

TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

ITI70LT

Mai Kraft 111605IVCM

**PERFORMANCE ANALYSIS OF ATTACKER PROFILING
IN QUANTITATIVE SECURITY RISK ASSESSMENT**

Master thesis

Aleksandr Lenin

M.Sc

Researcher

Tallinn 2014

Declaration

Hereby I declare that I am the sole author of this thesis. The work is original and has not been submitted for any degree or diploma at any other University. I further declare that material obtained from other sources has been duly acknowledged in the thesis.

.....

(date)

.....

(signature)

Annotation

We are surrounded by information systems everywhere. Today people depend on them as never before. Due to the fact that threats and attacks on information systems have become massive, their owners have to apply security measures to protect their property. This is very expensive and therefore the threats need to be accurately assessed in order to protect systems without overspending on them. Nowadays, it is hard to quantify how difficult it would be to attack the information systems. Thus, it would be very helpful if there existed an appropriate conceptual framework that accurately assessed system's security measures.

The attack tree analysis is one method attempting to solve this problem. Attack trees provide a formal and methodical way of describing possible attack scenarios in the considered environment. Attacks against a system are represented in a tree structure, where the goal of the attacker is the root node and leaf nodes are different ways of achieving the goal. Two types of refinements are commonly used: the AND- refinement where all sub-attacks must be satisfied in order to satisfy the root goal and the OR-refinement where any of the sub-attacks is sufficient to satisfy the goal.

This thesis studies the ApproxTree tool introduced by Jürgenson-Willemsen [1] and the ApproxTree+ tool proposed by Lenin et al.[2]. The aim of this thesis is to study the profiling effect on the genetic algorithm performance. Firstly, the hypothesis was validated whether profiling introduces any significant performance penalty and if the profiling can be integrated into existing risk assessment tools. Secondly, it was observed whether the genetic algorithm parameters that are optimal for ApproxTree are also optimal for the ApproxTree+ approach. As the current ApproxTree+ approach has some shortcomings, an improvement how to make this model more reliable and the computational method faster was proposed.

Annotatsioon (in Estonian)

Infosüsteemid ümbritsevad meid igalt poolt. Tänapäeva inimene on sellest sõltuv rohkem kui kunagi varem. Seetõttu on ohud ja rüüanded infosüsteemidele muutunud ka massiliseks ning infosüsteemide omanikud ja haldajad peavad rakendama turvameetmeid, et oma vara kaitsta. Turvameetmete rakendamine on kallis ning olemaks vähegi kuluefektiivne peab oskama hästi hinnata oma süsteemide turvalisust ja vastupidavust ründajate tegevusele. Selleks oleks aga vaja sobivat raamistikku, mis aitaks hinnata kui turvaline on süsteem erinevate rünnete vastu.

Ründepuu analüüs on küll küllaltki noor teadusvaldkond, kuid siiski proovib leida viise, kuidas hinnata infosüsteemide turvalisust. Ründepuu esitatakse puukujulises struktuuris kõige tipus ründaja põhieesmärk, mis toob talle materiaalselt kasu. Ründepuu lehed tähistavad elementaarründeid, mida enam väiksemateks rünneteks jaotada ei ole otstarbekas. Ründepuude meetodika populariseeriti 1999 aastal Bruce Schneieri poolt ning kuni tänaseni on aktiivne teadusuuringute objekt.

Käesolevas magistritöös uuritakse lähemalt Jürgenson-Willemsoni poolt välja töötatud ApproxTree ja Lenin-Willemsoni ApproxTree+ mudelit. Töö eesmärgiks oli uurida, kuidas mõjutab ründaja profiili integreerimine olemasolevatesse ründepuu arvutamismeetoditesse geneetilise algoritmi jõudlust. Valideeritakse hüpoteesi, kas profileerimise kasutamine suurendab geneetilise algoritmi arvutuste mahtu. Lisaks hinnatakse, kas profileerimiseta geneetilise algoritmi jaoks valitud parameetrid on sobilikud ka profileerimisega geneetilise algoritmi jaoks. Olemasoleval profileerimisega geneetilisel algoritmil on mõningad puudujäägid. Töös pakutakse välja lahendus, kuidas neid puudujääke kõrvaldada ning muuta profileerimisega geneetiline algoritm täpsemaks.

ANNOTATION	3
ANNOTATSIOON (IN ESTONIAN).....	4
LIST OF TABLES	6
LIST OF ABBREVIATIONS AND SYMBOLS	7
1 INTRODUCTION	8
2 INTRODUCTION TO ATTACK TREE ANALYSIS	10
2.1 ATTACK TREES FOR MODELING SECURITY	10
2.2 PARALLEL ATTACK TREE MODEL	14
2.3 OPTIMIZATIONS USING THE GENETIC ALGORITHM	16
2.4 ATTACKER PROFILING IN ATTACK TREE ANALYSIS	17
2.5 APPROXTREE+ TOOL	19
3 CASE STUDY: STEAL SENSITIVE INFORMATION BY COLLECTING NETWORK TRAFFIC OF AN ENTERPRISE	22
3.1 THE ATTACK TREE MODEL	24
3.2 ESTIMATED VALUES FOR THE ATTACK TREE LEAF NODES	30
4 ASSESSMENT OF ATTACKER PROFILING EFFICIENCY	34
4.1 PERFORMANCE ANALYSIS.....	39
4.2 IMPROVEMENT OF THE APPROXTREE+ METHOD.....	41
5 CONCLUSIONS AND FUTURE RESEARCH.....	44
REFERENCES.....	46

List of Figures



FIGURE 1. ATTACK TREE REPRESENTED BY B. SCHNEIER IN [5]	11
FIGURE 2. ATTACK TREE TAKING INTO ACCOUNT THE DIFFICULTY AND COST OF ATTACK BY B. SCHNEIER IN [5].....	11
FIGURE 3. THE INTERCONNECTION OF AN IP PBX SYSTEM WITH ITS COMPONENTS DESCRIBED IN [11]	23
FIGURE 4. MAIN GOAL OF THE ATTACKER: “STEAL SENSITIVE INFORMATION BY COLLECTING NETWORK TRAFFIC OF THE ENTERPRISE”	24
FIGURE 5. OR-REFINEMENT “GET ACCESS TO THE NETWORK TRAFFIC”	25
FIGURE 6. AND-REFINEMENT “GET IN THE NETWORK PATH”	25
FIGURE 7. OR-REFINEMENT “COMPROMISE LOCAL SYSTEM”	26
FIGURE 8. AND-REFINEMENT “INSTALL MALWARE TO THE ENTERPRISE’S COMPUTER”	26
FIGURE 9. AND-REFINEMENT “INSTALL MALWARE TO THE IP PBX”	26
FIGURE 10. AND-REFINEMENT “SOCIAL ENGINEER EMPLOYEE TO COLLECT NETWORK TRAFFIC”	27
FIGURE 11. AND-REFINEMENT “BRIBE EMPLOYEE TO GET TRAFFIC”	27
FIGURE 12. AND-REFINEMENT “THREATEN EMPLOYEE TO GET THE TRAFFIC”	28
FIGURE 13. AND-REFINEMENT “COLLECT DATA”	28
FIGURE 14. AND-REFINEMENT “DECODE MEDIA TRAFFIC”	28
FIGURE 15. ATTACK SUITE COMPUTED BY USING THE APPROXTREE METHOD. AND-REFINEMENT “STEAL SENSITIVE INFORMATION BY COLLECTING NETWORK TRAFFIC OF AN ENTERPRISE” AND “COLLECT DATA” AND “DECODE MEDIA TRAFFIC” AND-REFINEMENTS	35
FIGURE 16. ATTACK SUITE COMPUTED BY USING THE APPROXTREE METHOD. AND-REFINEMENT “STEAL SENSITIVE INFORMATION BY COLLECTING NETWORK TRAFFIC OF AN ENTERPRISE” AND “GET ACCESS TO THE NETWORK TRAFFIC” OR-REFINEMENT	36
FIGURE 17. COMPUTED ATTACK SUITE WITH APPROXTREE METHOD OF AND-REFINEMENT “GET IN THE NETWORK PATH”	36
FIGURE 18. COMPUTED ATTACK SUITE WITH THE APPROXTREE METHOD OF REFINEMENT “COMPROMISE LOCAL SYSTEM”	37
FIGURE 19. COMPUTED ATTACK SUITE WITH THE APPROXTREE METHOD OF AND-REFINEMENT “SOCIAL ENGINEER EMPLOYEE TO COLLECT NETWORK TRAFFIC”	37
FIGURE 20. COMPUTED ATTACK SUITE WITH THE APPROXTREE METHOD OF AND-REFINEMENT “THREATEN EMPLOYEE TO GET THE TRAFFIC”	37
FIGURE 21. INITIAL POPULATION SIZE’S EFFECT ON THE CONVERGENCE SPEED (# OF GENERATIONS).....	40
FIGURE 22. MUTATION RATE’S EFFECT ON THE CONVERGENCE SPEED (# OF GENERATIONS)	40
FIGURE 23. GENERATIONS’ EFFECT ON CONVERGENCE SPEED (# OF GENERATIONS)	41
FIGURE 24. EXECUTION TIME OF ATTACK TREES WITH DIFFERENT SIZE.	42
FIGURE 25. THE ATTACK TREE BEFORE APPLYING THE ATTACKER PROFILE ON THE LEFT AND THE ATTACK TREE AFTER APPLYING ATTACKER PROFILE ON THE RIGHT.	43

List of Tables

TABLE 1. ATTACKER PROFILING PARAMETERS.....	18
TABLE 2. PARAMETERS FOR DESCRIBING THE ATTACK TREE “STEAL SENSITIVE INFORMATION BY COLLECTING NETWORK TRAFFIC OF THE ENTERPRISE” ELEMENTARY ATTACKS.....	30
TABLE 3. ATTACK TREE "STEAL SENSITIVE INFORMATION BY COLLECTING NETWORK TRAFFIC OF AN ENTERPRISE" LEAF NODE'S ESTIMATED PARAMETERS	31
TABLE 4. ATTACKER PROFILES	33
TABLE 5. CALCULATION RESULTS OF THE APPROXTREE+ METHODS FOR ATTACKER PROFILES 1, 2, 3, 4, 5	38

List of Abbreviations and Symbols

Abbreviation	Description
SAT	Satisfiability problem of Boolean formulas
PDAG	Propositional directed acyclic graph
IP PBX	IP based telephone switching system within the enterprise to route internal and external calls
VoIP	Voice over IP

Symbol	Definition
\mathcal{T}	Attack tree \mathcal{T} with AND- and OR- refinements and set of leaves $\mathcal{X}=(\mathcal{X}_1,\dots, \mathcal{X}_n)$ and parameter Gains
\mathcal{F}	Boolean formula that describes the attack tree \mathcal{T}
\mathcal{X}_i	Leaf (Elementary attack) \mathcal{X}_i of attack tree \mathcal{T}
$Expenses_i$	Expected cost of launching the elementary attack \mathcal{X}_i (includes preparation costs and expected penalties)
p_i	Probability of succeeding when performing elementary attack \mathcal{X}_i
$Gains$	Reward of an attacker if the attack tree \mathcal{T} is realized
S	Attack suite S of the elementary attacks in the attack tree \mathcal{T}
$Outcomes_S$	Outcome value of the attack suite S
\mathcal{P}_S	Probability of $\mathcal{F}=\text{true}$ after executing the attack suite S
	AND-node represented with ADTool [3]
	OR-node represented with ADTool [3]
*	Boolean function conjunction operator
+	Boolean function disjunction operator

1 Introduction

Computer networks and systems are ubiquitous in our everyday life and are the core of modern communication. As today people depend on information systems more than ever before, information security has become very important. Information and information systems have to have certain levels of confidentiality, integrity, and availability to meet the targets of the system owners and users. The fact that computer networks around the world are constantly probed and attacked with the purpose to violate the security defenses and gain access to information shows that people who maintain those networks have to find ways to protect it.

The threats and attacks against computer systems have become more widespread, security measures more expensive, thus the risks are higher, which bring about the need to assess the threats. Assessing the threats and quantifying the difficulty of attacking the information systems is very difficult. An appropriate conceptual framework that suggests most optimal security measures would be very helpful for information systems' owners, security experts and software designers to assess security and provide a better overview of the security threats of such complex systems. The attack tree analysis and quantitative security assessment is a relatively young field that tries to solve this problem.

Since 1999 when Schneier popularized the attack tree analysis concept, it has been an active research subject in order to find ways to utilize it in real life environments. The main issue with the attack tree models is their computational complexity and calculation speed that limit their use in practice as the trees grow big and calculations are very time consuming.

The attack trees provide a formal and methodical way of describing possible attack scenarios in the considered environment. Attacks against a system are represented in a tree structure, where the goal of the attacker is the root node and different ways of achieving that goal are leaf nodes. Two types of refinements are used: the conjunctive refinement where all sub-attacks must be satisfied in order to satisfy the root goal and the disjunctive refinement where any of the sub-attacks are sufficient to satisfy the goal. The attack tree with AND- and OR-nodes may be represented as a monotone Boolean function. Satisfying assignments of this function represent possible attacks.

Several methods are proposed for performing quantitative security risk analysis based on attack trees. Calculating the attack tree is time consuming and not very rational in real life scenarios where the attack trees may have thousands of nodes. Therefore Jürgenson

et al. [1] proposed a genetic algorithm for fast and approximate calculations of attack trees. Later Lenin et al. [2] suggested to consider the attacker profile in calculations and for more real life scenarios.

In this thesis, the performance and precision of profiling in the scope of the ApproxTree+ tool, created by Lenin and Willemsen [2] compared to the ApproxTree tool [4] is assessed. Furthermore, it is estimated what effect the integration of profiling has on the genetic algorithm parameters. In particular, it is attempted to determine if the choice of optimal genetic algorithm parameters derived for ApproxTree by Jürgenson et al. remain optimal for the ApproxTree+ approach.

The thesis is organized as follows. Chapter 1 introduces the topic of research. Chapter 2 gives an overview of the current state of the art of attack tree security modeling, computations of parallel attack trees and optimizations of attack trees using genetic algorithm. Chapter 3 describes the case study attack tree “Steal sensitive information by collecting network traffic of an enterprise” in detail. Chapter 4 assesses the attacker profiling efficiency compared to the genetic algorithm without profiling. Chapter 5 studies the genetic algorithm parameters like initial population size, genetic algorithm termination condition and mutation rate effect on the convergence speed of the method. An improvement how to make ApproxTree+ computations faster and more accurate is proposed. Chapter 6 provides conclusions and proposes suggestions for future improvements.

2 Introduction to attack tree analysis

2.1 Attack trees for modeling security

Attack trees are used for analyzing computer systems security. The concept of attack trees is not new. Before the end of the 1990s the attack trees were known as threat logic trees. In 1991 the threat logic tree analysis was applied to information security and was based on the fault tree analysis where the tree's root node was the high-level potential threat which was subdivided into the tree's structure using AND and OR nodes. Tree leaves that did not require further division represented the attackers actions.

Bruce Schneier was one of the first to describe and popularize attack trees in his papers and articles. In his article [5], he points out that the attack trees can be used for analyzing the security of systems and subsystems and provide a way of thinking about security. Attack trees establish the basis of understanding the process of assessing security. He proposed to represent attacks against a system in a tree structure, with the goal as the root node and distinct attack steps as leaf nodes.

For example, Figure 1 represents a simple attack tree the goal of which is to open the physical safe. To reach the goal the attacker can pick the lock, learn the combination, cut open the safe or install the safe improperly so that he can easily open it later. There are two ways to learn the combination: to find the combination written down or to get the combination from the safe's owner. For getting the combination from the safe's owner the attacker could threaten, blackmail, bribe or eavesdrop. The key is to continue refining the nodes to the point where elementary attacks have been reached. For refining nodes AND and OR refinements are used. AND nodes represent different steps how the attacker can achieve the main goal, OR nodes show alternatives for carrying on the sub-goals.

The security of the system can be assessed and calculated when assigning a Boolean or continuous domain to the leaf nodes and propagating them up to the tree structure in the same way. Figure 1 contains Boolean values like "impossible" and "possible", but Figure 2 shows that it is possible to assign some other Boolean value or even some continuous value like the cost of the attack. Schneier [5] also made the point that sometimes it is important to determine the characteristics of the attacker to know which part of the attack tree is the one to worry about.

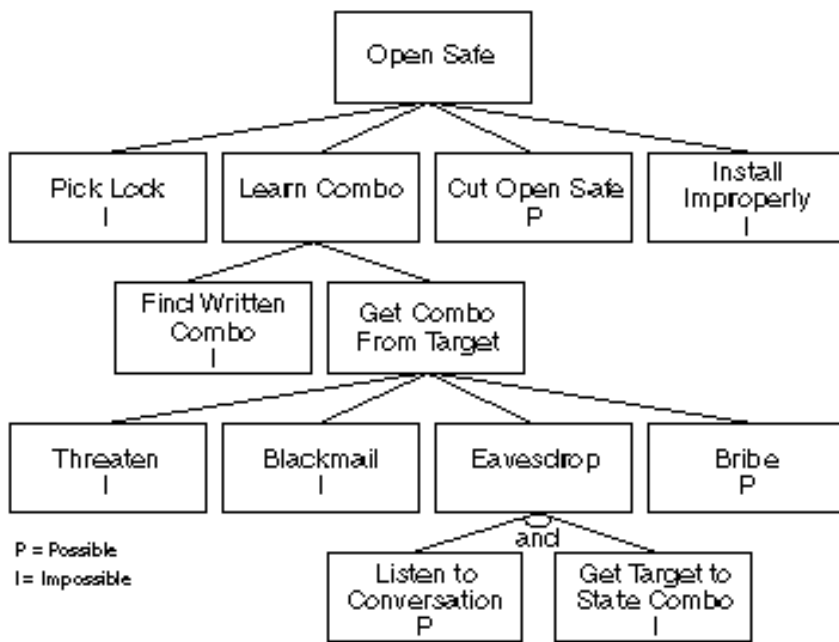


Figure 1. Attack tree represented by B. Schneier in [5]

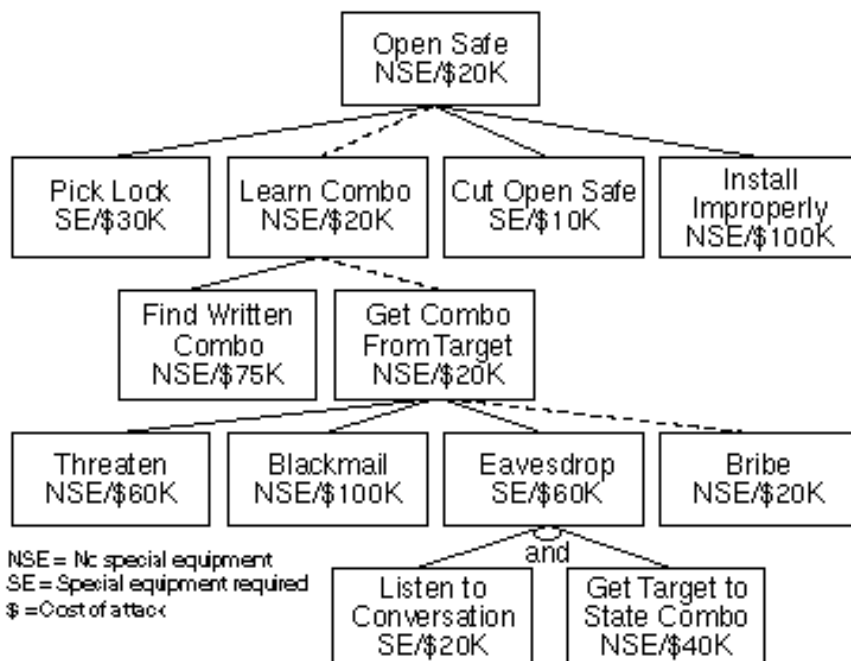


Figure 2. Attack tree taking into account the difficulty and cost of attack by B. Schneier in [5]

The concepts introduced by Schneier were formalized by Mauw and Oostdijk in [6]. They argued that formal interpretation of the attack trees is absolutely necessary to understand how the attack trees can be manipulated during the construction and analysis phases. Therefore they proposed formal definitions of how to compose an attack tree from elementary attacks and nodes, the semantics of the attack tree itself and associativity and distributivity properties of the nodes and suggested ways how to compute the analysis outcome when attribute values are assigned to the elementary attacks.

Mauw and Oostdijk presented the compatibility notion between the semantics and attributes for the attack trees and introduced multiset semantics based on a semiring. They found out that in some cases propositional interpretation of the attack trees is inappropriate, because the law of distributivity does not apply.

Moreover, using statements that express a concept that can be true or false brings about certain problems. For example, it is not suitable for modeling sequential semantics. Furthermore, the bottom-up approach is valid under the assumption that all attack steps are mutually independent which is not often the case in real-life scenarios. In reality, attackers execute some attacks and if they fail or succeed, they use additional information before choosing another line of action.

Buldas et al.[7] introduced the game-theoretic approach to the attack tree analysis. The theory was based on the assumption that the attacker is thinking in rational and economic terms. The proposed multiparameter attack tree model works with multiple parameters and analysis attacks from the attacker's viewpoint. The authors introduced *rational attacker's paradigm* that stated the following: rational attackers do not attack if it is unprofitable and the attacker chooses attack vectors that are the most profitable for him. The attacker's decision-making process was modeled using this paradigm. Firstly, the attacker has to have an overview of all the ways for attacking. Secondly, he/she will make possible plans for the attack by using the same approach as Schneier proposed - constructing an attack tree where the primary threat and sub-attacks are defined. Sub-attacks are refined using AND and OR-refinements until atomic threats get to a level where it does not make sense to divide them further. The attacker finds out whether any of these is profitable by evaluating all possible plans. In order to decide whether the attack is profitable for the attacker, the new parameter "outcome" (the difference between the expected reward and expected expenses) was introduced. The attacker calculates the outcome using the following parameters:

- What the attacker gains in case the attack is successful (*Gain*)
- How much money the attacker has to spend to launch an attack (*Cost*)
- Success probability of the attack (*p*)
- Probability of getting caught if the attack was successful (*q*)
- Expected penalties if the attacker is caught but the attack was successful (*Penalties*)
- Probability of getting caught in case the attack was not successful (*q₋*)
- Expected penalties if the attacker is caught but the attack was not successful (*Penalties₋*)

The formula (1) shows how to calculate the Outcome.

$$Outcome = -Cost + p \cdot (Gains - q \cdot Penalties) - (1-p) \cdot q_{-} \cdot Penalties_{-} \quad (1)$$

Buldas et al. stated that the system is secure against rational attackers if *Outcome* is less than or equal to 0. This means that the primary threat is not profitable for attackers.

Aivo Jürgenson has studied several attack tree models in more detail and found that those models have several shortcomings. In his thesis [4] he points out that the attack tree model used by Buldas et al. uses the node parameter propagation from child nodes to the parent nodes and propagation process in the OR-nodes relies on local optimum decisions which means that the computed utility value is not always the global maximum and the best attack suite might not be found. What's more, the model was not consistent with Mauw and Oostdijk attack tree foundations that stated that the equivalent attack trees have to result in the same utility value.

Jürgenson and Willemsen introduced the new attack tree model in [8] which was consistent with Mauw and Oostdijk work [6] and gave more reliable outcome values than model of Buldas et al. [7]. However, their outcome computation routine is very complex and time consuming and is applicable for analyzing the attack trees containing no more than 20 leaf nodes. This means that it cannot be used for analyzing the security of real life systems, because the attack trees there have thousands of leaves. The authors addressed the need for optimizations. Subsequently, in 2010 Jürgenson et al. [4] proposed a way to optimize and approximate calculations and managed to compute attack trees with 100 leaves in reasonable time.

2.2 Parallel attack tree model

The attack tree analysis begins with identifying one primary threat and continues by dividing the threat into sub-attacks so that all or some of them are necessary to materialize the primary threat. The sub-attacks are split until the state is reached when it does not make sense to divide resulted attacks any more. Those non-splittable attacks are called elementary attacks. During the splitting process AND- and OR-nodes are used. Having the primary threat in its root and elementary attacks in its leaves the AND-OR-tree is formed.

Jürgenson et al. in [4] have proposed two models for following the behavior of the attackers- parallel and serial attack tree models. In the parallel model attack steps are launched simultaneously. It assumes that the attacker decides on the list of attacks before starting and then all attacks are tried in a parallel manner. In the serial model attack steps are launched in a predefined order. The attacker starts attacking and then adaptively makes decisions based on the success or failure of preceding attacks.

As this thesis focuses on parallel attack tree model approach it is firstly important to point out the formal definitions of Jürgenson- Willemson's parallel attack tree model and discuss their optimizations and approximations proposed in [1]:

DEFINITION 2.1 (Elementary attack): *Elementary attack is the lowest level of abstraction of attacks, which do not have any internal structure within the scope of the particular attack tree. Elementary attacks are the leaves of the attack tree.*

DEFINITION 2.2 (Attack tree): *Attack tree \mathcal{T} is a simplified PDAG structure $(V = N \cup X, n_0, E)$, of the following elements:*

1. *the set of leaves $\mathcal{X} = \{X_1, \dots, X_n\}$ represents the elementary attacks, which are considered as propositional variables having values of true or false, correspondingly, if the elementary attack has been tried and was successful or has been tried and failed,*
2. *the set of nodes $\mathcal{N} = \{N_1, \dots, N_m\}$ represents the logical functions of either $\&$ and \vee . The function $\&$ evaluates to true if all of its children evaluate to true and function \vee evaluates to true, if some of its children evaluate to true,*
3. *$n_0 \in \mathcal{N}$ is the root node of the PDAG, which does not have any parents,*

4. $\mathcal{E} = \{(a, b) : a \in V \text{ and } b \in N\}$ is the set of directed edges between leaves \mathcal{X} and nodes \mathcal{N} or between nodes \mathcal{N} themselves.

DEFINITION 2.3 (Attack suite): *Attack suite $S \subseteq \mathcal{X}$ is the set of elementary attacks, which have been chosen by the attacker to be launched and used to try to achieve the attacker goal.*

DEFINITION 2.4 (Satisfying attack tree): *The attack tree \mathcal{T} is satisfied by the attack suite S and the goal of the attacker is achieved if the Boolean function corresponding to the root node n_0 evaluates to true when all elementary attacks from the attack suite S have been tried and they have been evaluated to true and false values, correspondingly, if the elementary attack was successful or failed.*

Only monotone Boolean formulas are considered, so that the trivial assignment $X_1 := \text{true}, \dots, X_n := \text{true}$ always evaluates \mathcal{F} to true. In addition to that, the basic game-theoretic approach of the original multi-parameter attack tree model was followed [7].

1. The attacker has to spend resources ($Cost_i$) to prepare and launch the elementary attack.
2. With the probability p_i the attack succeeds and probability $1 - p_i$ the attack fails
3. The attacker sometimes has to carry additional costs after failing or succeeding, this parameter is called $Expenses_i$.
4. There is the global parameter $Gains$ for the whole attack tree and it describes the utility of the attacker if the root node is achieved.

The attacker's game for the whole attack tree can be described as follows:

1. The attack tree with AND-nodes and OR-nodes is constructed and the attacker evaluates the parameters of the elementary attacks.
2. The attacker considers all potential attack suites. For those attacks, which allow the root node to be reached, he calculates the outcome value ($Outcome_S$).
3. Finally the attacker chooses the attack suite with the greatest outcome and launches the corresponding elementary attacks.

The outcome value for attack suite S can be computed as:

$$Outcome_S = p_s \cdot Gains - \sum_{x_i \in S} Expenses_i \quad (2)$$

P_S denotes here the success probability of an attack suite and it can be calculated as

$$p_s = \sum_{\substack{\mathcal{R} \subseteq S \\ \mathcal{F}(\mathcal{R} := true) = true}} \prod_{x_i \in \mathcal{R}} p_i \prod_{x_j \in S \setminus \mathcal{R}} (1 - p_j). \quad (3)$$

Formula (3) shows that when calculating the success probability of an attack suite S , the redundancy of the attack suite S is taken into account. It is done so because, there may be subsets $\mathcal{R} \subseteq S$ sufficient for materializing the root attack. In the parallel model the redundancy increases the success probability of attack suites. To prove that, let us assume we have an attack tree expressed by the Boolean function $\mathcal{F} = (A + B) * C$. The attacker can be successful in 3 ways: using attacks A and C, B and C, or A, B and C. As in the parallel model all elementary attacks are independent events, the success probabilities of the corresponding attack suites can be calculated as following:

$$\Pr[AC] = \Pr[A] * \Pr[C]$$

$$\Pr[BC] = \Pr[B] * \Pr[C]$$

$$\Pr[ABC] = \Pr[A] * (1 - \Pr[B]) * \Pr[C] + (1 - \Pr[A]) * \Pr[B] * \Pr[C] + \Pr[A] * \Pr[B] * \Pr[C]$$

The last formula shows that the attack tree can be successful in 3 cases: firstly, when A and C are successful and B is not; secondly, B and C succeed and A not; or thirdly, all A, B and C are successful. The redundant attack suites have greater success probability than the non-redundant ones and this in turn might increase the utility of the attack.

The complexity of the method comes from the necessity to solve the SAT (satisfiability) problem, which is complex. Even with all optimizations introduced in [8] Jürgenson-Willemson still faced an exponential complexity burst in formula (3). The fact that n attack steps have 2^n subsets shows the method is inappropriate to be used in real cases.

2.3 Optimizations using the genetic algorithm

Verifying if there are cases that satisfy the Boolean formula \mathcal{F} and computing the outcome by formulae (2) and (3) is very time-consuming. Therefore, Jürgenson et al. suggested using the genetic algorithm for optimizations and finding the optimum attack suite in [4]. The idea of the genetic algorithm is to generate the initial population, then cross the individuals, mutate them and sort out the best solutions, thus continuing to improve the result by reproduction:

1. Generate the first generation of \hat{h} individuals (attack suites) that satisfy the Boolean formula.
2. Cross \hat{h} attack suites with each other, producing $\binom{\hat{h}}{2}$ new attack suites.
3. Mutate each new individual with probability p .
4. Join the mutated population with the current population.
5. Choose those individuals, who are alive (satisfy Boolean function \mathcal{F} - i.e. $\mathcal{F}(S_j := t) = t$).
6. Compute the $Outcome_{S_j}$ for each of the remaining individuals and choose \hat{h} best individuals that produce the greater outcome for the next generation.
7. Reproduce until the determined number of generations is reached and choose the best attack suite for attack tree \mathcal{T} and its outcome.

In the work of Jürgenson et al. [4], an individual is an attack suite S and it is a bit array of all the attack tree leaves. The quality of individuals is measured by the outcome value of the attack (fitness function).

Population is a set of attack suites that are under consideration. When generating the initial population, it is important to choose those individuals that satisfy the Boolean formula \mathcal{F} , this means $\mathcal{F}(S_j := t) = t$. Starting from the root of the \mathcal{T} and choosing all children from AND-node and choosing randomly at least one child from OR-node.

The crossover operation crosses two attack suites σ_1 and σ_2 by randomly flipping the values of the attack steps from FALSE to TRUE or the opposite throughout all the elementary attacks $\mathcal{X}_i=(i=1, \dots, n)$. For example, first attack suite values are **11001011** and for the second **11011111**. If in the crossover phase we randomly cross for three bits of the attack suites, for example, then the result will be **11001111**.

2.4 Attacker profiling in attack tree analysis

A further development of the genetic algorithm introduced by Jürgenson et al. has been proposed by Lenin- Willemson in [2]. They introduced the attacker profiling concept, the reason behind which was to provide a more realistic insight into attacks. Classical risk analysis assumes that the attacker is almighty. However, an overpowered attacker is often not the case in reality. Attacker profiling limits adversarial capabilities and due to that makes the strategic behavior closer to the one which is likely to be observed in real

life. Moreover, attacker profiling separates the infrastructure properties from the properties of the threat. It enables a more flexible and comprehensive look at the ever-changing risk landscape and enables more reliable risk assessment. Attacker profiling is a way forward in dealing with complexities of security metrics; the parameters of the attack tree leaves cannot be estimated in a meaningful way without specifying attacker properties and capabilities. Lenin et al. [2] suggested using attacker profile for describing the attacker’s skills and resources available for performing malicious actions to achieve the main goal. The attacker profile considers several parameters like skill or proficiency, time and the attacker’s budget, described in Table 1.

Table 1. Attacker profiling parameters

Parameter	Description	Values
Proficiency	Attacker's skills level	Very High (V), High (H), Medium (M) Low (L)
Budget	Amount of financial resources available to the attacker	Currency units
Time	Amount of time the attacker can invest in attacking	Second (S), Minutes (MT), Hours (H) Days (D)

The attacker’s skill or proficiency is a parameter that describes the attacker’s skill level and influences the techniques that may be chosen for performing the attacks. This parameter uses value units Very High (V), High (H), Medium (M), and Low (L).

The second parameter is called the attacker’s budget. This is related to the amount of financial resources available to the attacker. For example, it might be the monetary value of hardware or software used to support the attack steps. The value unit for the attacker’s budget parameter is some specific currency value.

The third parameter is the time for attacking. This parameter describes how much time the attacker can invest in attacking. The value unit used for it can be Seconds (S), Minutes (MT), Hours (H), and Days (D).

Lenin and Willemson formalized the definition of the attacker profile in [2] and it is the following:

DEFINITION 2.5 (Attacker profile): *An attacker profile is a set of characteristics and properties uniquely describing the attacker under consideration:*

1. Budget- *the monetary resource of the attacker, measured in currency units.*
2. Proficiency- *the skill level of the attacker, measured on an ordinal scale (Low, Medium, High, Very High).*
3. Time- *the available time resource of the attacker, measured on an ordinal scale (Seconds/Minutes/Hours/Days)*

DEFINITION 2.6 (Profile satisfying attack suite): *A profile satisfying attack suite σ is a satisfying attack suite, which satisfies all the constraints of the chosen attacker profile \mathcal{P}_f*

DEFINITION 2.7: (Derived function). *If $\mathcal{F}(x_1, \dots, x_m)$ is a Boolean function and $v \in \{0, 1\}$, then by the derived Boolean function $\mathcal{F}|_{x_j=v}$ we mean the function $\mathcal{F}(x_1, \dots, x_{j-1}, v, x_{j+1}, \dots, x_m)$ derived from \mathcal{F} by the assignment $x_j := v$. [9].*

Attacker profiling was observed and analyzed also by Sari in [10]. She stated that the effect of applying the attacker profile to the attack tree will invalidate some nodes, thus it eliminates sub-trees from overall attack trees and provides possible attack steps for the particular attacker. Based on her calculation results, the final outcome calculated using attacker profiling is not significantly different from the outcome obtained without attacker profiling. The results calculated with the attacker profiling parameter were up to 20% smaller than the results without the profiling. Therefore she drew conclusions that the attacker profile is a useful concept to add to the attack tree for quantitative security assessment based on the attack tree methodology.

Sari also pointed out that without attacker profiling the result of the analysis might give a “False Negative” in case we underestimate the attacker’s strength, but results of the analysis with certain attacker profiles might result in “False Positives” in case we overestimate attacker’s strength. Based on the case study she declared that 20% is the required investment that companies or owners of the analyzed system have to spend in order to upgrade their systems towards the real or near to the ideal preferred system.

2.5 ApproxTree+ tool

Lenin et al. [2] demonstrated the possibility of integrating attacker profiling considerations into existing risk assessment tools. As an example of such integration, the authors introduced the analysis tool named ApproxTree+ [2] which is an extension of the existing ApproxTree tool [1] enhanced by integrating the attacker and victim

profiling considerations into it. This enables to assess the security of the considered infrastructure against the entire set of threats enabling plug-and-play behavior of the analysis approach. This adds flexibility to the existing risk assessment practices.

The ApproxTree+ method uses the genetic algorithm to facilitate the usage of the computational method for large attack trees:

1. Generate initial population of n attack suites that satisfy the Boolean function and correspond to the attacker profile so that every attack step's strength is not greater than the attacker's skill, time for attacking is not longer than the attacker's time and the cost of all nodes does not exceed the attacker's budget.
2. Cross all the individuals in the initial population with everybody else and produce new individuals.
3. Mutate each individual with probability p .
4. The mutated population is joined with the current population.
5. Choose the fittest individuals that satisfy the attacker profile and the Boolean function and form the next generation.
6. Reproduce until k last generations do not increase the outcome.

Lenin et al. conducted performance analysis for comparing the ApproxTree and ApproxTree+ calculation methods [2]. They found that attacker profiling did not add any significant computational overhead. In both methods the initial population generation phase and mutation phase were almost immediate. The main workload was performed by the crossover phase and consumed approximately 85-99% of the cumulative time distribution among all the phases. The best individual selection phase did not bring along any significant workload.

Additionally, Lenin et al.[2] analyzed the effect of the generic algorithm parameters mutation rate and initial population size on the convergence speed for the attack trees of different sizes to assess whether the parameters of the genetic algorithm used by ApproxTree were optimal for the ApproxTree+ method. It turned out that the speed of convergence of ApproxTree+ did not exceed the speed of convergence of ApproxTree and the mutation step had no significant effect on the convergence speed. The results also indicated that increasing the initial population size increased the convergence speed. Moreover, the convergence speed did not depend on the size of the attack tree.

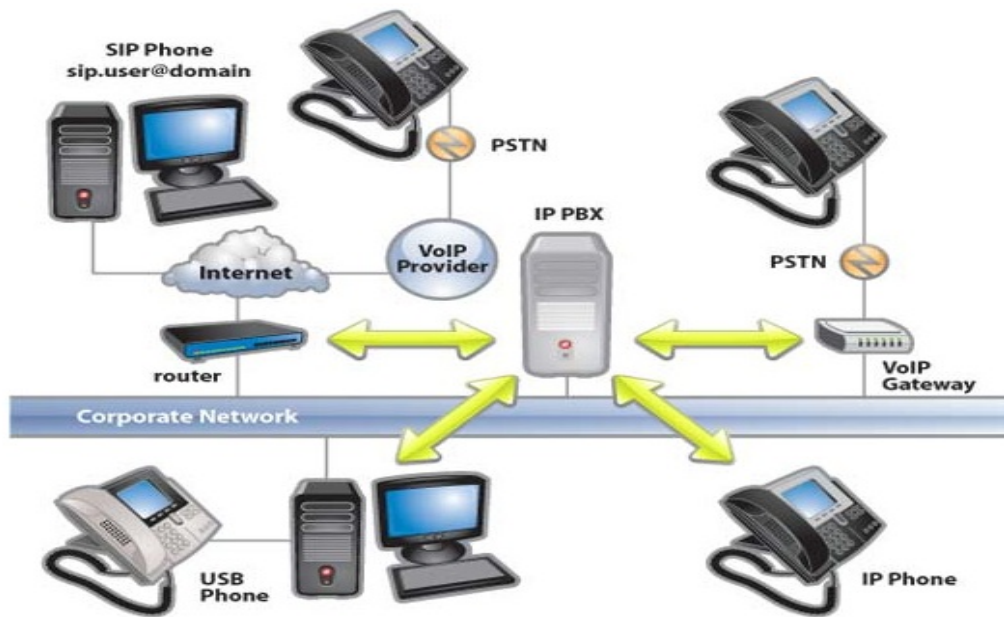
ApproxTree and ApproxTree+ computation results showed that the final outcome converged to the most profitable attack suite (global optimum) or the computational method failed to generate the initial population of individuals. If the ApproxTree+ was unable to generate the initial population, this might mean that either the profile constraints were too strict, so that no profitable attack suites exist at all or the method failed to generate the initial population due to its stochastic nature. When ApproxTree+ produces such results, it could mean any of the cases since the exact reason remains unknown.

3 CASE STUDY: Steal sensitive information by collecting network traffic of an enterprise

In the recent years technology has evolved tremendously. Today, all enterprises, governments, the military, hospitals, financial institutions, private businesses gather a lot of confidential information about their customers, employees, products, research and financial status which are processed and stored on computers and transmitted across networks. This information needs to be protected, because if some of the confidential information should get into the hands of a competitor or a black hat hacker, it could cause financial loss and damage to the company's reputation. Therefore, protecting confidential information is any business's requirement and in many cases also an ethical and legal requirement.

There are many different ways how sensitive information can be collected by attackers, but this thesis focuses on the case where the attacker misuses some of the vulnerabilities of IP PBX systems. A brief overview of how IP PBX systems work will be given next.

Figure 3 illustrates the typical IP PBX network. It consists of an IP PBX server, phones and a VoIP gateway. The IP PBX server registers all its clients (VoIP phones). When a client needs to make a call, the IP PBX should give permission to establish the connection. The IP PBX has a directory of all clients and their corresponding SIP addresses. That makes it possible to route internal and external calls. For placing calls to external numbers, VoIP gateways or VoIP service providers are needed. VoIP gateways connect the IP PBX with a traditional PSTN network. They digitize traffic from the standard PSTN lines so that the IP PBX could handle the traffic and so it could travel over the computer network. VoIP providers handle the digitalizing of traffic on their end and send the call via a network link [11].



How an IP PBX integrates on the network and how it uses the PSTN or Internet to connect calls

Figure 3. The interconnection of an IP PBX system with its components described in [11]

There are many different types of VoIP services and technologies available. Most enterprises implementing voice transmission over IP often overlook the security risks associated with it. They are mostly concerned about voice quality, latency and interoperability rather than seeing what threat VoIP can pose to the security of sensitive information. VoIP can be targeted in the ways similar to traditional network resources and the main threats may be the denial of service (DOS), intercepted communication and theft of service [12].

Let us take a look at an example of a company that wants to launch a service that is new to the market and will bring a 70% market share. The company does not want that the information about the service to become public or reach the competitor before the company has launched it. The attacker (for example the competitor) has the idea that the company has been developing something and sets a goal to find out what the company is up to. The company uses the IP PBX service between different branches for saving in management, maintenance and ongoing call costs.

For stealing sensitive information there are several ways like phishing scams, network malware, network and e-mail hacks etc. that an attacker could use. Compromising VoIP infrastructure can be accomplished in many different ways, but this thesis concentrates on a more rare case where information is gathered by collecting the VoIP traffic and playback of the calls.

In the process of constructing the attack tree, it is assumed that the company is using the IP PBX service and voice and network traffic are kept in the same VLAN. VoIP media traffic is encoded and encrypted.

The previous chapter introduced the concept of attack trees. This chapter gives an overview of the attack tree that was constructed for the case study. It will not analyze all possible ways of how the attacker can achieve the main goal, as this is not the main topic of the thesis. The aim of constructing the attack tree was to give an overview of its main concepts and show how attack trees are constructed. Moreover, it will be shown how it is possible to combine social, physical and technical attacks into one attack tree. In subsequent chapters the same attack tree is used for validating the genetic algorithm computational methods and analyzing their performance.

3.1 The attack tree model

The main goal of the attacker is to gather VoIP traffic and playback the calls in order to get access to the sensitive information. For getting the VoIP traffic, the attacker has to collect the network traffic of the enterprise. The primary threat was named “Steal sensitive information by collecting network traffic of the enterprise”, as shown in Figure 4. In order to achieve it, the attacker has to succeed in three activities. It is necessary to get access to the network traffic, collect data and decode media traffic. This way the attacker can playback the VoIP communication.

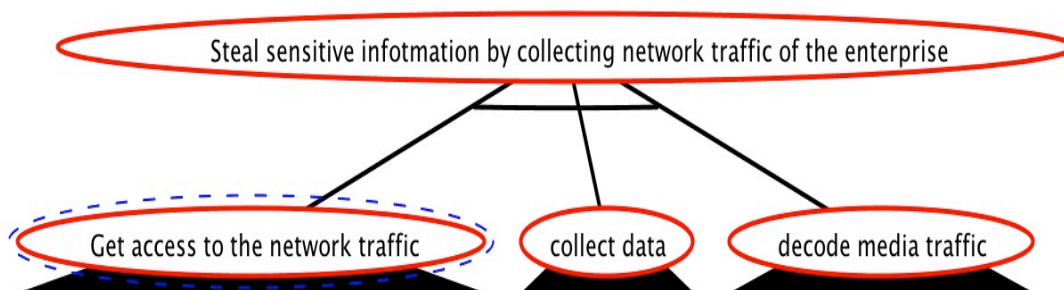


Figure 4. Main goal of the attacker: “Steal sensitive information by collecting network traffic of an enterprise”

The most complex part of the attack is getting access to the network traffic. For realizing that sub-goal the attacker has many options, which are presented in Figure 5. He/she could try to get in the network path by compromising some of the network devices like routers, switches, firewall etc. or compromise somehow the local system of the enterprise. (Figure 7 and Figure 8)



Figure 5. OR-refinement “Get access to the network traffic”

Figure 6 explains the refinement “Get into the network path”. To the successful end of getting into the network path the attacker has to hack into the enterprise network. First, he/she has to find a device to compromise. Network scans, banner grabbing and fingerprinting or social engineering attacks help to collect information about network devices like routers, switches, and firewalls. The bigger challenge is to find vulnerability and use it for obtaining access to the network. More often vulnerability scanners that use databases of known vulnerabilities are used to find vulnerabilities in the products and services of the target infrastructure

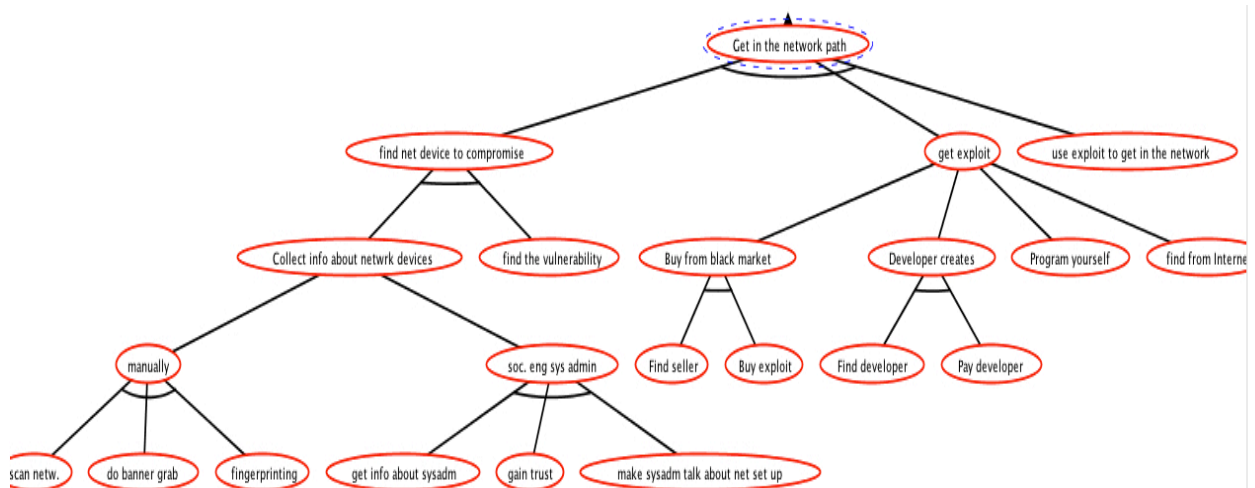


Figure 6. AND-refinement “Get in the network path”

The second method for accessing the traffic is to compromise local systems. Figure 7, Figure 8 and Figure 9 illustrate it. To realize that refinement, it is necessary either install malware onto the local computer or install malware to the IP PBX. VoIP equipment like IP PBXs or even softphones are vulnerable to malware just like any other Internet

applications. In both cases the attacker has to obtain proper malware by buying it from somewhere or creating it by him/herself, finding a suitable target like a local computer or IP PBX and infecting it.

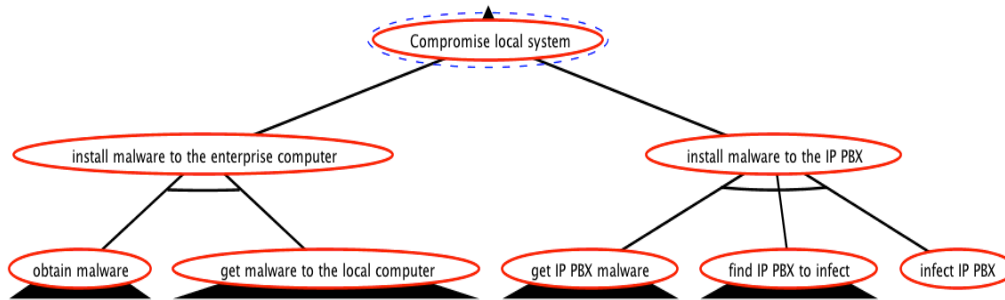


Figure 7. OR-refinement “Compromise local system”

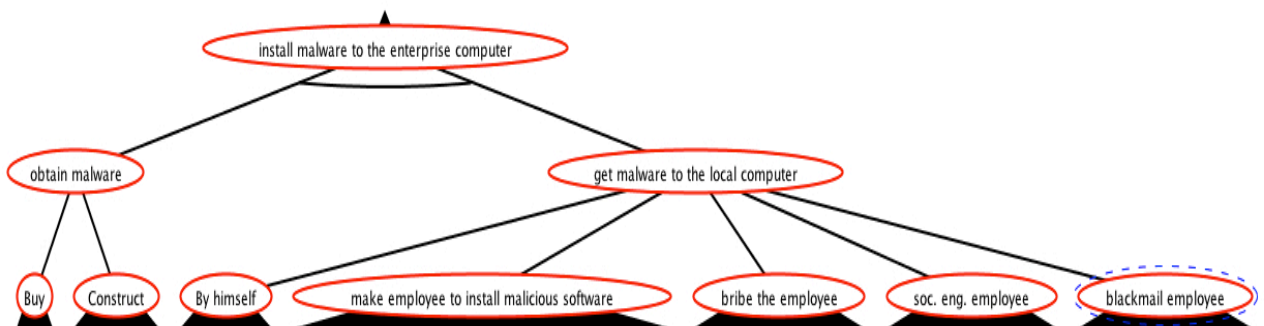


Figure 8. AND-refinement “Install malware to the enterprise’s computer”

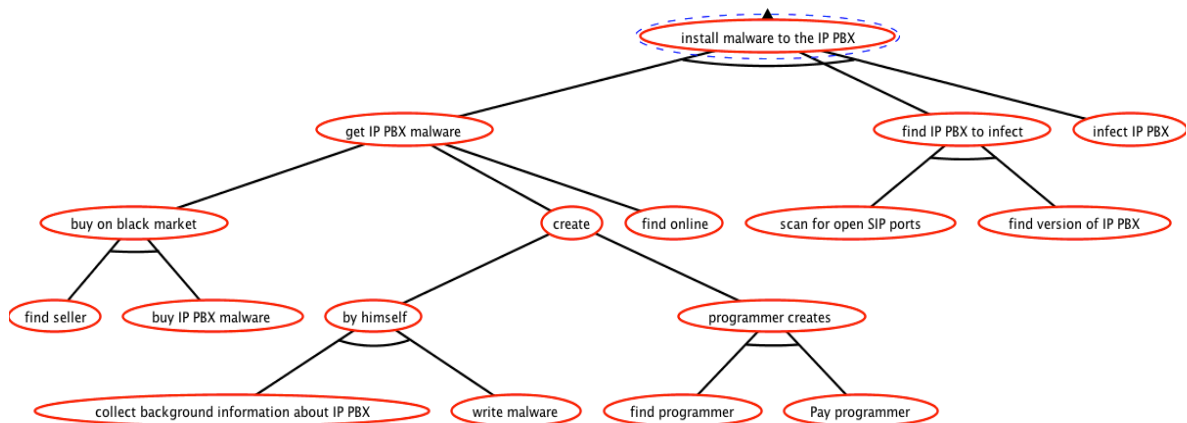


Figure 9. AND-refinement “Install malware to the IP PBX”

The solution for obtaining the network traffic does not have to be technical. The attacker can acquire it by social engineering, bribing or threatening the employees. The ways of how those could be achieved are shown in Figure 10, Figure 11, and Figure 12.

In social engineering attacks the attacker approaches an employee, impersonates being, for example, a representative of legal institution or a technician and then persuades the victim to collect and give the traffic. Usually social engineers rely on the natural helpfulness of people as well as on their weaknesses. Sometimes they take advantage of the fact that people are not aware of the value of the information they have and are careless in protecting it. However, if employees have received an awareness training, the social engineering step may be harder for an attacker.

If social engineering attacks do not work, the attacker has the possibility to bribe or threaten the company's employee. This attack includes finding personal information about the employee, approaching and offering a bribe or threatening the employee or his/her family.

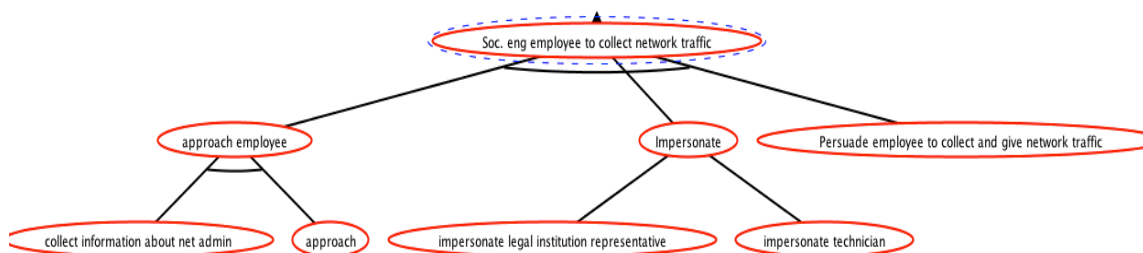


Figure 10. AND-refinement “Social engineer employee to collect network traffic”

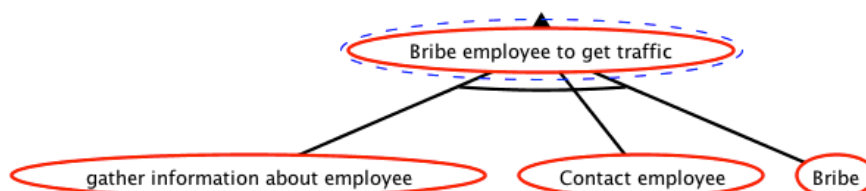


Figure 11. AND-refinement “Bribe employee to get traffic”

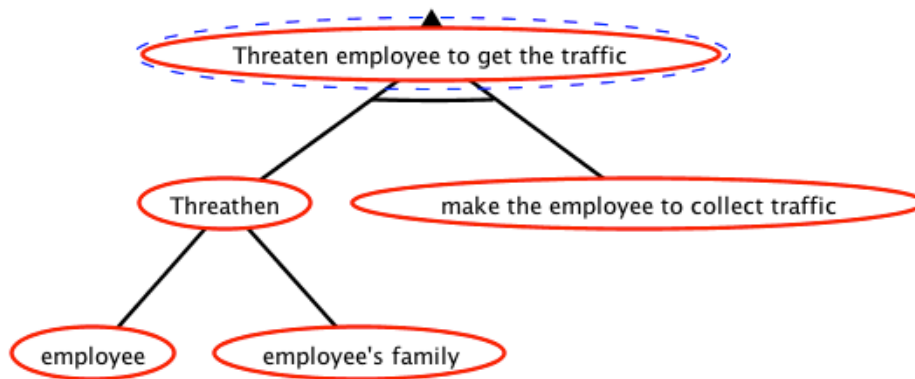


Figure 12. AND-refinement “Threaten employee to get the traffic”

Once access to the traffic is obtained, the attacker has to collect data and distinguish the media data. Figure 13 shows that for collecting data it is necessary to find proper tools for it. Usually the media traffic is encoded and in some cases also encrypted. The attacker has to find ways to decode and decrypt it, Figure 14.

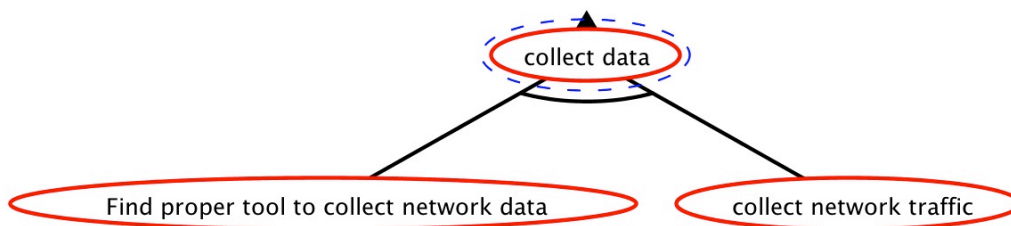


Figure 13. AND-refinement “Collect data”

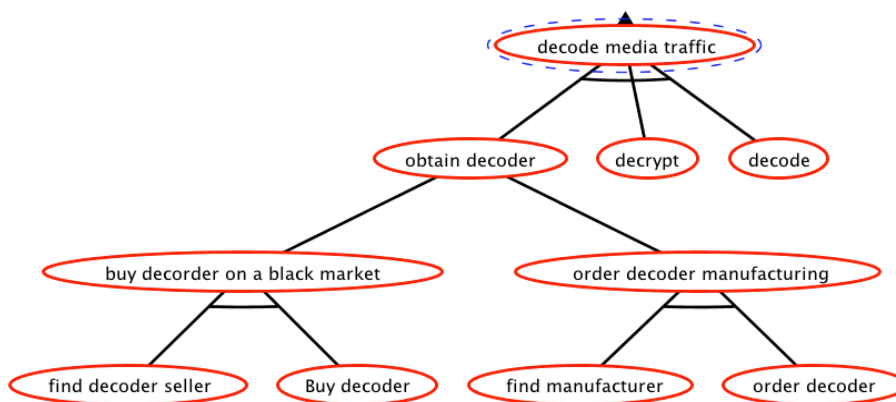


Figure 14. AND-refinement “Decode media traffic”

The case study attack tree consists of 77 elementary attacks. In order to be able to assess and make calculations, the attack tree has to be represented as a Boolean formula. The Boolean formula for the attack tree “Steal sensitive information by collecting network traffic of an enterprise” is the following:

$$\mathcal{F} = ((((((A35 * A36) + ((A54 * A55) + (A56 * A57)))) * (((A58 + ((A72 * A73) + (A74 * A75) + A69 + A70) + A59) * (A60 * ((A76 * A77) + A71)) * A37) + (A38 * A39 * A40 * A41) + (A42 * A43 * A44) + ((A61 * A62) * (A63 + A64) * A45) + (A46 * A47 * A48))) + (((A49 * A50) + ((A65 * A66) + (A67 * A68)) + A29) * (A30 * A31) * A14)) + (((((A51 * A52 * A53) + (A32 * A33 * A34)) * A11) * ((A25 * A26) + (A27 * A28) + A12 + A13) * A5) + (A7 * A8 * A9) + ((A19 + A20) * A10) + ((A15 * A16) * (A17 + A18) * A6)) * (A1 * A2) * (((A21 * A22) + (A23 * A24)) * A3 * A4))$$

3.2 Estimated values for the attack tree leaf nodes

In order to apply ApproxTree and ApproxTree+ calculation methods to the case study “Steal sensitive information by collecting network traffic of an enterprise”, it was necessary to estimate the attack tree’s elementary attack parameters and “attacker profile”. The attack tree leaf node’s parameters are Cost, Likelihood, Strength and Time. These parameters were described in Table 2.

Table 2. Parameters for describing the attack tree “Steal sensitive information by collecting network traffic of the enterprise” elementary attacks

Parameter	Description	Values
Cost	Monetary resource that the attacker has to spend to prepare or launch the elementary attack. For example buying specific software or hardware, hiring or bribing an employee, etc.	Estimated numerical value
Gain	Economic profit that the attacker receives after achieving the final goal	Estimated numerical value
Likelihood	Probability that the attack step will succeed with-in a single trial. Could be based on heuristics of similar attacks or cognitive estimations	Specific numeric value between [0...1]
Strength	The attacker's technical excellence or proficiency along with social skills that are needed for performing the attack successfully.	Very High (V): Beyond the known capability of best attackers
		High (H): Requires high degree of technical expertise and lots of experience, usually criminal cracker/hacker by profession.
		Medium (M): Requires a bit of technical knowledge, lacks experience, and heavily depends on available hacking tool resources.
		Low (L): Does not need to have technical skills to perform the specific attack.
Time	Time resource the attacker has to spend to perform the attack, apart from the difficulty and the cost of attack	Estimated in Days (D) Hours (H) Minutes (MT) Seconds (S)

Elementary attacks' estimated parameters are shown in Table 3. The estimation values for all leaf nodes were done based on cognitive estimations.

Table 3. Attack tree "Steal sensitive information by collecting network traffic of an enterprise" leaf node's estimated parameters

Leaf node	Node	Cost	Likelihood	Strength (H/M/L/V)	Time (H/M/S/D)
Find proper tools to collect network data	A1	0	0.85	L	HR
Collect internet traffic	A2	0	0.65	L	D
Decrypt	A3	0	0.65	V	D
Decode	A4	0	0.65	H	D
Use exploit to get in the network	A5	0	0.65	M	D
Persuade an employee to collect and give network traffic	A6	0	0.65	V	D
Gather information about the employee	A7	0	0.85	M	D
Contact the employee	A8	0	0.85	L	D
Bribe	A9	1000	0.85	M	D
Make the employee collect traffic	A10	0	0.85	L	D
Find vulnerability	A11	0	0.65	V	D
Program yourself	A12	0	0.85	V	D
Find exploitation means from Internet	A13	0	0.85	H	D
Infect the IP PBX	A14	0	0.65	V	D
Collect information about the employee	A15	0	0.85	M	D
Approach the worker	A16	0	0.85	L	D
Impersonate a legal institution representative	A17	100	0.65	M	MT
Impersonate a technician	A18	50	0.65	M	MT
Threaten the employee	A19	0	0.65	M	MT
Threaten the employee's family	A20	0	0.65	M	MT
Find a decoder	A21	0	0.65	V	HR
Buy a decoder	A22	100	0.85	L	MT
Find a manufacturer	A23	0	0.65	V	D
Order a decoder	A24	200	0.85	L	HR
Find a seller	A25	0	0.65	H	D
Buy the exploit	A26	1000	0.85	L	HR
Find a developer	A27	0	0.85	V	D
Pay a developer	A28	1000	0.85	L	D
Find online	A29	0	0.65	V	D
Scan for open SIP ports	A30	0	0.85	L	HR
Find a version of the IP PBX	A31	0	0.85	M	HR
Get info about sys admin	A32	0	0.85	M	D
Gain trust	A33	0	0.65	V	D

Make sys admin talk about the network's set up	A34	0	0.65	M	D
Find a malware seller	A35	0	0.85	H	D
Buy the malware	A36	500	0.85	L	HR
Install the malware	A37	0	0.65	V	HR
Find a victim	A38	0	0.85	M	D
Collect information about the victim	A39	0	0.85	M	D
Inject webpage with malware	A40	0	0.85	M	HR
Make an employee visit the malicious webpage and install SW	A41	0	0.85	M	HR
Search information about the target	A42	0	0.85	L	D
Approach the employee	A43	0	0.85	L	MT
Bribe	A44	1000	0.85	M	MT
Persuade the employee to install SW	A45	0	0.85	M	D
Collect sensitive information about the employee	A46	0	0.65	V	D
Blackmail	A47	0	0.65	M	D
Force the employee to inject computer	A48	0	0.65	M	D
Find a malware seller	A49	0	0.05	V	D
Buy IP PBX malware	A50	700	0.85	L	HR
Scan network	A51	0	0.85	L	HR
Do banner grabbing	A52	0	0.85	L	HR
Fingerprinting	A53	0	0.85	L	HR
Obtain necessary information	A54	0	0.65	V	D
Write a code yourself	A55	0	0.85	V	D
Find a developer	A56	0	0.85	V	D
Bribe a developer	A57	1000	0.85	M	HR
Physically	A58	0	0.65	M	MT
Unattended guest	A59	0	0.65	H	MT
Find suitable computer	A60	0	0.85	L	HR
Collect background information	A61	0	0.85	H	D
Approach	A62	0	0.85	L	MT
Impersonate a helpdesk assistant	A63	70	0.65	M	MT
Impersonate a higher executive	A64	300	0.65	H	MT
Collect background information about the IP PBX	A65	0	0.65	M	D
Write malware	A66	0	0.95	V	D
Find a programmer	A67	0	0.85	H	D
Pay the programmer	A68	1000	0.85	L	D
Impersonate a janitor	A69	50	0.65	L	MT
Impersonate a computer technician	A70	50	0.65	M	MT
Guess the password	A71	0	0.05	M	D
Find a suitable victim	A72	0	0.85	M	D
Impersonate a branch office employee	A73	50	0.65	M	MT
Cut network wires	A74	0	0.85	L	MT

Impersonate an ISP technician	A75	70	0.65	M	MT
Obtain a dictionary	A76	0	0.85	M	HR
Use brute force	A77	0	0.85	L	D

Furthermore, the ApproxTree+ calculation method uses Attacker and Victim profiles for computations. For profiling the attacker the parameters proposed by Lenin et al. [2]- budget, skill and time were used. When looking the attack tree estimated parameters in Table 3, it seems that time and skills are the most important factors for the considered attacker profiles. As for validating the hypothesis, different attacker profiles were chosen where the attacker had very high skills, the time range of days to perform attack and varied with budget parameter. Also, the results with different skills and time parameter when the Budget was very high were observed. As in attack tree there were no attacks that could be performed in Seconds then we did not choose that for any attacker profile time parameter. It was obvious that this would not have given any results. Table 4 shows 12 attacker profiles used for case study for further analysis.

Table 4. Attacker profiles

Attacker	Budget	Skill	Time
Attacker1	20000	V	D
Attacker2	5000	V	D
Attacker3	1000	V	D
Attacker4	7000	V	D
Attacker5	500	V	D
Attacker6	0	V	D
Attacker7	20000	V	MT
Attacker8	20000	H	D
Attacker9	20000	L	D
Attacker10	200	H	MT
Attacker11	5000	M	D
Attacker12	20000	M	D

4 Assessment of attacker profiling efficiency

In this chapter, the efficiency of attacker profiling was examined. The Boolean function of the attack tree introduced in previous chapter, estimated elementary attack parameters shown in Table 3 and attacker profiles outlined in Table 4 for utilizing ApproxTree and ApproxTree+ computation methods were used.

Firstly, the ApproxTree program was executed more than 5 times and the highest outcome given was considered. The mutation rate of 0.1 and the population factor of 2, the gain for the attacker as 50 000 EUR were used. The results of computation showed that the maximum utility for attacker would be 8216,58 EUR and the most profitable attack suite is the following:

A1, A2, A3, A4, A5, A6, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A29, A30, A31, A32, A33, A34, A51, A52, A53. This attack suite is graphically represented in Figures 15-20.

In order to steal sensitive information by collecting the network traffic of the enterprise, the attacker has to perform the following steps: “get access to the network traffic”, “collect data” and “decode media traffic”. For getting access to the traffic, attacker can make some decisions. He/she could perform one or all of the attacks: “get in the network path”, “compromise the local system”, “social engineer an employee to collect network traffic” or “threaten an employee to get the traffic”.

“Get in the network path” node, Figure 17, is realized when refinements “Find net device to compromise”, “Get exploit” and “Use exploit to get in the network” are successfully realized. “Find net device to compromise” consists of “Collect info about network devices” and the elementary attack “find vulnerability”. The information about network devices can be collected in two ways: manually or by social engineering the system admin. Manually, the attacker could use methods like “network scan”, “banner grabbing”, and “fingerprinting”. Social engineering the system administrator requires more skill. The attacker has to “get info about the sys admin”, “gain trust” and “make the system admin to talk about network set up”.

The second method of getting access to the network traffic of an enterprise is “Compromise local system” by installing malware to the IP PBX, Figure 18. For that, the attacker has to perform the following elementary attacks: get the IP PBX malware by finding it online, finding the IP PBX to infect by scanning for open SIP ports and that way collecting the knowledge of its version, and then infecting the IP PBX.

Social engineering and threatening employees are similar attacks in the sense that the attacker has to physically approach employees and persuade them to collect and give network traffic. In this case, Figure 19 shows how attacker could perform a social engineering attack. He/she has to approach the employee, introducing him/herself as somebody else like a representative of a legal institution or a technician and then persuade the employee to collect and give traffic. The attacker could threaten the employee or the employee’s family to get wanted information.

After gaining access to the network traffic, the attacker has to collect data and decode its media traffic. Collecting data assumes that the attacker has proper tools for it and before decoding traffic media traffic, it is necessary to buy a decoder from the black market or order it from a manufacturer. In case the data is encrypted, the attacker has to be prepared to decrypt it.

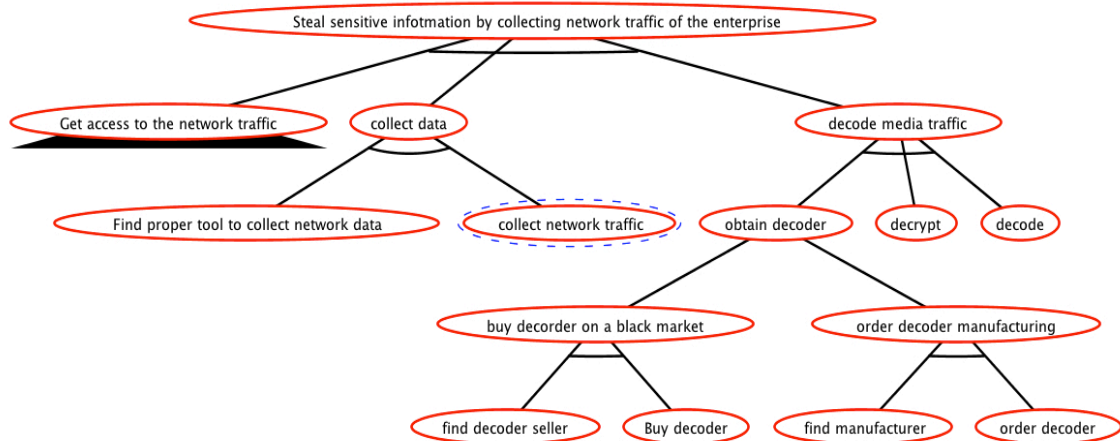


Figure 15. Attack suite computed by using the ApproxTree method. AND-refinement “Steal sensitive information by collecting network traffic of an enterprise” and “Collect data” and “Decode media traffic” AND-refinements

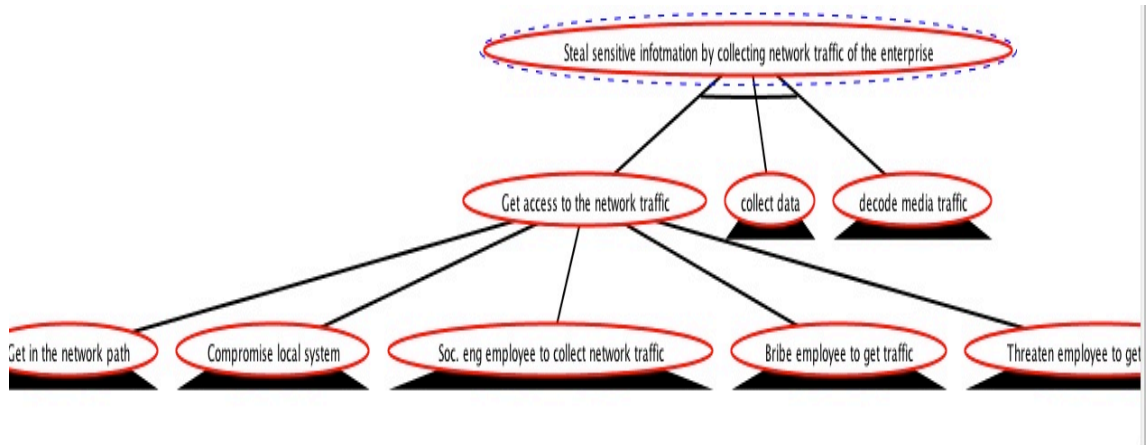


Figure 16. Attack suite computed by using the ApproxTree method. AND-refinement “Steal sensitive information by collecting network traffic of an enterprise” and “Get access to the network traffic” OR-refinement

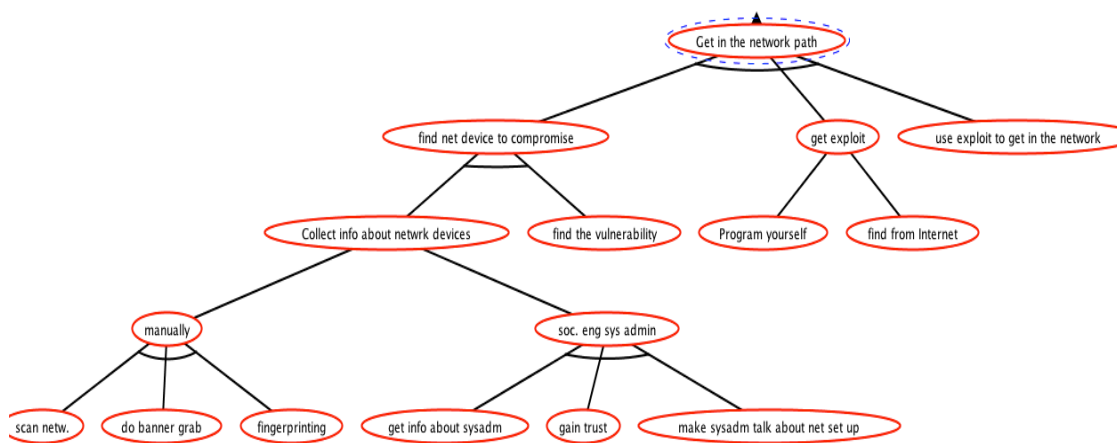


Figure 17. Computed attack suite with ApproxTree method of AND-refinement “Get in the network path”

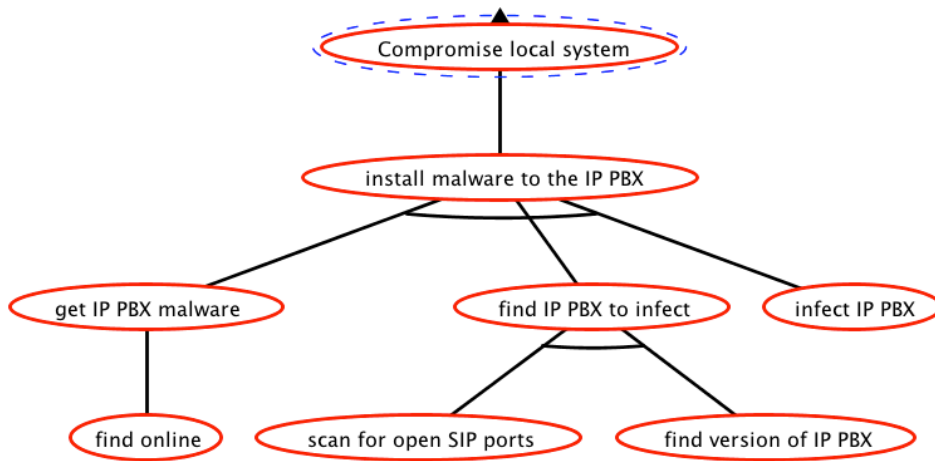


Figure 18. Computed attack suite with the ApproxTree method of refinement “Compromise local system”

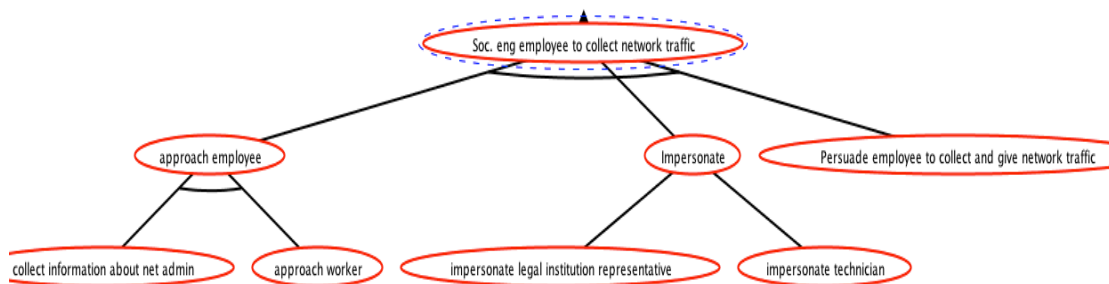


Figure 19. Computed attack suite with the ApproxTree method of AND-refinement “Social engineer employee to collect network traffic”

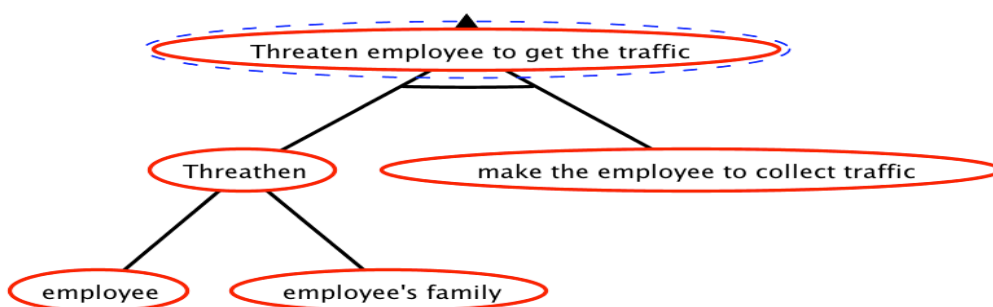


Figure 20. Computed attack suite with the ApproxTree method of AND-refinement “Threaten employee to get the traffic”

Calculated attack suites have a redundancy - attacker has an option which attacks to realize in order to steal the sensitive information. This way, the attacker could increase the success probability of the attack vector and that in turn can increase the utility of the attack.

The ApproxTree+ calculation method gave lower results than the ApproxTree method. This was expected because, in her thesis Sari [10] mentioned that using the attacker profile lowers the computed outcome up to 20%. Also, the results about all of the attacker profiles were not computed - the initial population could not be generated for the attacker profiles 6, 7, 8, 9, 10, 11, 12. The maximum outcome over all profiles was 8177,65 and the most profitable attack suite was the same as with the ApproxTree method: A1, A2, A3, A4, A5, A6, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A29, A30, A31, A32, A33, A34, A51, A52, A53, graphically represented in Figures 15-20.

Table 5 shows the calculated outcome results for the attacker profiles 1, 2, 3, 4, 5 and attack suites that corresponded to the highest outcome.

Table 5. Calculation results of the ApproxTree+ methods for Attacker profiles 1, 2, 3, 4, 5

Attacker	Budget	Skills	Time	Gain and attack suite
Attacker 1	20000	V	D	GAIN: 7866,67 max AS: A1,A2,A3,A4,A5,A6,A10,A11,A12, A13,A14,A15,A16,A17,A18, A19, A20,A21,A22,A23,A24,A29,A30,A31,A32,A33,A34,A51,A52,A53.
Attacker2	5000	V	D	GAIN: 8177,65 max AS: A1,A2,A3,A4,A5,A6,A10,A11,A12, A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A29,A30,A31, A32,A33,A34,A51,A52,A53.
Attacker3	1000	V	D	GAIN: 7894,12 max AS: A1,A2,A3,A4,A6,A10,A14,A15,A16 ,A17,A18,A19,A20,A21,A22,A23,A24,A30,A31,A65,A66
Attacker4	7000	V	D	GAIN: 7866,67 max AS: A1,A2,A3,A4,A5,A6,A10,A11,A12, A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A29,A30,A31, A32,A33,A34,A51,A52,A53.
Attacker5	500	V	D	GAIN: 7696,54 max AS: A1,A2,A3,A4,A5,A6,A10,A11,A12, A13,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A32,A33,A34

4.1 Performance analysis

The ApproxTree+ tool uses the genetic algorithm like the ApproxTree method proposed by Jürgenson et al. [1], but with the difference that attacker profile considerations are taken into account. It facilitates the usage of the computational method for large attack trees. This section analyzes the performance of the ApproxTree+ compared to the ApproxTree with different genetic algorithm parameters and assess whether the genetic algorithm parameters estimated for the ApproxTree by Jürgenson et al. [1] are still optimal for the ApproxTree+.

The effect of the genetic algorithm parameters such as the initial population, mutation rate and the termination condition of the reproduction process of the genetic algorithm (generations) on the attack tree “Steal sensitive information by collecting network traffic of the enterprise” with 77 elementary attacks in it was analyzed and the results of computations were validated. One parameter at a time was varied to see its effect on the convergence speed.

Firstly, the initial population parameter was investigated. In experiments the initial population ranged from $1n$ up to $10n$ (n being the number of leaf nodes in the attack tree), the results are shown in Figure 21. It demonstrates that in both cases the convergence speed decreases with the increase in the initial population size. The results show that with the smaller initial population size the convergence speed is faster, but the average outcome values are also smaller than with a greater initial population. As with the ApproxTree+ the initial population size $1n$'s average outcome value was 7655,73 then with the population size $10n$, the average outcome value was 8262,51. However, the ApproxTree+ method gives smaller outcome values than the ApproxTree, the tendency described was the same. Thus, it is possible to conclude that with a smaller initial population size, there is a greater probability that the algorithm will get stuck to the local optimum.

The initial population size value 2 chosen by Jürgenson- Willemsen [1] is sufficiently good for the ApproxTree+ approach, however the initial population value 1 would also

be suitable because with those two values the convergence speed is the quickest.

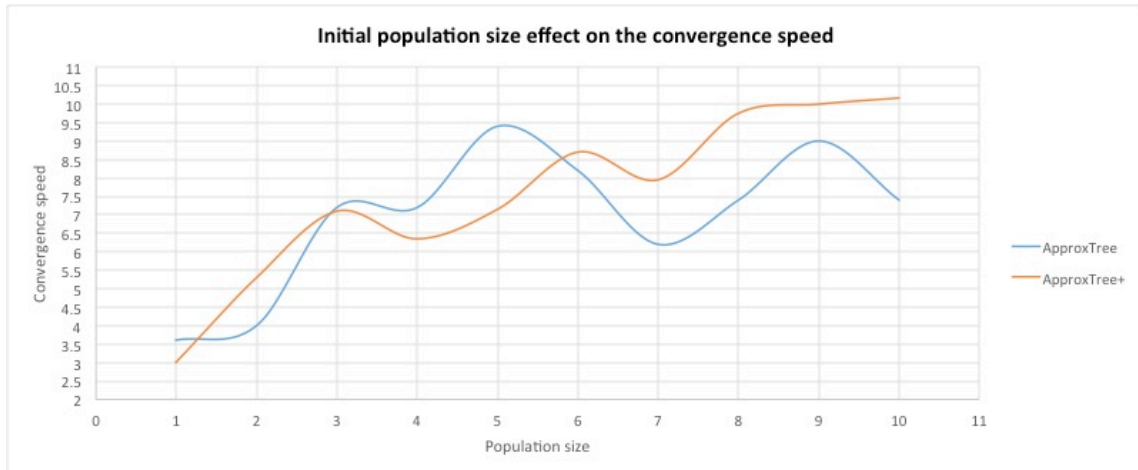


Figure 21. Initial population size’s effect on the convergence speed (# of generations)

In Figure 22, it is seen that the percentage of mutation rate has no significant effect on the convergence speed of both methods. Applying different mutation rates does not significantly change the convergence speed of the attack tree. Moreover, the precision analysis shows that the outcome does not depend on the mutation rate extensively.

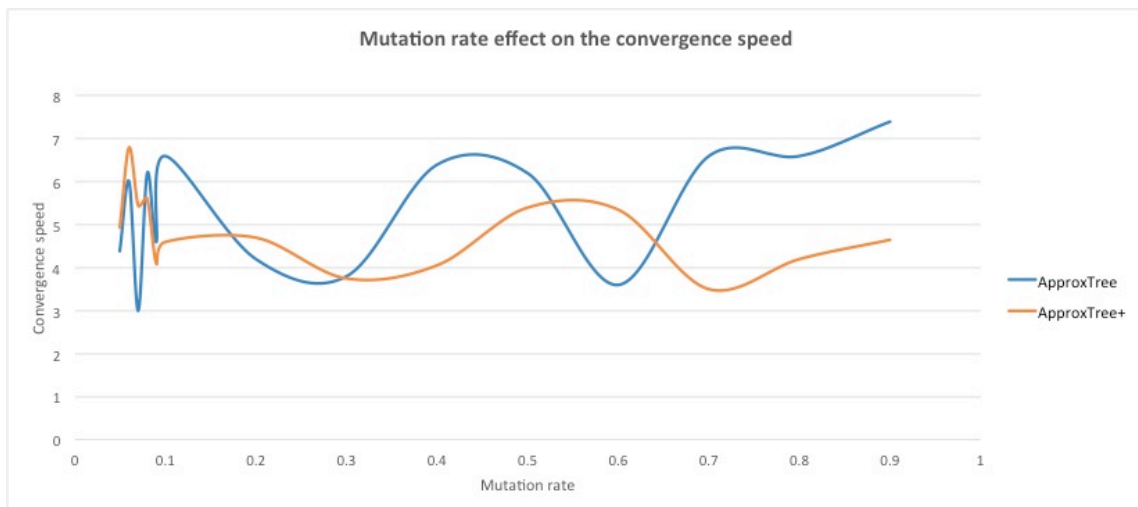


Figure 22. Mutation rate’s effect on the convergence speed (# of generations)

The genetic algorithm’s termination condition’s effect on convergence speed was also studied. The generation parameter for the ApproxTree determines the number of generations the genetic algorithm goes through before selecting the most profitable attack suite, but for the ApproxTree+ approach the generations in the genetic algorithm determines the parameter that stops the algorithm when the specified number of generations has been tried and the result did not improve. Figure 23 shows that the increase in the genetic algorithm’s generations, the convergence speed decreases. This is the expected behavior, because the more generations the algorithm has to reproduce, the

more time it takes. Based on the observations, the chosen value 10 is sufficiently good for the ApproxTree+ because that value could give us global optimum results and the algorithm's calculation time has to be sufficiently fast.

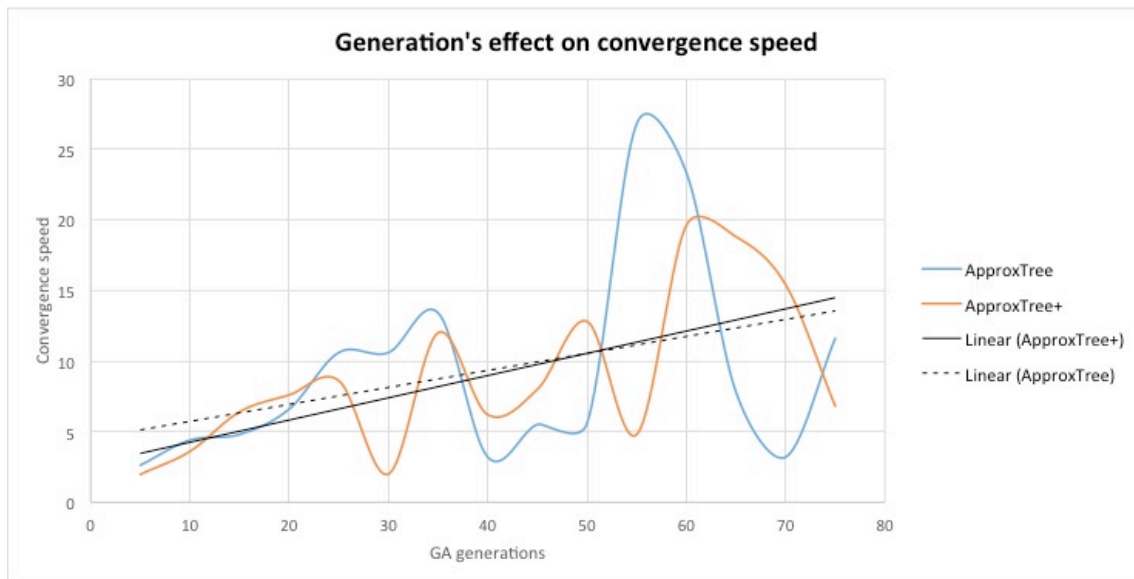


Figure 23. Generations' effect on convergence speed (# of generations)

Based on the performance analysis results, the effect of genetic algorithm parameters on convergence speed in the ApproxTree and ApproxTree+ is similar in both cases. This means that profiling has no significant effect on the performance of the genetic algorithm and integration of profiling does not introduce any significant computational overhead. When increasing the initial population size of the algorithm, the convergence speed of the ApproxTree and ApproxTree+ methods decreases. The same effect occurs when varying the generation's number. In both cases the convergence speed does not depend on the mutation rate. This might be possible due to that when the amount of the initial population exceeds the amount of possible solutions and thus all possible solutions are very likely to be present in the initial population already. In case of quite a small initial population the mutation rate might have some effect.

4.2 Improvement of the ApproxTree+ method

The existing implementation of the ApproxTree+ has a shortcoming- the results are not reliable. When the ApproxTree+ is unable to generate the initial population, it was not possible to determine the exact reason for it. It is unclear if it happened due to the profiling constraints being too strict and thus no solutions exist or the method failed to find any solution due to its stochastic nature. This section presents an improvement to the existing model, which makes the ApproxTree+ calculations more reliable and

precise. The suggested improvement made the computational method significantly faster enabling the analysis of larger attack trees (Figure 24)

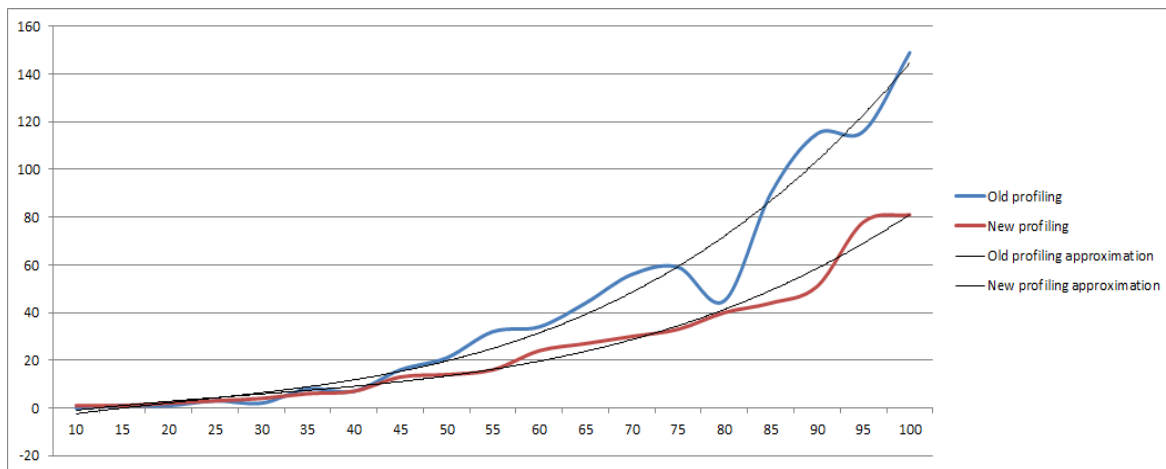


Figure 24. Execution time of attack trees with different size.

Real life attack trees could consist of thousands of leaf nodes and calculating the best attack suite could take days. The threat landscape has an ever-changing nature and analysis that takes days is not affordable, as the threat landscape will change quicker than we will be able to analyze it. Thus, the entire set of analysis will become worthless and outdated. This makes the need to find optimal and more efficient ways to compute attack trees salient.

It is proposed that the attacker profile should be applied to the initial attack trees and the derived attack tree is produced. The genetic algorithm for fast approximations should be launched on the derived attack tree. This method is noticeably faster and more accurate than the current ApproxTree+ because initially applying the profile to the attack tree excludes the leaf nodes that attacker cannot execute. If this results in an empty attack tree as a product, there can be stated that the considered class of attackers will not be able to attack the considered system.

The ApproxTree+ improved model looks as follows:

1. Apply the attacker profile to the attack tree. Leaf nodes, where the component strength is greater than the attacker's skills and time, are excluded from the attack tree. The remaining leaf nodes form a sub tree that the attacker is able to attack.
 - 1a. If the result of step 1 is an empty tree- notify the user that the considered attacker class is unable to attack the analyzed system and quit.

2. Create the first generation of n individuals from the attack suites that satisfy the Boolean formula and verify that the total cost of the attack suites in the generation is not greater than the attackers budget.
3. Cross all the individuals in the initial population with everybody else and produce new individuals.
4. Mutate each individual with probability p .
5. Unite the mutated population with the initial population.
6. Finally, choose the n fittest individuals that satisfy the Boolean function and attacker profile budget constraints and form the next generation.
7. Continue the reproduction process until ℓ last generations do not increase the outcome.

For an example, let us take an attack tree of OR-root node with five elementary attacks G, H, I, J, K, illustrated in Figure 25. This attack tree can be represented with the following Boolean function: $\mathcal{F} = (G * H) + (I * J * K)$. The attacker profile's skills parameter is High.

First, when applying the attacker profile the H node is excluded because that leaf nodes requires more skills than the attacker profile has. In the algorithm those attack steps that attacker cannot launched is expressed with $\mathbf{0}$. Therefore, the Boolean function will be $\mathcal{F} = ((G * \mathbf{0}) + (I * J * K))$. The derived Boolean function in which the genetic algorithm is used for the finding the most profitable attack suit for attacker will be $\mathcal{F} = (I * J * K)$. (Figure 25).

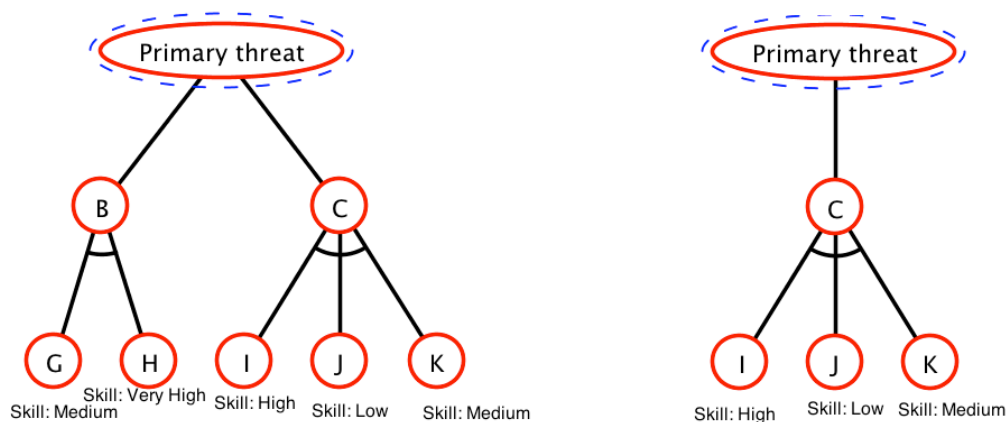


Figure 25. The attack tree before applying the attacker profile on the left and the attack tree after applying attacker profile on the right.

5 Conclusions and Future Research

This thesis studied the parallel model of the attack tree computations method using the ApproxTree and ApproxTree+ models. The ApproxTree is a model for quantitative security assessment proposed by Jürgenson et al. [4]. Lenin et al. [2] proposed a new model known as the ApproxTree+ which is the further development of the ApproxTree which takes attacker profiling into account. For the purposes of this thesis, the case study attack tree “Steal sensitive information by collecting network traffic of the enterprise” was constructed, which was used for obtaining and validating the results of the current state of the art in quantitative security assessment. In addition, the genetic algorithm parameters such as initial population size, mutation rate and number of generation’s effect on the convergence speed of the attack tree were analyzed. Finally, improvements to the ApproxTree+ method were proposed that make the analysis results more reliable and the method itself more efficient.

Also, the attacker profiling efficiency was assessed. The results were consistent with Sari’s thesis [10]. The ApproxTree+ computed outcome value is not significantly lower than the outcome calculated with the ApproxTree method. The effect of applying attacker profile to the attack tree is the elimination of sub-trees, which do not meet the attacker profile, from the general attack tree and gives feasible attack steps to the attacker who has suitable parameters to perform the attack. The attacker profile considered the attacker’s Budget, Skills and Time. It became evident that some parameters are more important for the certain attack tree than the others. In this case study, the important parameters were firstly Time and secondly Skills. If the attacker profile had the Time parameter’s value other than Days (D) and skills lower than Very High (V), the ApproxTree+ method did not find suitable attack suites.

In addition, the effect of the genetic algorithm parameters like the initial population size, mutation rate and the number of generations on the convergence speed were analyzed to assess whether the genetic algorithm parameters used by the ApproxTree are optimal for the ApproxTree+ approach. The results showed that the effect of genetic algorithm parameters on the convergence speed was similar in both cases. The increase in the initial population size decreased the convergence speed. Increasing the number of generations also decreased the convergence speed. Changing the mutation rate parameter did not have any effect on the convergence speed in either of the methods.

Based on computation in the considered case study, it became evident the increase in the initial population size from $1n$ to $10n$ decreased the convergence speed approximately from 3 to 10 generations. The optimal initial population size could be $2n$ (n is the number of the attack tree's leaves) because the convergence speed decreased with greater initial population sizes. The mutation rate parameter 0.1 is optimal because increasing its value did not have any effect on the convergence speed. The optimal number of generations is 10 because increasing number of generations will not increase the convergence speed of the genetic algorithm, but it does not affect its precision. In general, profiling has no effect on the genetic algorithm's performance and convergence speed. Attacker profiling does not introduce any significant computational overhead and thus may be integrated into existing risk assessment tools.

In order to make the analysis's results more reliable and speed up the computations of the ApproxTree+ model, the attacker profile should be applied before applying the genetic algorithm. Thus, it is possible to eliminate attack steps that do not correspond to the attacker profile and get the derived attack tree which is, as a rule, smaller than the initial attack tree.

The attacker profile in the attack tree analysis gives a possibility to observe a more real-life attacker concept as in real life situations the attackers have certain constraints and cannot perform all possible attack scenarios against targeted systems. However, when constraining the attacker, the risk of underestimating the opponent and suffer damage might be faced. Therefore, it is necessary to find some kind of a balance between the risk of excessive investments into security measures and the risk of getting damage due to underestimating the attacker resources when specifying the attacker profiles to consider.

Regarding future work, it is necessary to study attacker profile parameters that might affect the attackers' behavior more in depth and add those to the attacker profile. For instance, the considered parameters might be the attacker's motivation or the quantity of attackers. Some steps are being made by Lenin et al. towards the new parameter "insiderness" which estimates the level of trust the attacker has. If the attacker is an employee, some attack steps are easier to perform.

Some achievements have been made in estimating the set of underlying parameters, which such parameters like "success probability" or "time required for attacking" are dependent on. Lenin, et al. suggests to use Item Response Theory to derive the dependencies.

References

- [1] A. Jürgenson and J. Willemsen, “On fast and approximate attack tree computations,” *Springer*, vol. 6047 of LNCS, pp. 56–66, May 2010.
- [2] A. Lenin, J. Willemsen, and D. P. Sari, “Attacker profiling in quantitative security assessment based on attack trees”, NordSec, 2014 (to appear)
- [3] P. Kordy, “ADTool,” 2010. [Online]. Available: <http://satoss.uni.lu/members/piotr/adtool/jnlp/adtool.jnlp>.
- [4] A. Jürgenson, “Efficient Semantics of Parallel and Serial Models of Attack Trees,” Tallinn University of Technology, 2010.
- [5] B. Schneier, “Attack Trees: Modelling security threats,” *Dr. Dobb’s J. Softw. Tools*, pp. 21–22, 24, 26, 28–29, 1999.
- [6] S. Mauw and M. Oostdijk, “Foundations of Attack Trees,” *Springer*, vol. 7638 of LNCS, pp. 186–198, 2005.
- [7] A. Buldas, P. Laud, J. Priisalu, M. Saarepera, and J. Willemsen, “Rational choice of security measures via multi-parameter attack trees,” *Springer*, vol. 4347 of LNCS, pp. 235–248, Aug. 2006.
- [8] A. Jürgenson and J. Willemsen, “Computing exact outcomes of multiparameter attack trees,” *Springer*, vol. 5332 of LNCS, pp. 1036–1051, 2008.
- [9] A. Buldas and A. Lenin, “New Efficient Utility Upper Bounds for the fully adaptive model of attack trees,” *Springer*, vol. 8252 of LNCS, pp. 192–205, 2013.
- [10] D. P. Sari, “Attacker Profiling in Quantitative Security Assessment,” Tallinn University of Technology, 2014.
- [11] O. Ajasa, A.A.A Shoewu, “Exploiting VoIP telephony in IP PBX Solution,” *Pacific J. Sci. Technol.*, vol. 13, no. 2.
- [12] M. Paquette, “VoIP becomes target.(VoIP Security),” Aug. 2007.