IT University of Copenhagen

Master Thesis

Software Development and Technology

Role-Mining of Access Control Policies with an Evolutionary Computation Approach

Author:
Theresa Ragna Elisabeth
Brandt von Fackh

Supervisor:
Sebastian
Risi

1. December 2015

Dedicated to ...

1 Abstract

In this thesis I randomly research how an EA can be applied to solve the Role-Mining Problem, where I don't know the outcome. But I learned something about Role Mining and EAs.

2 Acknowledgement

SECTION UNDER CONSTRUCTION

Contents

3 Introduction

SECTION UNDER CONSTRUCTION

3.1 Motivation

Why is Role Mining done?

3.2 Thesis question

How can the role mining problem be solved with EA

3.3 Roadmap

Overview of the chapters

4 Domain and problem

In this section a basic introduction to the domain is given. It is important that the motivation and concepts of the domain are known in order to scope the problem definition and for guidance in the implementation of the approach.

4.1 Access Control Models

Access control ensures through policy definitions and enforcement that users[1] can only access resources (e.g. files, applications, networks) they are authorized for. The policy definitions describe which user is authorized for which permission. A permission describes which action (e.g. read, write, execute) can be done on a resource.
There are business, security and regulatory drivers for managing access control policies [19]. The interest of the business is to lowering the costs for managing the permissions for employees and quickly equip employees with necessary permissions such that they can efficiently full-fill their tasks. The security driver is to ensure information security, integrity, and availability to prevent accidental and intentional security breaches. But also to be compliant with regularities, such as the Data Protection Directive in Europe (Directive 95/46/EC)[2], Basel II[3] or company regulations, have an impact on the access control policies.
The National Institute of Standards and Technology (NIST) [11] describes various logical access control models, which provide a policy framework that specifies how permissions are managed and who, under what circumstances, is entitled to which permission and enforce these access control decision. Some of these logical access control models are briefly described in the following to enable the reader to differentiate between the concepts.

- **Identity-based Access Control (IBAC)**
  In IBAC a user gets a certain access to a resource by being assigned directly to a permission, which is connected to a resource. On a low-level Access Control Lists (ACLs) are implementations of IBAC. While IBAC may be manageable in companies with a small amount of users and permissions, the maintenance can

[1]For simplicity the term "User" is used throughout the thesis although the term "subject" would be more general to also include e.g. service accounts.
[2]`http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML` (07-10-2015)
[3]`https://en.wikipedia.org/wiki/Basel_II` (07-10-2015)

be quickly overwhelming in consideration of satisfying all drivers of access control policies (see above).

- **Role-based Access Control (RBAC)**
  In RBAC permissions are bundled to roles, which then are assigned to users. A user gets a certain access to a resource by being assigned to a role, which contains the corresponding permission to that access. In other words users inherit all permissions of the roles they are assigned to. The motivation of this extra abstraction layer of a role is to easier maintain the access control of users. Even more abstraction can be introduced by role hierarchies (see section 4.2.2). The RBAC model, which contains the roles, the user assignments to roles, the permission assignments to roles and the role-hierarchy, needs to be defined before the access control mechanism can enforce the access control. The RBAC model can be an ease of administration in comparison to direct assignments in IBAC. The degree of the benefit is dependent on the RBAC model (see section 4.2).

- **Attribute-based Access Control (ABAC)**
  In [11] the authors try to guide to a standard definition of ABAC, since there seems to be no consensus. The basic concept of ABAC is to introduce an additional abstraction layer in form of ABAC policies, which are basically complex boolean rule sets that can evaluate different kinds of attributes. These attributes could be user information (e.g. department, job title), resource information (e.g. threat level) or environment information (e.g. current time, current location). A user gets a certain access to a resource if the rule set is satisfiable. The rule sets need to be defined before the access control mechanism can enforce the access control. Compared to RBAC ABAC seems more flexible, but it also comes with challenges regarding risk and auditing[1].

The scope of this thesis is to research an approach which outputs an RBAC model. The RBAC model is often used in bigger organizations[19] and is leveraged by many Identity- and Access Management (IAM) systems. Some ideas of the ABAC model are exploited to some extent in the implementation of the thesis approach.

4.2 RBAC Reference Model

There are different functional capabilities of RBAC models, which are used to distinguish different RBAC models. In [23] four conceptual RBAC models are distinguished, which will be introduced in the following.

4.2.1 RBAC$_0$ - Base Model

The base model describes the minimum requirement for an RBAC model. The minimum requirement is that roles are bundles of permissions and users are assigned to these roles in order to get the according permissions. A user can be assigned to several roles and roles can be assigned to several users. The same many-to-many relation exists between roles and permissions. Therefore a user get can get the same permission several times by different roles. Figure 1 demonstrates the relation between users, permissions and roles. Furthermore sessions are also considered in the base model in [23]. A session is a mapping of one user to many roles. With a session a user can activate a subset of roles he or she is assigned to.

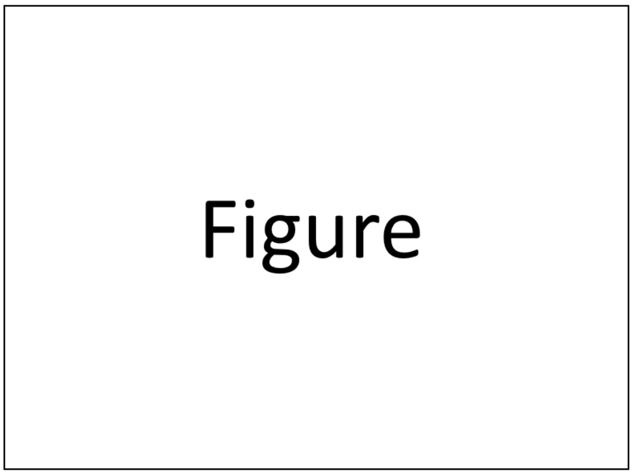


Figure 1: The relation between users, permission and roles in RBAC.

4.2.2 RBAC$_1$ - Role Hierarchies

The RBAC$_1$ model is based on RBAC$_0$ and introduces the concept of role hierarchies. In a role hierarchy roles can be assigned to roles and inherit their permissions to the roles they have been assigned to. The role hierarchy can be restricted as a tree or inverted tree, where a role inherits only the permissions of its child-roles, or a partially ordered set, where a role inherits the permissions from any other junior-role, which is assigned to it. It is also possible to limit inheritance in role hierarchies to control the power of impact of roles. In figure 2 shows one example of a role hierarchy. In [17] role hierarchies and how they can be in conflict with some control principles, such as the Separation-of-Duty (SoD) principle, decentralisation, supervision and review, are discussed.

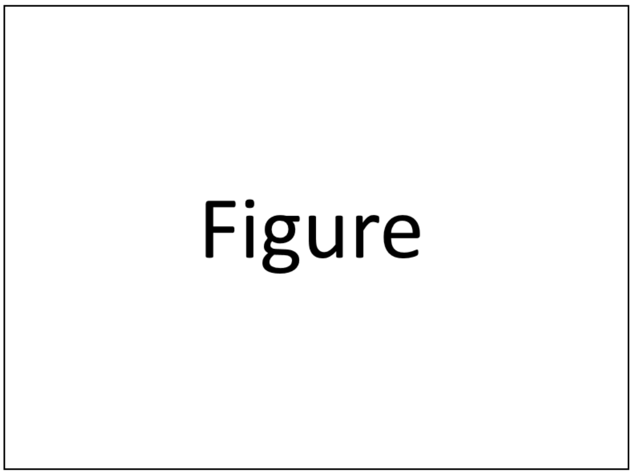


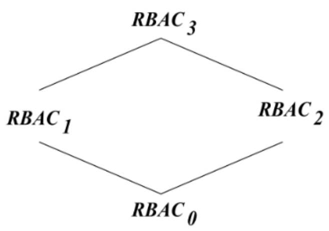
Figure 2: Example of a role hierarchy

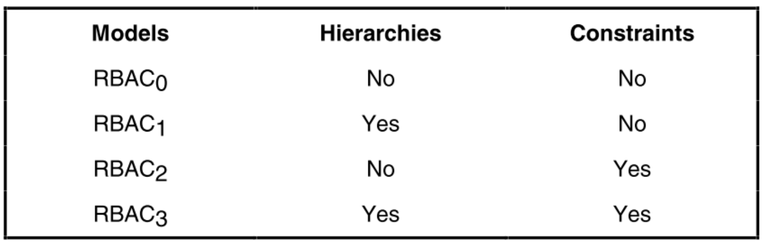
4.2.3 RBAC$_2$ - Constraints

The RBAC$_2$ model is based on RBAC$_0$ and introduces the concept of constraints. With the help of constraints Separation-of-Duty (SoD), also known as Segregation-of-duty, can be achieved. SoD is a security principle for preventing fraud and errors by splitting tasks and associated permissions among multiple users. For example a user who has the permission to request the purchase of goods or services should not have the permission of approving the purchase. Roles in an RBAC model should therefore not have permissions bundled, which violate each other due to SoD. But also user assignments to multiple roles should not violate the SoD requirements.

4.2.4 RBAC$_3$ - Consolidated Model

The consolidated model combines RBAC$_1$ and RBAC$_2$ and therefore supports role hierarchies and constraints. Bringing these two functional capabilities together come with additional functionality. Constraints can also be applied on the role hierarchy. For example, the number of parent roles for child roles can be limited or different child roles can be constrained to have different parent roles.





Models	Hierarchies	Constraints
RBAC$_0$	No	No
RBAC$_1$	Yes	No
RBAC$_2$	No	Yes
RBAC$_3$	Yes	Yes

Figure 3: Relationship among RBAC models[23]

The relation of the four introduced models can be seen in figure 3. In this thesis the main focus will be on the RBAC$_0$ model. Like in [24] and [8] sessions will not be considered for simplicity reasons.
The NIST RBAC model [22] distinguished between four levels of RBAC models: Flat, Hierarchical, Constrained and Symmetric RBAC. Each level introduces an additional functional capability to the previous level. In terms of the NIST RBAC model the Flat RBAC model will be the focus of the thesis.

4.3 Role Model Quality

A basic role model consists of users, permissions, roles, user-role assignments and role-permission assignments. A role model is most desirable if it has a state, which best leverages the RBAC model by satisfying the business, security and regulatory drivers: Lowering the costs for the administration of access control and equip employees with necessary permissions such that they can efficiently full-fill their tasks while keeping security requirements and being compliant. How this goal is measured in the role model has been described by several quality criteria[14][10]. The criteria concern the role model state, individual roles or both. In the following the different criteria is summarized into the categories: Completeness, complexity and comprehension.

4.3.1 Completeness

By completeness it is meant to rebuild the current user-permission assignment state by the role model. When the user-role and the permission-role assignments of the role model are resolved, it should cover all current user-permission assignments.
It is assumed that the current user-permission assignments are in a state, where users get the necessary access to efficiently perform their tasks and no security or compliance regulations are violated. This of course is an ideal situation, but in reality the current state often has quite some noise, especially when no IAM solution has been in place before. Users tend to have more permissions than they actually need, since it is unlikely that someone will claim that he has too many permissions. The measure of completeness of the role model state is therefore in accordance to the initial access control configuration.
A certain amount of overentitlements (users get too many permissions) or underentitlements (users get too few permissions) in the access control policies can be acceptable, if the least privilege principle is too costly to implement in practice.
The combinations of permissions within roles or the combination of user-role assignments might violate security regulations such as SoD. This measure is therefore not only taking individual roles but also the role model state into account.
Constraints could also be ensured in a mechanism posterior to the role model, where individual permissions are detracted from users again, if it violates a constraint. Allowing constraint violations in the role model increases not only the processing and reliance on the posterior mechanism but also the auditing effort.

4.3.2 Complexity

The complexity of a role model is measured in its number of roles and the number of user-role and permission-role assignments. It is often connected to the maintenance costs of a role model.
The more roles the role model has, the more maintenance effort is expected. Hence, a minimal set of roles is preferable. Furthermore it should be obvious that the total number of roles should be smaller than the total number of users or total number of permissions. Otherwise there would be no use of the advantage of having RBAC in

comparison to IBAC in terms of administration costs. When each user has its individual role with the according individual bundle of permissions, the abstraction layer of the role becomes obsolete.
Also the more user-role and permission-role assignments are needed, the more maintenance effort is expected. Large roles with many permissions may reduce the number of user-role assignments, but may lead to more confidentiality/accessibility violations (conflicting Completeness). The same applies for if each role is used by many users. Small roles with few permissions on the other hand can lead to more administration effort as mentioned above, since many roles are necessary for achieving completeness. The same applies if each role is only used by very few users.
A role model probably consists of very general large roles and specialized small roles. Determining a fix boundary of how many permissions or users can be assigned to a role requires knowledge of the role model or is given by company or security regulations.

4.3.3 Comprehension

A recently more discussed topic is the "meaningfulness" of roles [26][10]. It is important that the administrators, which are maintaining the role model, can logically understand the role model for maintaining the roles and assignments confidently. Otherwise it might happen that they avoid to work with the role model since they feel not confident to stay in line with security and compliance regulations. Or it will cause them extra effort and costs to work with the role model. The roles should be therefore comprehensive, which can be achieved by giving them a meaning close to business roles, e.g. a role "Employee", which contains all permissions every employee will get and is assigned to every employee-user.
This criteria can loosen some of the other criteria, which rather concentrate on the compression of the access control information [9]. For achieving more intuitive meaningful roles it might be necessary to allow more roles than the minimal number of roles resulting from compression. More roles might result in more assignments, which are necessary to keep the role model more comprehensive. Lowering costs by having a more comprehensive role model may contradict lowering costs by having a less complex role model.

4.4 Role Engineering

Role engineering [2] describes the process to create a role model for RBAC. This task is proved to be very difficult in large enterprises.
There are three different approaches to conquer the goal of finding the right role model: Top-Down, Bottom-Up and the Hybrid approach[2][10].

- **Top-Down Approach**
  One approach is to do a top-down analysis, where the roles are built out of business information. Business processes, business roles and security policies are analysed to build a suitable role model. The resulting roles contain high-level permissions, which need to be mapped to technical permissions that are used by IT systems. The roles are easy to understand as they are derived from business concepts. The analysis of the business information by experts to a high-level role model and the mapping into low-level accesses by IT Specialists are very time-consuming and costly. Furthermore the resulting role model could lead to a different access control configuration than the current one. It is likely that users get less permissions than they used to have, which might prevent them of doing their tasks efficiently.

- **Bottom-Up Approach**
  The bottom-up approach exploits the current user-permission assignments and tries to gather a role model out of it. Since this approach often uses Data Mining Techniques, the method is also called Role Mining[13]. This approach on the other hand is often failing in generating a comprehensive role model, which is accepted by the administrators[9].

- **Hybrid approach**
  Since the advantages and disadvantages of the top-down and the bottom-up approach are mirrored, hybrid approaches have been suggested [10][16]. In these approaches the business information is leveraged to guide the computational generation of a role model out of user-permission assignments.

4.5 Role Mining

Role Mining is a bottom-up role engineering approach. The basic input of role mining are users, permissions and the current assignments to each other. The input can be expressed as tuple $< U, P, UPA >$, where

- U is a set of users
- P is a set of permissions
- $UPA \subseteq U \times P$ are user-permission assignments

On top additional information can be taken as input like constraints, top-down information (business information) and already existing roles. The output of role mining is the role model. A basic role model state can be described as tuple $< U, P, R, UA, PA >$, where

- U is a set of users
- P is a set of permissions
- R is a set of roles
- $UA \subseteq U \times R$ are user-role assignments
- $PA \subseteq R \times P$ are permission-role assignments

Optionally a role hierarchy is also part of the role model. In some papers, such as [18] and [5], direct user-role assignments ($DUPA \subseteq U \times P$) are added to the role model tuple in order to provide more flexibility to handle anomalous permissions which do not fit into the role structure. Alternatively a new role for each of the individual direct assignments of DUPA can be defined, although such roles are not desirable in a role model. By adding DUPA to the tuple it can be distinguished between DUPA and undiserable roles. Figure 4 shows the inputs and outputs of the role mining process.

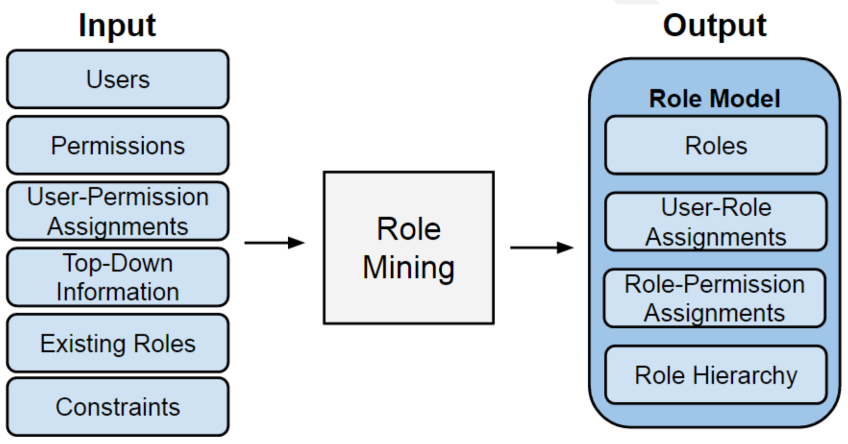


Figure 4: The inputs and outputs of the role mining process

A specific example in matrix representation is shown in figure 5.

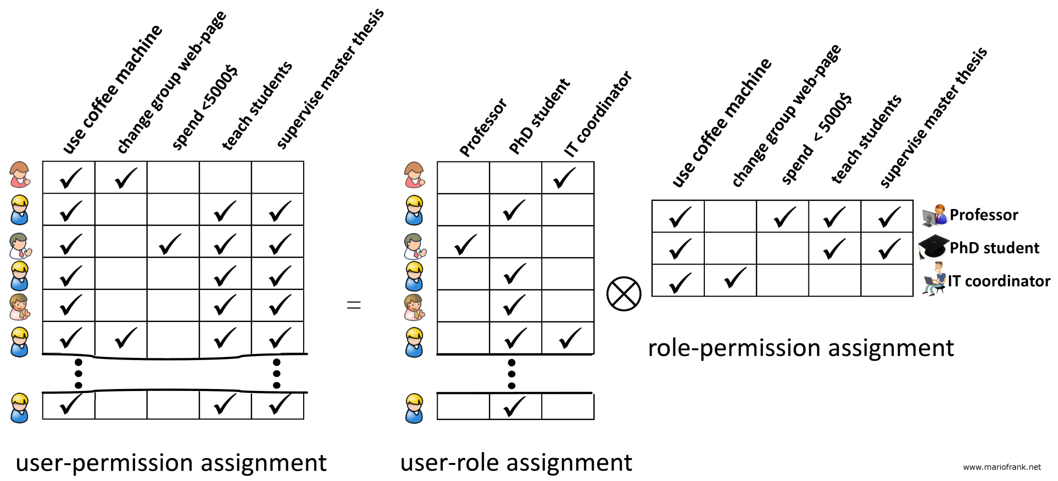


Figure 5: University toy example taken from http://www.mariofrank.net/rolemining.html, which demonstrates the transformation from UPA to $UA \times PA$

4.5.1 Role Mining Problem Definitions

SECTION UNDER CONSTRUCTION

There are several role-mining problem (RMP) definitions existing. The problem definitions, which will be considered in this thesis, are introduced in the following.

- **The Basic Role-Mining Problem**[24]
  "Given a set of users U, a set of permissions P and a user-permission assignment UPA, find an RBAC configuration RC that minimizes the number of roles k and does not deviate from UPA."

- **The δ-approx. Role-Mining Problem**[24]
  "Given a set of users U, a set of permissions P and a user-permission assignment UPA, find an RBAC configuration RC that minimizes the number of roles k and deviates from UPA with less than δ assignments."

- **The Min-Noise Role-Mining Problem**[24]
  "Given a set of users U, a set of permissions P and a user-permission assignment UPA, and the number of roles k, find an RBAC configuration RC with k roles, minimizing the deviation between UPA and RC."

- **The Min-Edge Role-Mining Problem**[15]
  "Given a set of users U, a set of permissions P and a user-permission assignment UPA, find an RBAC configuration RC that is consistent with UPA and minimizes the number user-role assignments and role-permission assignments."

- **The Interference Role-Mining Problem**[8]
  "Let a set of users U, a set of permissions P, a user-permission relation UPA, and, optionally, part of the top-down information TDI be given. Under Assumption 1-3, infer the unknown RBAC configuration $RC = (R*, UA*, PA*)$.*
  Assumptions:
  1. RC$*$ generated UPA
  2. RC$*$ reflects top-down information (TDI).
  3. Exceptions (errors) might exist."

4.5.2 Related Problems

SECTION UNDER CONSTRUCTION
The role-mining problems have related problems like e.g. the Boolean Decomposition Problem (BDM) or the Tiling Problem.

5 Background: Evolutionary algorithms (EAs)

Evolutionary algorithms (EAs) are random searches inspired by the concept of the natural evolution. The general idea is to have a given population of individuals, which is evolving to new fitter generations of the population by natural selection (survival of the fittest). There are different variants of EAs: Genetic algorithms (GA), evolution strategies (ES), evolutionary programming (EP) and genetic programming (GP). All variants follow the same common concept described in the following with differences in technical details such as the representation of individuals[6].
The individuals in a population represent candidate solutions for the problem. A fitness function is evaluating each candidate solution in the population. Candidate solutions with a high-rated fitness are more likely to be chosen to seed the next generation than candidate solutions with low-rated fitness. The next generation is generated by applying variation operators like recombination and mutation on the selected candidate solutions. Recombination is applied to two or more candidate solutions (parents) and result in one or more new candidate solutions (children). During the recombination new candidate solutions are generated by merging random parts of the parents. The mutation operator is applied on only one candidate solution. The output is a copy of the candidate solution, where a random part is changed. The new candidate solutions, which are the output of the variation operators, are called the offspring and compete with the candidate solutions in the current population for a spot in the next generation of the population. In a survival selection the candidate solutions for the next generation are chosen. This process is repeated until a sufficient candidate solution is found or a previously set limit (e.g. number of generations) is reached. A flow chart of an evolutionary algorithm process can be seen in figure 6.

The driving forces of evolutionary algorithms are the variation operators, which are generating new diverse candidate solutions (novelty), and the natural selection, which is guiding to better candidate solutions (quality)[6].
By preserving the possibility that candidate solutions with a lower fitness can be selected as seed for the offspring, the chance of getting into a local optimum should be minimized like in other meta-heuristics.

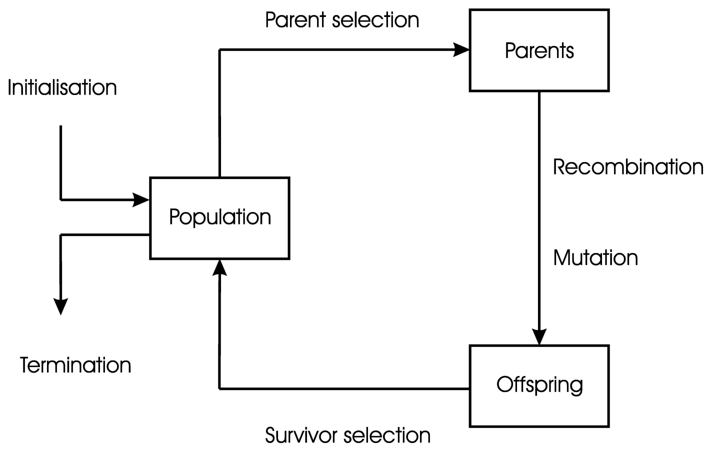


Figure 6: The general scheme of an evolutionary algorithm as a flow-chart [6]

5.1 Components of Evolutionary Algorithms

An EA is defined by its components, procedures and operators, which will be introduced in the following.

- **Individuals**
  The individuals of a population in EA represent candidate solutions. The candidate solutions within the original problem context are called phenotypes, while their representation in the EA are called genotypes or chromosomes. A building block of a chromosome is called a gene, where the possible values for a gene are called alleles. The mapping from the phenotype to the genotype is called encoding and the inverse mapping is called decoding. A representation could be for example a binary string or a string with real values. Both also more complex structures could be the right representation. Finding the right representation of the phenotypes is problem-specific and a difficult task. It is crucial for the success of the EA. The representation influences the variation operators.

- **Population**
  The population is a a list of individuals (genotypes), which can occur several times within the list. The individuals are static objects, which do not change. The population is dynamic, which will change over time due to the exchange of individuals.

The parent selection of the individuals, which will be the seed for the offspring, is mostly carried out on the whole population size. In most EAs the population size remains constant. The diversity of a population describes the measure of how many different individuals are present. The measure can be based on the fitness values, the phenotypes or the genotypes of the individuals. For example two individuals can represent two different phenotypes, but are evaluated with the same fitness value.

The first population consists of randomly generated individuals or of chosen individuals with higher fitness.

- **Evaluation Function (or Fitness function)**
  The evaluation function evaluates the individuals of a population and assigned the fitness of a candidate solution. Since the fitness influences the selection of an individual, the evaluation function encourages improvements. The goal could be to minimize or to maximize the fitness of individuals. Mathematically a minimization function can be easily transformed to a maximization problem and vice versa. The evaluation function in an EA is constructed from the objective function in the phenotype space.

- **Parent Selection**
  The parent selection is the selection of individuals, which will be the seed for the next generation. Typically individuals with a better fitness value get more often selected to further improve the individuals. But also individuals with a low quality fitness get a chance to pass on their genes into the next generation. This ensures that the search is not too greedy and get into a local optimum. The balance between parents with a high-quality fitness and parents with a low-quality fitness is often probabilistic.

- **Variation Operators (Recombination and Mutation)**
  The variation operators are responsible for discovering the search space by creating new individuals on the bases of existing individuals in the current population. All newly created individuals of a generation is called the offspring.
  There are different types of variation operators. While the mutation oparator only takes one individual as input, the recombination (or crossover) operator takes at least two individuals as input. The mutation operator creates a new individual

(mutant), which slightly differs from the input-individual (original). The change from the original to the mutant is chosen randomly. If the change is not random but rather guided, it is defined as an heuristic operator. The recombination operation is creating one or more new individuals (children) from its parent individuals by mixing randomly genes of these. A child might have a lower, equal or higher fitness value than its parents depending on the combination of genes.
Variation operators are depending on the representation of the phenotypes.

- **Survivor Selection (Replacement)**
  The survivor selection is the replacement strategy of the population and happens after the creation of the offspring. The current population is replaced by a new population - the new generation - which is mostly the same size as the population, which is being replaced. Like in the parent selection, the selection is based on the fitness values of the individuals and is often deterministic. In a fitnesss-biased selection for example the individuals of the population and the offspring are sorted by their fitness values and then the top segment is selected for the next population generation. In an age-biased selection only the individuals from the offspring are considered for the new population generation. If the current fittest individual of a population is kept in the next population, the concept of elitism is used.

- **Termination Condition**
  The EA is either terminated when one individual is reaching a known optimal fitness value or when predefined computation conditions are met. These conditions can be for example the number of generations, number of fitness evaluations, maximum allowed CPU time or a threshold under which the population diversity has to fall.

5.2 Multiobjective Evolutionary Algorithms (MOEAs)

The solution of an optimization problem might not only be measured regarding one objective but several, possibly conflicting objectives. These problems are called multiobjective problems (MOPs). It is unlikely that there is one optimal solution for such kind of problems. It is rather likely to have a set of optimal solutions also known as Pareto-optimal solutions.
An example of a MOP could be buying an airplane ticket for a flight from city X to destination Y, where the buyer takes several factors into account. For simplification lets

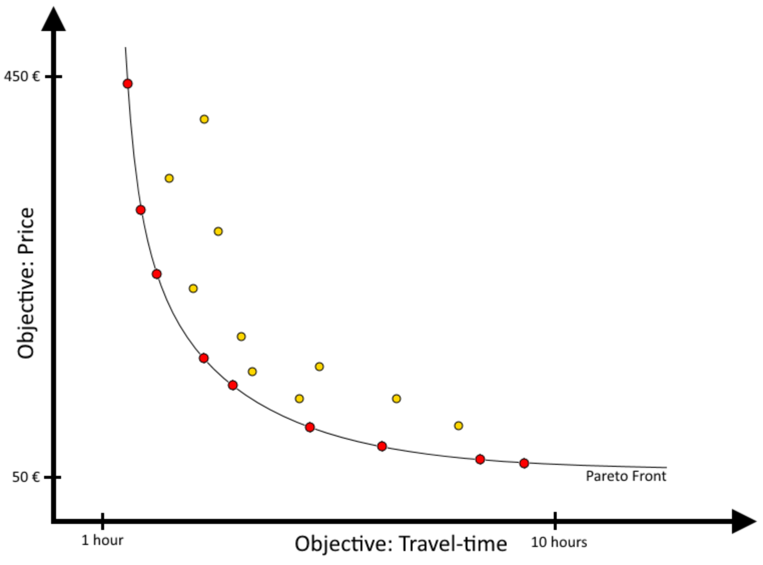


Figure 7: MOP Example: Buying an airplane ticket for a flight from X to Y with the objectives price and travel-time. A dot represents an offer from an airline. The red dots represents the optimal offers regarding the objectives, while the yellow dots are suboptimal offers.

say the factors are price and travel time, where the rule is: The shorter the travel time, the higher the price. The buyer wants the cheapest airplane ticket and the shortest travel time, so he probably has to make a compromise between these two factors. There is a set of solutions for the two objective problem: From cheap long-travel to expensive short-travel solution. The graph in figure 7 demonstrates the different options and outlines the set of pareto-optimal solutions on the so-called Pareto Front. The buyer can pick one of the cheaper solutions, but has to make a trade-off regarding the travel-time.

The fitness values of several objectives can also be combined in one single objective function, where the fitness values for each objective are weighted. This approach is also called scalarisation[6]. The weights have to be pre-determined and adjust the preference of an objective. Sometimes these preferences are not known beforehand. This is where multiobjective optimization approaches become interesting.

The goal in multiobjective optimization is to find all pareto-optimal solutions. The strength of EAs can be seen in solving MOPs. An EA can find several pareto-optimal solutions in one single run, since it works with a population of solutions simultaneously[4].

Why do I use MOEA this in my thesis??

5.2.1 Nondominated Sorting Genetic Algorithm II (NSGA-II)

One EA approach to multiobjective Optimization is the nondominated sorting genetic algorithm II (NSGA-II)[4].
First find argumentation why you do MOEA on your domain

5.3 Co-Evolution

SECTION UNDER CONSTRUCTION
A special form of evolution in EAs is co-evolution. In co-evolution several separate populations are influencing the fitness of the individuals of each other.

5.3.1 Symbiotic, Adaptive Neuro-Evolution (SANE)

Results not good enough

5.3.2 Enforced Sub-Populations (ESP)

6 Related Work

6.1 Role Mining with Data Mining

SECTION UNDER CONSTRUCTION

Role Mining has been first coined in [13]. In the following years several researchers have analyzed Role Mining further and defined several Role Mining Problems, Quality Measures, Cleaning Techniques and Algorithms.

6.2 Role Mining of "Meaningful Roles"

SECTION UNDER CONSTRUCTION

In the recent years several researchers are investigating the problem finding "meaningful" roles, since the classic Role Mining approaches outputs RBAC models, which are often not accepted in practice.
[26]

6.3 Role Mining with Bio-inspired Techniques

SECTION UNDER CONSTRUCTION

In [20] and [21] the Basic RMP and the Min-Edge RMP are tackled with a genetic algorithm (GA). For the evaluation function of the GA the authors combine several objectives in one single objective maximization function, where the fitness values for each objective are weighted. The objectives for the Basic RMP are the minimization of roles, confidentiality violations and availability violations. A confidentiality violation expresses a situation where a user would get an access right due to the generated role model, which he or she did not had in the current access control configuration. An availability violation on the other hand expresses if a user gets an access right less than he or she did had in the current access control configuration. For the Min-Edge RMP the objective of minimizing roles is exchanged with minimizing the number of user-role- and role-permission assignments.
In a first version of the approach the authors choose a representation of role models as individuals consisting of three chromosomes: A chromosome X, which represents the UA-Matrix, a chromosome Y, which represents the PA-Matrix, and a control-chromosome Z, which controls if a role is active or passive. With the control-chromosome they influence how many roles a role model has and make it possible to vary the amount with

variation operators of the GA. The authors suggest a crossover function in three phases, which is particularly designed for the suggested multi-chromosomal representation. The drawback of this approach are unnecessary passive genes. The chromosomes X and Y, respectively the UA- and PA-Matrix, contain roles, which are passive (controlled by chromosome Z).
Due to this drawback the authors came up with an improved representation where they combine the X and Y chromosome into one variable-length chromosome and get rid of the control-chromosome Z. A role model is now representation as chromosome where a gene represents a role. A gene (role) consists of a user-list (L_X) and a permission-list (L_Y). Due to the change of representation the crossover method is changed accordingly. As shown in figure 8 the author suggest to use a traditional one-point crossover, where the points of crossover of two individuals are randomly chosen. After the crossover operation a local optimization of offspring chromosomes is executed, where genes with same user-lists or permission-lists are combined.

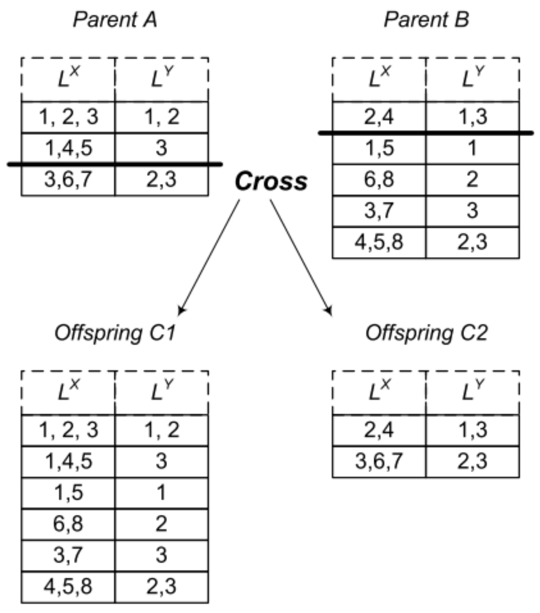


Figure 8: Examples of the crossover in the improved GA for the RMP by [21]

In [20] the first approach is tested on randomly generated data with three different dimensions. The evaluation is on the number of generations needed to obtain a solution

which meet the suggested evaluation function. In [21] the performance of the two approaches are compared with the result that the improved GA has a better performance in all dimensions of UPA, especially in greater dimensions. In the recent paper [12] the authors focusing on the multi-chromosomal approach again.
In this thesis the suggested improved approach is used as starting point in my approach to deal with the basic RMP.

In [5] the authors propose a genetic algorithm (GA)-based and an ant-colony-optimization (ACO)-based algorithm for the ideas in [26] (see section 6.2), where a previously mined set of candidate roles is optimized. The focus is on the objectives of the basic RMP and on the interpretability of roles, where a role is considered meaningful if its assigned users can be characterized by an expression of user attributes. The goal is finding optimal candidate roles for an $RBAC_1$ role model (role model with role hierarchy).
In the GA-based algorithm a set of all candidate roles is reduced by repeatedly removing roles, while in the ACO-based algorithm it is started with an empty candidate role set, where roles are repeatedly added.

7 Solution Approaches

SECTION UNDER CONSTRUCTION

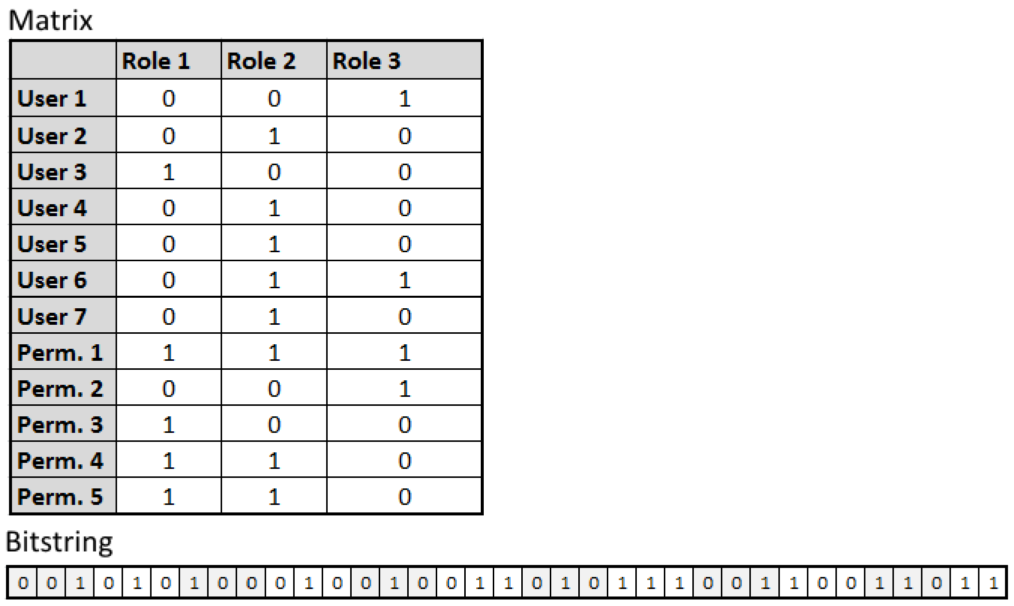
Implementation with the DEAP Framework for Python [3]

7.1 Evolutionary algorithm for the Basic Role-Mining Problem

SECTION UNDER CONSTRUCTION

7.1.1 Representation of individuals

In a first attempt I tried to represent an RBAC model as individual with a bit-string. The bit-string is derived from a combined UR- and RP-Matrix. Figure 9 shows an example. The motivation of this initial idea was that the mutation operator can be easily executed by just flipping a bit within the bit-string. However, the problem with this representation is that the number of roles needs to pre-defined.



Matrix

	Role 1	Role 2	Role 3
User 1	0	0	1
User 2	0	1	0
User 3	1	0	0
User 4	0	1	0
User 5	0	1	0
User 6	0	1	1
User 7	0	1	0
Perm. 1	1	1	1
Perm. 2	0	0	1
Perm. 3	1	0	0
Perm. 4	1	1	0
Perm. 5	1	1	0

Bitstring

0	0	1	0	1	0	1	0	0	0	1	0	0	1	0	0	1	1	0	1	0	1	1	1	0	0	1	1	0	0	1	1	0	1	1

Figure 9: *Example of a combined UR- and RP-Matrix with its bit-string representation*

In [21] two different representations are introduced. In the first representation the authors suggest to build an individual out of three chromosomes X, Y and Z. Chromosome X describes the UR-Matrix, chromosome Y describes the RP-Matrix and chromosome Z is a vector, which marks roles as active or passive. With this representation the number of roles becomes part of the search. An example can be seen in figure **??**. The drawbacks of this representation are unnecessary genes (roles) and complex crossover operations[21]. The second improved representation eliminates these drawbacks by having one chromosome for an individual, where no unnecessary genes occur. The chromosome consists of a list of roles, which contain a user-set and a permission-set (see figure **??**).
Due to the advantages of the last mentioned representation, this is chosen to be the representation for the research of single-objective EAs for solving the RMP.

Description of data structure

7.1.2 Initialization of the Population

The initial population is randomly generated by creating chromosomes (RBAC models) of random gene size (role size), which contain a random set of users and permissions. It is ensured that each role has at least one user and permission assigned.

7.1.3 Mutation methods

For the mutation six different mutation types are implemented, which are randomly executed.

- 1. Add a new role
- 2. Remove a role
- 3. Add a user to a role
- 4. Remove a user from a role
- 5. Add a permission to a role
- 6. Remove a permission from a role

Pseudocode?

7.1.4 Crossover methods

The recombination is executed on two individuals with a traditional one-point crossover. In the crossover operation the roles of the parents are merged to new individuals.
Pseudocode?

7.1.5 Evaluation functions

Several objectives of the role mining problem are separately implemented as evaluation function and their impact on the EA is analyzed. In [21] a weighted function is suggested which takes the number of roles, access-violations and availability-violations into account. The function is formulated as maximization function. Another commonly used objective function, the weighted structural complexity, is also implemented and compared with the first function. Different weights for the functions are tested and reveal that the objective of minimizing the number of roles has a strong influence.
To analyze the different objectives a MOEA has been implemented. The NSGA-II algorithm is applied and reveal the algorithm is not good in handling individuals with the same fitness. The Fortin improvement is applied. Since the objective of number of roles has a too strong impact, a NSGA-II algorithm with weights is implemented.

Pseudocode?

- NSGA2
  Why? The higher the role number (1 Role for each user), the more likely it is to have no violations. The lower the role number, the more violations
- Improved NSGA2 (Fortin)
  Why? Different Individuals have same fitness
- Weighted NSGA2
  Why? 2nd objective is less important
  Issue? Skipped fronts, no symmetry in domination matrix

7.1.6 Selection mechanisms

Which selection mechanisms are chosen is dependent on if a single-objective or a multi-objective EA is applied.

For the parent selection ...

For the survivor selection ...

DEAP Build-in selection methods

Improvements of DEAP Build-in selection methods

Pseudocode?

7.2 Evolutionary algorithm for the Inference Role-Mining Problem

SECTION UNDER CONSTRUCTION

Not implementation yet

7.3 Alternative approach with Co-Evolution

SECTION UNDER CONSTRUCTION

Instead of that individuals represent RBAC models, the individuals represent roles. The motivation behind this approach is to involve human interaction in the survival selection of the EA. Since a whole RBAC model can be hardly evaluated at once, the individuals have to be a smaller fraction of the RBAC model.

7.3.1 Symbiotic, Adaptive Neuro-Evolution (SANE)

7.4 Human interaction

SECTION UNDER CONSTRUCTION

Not implemented

8 Experiments

SECTION UNDER CONSTRUCTION
This chapter describes the setup of the experiments executed. First the data sets used in the experiments are described. Then the measures and the setup of the experiments are introduced.

8.1 Data Sets

8.1.1 Synthetic Datasets

SECTION UNDER CONSTRUCTION
Some papers are providing data generators for synthetic datasets for the performance evaluation of the various role mining algorithms. The data generator in [25] takes as input the number of users, permissions and roles to generate a pair of User-Role-Assignment and Role-Permission-Assignment matrices. Combining these two matrices, the corresponding User-Permission-Assignment matrix is obtained. In the data generator introduced in [Lu et al. 2014]...
"In [Lu et al. 2014], several sets of synthetic data of varying parameters are used. The data gen- erator takes as input the number of users and permissions, density of 1s in the PA, density of 1s in the UA and the noise level to produce the UA and PA matrices."

These data generators only consider users, permissions and their assignment to each other. They do not consider user information. Due to this a new synthetic data generator was created for this thesis, which is described in the following. The data generator

8.1.2 Real Datasets

SECTION UNDER CONSTRUCTION
In many research papers the same datasets are used for performance evaluation of role mining Algorithms. Table 1 lists some of these data sets and their components.

The authors of [7] obtained these datasets from Cisco firewalls and the Lotus Domino server of the Hewlett Packard (HP) networks. The healthcare dataset was collected from the US Veteran's Administration.

Also see [26]

Algorithm 1 Algorithm for creating an synthetic dataset for testing of role mining algorithms

```
1: procedure CreateRoles($roleCnt$,$permissionCnt$,$maxPermissionForRole$,$maxPermissionUsage$
2:     $roles$ = [ ] for each $roles[r]$ with $r = 0..roleCnt$-1
3:     $permissionBucket = maxPermissionUsage$ for each $permissionBucket[p]$ with $p = 0..permissionCnt$-1
4:     for $role$ in $roles$ do
5:         $permissionForRoleCnt$ = pick randomly between $1..maxPermissionForRole$
6:         $tempPermissionBucket = permissionBucket$ - $role$
7:         if length of $tempPermissionBucket > 0$ then
8:             $permission$ = draw randomly number of $permissionForRoleCnt$ out of $tempPermissionBucket$
9:             $role$.add($permission$)
10:        else
11:            Error
12:        end if
13:    end for
14:    return $roles$
15: end procedure
16:
17: procedure CreateUsers($userCnt$,$attribute$)
18:    $users$ = [ ] for each $users[u]$ with $u = 0..userCnt$-1
19:    for $user$ in $users$ do
20:        for $a$,$attrValues$ in $attributes$ do
21:            $user[a]$ = pick randomly a value of $attrValues$
22:        end for
23:    end for
24:    return $users$
25: end procedure
26:
27: procedure CreateRules($roleCnt$,$attributes$,$maxRuleConditionCnt$)
28:    $rules$ = [] for each $rules[r]$ with $r = 0..roleCnt$-1
29:    for $rule$ in $rules$ do
30:        $selectedAttributes$ = pick $maxRuleConditionCnt$ random $attributes$
31:        for $a$,$attrValues$ in $selectedAttributes$ do
32:            $rule[a]$ = pick randomly values of $attrValues$
33:        end for
34:    end for
35:    return $rules$
36: end procedure
```

[HTML]EFEFEF **Dataset**	**Users**	**Permissions**	**User Permission Assignments**
healthcare	46	46	1486
domino	79	231	730
emea	35	3046	7220
apj	2044	1146	6841
firewall-1	365	709	31951
firewall-2	325	590	36428
americas-small	3477	1587	105205
americas-large			

Table 1: Real Datasets

8.2 Visualisation of the RBAC Model

SECTION UNDER CONSTRUCTION

8.3 Experiments with Single-Objective EAs

SECTION UNDER CONSTRUCTION

8.3.1 Setup

- Number of turns
- Random Start-population
- Fixed Start-population

8.3.2 Measures

- Fitness (Min, Max, Avg)

8.4 Experiments with Multi-Objective EAs

8.4.1 Setup

8.4.2 Measures

8.5 Experiments with Co-Evolution

8.5.1 Setup

8.5.2 Measures

9 Results and Evaluation

SECTION UNDER CONSTRUCTION

10 Discussion and Future Work

SECTION UNDER CONSTRUCTION

11 Conclusion

SECTION UNDER CONSTRUCTION

12 Bibliography

[1] Ed Coyne and Timothy R. Weil. Abac and rbac: Scalable, flexible, and auditable access management. *IT Professional*, 15(3):14–16, 2013.

[2] Edward J Coyne, Timothy R Weil, and Rick Kuhn. Role engineering: Methods and standards. *IT Professional*, (6):54–57, 2011.

[3] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: A python framework for evolutionary algorithms. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 85–92, New York, NY, USA, 2012. ACM.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comp*, 6(2):182–197, April 2002.

[5] Xuanni Du and Xiaolin Chang. Performance of AI algorithms for mining meaningful roles. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*, pages 2070–2076, 2014.

[6] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* SpringerVerlag, 2003.

[7] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, pages 1–10, New York, NY, USA, 2008. ACM.

[8] Mario Frank, Joachim M. Buhman, and David Basin. Role mining with probabilistic models. *ACM Trans. Inf. Syst. Secur.*, 15(4):15:1–15:28, April 2013.

[9] Mario Frank, Joachim M. Buhman, and David Basin. Role mining with probabilistic models. *ACM Trans. Inf. Syst. Secur.*, 15(4):15:1–15:28, April 2013.

[10] Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. A probabilistic approach to hybrid role mining. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 101–111, New York, NY, USA, 2009. ACM.

[11] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Arthur R. Friedman, Alan J. Lang, Margaret M. Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, and Scarfone Cybersecurity. Guide to attribute based access control (abac) definition and considerations (draft), 2013.

[12] Igor Kotenko and Igor Saenko. Improved genetic algorithms for solving the optimisation tasks for design of access control schemes in computer networks. *Int. J. Bio-Inspired Comput.*, 7(2):98–110, May 2015.

[13] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, SACMAT '03, pages 179–186, New York, NY, USA, 2003. ACM.

[14] Michael Kunz, Ludwig Fuchs, Michael Netter, and Günther Pernul. Analyzing quality criteria in role-based identity and access management. In *Proceedings of the 1st International Conference on Information Systems Security and Privacy*, pages 64–72, 2015.

[15] Haibing Lu, J. Vaidya, and V. Atluri. Optimal boolean matrix decomposition: Application to role engineering. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 297–306, April 2008.

[16] S. Mandala, M. Vukovic, J. Laredo, Yaoping Ruan, and M. Hernandez. Hybrid role mining for security service solution. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 210–217, June 2012.

[17] Jonathan D. Moffett. Control principles and role hierarchies. In *Proceedings of the Third ACM Workshop on Role-based Access Control*, RBAC '98, pages 63–69, New York, NY, USA, 1998. ACM.

[18] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with semantic meanings. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, pages 21–30, New York, NY, USA, 2008. ACM.

[19] Alan C O'Connor and Ross J Loomis. 2010 economic analysis of role-based access control. *NIST, Gaithersburg, MD*, 20899, 2010.

[20] Igor Saenko and Igor Kotenko. Genetic algorithms for role mining problem. In *Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, PDP '11, pages 646–650, Washington, DC, USA, 2011. IEEE Computer Society.

[21] Igor Saenko and Igor Kotenko. Design and performance evaluation of improved genetic algorithm for role mining problem. In *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on*, pages 269–274. IEEE, 2012.

[22] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The nist model for role-based access control: towards a unified standard. In *ACM workshop on Role-based access control*, volume 2000, 2000.

[23] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, February 1996.

[24] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: Finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, SACMAT '07, pages 175–184, New York, NY, USA, 2007. ACM.

[25] Jaideep Vaidya, Vijayalakshmi Atluri, and Janice Warner. Roleminer: Mining roles using subset enumeration. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 144–153, New York, NY, USA, 2006. ACM.

[26] Zhongyuan Xu and Scott D. Stoller. Algorithms for mining meaningful roles. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, pages 57–66, New York, NY, USA, 2012. ACM.

13 Appendix