Analyzing the Security in the GSM Radio Network using Attack Jungles

Parosh Aziz Abdulla[1], Jonathan Cederberg[1], and Lisa Kaati[2*]

[1] University of Uppsala, Sweden, email: {parosh, jonathan.cederberg}@it.uu.se
[2] FOI Swedish Defence Research Agency, Sweden, email: lisa.kaati@foi.se

Abstract. In this paper we introduce the concept of attack jungles, which is a formalism for systematic representation of the vulnerabilities of systems. An attack jungle is a graph representation of all ways in which an attacker successfully can achieve his goal. Attack jungles are an extension of attack trees [13] that allows multiple roots, cycles and reusability of resources. We have implemented a prototype tool for constructing and analyzing attack jungles. The tool was used to analyze the security of the GSM (radio) access network.

1 Introduction

Analyzing the security of complex systems is a difficult but important task. In industrial projects, security analysis is usually performed by a team of experts that create a model of the system and then analyze the security manually. The drawback of this approach is that, for complex systems, the model grows to a size that no longer permits manual analysis, thus creating a need for tools that partially automate the process.

In collaboration with a team of security experts at Ericsson Research the security of the GSM radio network was analyzed. To do this, we used a formalism called *attack trees* where the vulnerability of a system is modeled by finding all possible attacks towards a system. Attack trees were first introduced by Bruce Schneier [13] and later formalized by Mauw and Oostdijk in [11]. An attack tree is an acyclic graph with one root node. The root node represents the global goal and the tree itself encodes all attacks that are possible on the system. There are two types of nodes in an attack tree, and-nodes and or-nodes. Satisfying a node means either satisfying all predecessor nodes (and-node) or satisfying one predecessor node (or-node).

When using the formalism described in [11], we encountered problems regarding both the modeling and the analysis of attacks. One problem that occurred when modeling the attacks was the appearance of cycles. A problem that occurred when analyzing attacks was the fact that some nodes could, once they hade been accomplished, be re-used over and over again.

* This work was supported by a grant from The Swedish Governmental Agency for Innovation Systems.

To overcome these problems we have extended the notion of attack trees to *attack jungles*. There are three aspects that distinguish attack jungles from attack trees. First of all, an attack jungle may have several root nodes (an attack tree only has one root), where each root is a global goal of an attacker. Secondly, we introduce the possibility to use nodes that are *reusable*. A reusable node reflects the fact that the knowledge gained from a node can be used in several attacks. An example of reusability is obtaining a secret cryptographic key. If an attacker obtains a secret key, all messages encrypted using the key can also be decrypted by the attacker. Reusability is an important property since some resources only need to be exploited once but can affect several of the identified threats. Finally, we allow an attack jungle to contain cycles. Cycles are useful when modeling attacks that depend on each other. One example is if you have an address you can easily find the name of the person living at the address and vice versa.

Once the attack jungle is created, it can be analyzed to gain information about the security of the system. The analysis is done using an algorithm that relies on the principle of backward reachability analysis.

The analysis of an an attack jungle can answer a number of different questions about the attack jungle, such as "Is any attack possible?", "What is the minimum cost of performing an attack?" and "What is the minimum required skill level for an attack?"

The formal representation of an attack jungle enables tools to both identify and analyze possible threats to a specific system. Good tool support is required since attack jungles can become large and complex (a full attack jungle may contain thousands of different paths all leading to completion of the attack).

We have implemented a prototype tool that can be used to model and analyze attack jungles. The tool was used in a study conducted to analyze the security of the GSM radio access network where attacks towards the network were both modeled and analyzed. For attacks that had a high risk of occurring, countermeasures were designed and the tool was used to evaluate the impact/benefit of implementing the different countermeasures.

1.1 Related work

Attack trees was introduced by Schneier [13] and later formalized by Mauw and Oostdijk in [11] where the authors provide a framework for attack trees including algorithms to analyze the attack trees.

Defence trees [5] are an extension of attack trees with attack countermeasures. In a defence tree each leaf is decorated with a set of countermeasures. Each countermeasure represents a possible risk relaxation when the specific vulnerability is used. In [4] defence trees are used to represent attack scenarios and game theory to analyze the possible strategies of the attacker and the defender (or system administrator). In [6] defence trees that are enriched with conditional preferences are introduced. These structure called are called CP-defence trees.

Attack trees are not only used to analyze the security of existing computer systems. A similar formalism called obstacle trees [9] are used when caring for security at requirements engineering time. Obstacle trees are a part of a goal-oriented framework, where anti-goals that threatens the security of the system are setup by an attacker [15]. Another way to use attack trees are described in [7] were they are used as a tool by intelligence analysts and decision-makers in their work.

Another approach to analyze the security of a system is to use a formalism with a similar name called attack graphs [14] [12]. Attack graphs are automatically generated from a model of the system and the attacks towards the system are found automatically using model checking techniques.This approach is completely different from ours since the attacks themselves are not modeled, instead the system is modeled and then the attacks are found automatically.

There are some commercial tools that use attack trees. One is AttackTree+ by isograph [3] and another is SecurITree by Amenaza Technologies [10]. These tools do not allow the attack tree to contain cycles and the concept of reusability of resources can not be modeled.

1.2 Outline

In the next section we introduce the concept of an attack jungle. Section 3 presents an algorithm used for analyzing an attack jungle based on backward reachability analysis. Section 4 describes how to use the result of the analysis to automatically derive certain attributes of the modeled system. Section 5 contains a case study of the security in the GSM network, we give a short description of the different components in the GSM network and describe some of the threats towards the GSM radio access network that was identified. Finally, in section 6 we give some concluding remarks and some directions for future work.

2 The Attack Jungle formalism

In this section we give some preliminaries of *Attack Jungles*. The presentation follows that of [11] in many respects, but we have reformulated some concepts to fit our more general formalism.

For a set A, we use $\mathcal{M}(A)$ to denote the set of multisets over A. For a multiset $m \in \mathcal{M}(A)$, we write $m(a)$ to denote the number of occurrences of a in m. Sometimes, we write multisets as lists, so $[a, a, b]$ denotes the multiset with two occurrences of a and one of b (i.e. $m(a) = 2$ and $m(b) = 1$). For a multiset m and an element a, we write $a \in m$ to denote that $m(a) > 0$. For multisets m_1 and m_2, we let $m_1 + m_2$ be the multiset such that $(m_1 + m_2)(a) = m_1(a) + m_2(a)$, and we let $m_1 - m_2$ be the multiset such that $(m_1 - m_2)(a) = \max(0, m_1(a) - m_2(a))$. For multisets m_1 and m_2, we use $m_1 \leq m_2$ to denote that $m_1(a) \leq m_2(a)$ for all a. We define $|m| := \sum_{a \in m} m(a)$ i.e., $|m|$ is the sum of the multiplicities of all elements in m.

We show a typical example of an attack jungle in Figure 1. An attack jungle is a graph with two types of nodes: and-nodes and or-nodes. An attack jungle encodes in a hierarchical manner a set of attacks on a specific target. More formally, an attack jungle is a multigraph $J = (V, E, A)$, where

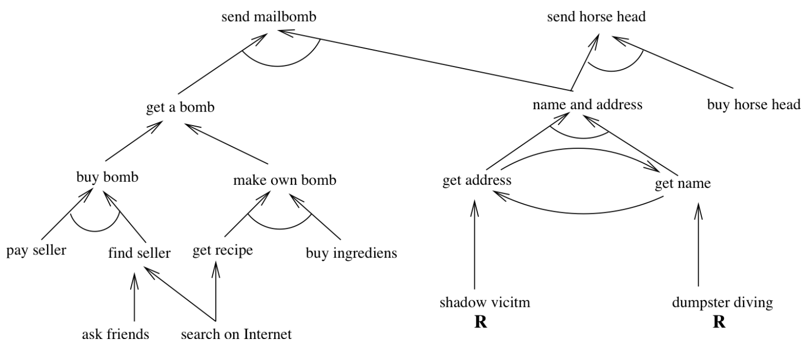

Fig. 1. An Attack Jungle

- V is a set of nodes, each describing an attack goal.
- $E \in \mathcal{M}(V \times V)$ is a multiset of edges
- $A \subseteq V$ is a unary relation on the nodes, describing which nodes are and-nodes. The complement $V \setminus A$ a set of all or-nodes.

Note that some of the nodes in the attack jungle have no incoming edges. These nodes are called *attack components* and are the lowest level of abstraction that the model provides. The set of attack components is denoted by $\mathbb{C}$.

We illustrate how to interpret an attack jungle using the example of Figure 1. Consider the attack jungle in Figure 1. Each node in the graph describes an attack goal. Most such goals are not what we intuitively would think of as the goal of an attack, but rather some component (intermediate step) of an attack. For example, “get address” is not really an attack in itself, but a necessary component of an attack where something is to be sent to the victim. To realize the attack goal “buy bomb”, the attacker has to “find seller” and “pay seller”. The attack goal “find seller”, in turn, can be accomplished either by a “search on Internet” or “ask friends”. When realizing “buy bomb” we have to accomplish all attack goals below, but for “find seller” any one will suffice. This is because the former node is an and-node, whereas the latter is an or-node. This is illustrated in Figure 1 by the existence of a small arc between the incoming arrows of “buy bomb”, whereas there is no such arc for “find seller”.

Note that in the definition of the attack jungle, the set of edges is a multiset rather than an ordinary set, making the graph a multigraph rather than a standard graph. The reason for this is to capture situations where one needs multiple instances of a resource to accomplish a goal.

With this interpretation of an attack jungle, a question arises: Given an attack goal v, how does one identify all multisets of attack components from which we can accomplish v? To answer this question, we first need to make precise the notion of accomplishing an attack goal.

For an attack jungle (V, E, A), a *state* is a multiset $s \in \mathcal{M}(V)$. Being in a specific state, means that all the attack goals corresponding to the nodes in the state are accomplished. To formalize the notion of accomplishing an attack goal, we define a transition relation on states that describes how a specific attack goal can be accomplished from other attack goals. We first define a relation $\Gamma \subseteq \mathcal{M}(V) \times V$ describing how we can accomplish a specific attack goal from its direct predecessors in the attack jungle. As described above, we need only accomplish one of the direct predecessors of an or-node, whereas all direct predecessors of an and-node have to be accomplished. More formally,

$$\Gamma(m, v) \text{ iff } \begin{cases} v \in A \text{ and } m(u) = E((u, v)) \text{ for all } u \in V \\ v \notin A, m = [u] \text{ for some } u \in V, \text{ and } (u, v) \in E \end{cases}$$

We are now ready to define a transition relation $\longrightarrow$ on states. For states s, s', we have that $s \longrightarrow s'$ iff there is a multiset $m \leq s$ such that $\Gamma(m, v)$ and $s' = (s - m) + [v]$.

The meaning of the transition relation is illustrated by the following example. If the attack jungle in question is the one depicted in Figure 1, we can make a transition from the state ["name and address", "make own bomb"] to the state ["name and address", "get a bomb"], and then to the state ["send mailbomb"].

Given this formalism, we can see that the problem of identifying sets of attack components from which we can accomplish a goal node $g \in V$ reduces to the problem of determining reachability in the above transition system. Formally, this amounts to finding the set $A \subseteq \mathcal{M}(\mathbb{C})$ such that for each $a \in A$, $a \xrightarrow{*} G$ where $G \geq [g]$, i.e., the set of initial states in the transition system is $\mathcal{M}(\mathbb{C})$, and the set of final states is $\{G \in \mathcal{M}(\mathbb{C}) : G \geq [g]\}$ In the next section, we will show how to solve this problem.

3 Algorithm

To solve the problem described in the previous section, we will employ the technique of *Symbolic Backwards Reachability*. We first identify an interesting property of the transition system: its *monotonicity*.

A monotonic transition system has the property that for states s_1, s_2, s_3, such that $s_1 \longrightarrow s_3$ and $s_1 \leq s_2$, there is a state s_4, such that $s_2 \longrightarrow s_4$ and $s_3 \leq s_4$. The monotonicity of our transition system follows directly from its definition.

Monotonicity allows for efficient representation of sets of states through the concept of *upward-closed* sets of states, using the minimal element of such sets as representative. The upward closed set represented by a state s is the set $\{s' | s \leq s'\}$. For a monotonic transition system, upward-closedness is preserved by the transition relation. For more details on monotonic transition systems and their properties, see [2].

The idea behind backward reachability analysis is to start with the final state we want to reach, and then successively compute all states from which this target is reachable. If we reach an initial state during this analysis, we know that the target is reachable from the set of initial states.

Algorithm 1: Algorithm for analyzing an attack jungle

```
FIND-ALL-ATTACKS($J, t$)
 1  visited $\leftarrow \emptyset$
 2  working $\leftarrow \{t\}$
 3  while working $\neq \emptyset$
 4        do choose $v \in$ working
 5           for each $w \in Pre_{\longrightarrow}(v)$
 6                do if $\forall u \in$ (visited $\cup$ working) : $u \not\leq w$
 7                      then working $\leftarrow$ working $\cup \{w\}$
 8           visited $\leftarrow$ visited $\cup \{v\}$
 9           working $\leftarrow$ working $- \{v\}$
10  return visited $\cap \mathcal{M}(\mathbb{C})$
```

The algorithm takes as input an attack jungle J and a target state t, and returns the set of minimal initial states from which t is reachable. Typically t would be of the form $[v]$, where v is a sink (i.e. have no outgoing edges), or a node close to a sink. Returning to our example in Figure 1, a typical target would be [“send horse head”]. Taking t to include more than one element allows for analysis of simultaneous goals.

Termination Since the relation $\leq$ on multisets is a *well quasi-order*, our algorithm is guaranteed to terminate [2]. Note that the termination is independent of the structure of the attack jungle. In particular, the existence of cycles in the attack jungle is permitted.

4 Analyzing an Attack Jungle

When we have computed all possible states from which the target is reachable, we can use the framework of [11] to compute certain attributes of the system. The general idea is that we can assign attributes such as cost or difficulty to the attack components, and then automatically compute the cost or difficulty of the attack goal. This is formalized through the use of an *attribute domain*. Intuitively, an attribute domain describes the way in which values are assigned to states, and how to determine which among a set of such values that should be assigned to the final attack goal.

To faithfully model certain attacks, we need to extend the framework of [11]. The reason is that we believe that the concept of *reusability* is crucial to faithfully model certain attacks. Reusability is a property of an attack component, with the meaning that once used, it can be used multiple times at no extra cost. For example, we can think of some information that once acquired, can be taken into account multiple times without being destroyed. This is the case in the attack jungle in Figure 1, where the two attack components “shadow victim” and “dumpster diving” are annotated with a capital **R**, signifying that they are reusable. In particular, the reusability of “shadow victim” means that we can accomplish both “get address” and “get name” by shadowing the victim only once. Not having reusability can lead to problems in the analysis of the result of an attack jungle, since it can cause the analysis to overestimate the cost of attacks.

We now give some preliminaries of how to compute attributes, closely following the presentation of [11]. An attribute $\alpha : \mathbb{C} \to \mathbb{D}$ is a function which given a set $\mathbb{D}$ of attribute values, assigns a value to each attack component. To assign a value not only to individual attack components, but also to states, two functions called the *disjunctive* and *conjunctive combinator* are used. The idea is that for each state, we use the conjunctive combinator to combine the attribute values of the attack components. This way, we get an attribute value for each state. We then use the disjunctive combinator to combine the attribute values of the states into one attribute value for the whole attack jungle. For example, if the question we seek to answer is “What is the minimum cost of performing an attack?”, a reasonable choice for attribute domain would be $\mathbb{N}$. The attribute α would assign to each attack component a value reflecting the cost of accomplishing that attack component. We would use summation to combine the values for a state, as we need to accomplish all components, paying for each attack component individually. Finally, since we ask for the minimal cost, we would the minimum function to combine the values of all attacks into one value.

We now state this formally. The following definition is taken from [11]:
Let $\mathbb{C}$ be a set of attack components. An attribute domain is a structure $(\mathbb{D}, \triangledown, \triangle)$ where $\mathbb{D}$ is the set of attribute values, $\triangledown : \mathbb{D} \times \mathbb{D} \to \mathbb{D}$ is the disjunctive combinator for attribute values and $\triangle : \mathbb{D} \times \mathbb{D} \to \mathbb{D}$ is the conjunctive combinator for attribute values. We require that the combinators are associative and commutative, and that $\triangle$ distributes over $\triangledown$. The associativity and commutativity of the combinators make natural forward to generalize them to multisets.

To handle the concept of reusability, we introduce the *reducing function*, $\rho : \mathcal{M}(\mathbb{C}) \to \mathcal{M}(\mathbb{C})$. Intuitively, $\rho(m)$ is a multiset identical to m except that for all reusable attack components, all duplicates have been discarded. More formally, for each $r \in m$ such that r is reusable, $\rho(m)(r) = 1$ if $m(r) > 0$ and $\rho(m)(r) = 0$ if $m(r) = 0$. Note that we allow for ρ to depend on the particular attribute that we are considering. This allows an attack component to be reusable with respect to one attribute, but some other another attribute.

We can now define the answer to a question about the attack jungle in the following way. Assume an attribute domain $(\mathbb{D}, \triangledown, \triangle)$, an attack jungle J and a target state t. Let Ω be the set of all minimal states from which we can reach the state t, i.e. the output of our algorithm. Also assume a reducing function ρ. Under these conditions, the answer to the question represented by the attribute domain is

$$\triangledown_{s \in \Omega} \triangle_{c \in \rho(s)} \alpha(c)$$

Table 1 shows some examples of attribute domains and corresponding questions.

We conclude this section by applying our analysis to the attack jungle in Figure 2. In Table 2, we give possible attributes to the attack components of

Attribute Domain	Question
$(\{F, T\}, \vee, \wedge)$	Is any attack possible?
$(\{F, T\}, \wedge, \vee)$	Is [some property] needed for all attacks?
$(\mathbb{N}, \min, \max)$	What is the minimum required skill level for an attack?
$(\mathbb{N}, \min, +)$	What is the minimum cost of performing an attack?

Table 1. Examples of attribute domains and corresponding questions.

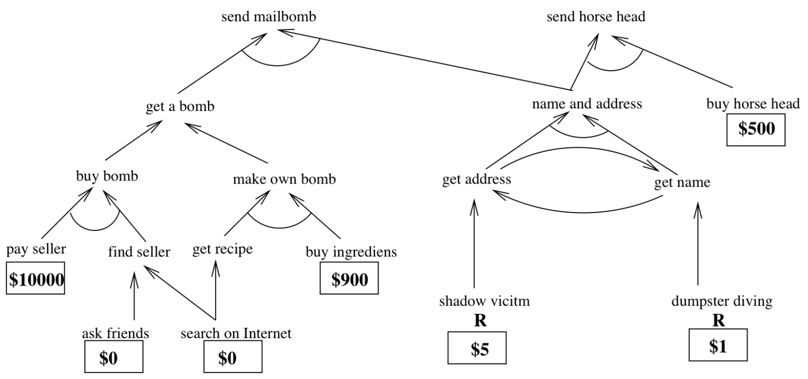

Fig. 2. An attack jungle with attributes.

the attack jungle. We also provide labels for the attack components to enable compact representation of states. The question we are trying to answer in this example is "What is the minimum cost of performing an attack?", and thus we have $(\mathbb{N}, \min, +)$ as our attribute domain. In Table 3, we give the result of the analysis for two different target states.

This example illustrates why reusability can be needed to model attacks. As can be seen from Table 3, the resulting number would be too large if we simply applied our combinator functions to the minimal initial states. This is very problematic since we are trying to find the *minimum* cost, and therefore have to be very conservative in our estimations.

Name of attack component	label	cost (α)
Pay seller	a	\$10000
Ask friends	b	\$0
Search on Internet	c	\$0
Buy ingredients	d	\$900
Shadow victim	e	\$5
Dumpster diving	f	\$1
Buy horses head	g	\$500

Table 2. Labels and costs for the attack components.

Name of attack goal	possible initial states	reduced initial states	cost
Send mailbomb	[a, b, c, d, e, e]	[a, b, c, d, e]	$10905
	[a, b, c, d, e, f]	[a, b, c, d, e, f]	$10906
	[a, b, c, d, f, f]	[a, b, c, d, f]	$10901
	[a, c, c, d, e, e]	[a, c, c, d, e]	$10905
	[a, c, c, d, e, f]	[a, c, c, d, e, f]	$10906
	[a, c, c, d, f, f]	[a ,c, c, d, f]	$10901
		Minimum cost:	$10901
Send horse's head	[e, e, g]	[e, g]	$505
	[e, f, g]	[e, f, g]	$506
	[f, f, g]	[f, g]	$501
		Minimum cost:	$501

Table 3. Initial states and costs for the possible threats

5 Case Study: The GSM network

This section gives a brief description on how attack jungles were used for analyzing the security of the GSM network with focus on the radio access network, called GERAN.

First we give a short description of the GSM system. For a more detailed description of the GSM system see [16, 8]

5.1 The GSM System

Global System for Mobile communications (GSM) is the most common standard for mobile phones in the world. This study focus on the parts of the GSM system that provides basic connectivity and hence leaves parts of the network related to higher layer services out. The GSM system can be grouped into three main parts: the user equipment, the radio access network, and the core network.

User equipment The user equipment (UE) is the part of the GSM network that a user operates. It consists of a smart card used in the subscriber authentication process and the device used to access the network. The device could be a mobile phone or some other equipment with at GSM modem.

The Radio Access Network (GERAN) The radio access network is the part of the GSM network that provides radio connectivity for the UE to the GSM network. It consists of two elements, the Base Transceiver Station (BTS) and the Base Station Controller (BSC). The BTS consists of the radio transmitter receivers, and their associated antennas that communicate directly with the user equipment and is the node that is sometimes referred to as a base station. The BTS provides encryption of the traffic to and from the UE. The BSC controls a group of BTSs. It manages the radio resources and controls items such as allocation of channels and handover within the group of BTSs.

The core network The core network contains a variety of different elements. It provides the main control and interfacing for the whole mobile network. There are two types of accesses, one circuit switched and one packet switched. On a very high level, the difference between circuit switched access and packet switched access is that in the case of circuit switched, a channel is allocated between the two communicating parties when the connection is set-up, and the capacity of the channel cannot be used for other purposes as long as the connection is up. In the case of packet switched, the data transmitted between two communicating parties is packetized and the connection only uses as much capacity as there is data to be transmitted. This gives a more effective utilization of the network resources. The packet switched service was added to the GSM system at a later stage, and introduced two new nodes in the architecture.

The main element of the circuit switched part of the core network is the Mobile switching Services Center (MSC). The MSC provides functionality to enable the requirements of a mobile user to be supported. These include registration, authentication, call location, and call routing to a mobile subscriber. To enable the MSC to perform its functions it requires data from a number of databases. One is the Visited Location Registry (VLR) that contains information about the identities of the UEs that are currently in the area that is served by this network. To make access faster and more convenient, the VLR is (commonly) integrated in the MSC. Another database is the Home Location Register (HLR). It contains administrative information about each subscriber together with their last known location.

The packet switched part of the core network mainly adds two new nodes compared to the circuit switched part; it contains one node called the Serving GPRS Support Node (SGSN). If the UE is attached to the packet switched domain, it is the SGSN that handles the authentication of the user equipment by using information from the HLR database. It is also the SGSN that performs encryption of the data to and from the UE in the packet switched domain. Typically the SGSN is located in a trusted, physically protected place. This gives a higher security compared to the circuit switched case where the encryption is terminated in the BTS, which can be located in physically insecure places (and the links from the BTS to the core network do not have a standardized protection mechanism; sometimes micro wave links are also used for this).

The Gateway GPRS Support Node (GGSN) is the node that provides the UE with an IP point of presence when it is attached to the packet switched domain, it interfaces to external IP networks. The GGSN also collects billing information, provides packet filtering and is responsible for GPRS session management. The architecture of the radio part of the GSM network is shown in Figure 3.

The GSM network provides two main security services to the user: subscriber authentication and encryption of the data sent to and from the UE. In addition to this, there is identity protection by the use of temporary identities.

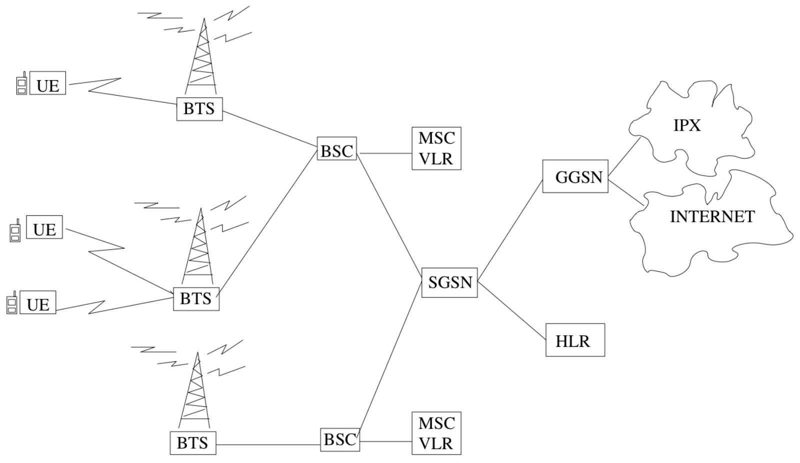

Fig. 3. The architecture of the GSM system

5.2 Creating The GSM Attack Jungle

Security experts studied the GSM network (focusing on GERAN) carefully and possible threats and corresponding attacks towards the security of the system were realized.

All attacks were modeled in an attack jungle using our tool. The complete attack jungle contained 323 nodes.

Table 4 provides a short description of some of the threats (and corresponding attacks) that were identified by the security experts as possible to realize. For a more detailed description containing all the identified threats (and attacks) that were identified in this case study see [1]. The threat "false base station", was already known to be a quite serious threat to GSM, which in fact lead to countermeasures taken in 3G networks (UMTS). Nevertheless, re-confirmation of known security issues is also useful as it helps to validate the model and analysis.

5.3 Analyzing the GSM Attack Jungle

In this particular study, each attack component in the created attack jungle was ranked with the probability that it might occur and each threat was ranked with the seriousness (how big the damage would be if the threat was realized).

Ranking the probability Each attack component was ranked on a scale from one to five, where rank one means "negligible" and rank five "almost certain". An example of an attack with rank one is an attack that is successful with a probability of guessing or breaking an 80bit (cryptographic) key. An attack of rank 5 is an attack that can be performed by single, averagely skilled "hackers" with standard PC/phone resources, possibly using "attacking tools" developed by someone else found on the Internet.

Threat	**Attack(s)**
Sensitive information regarding the user is revealed.	- The UE is fooled to reuse a previously compromised cryptographic key. - The UE is/will be fooled to reuse the same cryptographic key with an insecure encryption algorithm. - The cryptographic key is disclosed by other means.
False base station (BTS)	- Attacker disables (for example cutting antenna connection or cutting the power) a real base station and puts up a false base station. Attacker can now fake a base station towards the network and fake a network towards the UE. The attacker can then easily get the information that is sent.
The UEs are not able to use signalling towards the network.	- An attacker sets up a false BSC (could be implemented in a false base station), that broadcasts a "barred access class" message (unencrypted). This messige disables signalling between the network and a set of UEs. UEs does not try to reconnect after receiving theis message (except for emergency calls).
UE is forced to use a different network.	- An attacker changes an "location update accept" message to an "location update reject" with a cause code of "PLMN not available". This forces the UE to look for another network (this is a man-in-the-middle attack).
The UE accepts faked authentication signalling messages.	- The attacker sends a random number (RAND) that is already used (replayed) to the UE.
Successful impersonation of a subscriber	- For an attacker to sucessfully impersonate a subscriber, cryptanalysis of one of the encryption algorithms that are used has to be done.
Long-term subscriber key is disclosed	- The key is disclosed by preforming cryptanalysis of a encryption algorithm (AKA). - The key is leaked from manufacturer. - The key is leaked when it is installed in the autentication center.

Table 4. A small set of the identified threats and their corresponding attacks.

Ranking the Seriousness The seriousness of the threats (or attack goals) were estimated and ranked on a scale from one to five where one means "minimal" and five "very high". Attack goals of rank one are threats that would not imply anything for user privacy, QoS (Quality of Service), or charging, e.g. being able to occasionally increase a phones transmit power. Attack goals of rank five are attacks that if they are realized it would make frontpage news and seriously damage the trust in mobile networks, either from users or operators point of view.

Risk Assessment By computing the product of the seriousness and the probability for each threat a measure for the risk can be deduced.

For threats with the highest risk protection mechanism (countermeasures) to counter the identified risks can be designed.

Countermeasures Analyzing the attack jungle and determining the high risk threats gives the opportunity to design and evaluate possible countermeasures that are suitable for the system.

In this case study, some of the possible identified countermeasures for the high risk threats are:

– Add new encryption algorithms.
– Remove insecure encryption algorithms.
– Add integrity protection to broadcasted messages.
– Enlarge the size of encryption keys.
– Add integrity protection to broadcast signalling.

A more detailed description of all the identified countermeasures can be found in [1].

To analyze the effect of the identified countermeasures, each countermeasure was added to the attack jungle. The attack jungle was analyzed using our backward reachability algorithm and the security experts could assure that the countermeasures worked as expected. Each countermeasure was evaluated separately and possible advantages, disadvantages, main vulnerabilities and remaining security issues carefully analyzed.

6 Conclusion

We have introduced a formal, methodical way of describing the security of systems called attack jungles. In this model we allow multiple roots, cycles and reusability of resources. The analysis of an attack jungle is done in two steps; first we extract all possible paths in the attack jungle that leads to an attack goal. Secondly, we assign attributes to the attack components in the extracted paths. By decorating the attack components with different attributes we can answer questions such as "What is the minimum cost of performing an attack?" and "What attack is most likely to happen?"

Previously, analyzing the security of systems was traditionally done by hand. However when analyzing a large and complex system such as the GSM network this approach is impossible and the notion and implementation of attack jungles was a necessary and helpful tool for analyzing the security of such a system. Using the ability to model the security of system with cycles, multiple roots and reusable nodes both modeling and the analysis was significantly simplified. The backward reachability algorithm described in this paper easily verifies properties of the attacks. Countermeasures can be added to the model and the algorithm can verify that each countermeasure works as expected and also analyze the behavior of different combinations of several countermeasures.

We have tested our approach by implementing a prototype tool where we can both model and analyze attack jungles. The tool have been used to model possible attacks towards the security in the GSM radio network. A set of possible threats and corresponding attacks towards the security in the GSM radio network where modeled using the tool. For the attacks with the highest risk, countermeasures were designed and the tool was used to evaluate the impact/benefit of implementing the different countermeasures.

When analyzing the attack jungle and the designed countermeasures, the analysis showed that some of the countermeasures was not secure enough to remove all attacks. This should be considered when updating the system since the money and effort put into implementing the countermeasure might not be reasonable.

Several extension can be carried out to the attack jungle formalism. We plan to extend the model by adding weights to each edge in the tree. The weight could for example symbolize how important an attack component are for the ability to accomplish a specific attack. Another direction for future work is to encode attack jungles symbolically and extend our algorithm to work on a symbolic encoding of the attack jungle.

References

1. Specification of `GERAN` vulnerabilities. http://www.3gpp.org/ftp/Specs/html-info/33801.htm, 2007.
2. P.A. Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Inf. Comput.*, 160(1-2), 2000.
3. Isograph AttackTree+. http://www.isograph-software.com/atpover.htm, 2007.
4. S. Bistarelli, M. Dall'Aglio, and P. Peretti. Strategic games on defense trees. In *Formal Aspects in Security and Trust*, pages 1–15, 2006.
5. S. Bistarelli, F. Fioravanti, and P. Peretti. Defense tree for economic evaluations of security investment. In *In ARES06*, pages 416–423. IEEE Computer Society, 2006.
6. S. Bistarelli, P. Peretti, and I. Trubitsyna. Analyzing security scenarios using defence trees and answer set programming. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 197(2):121–129, 2008.
7. J. Brynielsson, A. Horndahl, L. Kaati, C. Mårtenson, and P. Svenson. Development of computerized support tools for intelligence work. In *In Proceedings of ICCRTS*, 2009.

8. Lillian Goleniewski and Kitty Wilson Jarrett. *Telecommunications Essentials, Second Edition: The Complete Global Source (2nd Edition)*. Addison-Wesley Professional, 2006.
9. A. Van Lamsweerde, S. Brohez, R. De Landtsheer, and D. Janssens. From system goals to intruder anti-goals: Attack generation and resolution for security requirements engineering. In *In Proc. of RHAS03*, pages 49–56, 2003.
10. Amenaza Technologies Limited. A quick tour of attack tree based risk analysis using `SecureITree`, 2002.
11. S. Mauw and M. Oostdijk. Foundations of attack trees. In *International conference on information security and cryptology – icisc 2005*, pages 186–198. Springer-Verlag, Berlin, 2005.
12. R. Lippmann O. Sheyner J. Haines, S. Jha and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.
13. B. Schneier. Attack trees. *Dr Dobbs Journal*, 24(12), 1999.
14. O. Sheyner and J.M. Wing. Tools for generating and analyzing attack graphs. In *Proceedings of Workshop on Formal Methods for Components and Objects*, pages 344–371, 2004.
15. A. van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *In Proc. of ICSE04*, pages 148–157, 2004.
16. Steve Wisniewski. *Wireless and Cellular Networks*. Prentice-Hall, Inc., 2004.