example, if each membership value indicating that "the distance between the input and my band places the input close to band 5", then with very high confidence we can say the input belongs to band 5. This is analogous to examine the input from 10 different perspectives to draw the final conclusion, which is more likely to be accurate than examining the input from one perspective.

Two setups with different pre-defined target memberships are carried out to validate this concept. In the first setup the 10 target memberships for M_j are defined as :

$$M_j(k) \ \equiv \ (k - j)/10,\ k = 0\ to\ 9 \qquad (7)$$

This is like mapping a traditional triangle fuzzy membership function, which has the range $[0, 1]$ and $M_j(j) \equiv 1$ as the tip of the triangle, to a straight-line membership function with the range $[-1, 1]$ and $M_j(j) \equiv 0$. Figure 3 shows the target membership curves for all 10 bands using (7). In the second setup, bell-shaped Gaussian distribution curves are mapped in a similar fashion and Figure 4 shows the membership curves for all 10 bands.

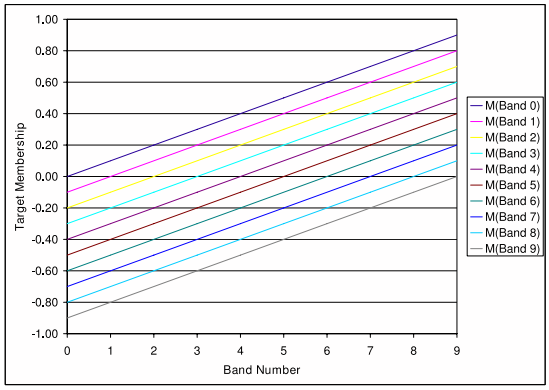

Fig. 3. The linear (modified triangle) membership functions, M_{linear} for of all the bands

B. De-fuzzification

Once all 10 membership functions are learnt and feeding an input $x \equiv (sl_x, ol_x)$ to these functions, a *de-fuzzification* mechanism is required to map all 10 values returned by these functions to a risk band number. Two voting based algorithms are used for de-fuzzification: One (Algorithm 1) uses the direction knowledge only [1] and the other one (Algorithm 2) uses both the direction and distance knowledge.

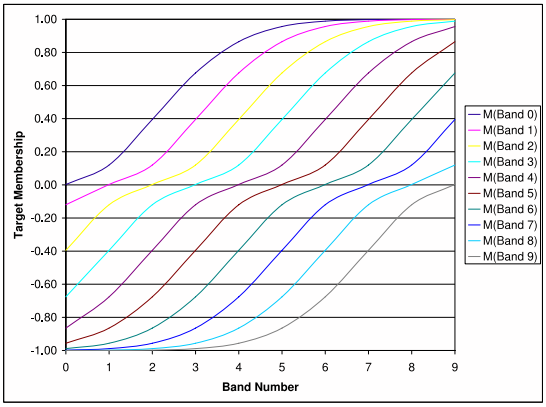

Fig. 4. The curve (modified Gaussian distribution) membership functions, $M_{Gaussian}$ for of all the bands

initialise an array $v[10]$ *with all elements set to* 0
forall *example* x **do**
  forall *band* j **do**
    if $M_j(sl_x, ol_x) > 0.05$ **then**
      forall $k > j$ **do**
        $v[k] \leftarrow v[k] + 1$
    else if $M_j(sl_x, ol_x) < -0.05$ **then**
      forall $k < j$ **do**
        $v[k] \leftarrow v[k] + 1$
    else if $-0.05 \leq M_j(sl_x, ol_x) \leq 0.05$ **then**
      $v[j] \leftarrow [j] + 1$
choose $v[i]$ *with the maximum value*
outputs i *as the risk band number*

Algorithm 1: Direction based de-fuzzification [1]

C. Individual representation

In this experiment, the terminal set T consists of sl, ol and a set of random real constant numbers in the range of $(-1, 1)$ whereas the function set F is reduced to $\{+, -, *, /, exp, log, sin, cos\ max, min, ceil, floor\}$ just as in Experiment 1. The individual structure used here is also similar as in Experiment 1, but they represent two different things. Each individual here resembles the membership function for the band in question whereas an individual in Experiment 1 resembles a function that corresponds to the policy as a whole.

D. Fitness evaluation

The fitness function uses per-band *normalised* distance: let $M_{j,i}$ represent an individual i in the search for the member-

initialise an array $v[10]$ *with all elements set to* 0
forall *example* x **do**
  forall *band* j **do**
    if $M_j(sl_x, ol_x) > 0.05$ **then**
      $p \leftarrow j + M_j(x) * 10$
      $k \leftarrow min(\lfloor p + 0.5 \rfloor, 9)$
      $v[k] \leftarrow v[k] + 3$
      if $k > p$ **then**
        $v[k-1] \leftarrow v[k-1] + 2$
        $v[k+1] \leftarrow v[k+1] + 1$
      else
        $v[k-1] \leftarrow v[k-1] + 1$
        $v[k+1] \leftarrow v[k+1] + 2$
    else if $M_j(sl_x, ol_x) < -0.05$ **then**
      $p \leftarrow j + M_j(x) * 10$
      $k \leftarrow max(\lceil p - 0.5 \rceil, 0)$
      $v[k] \leftarrow v[k] + 3$
      if $k < p$ **then**
        $v[k+1] \leftarrow v[k+1] + 2$
        $v[k-1] \leftarrow v[k-1] + 1$
      else
        $v[k+1] \leftarrow v[k+1] + 1$
        $v[k-1] \leftarrow v[k-1] + 2$
    else if $-0.05 \leq M_j(sl_x, ol_x) \leq 0.05$ **then**
      $v[j] \leftarrow v[j] + 3$
      $v[j+1] \leftarrow v[j+1] + 1$
      $v[j-1] \leftarrow v[j-1] + 1$
choose $v[i]$ *with the maximum value*
outputs i *as the risk band number*

Algorithm 2: Direction and distance based de-fuzzification

ship function for band j, the individual fitness, $fitness_j(i)$ is defined as follow:

$$fitness_j(i) = \sum_{\forall\ bands\ k=0\ to\ 9} score_k(x) \qquad (8)$$

where

$$score_k(x) = \frac{\sum_{\forall\ x\ in\ band\ k} |M_{j,i}(sl_x, ol_x) - M_j(band_k)|}{total\ number\ of\ band\ k\ examples}$$

Thus, the $fitness_j(i)$ represents the sum of *normalised* distances between the $M_{j,i}$ and M_j.

In contrast to the previous two experiments, here smaller $fitness_j(i)$ means better fit.

E. Summary

The experiment setup is summarised in Table III. This experiment is carried out using the ECJ Framework v16 [10] and the default values are used for unmentioned parameters.

Objective	Search for the fuzzy membership functions for all bands, $\forall j \in [0, 9]$, M_j characterised by the examples in the training set
Terminal set T	$\{sl, ol\} \cup \{r \mid -1.0 < r < 1.0\}$
Function set F	$\{+, -, *, /, exp, log, sin, cos\ max, min, ceil, floor\}$
Fitness function $f_j(i)$	$fitness_j(i)$ in (8)
Number of generation	500
Population size	1024 (default)
Population initialisation	Ramp half and half method with the minimum and maximum height of the tree set to be 2 and 6 respectively (default)
Genetic Operators	crossover, mutation, reproduction (default)
Maximum height of tree	17

TABLE III
SUMMARY OF EXPERIMENT 3 SETUP

VII. RESULT AND DISCUSSION

Each experiment is repeated for 10 times using different random seeds. The average of the policy performances over the 10 runs in terms of the distance from the target band is summarised in the Table IV. The training sets and testing sets are explained in Section IV-B.

In general, the experiments using fuzzy membership approach consistently perform the best in terms of accuracy on both the discrete and continuous testing sets, the experiments using IF–THEN rules approach have good performances on discrete testing set, but poor performances on continuous testing set and the experiments using the symbolic regression approach have poor performances on both testing sets.

The poor performances of the experiments using the symbolic regression approach can be explained from the function used in partitioning the risk scale (3) which results in uneven distribution of the examples in the training sets.The output band is strongly correlated to one of the inputs, ol. Approximately half of the training examples are mapped to band equal to ol. This becomes a high local optimum point which is difficult to escape. Indeed, a careful analysis on result reveals it is the case that the best function (policy) learnt maps every possible input pairs to ol in one third of the total runs. Another local optimum is the function that maps every possible input pairs to band 0.

In the experiments using IF–THEN rules inference approach, the performance of the policies improve significantly on the discrete testing set (Testing Set 1). The policy performances also improve as the training set size increases. In particular, the policy learnt using 500 random examples performs extremely well, it maps 93.8 examples in average to the bands correctly and the mean distance between the target bands and the results is 0.348. This is because the high local optimum points present in the last approach are removed by design ($band = ol$) or the use of the weighted function ($band = 0$). However, the performance of the policies on continuous testing set (Training Set 2) does remain similar to the policies learnt using symbolic regression approach.

Furthermore, analysis on the outputs of these two approaches show that there are numerous unusual cases such that some (sl, ol) pairs with $(high, low)$ values are mapped to band 9 (the highest band in our case). This suggests that the policy resolution mechanism has taken placed. In

other words, policies learnt are incomplete. This necessarily pessimistic policy resolution mechanism degrades the performance significantly.

The experiments using fuzzy membership functions inference approach consistently have the best performance in terms of accuracy on both the discrete and continuous testing sets. Also, the de-fuzzification method that uses both direction and distance information improves the performance of the experiments using the mapped triangle based fuzzification, the mean distance from the target for all cases is reduced to less than 0.8. However, this new de-fuzzification method degrades the performance of the experiments using the mapped Gaussian curve based fuzzification. This is because the de-fuzzification method makes the assumption that the distance from the target increases linearly in respect to the membership value. This is not the case in Gaussian curve based fuzzification. A possible further work is to design a compatible de-fuzzification mechanism using both distance and direction for Gaussian curve fuzzification.

VIII. Conclusion and Further Work

This report presents a comparison of three different Genetic Programming based approaches in which policies can be inferred from a set of previously made decision examples. These approaches are symbolic regression, IF–THEN rules inference and fuzzy membership functions inference approaches. The fuzzy membership functions inference approach is found to have the best performance in terms of accuracy. Also, the fuzzification and de-fuzzification methods are found to be strongly correlated; incompatibility between them can have strong negative impact on the result, e.g. the incompatibility between the Gaussian curve based fuzzification and linear distance based de-fuzzification.

Security policy inference is a new domain in which evolutionary algorithms can be employed. Envisaged future work are:

1) Injection of false examples – Instead of just using training examples with correct decisions, we could include some examples with wrong decisions to see if this learning approach is sufficiently robust.
2) Multi-objective genetic programming – In our experiments, a weighted sum is used to calculate the individual fitness. The weights are defined based upon the approximated distances from the expected result. This is not always possible in other policies. For example, a binary-decision model like the Bell-LaPadula model only determines whether a decision is right or wrong, but can not tell how good (or bad) a decision is, i.e., the distance is constant. Also, the weight is difficult to justify. A radical way to put this matter forward is to employ multi-objectives genetic programming (MOGP).
3) Other evolutionary paradigms – Most of the real world policy can be much bigger than the Fuzzy MLS model. For security concerns, it is good to have a defined structure for the policy to make it more analysable. Grammatical Evolution which uses a set of grammar rules to define the individual structure and an evolutionary algorithm to evolve a string of indices which is then used to index which grammar rules to be "expanded" can be a good starting point.
4) Fuzzification – While the policies learnt using fuzzy membership functions inference approach have superior performance compares to others, different fuzzification methods seem to result different performances. A possible avenue of further work include the way in choosing these pre-defined points in a systematic fashion.

Our GP-based inference approach has the potential to make a significant contribution to the field of policy inference. Everyone accepts that policy specification is currently hard, and things are set to worsen as systems are deployed in ever more complex environments with increasing sophistication and subtlety of decision making needed. The work reported here and in [1] shows that the technique has very considerable promise. We recommend this important application area to fellow researchers.

Appendix A: Risk Computation

In [7], the risk resulted from the "gap" between a subject's and an object's sensitivity levels (sl and ol) is estimated using the following formula:

$$risk(sl, ol) \;=\; Val(ol) \times P_1(sl, ol) \tag{9}$$

$Val(ol)$ is the estimate value of damage and is defined in [7] as

$$Val(ol) = a^{ol}, \quad a > 1$$

The object sensitivity level is considered to be the *order of magnitude* of damage and hence $Val(ol)$ is defined as an exponential formula. In our experiments we set a to be 10. $P_1(sl, ol)$ is the probability of unauthorised disclosure and is defined in [7] as a sigmoid function:

$$P_1(sl, ol) = \frac{1}{1 \;+\; exp(-k(TI(sl, ol) - mid))}$$

$TI(sl, ol)$ is called the *temptation index* which indicates how much the subject with sensitivity sl is tempted to leak information with sensitivity level ol; it is defined as:

$$TI(sl, ol) = \frac{a^{(ol-sl)}}{M - ol}$$

The intuition for $P_1(sl, ol)$ and $TI(sl, ol)$ can be found in [7]. The value mid is the value of TI that makes P_1 equal 0.5; the value k controls the slope of P_1. The value M is the *ultimate object sensitivity* and the temptation TI approaches infinity as ol approaches M; the intuition is that access to an object that is as sensitive as or more sensitive than M should be controlled by human beings and not machines. In our experiments, the maximum value for sl and ol is 10; the settings for k, mid and M are $k = 3,\ mid = 4,\ M = 11$.

Experiment	Training set	Testing set	Distance from target band				Mean distance
			0	1	2	≥ 3	
Symbolic Regression	1	1	62.3	9.6	6.0	22.1	1.248
		2	43.6	20.7	7.3	28.4	1.703
	2	1	71.9	8.0	4.2	15.9	0.895
		2	52.3	21.0	5.7	21.0	1.411
	3	1	63.2	10.7	5.7	20.4	1.203
		2	38.5	29.5	7.5	24.5	1.667
IF–THEN Rules	1	1	88.1	0.4	0.7	10.8	0.797
		2	48.1	19.0	5.5	27.4	1.831
	2	1	82.7	3.7	2.3	11.3	0.746
		2	52.7	20.9	5.4	21.0	1.500
	3	1	93.8	0.4	0.0	5.8	0.348
		2	49.6	20.9	4.5	25.0	1.824
Mapped triangle (Direction only)	1	1	50.7	26.4	11.5	11.4	0.868
		2	40.5	33.0	13.1	13.4	1.076
	2	1	56.3	23.9	9.2	10.6	0.826
		2	45.0	31.7	8.6	14.7	1.056
	3	1	53.9	23.5	11.1	11.5	0.863
		2	43.4	36.6	10.9	9.1	0.921
Mapped Gaussian curve (Direction only)	1	1	63.9	20.6	8.3	7.2	0.620
		2	51.8	29.7	12.7	5.8	0.757
	2	1	56.9	20.9	14.4	7.8	0.772
		2	41.8	25.4	18.9	13.9	1.173
	3	1	71.4	19.4	6.7	2.5	0.414
		2	60.1	24.2	9.2	6.5	0.638
Mapped triangle	1	1	91.6	4.8	1.0	2.6	0.178
		2	62.2	23.5	3.6	10.7	0.781
	2	1	76.0	13.3	3.6	7.1	0.514
		2	62.1	23.9	4.8	9.2	0.634
	3	1	78.5	10.8	2.5	8.2	0.535
		2	58.8	27.3	3.4	10.5	0.782
Mapped Gaussian curve	1	1	65.5	14.4	9.6	10.5	0.696
		2	52.6	18.8	12.7	15.9	1.013
	2	1	62.6	14.6	9.2	13.6	0.916
		2	47.9	17.6	11.9	22.6	1.492
	3	1	65.4	14.5	9.0	11.1	0.707
		2	50.3	19.7	12.3	17.7	1.097

TABLE IV

PERFORMANCE SUMMARY OF THE LEARNT POLICIES USING DIFFERENT APPROACHES

ACKNOWLEDGEMENT

This research is funded as part of the International Technology Alliance (ITA) Project 6: Risk and Trust Management in Dynamic Coalition.

REFERENCES

[1] Y. T. Lim, P. C. Cheng, J. A. Clark, and P. Rohatgi, "Policy Evolution with Genetic Programming," IBM Research Report RC24442, Tech. Rep., 2008.

[2] "Horizontal Integration: Broader Access Models for Realizing Information Dominance," The MITRE Corporation JASON Program Office, Mclean, Virginia, Tech. Rep. JSR-04-132, Dec 2004.

[3] P. D. McDaniel, "Policy Evolution: Autonomic Environmental Security," December 2004. [Online]. Available: www.patrickmcdaniel.org/talks/serc-12-04.pdf

[4] M. L. Wong and K. S. Leung, *Data Mining Using Grammar Based Genetic Programming and Applications*, ser. Genetic Programming. Kluwer Academic Publishers, Jan. 2000, vol. 3.

[5] R. R. F. Mendes, F. de B. Voznika, J. C. Nievola, and A. A. Freitas, "Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. San Francisco, California, USA: Morgan Kaufmann, 7-11 2001, p. 183. [Online]. Available: citeseer.ist.psu.edu/mendes01discovering.html

[6] G. Pappa and A. Freitas, "Towards a genetic programming algorithm for automatically evolving rule induction algorithms," in *Proc. ECML/PKDD-2004 Workshop on Advances in Inductive Learning*, J. Furnkranz, Ed., Pisa, Italy, September 2004, pp. 93–108. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2004/2023

[7] P. C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger, "Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control," IBM Research Report RC24190, Tech. Rep., 2007.

[8] D. Bell and L. LaPadula, "Secure computer systems: Mathematical foundations," MITRE Corporation, Tech. Rep. ESD-TR-73-278, 1973.

[9] P. C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger, "Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control," *IEEE Symposium on Security and Privacy*, pp. 222–230, 2007.

[10] S. Luke, "ECJ version 16 A Java-based Evolutionary Computation Research System," August 2007. [Online]. Available: http://cs.gmu.edu/ eclab/projects/ecj/

[11] D. J. Montana, "Strongly Typed Genetic Programming," *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, 1995.