Modelling and Analysing Network Security Policies in a Given Vulnerability Setting

Roland Rieke*

Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany
rieke@sit.fraunhofer.de

Abstract. The systematic protection of critical information infrastructures requires an analytical process to identify the critical components and their interplay, to determine the threats and vulnerabilities, to assess the risks and to prioritise countermeasures where risk is unacceptable. This paper presents an integrated framework for model-based symbolic interpretation, simulation and analysis with a comprehensive approach focussing on the validation of network security policies. A graph of all possible attack paths is automatically computed from the model of an ICT network, of vulnerabilities, exploits and an attacker strategy. Constraints on this graph are given by a model of the network security policy. The impact of changes to security policies can be computed and visualised by finding differences in the attack graphs. A unique feature of the presented approach is, that abstract representations of these graphs can be computed that allow comparison of focussed views on the behaviour of the system. This guides optimal adaptation of the security policy to the given vulnerability setting.

Keywords: threats analysis, attack simulation, critical infrastructure protection, network security policies, risk assessment, security modelling and simulation.

1 Introduction

Information and communication technology (ICT) is creating innovative systems and extending existing infrastructure to such an interconnected complexity that predicting the effects of small internal changes (e.g. firewall policies) and external changes (e.g. the discovery of new vulnerabilities and exploit mechanisms) becomes a major problem. The security of such a complex networked system essentially depends on a concise specification of security goals, their correct and consistent transformation into security policies and an appropriate deployment and enforcement of these policies. This has to be accompanied by a concept to adapt the security policies to changing context and environment, usage patterns and attack situations. To help to understand the complex interrelations of security policies, ICT infrastructure and vulnerabilities and to validate security

* Part of the work presented in this paper was developed within the project SicAri being funded by the German Ministry of Education and Research.

goals in such a setting, tool based modelling techniques are required that can efficiently and precisely predict and analyse the behaviour of such complex inter-related systems. These methods should guide a systematic evaluation of a given network security policy and assist the persons in charge with finally determining exactly what really needs protection and which security policy to apply.

A typical means by which an attacker or his malware try to break into a network is, to use combinations of basic exploits to get more information or more credentials and to capture more hosts step by step. To find out if there is a combination that enables an attacker to reach critical network resources or block essential services, it is required to analyse all possible sequences of basic exploits, so called *attack paths*. Based on such an analysis, it is now possible to find out whether a given security policy successfully blocks attack paths and is robust against changes in the given vulnerability setting.

For this type of security policy analysis, a formal modelling framework is presented that, on the one hand, represents the information system and the security policy, and, on the other hand, a model of attacker capabilities and profile. It is extensible to comprise intrusion detection components and optionally a model of the system's countermeasures. Based on such an operational model, a graph representing all possible attack paths can be automatically computed. It is called *attack graph* in the following text. Now security properties can be specified and verified on this attack graph. If the model is too complex to compute the behaviour, then simulation can be used to validate the effectiveness of a security policy. The impact of changes to security policies can be computed and visualised by finding differences in the attack graphs. Furthermore, abstract representations of these graphs can be computed that allow comparison of focussed views on the behaviour of the system. If there are differences in the detailed attack graphs but no differences in the abstract representations thereof, this proves that the different policies are equally effective on the enforcement of security goals on the abstract level, even if variations in the attack paths are covered by different policy rules. The subsequent paper is structured as follows. Section 2 gives an overview of related work. The modelling approach is described in Sect. 3, while Sect. 4 presents an exemplary analysis of network security policy adaptation aspects in a given scenario. Finally, the paper ends with an outlook in Sect. 5.

2 Related Work

The network vulnerability modelling part of the framework presented in this paper is adopted from the approach introduced in [1] and is similar in design to an approach by Phillips and Swiler in [2] and [3]. A major contribution of [1] was the use of abstraction methods to visualise compact presentations of the graph and the inclusion of liveness analysis. Related work of Jha, Sheyner, Wing et al. used attack graphs that are computed and analysed based on model checking in [4] and [5]. Ammann et al. presented an approach in [6] that is focussed on reduction of complexity of the analysis problem by explicit assumptions of monotonicity. Recent work in this area by Noel, Jajodia et al. in [7] and [8]

describes attack graph visualisation techniques while the work of Kotenko and Stepashkin in [9] is focussed on security metrics computations.

To model the ICT network, the vulnerabilities and the intrusion detection systems, a data model loosely resembling the formally defined M2D2 information model [10] is used. Appropriate parts of this model are adopted and supplemented by concepts needed for description of exploits, attacker knowledge and strategy and information for cost benefit analysis.

The model of the network security policies used in this paper is based on the Organisation Based Access Control (Or-BAC) model. A formal approach to use Or-BAC to specify network security policies was presented in [11]. This approach is used here to model the network security policies in the attack graph analysis framework.

The modelling framework is based on Asynchronous Product Automata (APA), a flexible operational specification concept for cooperating systems [12]. An APA consists of a family of so called elementary automata communicating by common components of their state (shared memory). The applied verification method is implemented in the SH verification tool [13] that has been adapted and extended to support the presented attack graph analysis methods.

Major focus of the combined modelling framework presented in this paper, is the integration of formal network vulnerability modelling on the one hand and network security policy modelling on the other hand. This aims to help adaptation of a network security policy to a given and possibly changing vulnerability setting. Recent methods for analysis of attack graphs are extended to support analysis of abstract representations of these graphs.

3 Modelling Critical ICT Infrastructures and Threats

The proposed operational model comprises, (1) an asset inventory including critical network components, topology and vulnerability attributions, (2) a network security policy, (3) vulnerability specifications and exploit descriptions, and (4) an attacker model taking into account the attackers knowledge and behaviour.

3.1 ICT Network Components

The set of all hosts of the information system consists of the union of the hosts of the ICT network and the hosts of the attacker(s). Following the M2D2 model, *products* are the primary entities that are vulnerable. A *host configuration* is a subset of products that is installed on that host and *affects* is a relation between vulnerabilities and sets of products that are affected by a vulnerability. A host is *vulnerable* if its configuration is a superset of a vulnerable set of products and the affected services are currently running.

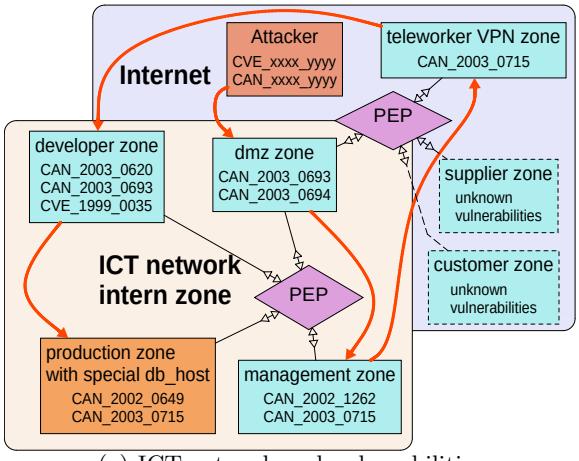
In order to conduct a subsequent comparative analysis of attack paths, an asset prioritisation as to criticality or worth regarding relative importance of a host is required.

3.2 Network Security Policies

The model of the network security policies is based on the Organisation Based Access Control (Or-BAC) model. The approach to use Or-BAC to specify network security policies as presented in [11] is adopted here to model the network security policies in the attack graph analysis framework. The advantage of this choice is, that it is possible to link the policies in the formal model at an abstract level to the low level vendor specific policy rules for the policy enforcement points (PEPs) such as firewalls in the concrete ICT network. Please refer to [11] for such a transformation concept exemplified on the iptables packet filtering mechanism used in Linux.

Following the Or-BAC based concept, the network vulnerability policy is given at an abstract level in terms of *roles* (an abstraction of subjects), *activities* (an abstraction of actions) and *views* (an abstraction of objects). A *subject* in this model is any host. An *action* is a network service such as snmp, ssh or ftp. Actions are represented by a triple of protocol, source port and target port. An *object* is a message sent to a target host. Currently only the target host or rather the role of the target host is used for the view definition here. To specify the access control policy using this approach, *permissions* are given between role, activity and view.

To illustrate the concept described here, a small example scenario is given in Fig. 1(a). Modelling concepts and typical analysis outcome will be illustrated using this example scenario throughout the paper. One possible attack path is sketched in the scenario. The policy rules for the example scenario are defined by the table in Fig. 1(b).



(a) ICT network and vulnerabilities

Role (source)	View (target)	Activity (service)
internet	*internet*	*any*
any	*dmz*	*ssh*
any	*dmz*	*smtp*
dmz	*intern*	*ssh*
intern	*any*	*net*
intern	*internet*	*ftp*
intern	*internet*	*rsh*
intern	*dmz*	*ssh*
db_host	*production*	*rpc*
teleworker	*dmz*	*any*

(b) Network security policy

Fig. 1. Scenario and network security policy

3.3 Vulnerabilities

Vulnerability specifications for the formal model are derived from the Common Vulnerabilities and Exposures (CVE/CAN) descriptions. The MITRE Corporation provides a CVE web site (`http://www.cve.mitre.org/`) with a list of virtually all known vulnerabilities. The CVE name is the 13 character ID used by the CVE standards group to uniquely identify a vulnerability. Additional information about the vulnerabilities also covers preconditions about the target host as well as network preconditions. Furthermore, the impact of an exploitation of a vulnerability is described. The specifications for the formal model of the vulnerabilities additionally comprise the vulnerability *range* and *impact type* assessments provided by the National Institute of Standards and Technology (NIST) (`http://nvd.nist.gov/`).

Vulnerability Severity. The Common Vulnerability Scoring System (CVSS) [14] provides universal severity ratings for security vulnerabilities. These ratings are used in the model as an example for a measure of the threat level. Another example for such a measure is the metric used by the US-CERT (cf. `http://www.kb.cert.org/vuls/html/fieldhelp#metric`). These measures are based on information about the vulnerability being widely known, reported exploitation incidents, number of infected systems, the impact of exploiting the vulnerability and the knowledge and the preconditions required to exploit the vulnerability. Because the approximate values included in those measures may differ significantly from one site to another, prioritising of vulnerabilities based on such measures should be used with caution.

To have a vulnerable product installed on some host, does not necessarily imply, that someone can exploit that vulnerability. A target host *is configured vulnerable*, if (1) the target host has installed a product or products that are vulnerable with respect to the given vulnerability, and (2) necessary other preconditions are fulfilled (e.g. some vulnerabilities require that a trust relation is established as for example used in remote shell hosts allow/deny concepts).

A second precondition to exploit a vulnerability is, that the target host *is currently running the respective products* such as a vulnerable operating system or server version. If a user interaction is required this also requires that the vulnerable product is currently used (e.g. a vulnerable Internet explorer).

The third necessary preconditions is, that the *network security policy permits* that the target host is reachable on the port the vulnerable product is using from the host the attacker selected as source.

3.4 Attacker and System Behaviour

Attacker Knowledge. The knowledge of exploits and hosts and the credentials on the known hosts constitute an attackers profile. Knowledge about hosts changes during the computation of the attack graph because the attacker might gain new knowledge when capturing hosts. On the other hand, some knowledge may become outdated because the enterprise system changes ip-numbers or other

configuration of hosts and reachability. In case a vulnerability is exploited, the model has to cover the *effects for the attacker* (for example, to obtain additional user or root credentials on the target host) and also the *direct impact on the network and host* such as, to shut down a service caused by buffer overflow.

Dynamic System Behaviour. The information model presented so far covers the description of a (static) configuration of an ICT network and its vulnerabilities. In the formal model such a configuration describing the *state* of the ICT network is represented by *APA state components* (APA representation of an ICT network is covered in more detail in [1]).

To describe how actions of attacker(s) and actions of the system can change the state of the ICT network model, specifications of *APA state transitions* are used. These state transitions represent atomic exploits and optionally the actions that the ICT network system can take to defend itself or to implement vital services. Formally, a state transition can occur, when all expressions are evaluable and all conditions are satisfied. So called *interpretation variables* are used to differentiate the variants of execution of the same transition. All possible variants of bindings of interpretation variables from the state components are generated automatically. So for example for a transition modelling an exploit, all possible combinations of bindings of source and target host are computed and further evaluated.

Attacker Behaviour. Attacker capabilities are modelled by the atomic exploits and by the strategy to select and apply them.

A state transition modelling an exploit is constructed from, (1) a predicate that states that the attacker *knows* this exploit, (2) an expression to select source and target hosts for the exploit, (3) a predicate that states that the target *host is vulnerable* by this exploit, (4) an expression for the impact of the execution of this exploit on the attacker and on the target host as for example the shut down of services. Optional add-ons are, an assignment of cost benefit ratings to this exploit and intrusion detection checks.

Several different attackers can easily be included because an attacker is modelled as a role not a single instance and the tool can automatically generate multiple instances from one role definition.

Modelling of Denial of Service (DoS) attacks aiming to block resources or communication channels either directly or by side effects require a much more detailed model of the resources involved. This could be accomplished using the presented framework but is out of scope of this paper.

Some experiments have been made to generate a set of known exploits for the attacker(s) from a given algorithm. If for example it is assumed that the attacker knows 3 different exploits, then all combinations of 3 exploits from the set of all specified exploits have to be computed and further analysed. Another example for an attacker strategy is, that the attacker uses only exploits for vulnerabilities with a severity above a given threshold. This is based on the assumption, that the vulnerability severity reflects the probability of exploitation of a vulnerability.

Composition of a Model and Computation of an Attack Graph. The SH verification tool [13] is used to analyse this model. It manages the components of the model, allows to select alternative parts of the specification and automatically "glues" together the selected components to generate a combined model of ICT network specification, vulnerability and exploit specification, network security policy and attacker specification.

After an initial configuration is selected, the attack graph (reachability graph) is automatically computed by the SH verification tool. Also, on the fly analysis allows, to stop computation automatically when specified conditions are reached (or invariants are broken), so called break conditions can be specified using regular expressions. A violation of a security property for example, can in many cases be specified as a break condition.

Attack Graph of the Example Scenario. The computed attack graph for the simple example scenario (assuming the attacker knows all exploits) has 500 nodes and 4136 edges. Now we assume as a more realistic attacker behaviour, that the attacker will only exploit vulnerabilities with a severity level above a given minimum. In the example scenario, a severity level of 4 results in an attack graph with 178 nodes and 1309 edges. This graph is still far too big to inspect it manually. Figure 2 shows a small section of it. Nodes with circle shape depict states where the successors are completely shown, nodes with rectangular shape depict nodes where the successors are cropped. For example the edge $M4 \longrightarrow M5$ depicts the application of an exploit where the ssh-vulnerability CAN_2003_0693 was used and the edge $M4 \longrightarrow M6$ depicts an exploit based on the same vulnerability but in this case operating stealth (not detected).

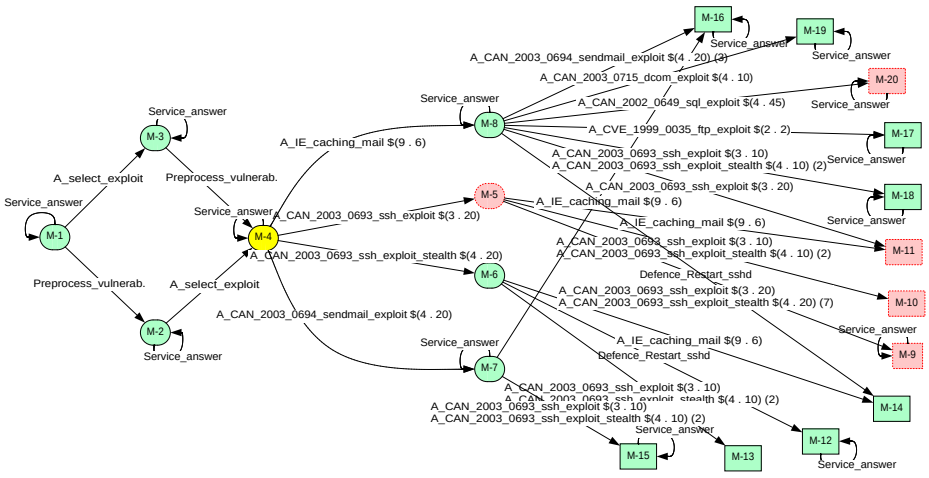


Fig. 2. Attack graph of example scenario (small section)

4 Evaluation of the Model

Abstractions. Abstract representations of the attack graph can be computed to visualise and analyse compacted information focussed on interesting aspects of the behaviour. The mappings used to compute the abstract representations of the behaviour have to be *property preserving*, to assure that properties are *transported* as desired from a lower to a higher level of abstraction and no critical behaviour is hidden by the mapping. Such properties, namely *simplicity*, are given in [15] and and a check for simplicity is implemented in the SH verification tool [13]. In some applications the SH verification tool already computed graphs of about 1 million edges in acceptable time and space. But it is impossible to visualise a graph of that size. So abstraction focussing on some interesting aspect is definitely a comfortable way to go in this case.

An Example for the Usage of Behaviour Abstraction. For this experiment, the vulnerability *range* and *impact type* assessments provided by NIST (cf. Sect. 3.3) are utilised. Range types of the vulnerabilities in the example scenario are *remote* (remotely exploitable) and *local* (locally exploitable). Impact types used here are *unspecific* (provides unauthorised access), *user* (provides user account access) and *root* (provides administrator access).

Step 1 - Define a Mapping. Figure 3 defines a mapping of all transitions representing the exploit of a vulnerability to the respective range and impact types of the vulnerabilities.

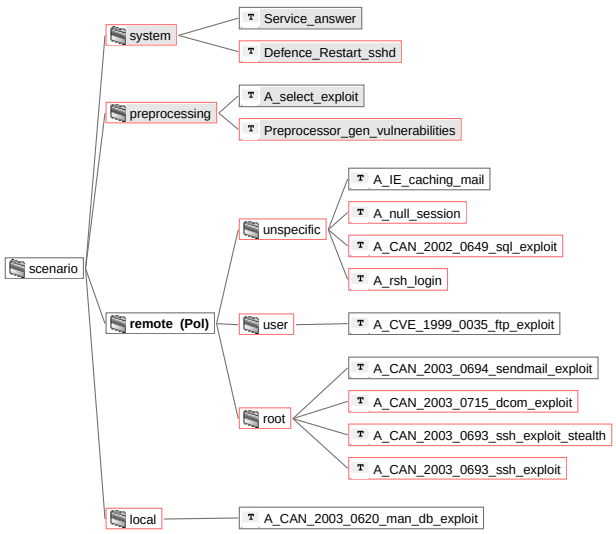


Fig. 3. Definition of an abstract representation of the attack graph

This mapping denotes, that all transitions (the leaves of the tree) are to be represented by their respective father nodes, namely *system*, *preprocessing*, *unspecific*, *user*, *root* and *local* in the abstract representation. The nodes *system* and *preprocessing* are coloured in grey, symbolising that they are mapped to ϵ, that means the transitions represented by these nodes are invisible in the abstract representation. Please ignore the notation (*Pol*) at the node *remote* for the moment.

Step 2 - Compute the Abstract Representation. Figure 4 shows the computed abstract view focussing on the transition types *root*, *user*, *unspecific* and *local*. This graph with only 20 states and 37 edges was derived from the attack graph (cf. Fig. 2) with 178 states and 1309 edges. The simplicity of this mapping that guarantees that properties are preserved was automatically proven by the tool.

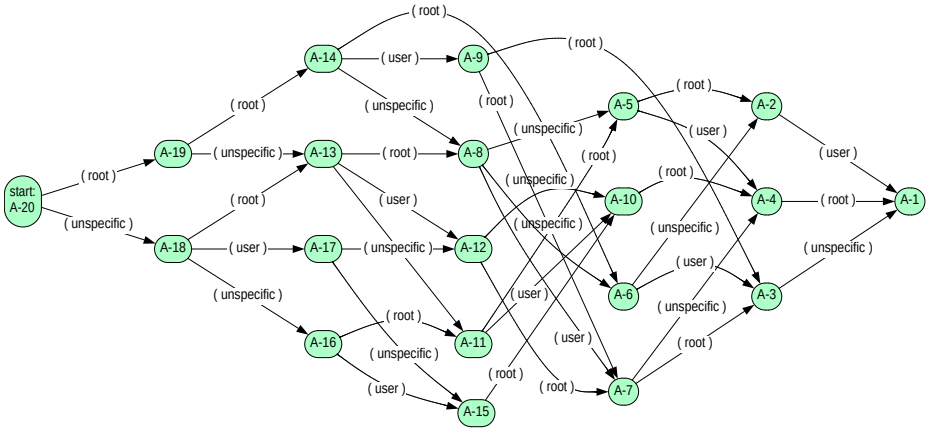
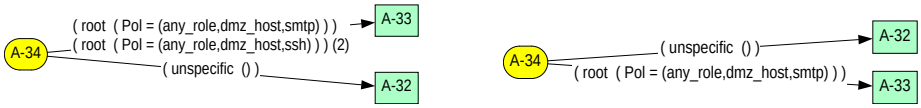


Fig. 4. Abstract view on an attack graph

Step 3 - Optionally Refine the Mapping. If you want to know for example, what policies are responsible to allow the attacks shown in Fig. 4 then a refinement of the abstraction defined in Fig. 3 is necessary. It is possible to "fine tune" the mapping so that the interpretation variables (cf. Sect. 3.4) stay visible in the abstract representation. In this case the binding of the interpretation variable *Pol* that contains the respective policy can be visualised. This is denoted by (*Pol*) in the node *remote* in Fig. 3. The corresponding refined abstract representation is a graph with 34 states and 121 edges when computed on the attack graph in Fig. 2. The initial nodes and edges of this graph are shown in Fig. 5(a). In comparison to the initial edges of the graph in Fig. 4 now the details on the related policies are visible.



(a) $(any\_role, dmz\_host, ssh/smtp)$ (b) $(any\_role, dmz\_host, smtp)$

Fig. 5. Details in the abstract view

Step 4 - Adapt/Optimise the System Configuration. Further analysis reveals, that, if the example policy given in Fig. 1(b) is changed to allow only *smtp* instead of *ssh* and *smtp* for *any_role* to *dmz_host* then the analysis yields a graph with only 94 states and 783 edges and performing the same steps as described above leads to the same graph (Fig. 4) in step 2 but a different one shown in Fig. 5(b) in the refinement step 3.
If alternatively the policy is restricted to allow only *ssh* instead of *ssh* and *smtp* in the above example, then again you get a different attack graph with 167 states and 1203 edges but the abstract view in step 2 is still the same.

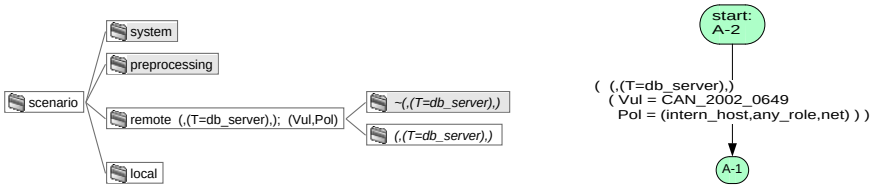
This stepwise analysis demonstrates that there may be differences in the detailed attack graphs but no differences in the abstract representations thereof. This indicates that the different policies are equally effective (or not) concerning the enforcement of security goals on the abstract level, even if variations in the attack paths are covered by different policy rules.

Using Predicates to Define Abstractions. Let us now assume that the host *db_server* in the scenario is the most valuable and mission critical host in the ICT network. So we want to know if in the given scenario, (1) attacks to the *db_server* are possible, (2) on which vulnerabilities they are based, and, (3) what policy rules are directly involved.

The abstraction in Fig. 6(a) exemplifies how predicates can be used to define such a mapping. In this mapping the predicate $(T = db\_server)$ matches only those transitions that model direct attacks to the target host *db_server*. The remote transitions that don't match that predicate are mapped to ϵ and so are invisible.

Evaluating this abstraction on the attack graph from Fig. 2 above results in the simple graph given in Fig. 6(b). This proves that, (1) in the current policy configuration attacks to the *db_server* are possible, (2) those attacks are based on exploits of the vulnerability CAN_2002_0649, and, (3) they are utilising the policy rule $(intern\_hosts, any\_role, net)$. So to prevent this attack, it has to be decided, if it is more appropriate to uninstall the product that is hurt by this vulnerability or to restrict the internal hosts in their possible actions by replacing the above policy with a more restrictive one.

Many further uses of these attack graphs are possible, such as cost benefit analysis or analysis of intrusion detection configurations.



(a) Abstraction used (b) Resulting graph

Fig. 6. Focus on attacks to the host *db_server*

Liveness properties in this context reflect survivability and business continuity aspects. When a system's countermeasures and the behaviour of vital services the system provides are included in the model, then these effects and the system's resilience can be analysed. Please refer to [1] for an example.

5 Further Research Objectives

The work presented in this paper brings together, (1) attack graph computation technology, (2) state-of-the-art policy modelling, and, (3) formal methods for analysis and computation of abstract representations of the system behaviour. The aim is, to guide a systematic evaluation and assist the persons in charge with optimising adaptation of the network security policy to an ever-changing vulnerability setting.

To seamlessly integrate the methods and tool presented here into a network vulnerability analysis framework, a tool-assisted transformation of up-to-date ICT system configuration and vulnerability databases into a formal specification of the model is required. This should preferably be based on automatically updated information of network scanners because administration databases are typically out-of-date. Recent work by Noel, Jajodia et al. in [7] and [8] already covers this aspect but more work is needed to facilitate the transformation of descriptions from vulnerability databases into formal vulnerability and exploit specifications.

A summarisation of severity ratings for single security vulnerabilities as provided by CVSS or US-CERT (cf. Sect. 3.3) based on attack graphs has been addressed in recent work of Kotenko and Stepashkin [9]. Interesting questions in such an approach are, which attacker strategy or bundle of strategies to apply and how to "condense" the information in the graph into a comprehensive measure of the security of an ICT network. Consideration of *resilience against unknown attacks* could also contribute to such a measure.

An even more advanced objective is, to extend this framework to support *policy-based, automated threat response* that makes use of alert information. Such a self-adaptive response mechanism could substantially improve the resilience of policy controlled ICT systems against network attacks.

References

1. Rieke, R.: Tool based formal Modelling, Analysis and Visualisation of Enterprise Network Vulnerabilities utilising Attack Graph Exploration. In: In U.E. Gattiker (Ed.), Eicar 2004 Conference CD-rom: Best Paper Proceedings, Copenhagen, EICAR e.V. (2004)
2. Phillips, C.A., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: NSPW '98, Proceedings of the 1998 Workshop on New Security Paradigms, September 22-25, 1998, Charlottsville, VA, USA, ACM Press (1998) 71–79
3. Swiler, L.P., Phillips, C., Ellis, D., Chakerian, S.: Computer-attack graph generation tool. In: DARPA Information Survivability Conference and Exposition (DISCEX II'01) Volume 2,June 12 - 14, 2001, Anaheim, California, IEEE Computer Society (2001) 1307–1321
4. Jha, S., Sheyner, O., Wing, J.M.: Two formal analyses of attack graphs. In: 15th IEEE Computer Security Foundations Workshop (CSFW-15 2002), 24-26 June 2002, Cape Breton, Nova Scotia, Canada, IEEE Computer Society (2002) 49–63
5. Sheyner, O., Haines, J.W., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: 2002 IEEE Symposium on Security and Privacy, May 12-15, 2002, Berkeley, California, USA, IEEE Comp. Soc. Press (2002) 273–284
6. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM conference on Computer and communications security, ACM Press New York, NY, USA (2002) 217–224
7. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, New York, NY, USA, ACM Press (2004) 109–118
8. Noel, S., Jacobs, M., Kalapa, P., Jajodia, S.: Multiple Coordinated Views for Network Attack Graphs. In: IEEE Workshop on Visualization for Computer Security (VizSec'05), Los Alamitos, CA, USA, IEEE Computer Society (2005)
9. Kotenko, I., Stepashkin, M.: Analyzing Network Security using Malefactor Action Graphs. International Journal of Computer Science and Network Security **6** (2006)
10. Morin, B., Mé, L., Debar, H., Ducassé, M.: M2d2: A formal data model for ids alert correlation. In: Recent Advances in Intrusion Detection, 5th International Symposium, RAID 2002, Zurich, Switzerland, October 16-18, 2002, Proceedings. Volume 2516 of Lecture Notes in Computer Science., Springer (2002) 115–137
11. Cuppens, F., Cuppens-Boulahia, N., Sans, T., Miège, A.: A formal approach to specify and deploy a network security policy. In: Second Workshop on Formal Aspects in Security and Trust (FAST). (2004)
12. Ochsenschläger, P., Repp, J., Rieke, R., Nitsche, U.: The SH-Verification Tool Abstraction-Based Verification of Co-operating Systems. Formal Aspects of Computing, The International Journal of Formal Method **11** (1999) 1–24
13. Ochsenschläger, P., Repp, J., Rieke, R.: The SH-Verification Tool. In: Proc. 13th International FLorida Artificial Intelligence Research Society Conference (FLAIRS-2000), Orlando, FL, USA, AAAI Press (2000) 18–22
14. Schiffmann, M.: A Complete Guide to the Common Vulnerability Scoring System (CVSS) (2005) http://www.first.org/cvss/cvss-guide.html.
15. Ochsenschläger, P., Repp, J., Rieke, R.: Verification of Cooperating Systems – An Approach Based on Formal Languages. In: Proc. 13th International FLorida Artificial Intelligence Research Society Conference (FLAIRS-2000), Orlando, FL, USA, AAAI Press (2000) 346–350