Evaluation of Risk for Complex Systems Using Attack Surface

Leanid Krautsevich, Fabio Martinelli and Artsiom Yautsiukhin
The Institute of Informatics and Telematics of The National Research Council
Via G. Moruzzi 1, Pisa, PI 56124, Italy
Email: {name.surname}@iit.cnr.it

***Abstract*—Many approaches for security assessment were recently proposed. In particular, attack graphs and attack surface gained a lot of attention. Nevertheless, these approaches suffer from several drawbacks. For example, attack graph operates only with known vulnerabilities and it is unclear how attack surface (metric) contributes to the risk picture for a complex system.**

We introduce a novel formal approach for modelling cyberattacks and evaluating of security of complex systems. Our formalisation unites attack surface and attack graph approaches and establishes an explicit link between these approaches and security risk assessment. In this way we are able to exploit the advantages of these three security evaluation approaches in a common framework overcoming many shortcomings of using these approaches separately.

Keywords*—*Attack Surface, Attack Graph, Risk, Complex Systems.

I. Introduction

Complex IT systems do not only significantly facilitate many functionalities of industrial products (corporate networks, electronic motors, SCADA, business processes, etc.) but also expose the products to new types of risk. The risk management practices state that in order to handle risk the organisation should first assess its risk and then define an appropriate strategy for risk mitigation. In this paper, we consider assessment of cyber security risks, i.e., the risks caused intentionally.

Despite a number of proposed methods for the assessment of security of IT systems there is still a need for a reliable, meaningful, cost-effective, widely applicable approach which is able to justify security investments. General risk assessment methods [1], [2], [3] mostly consider a high level model of the evaluated system and rely a lot on the statistics (for quantitative analysis) or on the opinion of experts (for qualitative analysis). The approaches analysing security using low level details (e.g., vulnerabilities) [4], [5], [6], [7] are not general enough to get the overall risk picture. Furthermore, the analysis of concrete software products is usually not considered during the analysis of the system as a whole where these products are applied [8], [9], [10]. Although all these techniques have their advantages when applied to analysis of specific security aspects, we propose a novel comprehensive framework for analysis of complex systems which aggregates the strengths of these existing approaches.

This paper brings together three main techniques for assessing of security: attack surface [9], attack graphs [24], [4], and risk assessment [3]. We start with the fact that despite all effort to secure a component of an IT system an attacker may still compromise the component using known and unknown vulnerabilities in the resources of the component. We understand resources as methods, channels and data items similarly to [9]. The attacker may need to successfully execute several exploits against several components of the system in order to reach her goal. We use a graph to show how a poorly secure components increases the possibility of the attacker to propagate in the the system and achieve her goal. We take into account all possible attacks to the system and formally link a specific attack with the overall risk picture for the system.

The main contributions of this paper are as follows.

- A novel scalable approach for security evaluation that is based on attack surface and attack graph and provides risk as an output.
- A way to gather the information for the evaluation out of system specification, which can be potentially automated or facilitated with a tool support.
- An evaluation method for computation of risk for the system using the low level information.
- An evaluation method for computation of probabilities to compromise the system using the software-related information (attack surface).

The paper is organised as follows. First, we describe the three existing approaches our approach is based on (Section II). In Section III we provide the details for the proposed security model of a complex system. Section IV contains the quantitative analysis of the model which explicitly shows how attack surface affects risk for the system. Conclusions and future work are outlined in Section V.

II. Background

Risk assessment [1], [2], [3] is an approach which computes risk for different threats in terms of losses in some period of time (e.g., Annualised Loss Expectancy (ALE)). Risk assessment usually considers a system as a whole (i.e., with little insight to the detailed structure of the system) and often relies on statistics or experience of the evaluator. The main advantage of the technique is that it is able to justify investments in security. In this work we focus on quantitative risk assessment to establish relations between different techniques formally. Since, quantitative risk assessment relies on statistics the available data should be correct and large enough, which is often not the case. Specific systems with small threat landscape

(e.g., e-motor) are not able to provide enough statistics for analysis. Moreover, it is assumed that past data can be used to approximate risk for the future and this assumption is not yet proven [11]. Finally, risk assessment is very costly and time consuming.

Attack graph [4], [5], [12], [24] is an approach that aggregates the vulnerabilities existing in the system in a graph, which can guide an attacker from an initial set of privileges to a target set. The main advantages of the attack graph is that it can be built automatically using the set of identified vulnerabilities (e.g., by a network scanner) [12], [4]. The graph building tools map the privileges required for exploiting a vulnerability with the resulting privileges received after exploiting other vulnerabilities. A number of qualitative and quantitative methods are proposed for the further graph analysis [13], [4], [14], [15], [22].

The main disadvantages of this technique are the following. An attack graph is a snapshot of a system at the moment of evaluation. Such graphs change after patching existing vulnerabilities, or discovering new vulnerabilities/exploits. Therefore, the graph (and the results of its analysis) changes after a short period of time. Thus, the only way to have up-to-date information is to perform the analysis continuously, which may be costly. Moreover, the technique can be applied only in case of "standard" targets of evaluation. In fact, currently it is applied only to network vulnerabilities. Other vulnerabilities can also be analysed with this approach in theory, but they require some sort of scanning tools and a database of possible exploits (see [16] for an example with social engineering vulnerabilities). Nevertheless, the technique can hardly be used for analysis of systems with a custom-made software, since their vulnerabilities are not known to the network scanners. Moreover, application of the technique in a special unique systems (like SCADA, energy networks, e-motors, etc.) is almost impossible. Finally, it is questionable whether the approach can take into account zero-day vulnerabilities (although some advances in this direction have been done [17]).

Attack surface [8], [18], [9] is an approach that evaluates a software from a point of view of available interactions with it: the more ways a user can interact with the system the less secure it is. Here we would like to note, that the approach itself has no specific quantitative metric for assessment and originally was applied to different versions of similar systems [8]. P. Manadhata and J. Wing [18], [9] advanced the idea and defined a method for security assessment with attack surface. The attack surface metric is similar to risk, but is evaluated using ad-hock parameters: damage-potential (assumed to be proportional to impact) and required effort (assumed to be reverse proportional to probability of success). In this paper we would like to extend the application of attack surface to the analysis of complex systems and show how attack surface for the installed software contributes to risk assessment of the system.

III. Build Graph

First, we define a number of terms used in this article in order to avoid misunderstanding. We do not insist on the common acceptance of these definitions but just define a vocabulary for this specific article. By a *privilege* we mean a special right to interact with the system (e.g., read records for a database, connect to a server, execute a program, etc.). By a *vulnerability* we mean such a configuration of a system which may lead to an unspecified increase of privileges (for an attacker). By an *exploit* we mean an atomic action which uses a vulnerability and leads to increase of privileges for an attacker. Thus, an exploit is bound with the exploited vulnerability. In general atomicity of an action may depend on the granularity of the applied method. By an *attack* we mean a specific sequence of exploits executed by an attacker to achieve its goal. In this paper we assume, that every attacker has only one, specific goal (i.e., reach some set of privileges). By a *threat* we consider all possible attacks leading to achieving a specific goal of the attacker.

Let P be a set of possible privileges in a system and p be an atomic privilege ($p \in P$). Let an account ac be a set of privileges $ac \in \mathcal{AC} = \mathbb{P}(P)$ (we use $\mathbb{P}(P)$ to denote the power set of privileges). Let also $AC \subseteq \mathcal{AC}$ be a set of such accounts relevant for the system. By "relevant" accounts we mean the real accounts which exist in the system (e.g., "guest" or "root" accounts for a Linux operational system). An account can be seen as a set of privileges, which is guarded by an access control system.

An attacker is able to increase her privileges by compromising the system. In other words, the attacker moves from one set of privileges to another, i.e., from one account to another one. The attacker is able to perform such a move by abusing the interactions between the processes (e.g., an attacker performs SQL injection to get the control over an administrator account for a database). The main difference here with attack graph approach is that we do not bind such a link with a specific vulnerability/exploit. We assume, that if there is an interaction between two accounts, then there could exist a possibility for an attacker to increase her privileges. This idea is similar to the one used for attack surface metric, where possible ways to penetrate into software are considered similar to data inputs.

Manadhata et al. [9] have defined three ways of interactions with a system that are called *resources* of a system: via directly exchanging data with a running method (entry and exit points), via sending data to a running service (via system channel), via untrusted data exchange (e.g., though writing and reading a file). In this work we will not go into these details and simply accept that a resource allows one account to interact with another account. Thus, by a resource we mean a running software. Moreover we can consider a user as a resource. People often are considered as the weakest link in IT security and many attacks are purely social (e.g., asking a user to provide his credentials) or socio-technical (e.g., asking user to run a malicious program on his computer). A user may be a system by itself (e.g., a partner), but in our paper it is treated as one account which is later transformed to a resource (see the discussion later in Section III-B).

Let A be a set of actions such that $a = \langle ac_b, r, ac_e \rangle \in A$, where ac_b is an account which the attacker uses to escalate its privileges, ac_e is the desired account, and $r \in R$ is a label denoting the exploited resource (e.g., a software running on behalf of the account to compromise) and R is a set of all available resources. Then we obtain super-actions $sa \in SA$ by simply grouping actions which have the same starting and

ending accounts:

$$sa = \langle ac_b, ac_e \rangle = \{a \in A \mid \forall r \in R,\ a = \langle ac_b, r, ac_e \rangle\} \quad (1)$$

Definition 1: Let AC be the set of all accounts in the system and SA be the set of all super-actions, then $\mathbb{AG} = (AC, SA)$ is an *account graph.*

In short, Definition 1 is a usual definition of a graph with existing accounts as nodes and super-actions as edges. An account graph shows which accounts the attacker should compromise before she is able to reach the desired privileges.

Thus, we start with identification of all possible accounts AC existing in the system. Identification of accounts may be possible by using the system specification. Then, it is required to find all software/modules/services running with the privileges of every account and all users who have credentials for this account. All these resources define the set R. The next step is to check which of these resources may interact with other accounts A (note, that for many social engineering attacks there is no need to have an account at all, i.e., we have an "empty" account). Then, we deduce super-actions SA from the actions as it is shown in Equation 1.

The account graph received with such procedure describes the system completely. Sometimes, such graph contains a lot of information which can be considered as redundant for the aim of the desired analysis. Similar to the attack graph methods [4], [19], we may analyse how the system is protected against a specific attacker. An attacker may be characterised with her initial privileges (e.g., network attacker may start with "empty" or just "guest" account when an initial account of an insider may be more privileged), target privileges and a set of means which the attacker can use for her attack (similar to [22], [20]). In this paper, we do not consider the means possessed by an attacker, but they easily can be added to the model for a more sophisticated analysis. Thus, we are able to reduce the analysed graph by considering only the subgraph formed by the paths from the initial account to the accounts, where the target privileges are satisfied.

A. Assumptions

We would like to discuss the following assumptions for our model.

- *The attacker moves from one account to another one*. Thus, we may think that the attacker loses her privileges, since she does not need the old ones any more. Such vision is applicable if an attacker has a clear plan and simply executes it (without possible deviations and retreats). Alternatively, we may simply keep track of the accounts compromised by the attacker to know all the privileges the attacker possesses at a certain point.
- *The attacker can transfer from one account to another one*, i.e., no additional privileges are needed for the transfer than the ones the attacker possesses at the current state. Keeping track of the compromised accounts may remove this assumption, but make the analysis more complicated.
- *Target accounts for an attacker contain all privileges she aims for*. In most cases the attacker has a specific goal, e.g., get root privileges on a server, which refer to the same account. Keeping track of compromised accounts may be useful in the rare cases, when this assumption does not hold.
- *Similar accounts* (e.g., accounts of the users playing the same role in the organisation) *are considered as separate accounts*. In theory, they can be considered as one, i.e., the same, but we deal with this problem in the future.
- *Access control systems and firewalls are considered robust, i.e., flowless*. This assumption can be easily removed by adding additional channels for access control systems (e.g., access control system may be broken by a brute-force attack) and do not remove resources "guarded" by firewalls.

The first, the second and the third assumptions are linked and deserve a particular attention. These assumptions follow from the type of the chosen graph (exact definition of the meaning for nodes and edges). The type of the graph we consider in the paper is one of the most simple possible graphs. Other two types of graphs used in the literature are:

- *With accumulating privileges* [14], [12], [21], [4], [22], [19], [5]. Every node denotes all privileges the attacker gained "so far". In other words, the attacker, moving through the graph, always accumulates her privileges.
- *With single privileges* [23], [24], [17]. Every node is considered as an atomic privilege and every (hyper)edge starts with several nodes (required set of privileges) and leads to the node(s) with received privilege(s).

All these three types of graphs may describe the same system but make more visible different aspects. The type of graph we use clearly shows the accounts compromised by an attacker and the structure of the system. The graph accumulating privileges is acyclic and allows for analysis with Markov Decision Process (MDP). The last graph helps to specify the minimal set of privileges which is required for compromising the system. Also the first and the third types of graphs do not have the state explosion problem. In fact, the first graph and the third can be considered as the same if we use accounts instead of atomic privileges. Moreover, it is possible to transform our graph to the second type.

The "classical" approach for attack graph starts with definition of existing vulnerabilities (by running a network scanning tool), i.e., edges. The nodes *are derived* from the initial conditions and results of exploiting these vulnerabilities. Our approach starts with determination of accounts (i.e., nodes) and resources (i.e., edges) extracting nodes and edges independently. Therefore, our nodes contain privileges which can be considered redundant for the further actions (but relevant for determination of the potential of the attacker and the potential damage she may cause). Note, that we do not need to reason on the level of atomic privileges (required for third type of attack graphs) since they are encapsulated by accounts.

B. Modularity

One more advantage of the used type of account graph is its modularity.

The complexity of an attack graph is a known problem [25] and the management of such graphs for complex systems may require huge resources. Similar to attack graphs we also would like to show that complexity of account graph may be reduced if we break the system into sub-systems and delegate evaluation of the parts to the corresponding local managers.

A sub-system in our model is just a set of accounts $AC^m \subseteq AC$, which belong to the same organisational unit m. In case we would like to hide the internal details of the sub-system, for secrecy reasons (e.g., the internal structure of a division should not be revealed to the overall system) or for easier management of the graph, we may leave only the accounts which interact with the parts of the complex system external to the module. Let input accounts AC_b^m be the accounts of the model to which there are incoming external edges from the system and let output accounts AC_e^m be the accounts that send data to the system.

The internal part can be hidden with simple rules:

- Substitute all paths from input accounts to output accounts with an action, i.e., $\forall ac_b \in AC_b^m, ac_e \in AC_e^m$ if there is a path π_i from ac_b to ac_e, then remove all edges and nodes which belong to π_i, apart of ac_b and ac_e themselves, from the graph and add an action $\langle ac_b, \pi_i, ac_e \rangle$.
- Collapse all actions from one account to another one into one super-action, i.e., for all such π_i get $\langle ac_b, ac_e \rangle$ (similar to Equation 1).

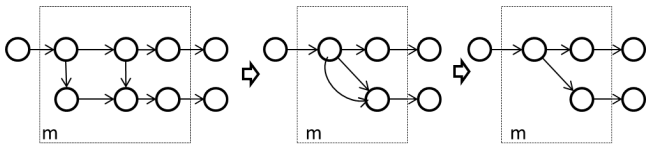

Fig. 1. Example of partial collapse.

With these simple reduction rules we easily hide the internal details and save the structure of the graph (paths). It is similar to actions (first reduction) and super-actions (second one). See Figure 1 for example. We only should keep track of such modifications in order to expand the collapsed parts if necessary. The most important point to note is that such simplification does not violate possible computations of risk as well (this will be shown in the next section).

Furthermore, if the analyst decides to leave only input or output accounts (depending on the analysis to perform), it is possible to collapse the paths in a similar way.

Finally, it is possible to hide the modules completely behind the edges passing "through" the module. See Figure 2 for example. Such simplification is reasonable in cases where the internal structure is unknown, e.g., taking into account a structure of a partner interacting with the system. In this case, we simply accept, that the partner may be compromised in any possible way and then consider the effect of such misuse case on our system.

IV. Risk analysis using account graphs

In this section we show how traditional risk assessment can be fine-grained with our model and how attack surface for a specific resource affects the risk for the whole system.

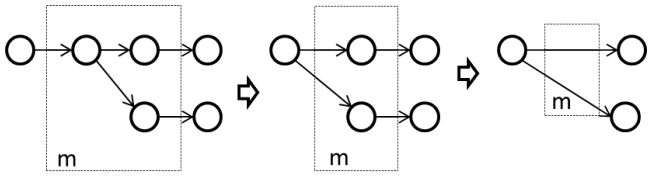

Fig. 2. Example of complete collapse.

A. Computation of risk with account graph

The usual formula for computation of risk $\boldsymbol{Risk}$ is as follows [26]:

$$\boldsymbol{Risk} = \sum_{\forall I} ARO_I \cdot SLE_I \tag{2}$$

where ARO_I and SLE_I are Annualised Rate of Occurrences and Single Loss Expectancy for a threat I correspondingly. Annualised Rate of Occurrences is a number of occurrences of the threat in some period of time (typically, a year). Single Loss Expectancy is a loss from a single threat occurrence.

Now, consider risk computed in our model. Let $\boldsymbol{risk}_I$ be a risk from a threat I. Regarding the definition of threat given in Section III we should consider all possible ways an attacker is able to reach the target account ac_e starting with initial account ac_b, i.e., all paths between the corresponding nodes. Let $\boldsymbol{risk}_i$ be a risk caused by an attacker executing a specific attack $i \in I$, i.e., traversing path π_i from ac_b to ac_e. Risk in this case is:

$$\boldsymbol{risk}_i = \boldsymbol{pr}_i^t \cdot d_i \cdot N_i \tag{3}$$

where d_i is a damage caused by the attack i to the organisation running the system. This damage may be seen as a cost caused directly by the attack, indirectly (via decreasing of reputation), the cost of removal (e.g., virus cleaning), etc. N_i is a number of attempts to execute attack i in a considered period of time (e.g., a year), when N is a total number of attempts for all attacks. The probability $\boldsymbol{pr}_i^t$ is the probability to successfully traverse path π_i. Let super-actions from π_i be ordered from 0 to n, and $\boldsymbol{pr}_{k,k+1}^t$ be the probability to successfully move from account ac_k to the next ac_{k+1}. Then:

$$\boldsymbol{pr}_i^t = \prod_{k=0}^{n-1} \boldsymbol{pr}_{k,k+1}^t \tag{4}$$

Now we show how to find risk for all attacks belonging to the same threat (similar to [20]). For doing this, we need to sum up all risks for specific attacks. Let $\boldsymbol{risk}_I$, $\boldsymbol{pr}_I^t$, d_I, N_I be the parameters for a threat I and let them have similar meanings to the ones defined for an attack i. Let $\boldsymbol{pr}_i^s$ be the probability for the attacker to select attack i. Then, we may say that $N_i = N \cdot \boldsymbol{pr}_i^s$. Let $\boldsymbol{pr}_{k,k+1}^s$ be the probability to select the next step to ac_{k+1} being at account ac_k. Then:

$$\boldsymbol{pr}_i^s = \prod_{k=0}^{n-1} \boldsymbol{pr}_{k,k+1}^s \tag{5}$$

Further:

$$\boldsymbol{risk}_I = \sum_{\forall i \in I} \boldsymbol{risk}_i = \sum_{\forall i \in I} \boldsymbol{pr}_i^t \cdot d_i \cdot N_i = \quad (6)$$

$$(\sum_{\forall j \in I} \frac{\boldsymbol{pr}_j^t \cdot \boldsymbol{pr}_j^s}{\sum_{\forall l \in I} \boldsymbol{pr}_l^s}) \cdot (\sum_{\forall i \in I} \frac{\boldsymbol{pr}_i^t \cdot \boldsymbol{pr}_i^s}{(\sum_{\forall j \in I} \boldsymbol{pr}_j^t \cdot \boldsymbol{pr}_j^s)} \cdot d_i) \cdot N_I$$

The first multiplier in Equation 6 is a mean probability of an attack belonging to threat I to be successful, i.e., $\boldsymbol{pr}_I^t$. The second multiplier is a mean damage among all these attacks, i.e., d_I (here we need a conditional probability that the damage is caused by the attack i if the threat I actually took place). Note, that $\sum_{\forall l \in I} \boldsymbol{pr}_l^s = \boldsymbol{pr}_I^s$ and $N_I = N \cdot \boldsymbol{pr}_I^s$. In short:

$$\boldsymbol{risk}_I = \boldsymbol{pr}_I^t \times d_I \times N_I \quad (7)$$

The result for a threat is of the same form as it is for an attack (see Equation 3). Note, that in classical risk assessment [3], [2] a threat is often defined in more general way (e.g., virus, insider attack). We can see such definition as a set of threats (threats which start with different initial accounts and lead to different target accounts). In this case we can simply apply the same reasoning as we did for aggregating risk of different attacks again, in order to match the definition used in classical risk assessment.

Finally, if we do some re-definitions $SLE_I = d_I$ and $ARO_I = \boldsymbol{pr}_I^t \times N_I$ and sum risks for all threats we will come to the Equation 2.

There are several methods, which use the structure of the graph and which find $\boldsymbol{pr}_{k,k+1}^s$, e.g., using MDP [22], [27], [5] (first we need to transform the graph to the second graph type to remove cycles). On the other hand, $\boldsymbol{pr}_{k,k+1}^t$ is usually considered as given. In this work we would like to make one step further and show how this value can be found if we know probabilities of exploitation for specific resources (see Section IV-B).

Finally, we would like to note, that since $\boldsymbol{pr}_I^t$ and $\boldsymbol{pr}_I^s$ are defined though specified paths, then the proposed methods of hiding details of the graph (Section III-B) do not violate such computation. We only need to re-compute $\boldsymbol{pr}_{k,k+1}^s$ and $\boldsymbol{pr}_{k,k+1}^t$ for the new edges according to the probability operations. For the first step of hiding we simply need to multiply the values $\boldsymbol{pr}_{k,k+1}^t$ for the hidden path to find the probability to traverse the path successfully (and $\boldsymbol{pr}_{k,k+1}^s$ value to find the probability to select the path). For the second step we should sum up values $\boldsymbol{pr}_{k,k+1}^s$ for alternative paths and take average probability of successful exploitation (using conditional probabilities of selection as weights).

B. Effect of attack surface on probability of exploitation

We see attack surface as a set of actions between two accounts, i.e., all actions for one super-action. The impact of an attack surface for a specific super-action on risk for the whole system is straightforward: the larger the attack surface, the more ways for the attacker to move from one account to another one exist, the higher is the probability of success and the higher is the risk for the system.

We are trying to quantify this effect. We cannot use the attack surface metric defined in [9] since the defined metric specify risk to compromise a specific software when we need to consider a global picture. Moreover, the metric defined in [9] even in theory only approximates risk, rather than computes it.

Since the attacker is able to compromise the considered super-action only through some pre-defined actions, then:

$$\boldsymbol{pr}_{k,k+1}^t = 1 - \prod_{\forall l} (1 - \boldsymbol{pr}_l^r) \quad (8)$$

where $\boldsymbol{pr}_l^r$ is the probability to compromise the action l from super-action between accounts ac_k and ac_{k+1}. In other words, the probability in equation is a selection out of probabilities to compromise the resources available for the attacker, i.e., *attack surface probability*. Moreover, now we have a parameter which is system specific ($\boldsymbol{pr}_{k,k+1}^t$) expressed with the probability, which, in most cases, is action specific ($\boldsymbol{pr}_l^r$).

In many cases a resource is a COTS software, e.g., Linux, Firefox, IMAP, etc. The probability to compromise such software can be estimated by external experts and re-used for different specific systems. For example, if the version of the installed software is old, many vulnerabilities are known for it and there are widely available exploits for them (such information could be found in a CVE database) then the probability should be high. The fully patched software can have much lower probability. Moreover, here we may take into account such measures as Common Criteria level [10]. Similar reasoning can be applied for analysis of custom made software.

V. Conclusions and Future Work

Our approach is based on the combination of attack surface, risk assessment and attack graphs. Although, our proposal has many similarities with these three approaches it also provides new features. First of all we have shown how attack graph, attack surface and risk relate to each other. Second, our approach may be used for a detailed analysis of a system even if the concrete actions of the attacker are unknown. This is particularly important, since often we do not know exactly the existing vulnerabilities in the system. Therefore, the proposed approach can be also applied to new or custom systems, where components are unique for the system and no automatic support for searching vulnerabilities exist. This is important for such systems as critical infrastructures where existence of vulnerabilities is usually not tolerated at all. Another advantage of our approach is that the analysis can be shared between different stakeholders, and the complexity of the analysis can be reduced. The whole graph may be seen as a composition of sub-graphs, for which the analysis is performed by local managers, who have deeper knowledge about their sub-systems. If a sub-system is changed then we do not need to redo the analysis completely, but just substitute the changed part and re-compute the affected paths. Finally, the results of the analysis should not only depend on the current security picture, but be also valid for a longer period of time (if we compare with the attack graph approach).

This paper is the initial step in defining the approach, which contains mostly general idea. Much has to be done to apply this approach in practice. In particular, a more practical way of defining accounts is required. Currently we consider

accounts mostly as accounts of an operational systems (e.g., "guest" or "root"), but other software components also have accounts (e.g., accounts for databases, limited access rights for plug-ins of browsers, accounts for enterprise management systems, etc.). On the other hand, many software which also have accounts are not relevant for the analysis, since either these accounts do not differ from security perspective or do not lead to further propagation of attacker's privileges.

Currently, the approach focuses mostly on seizing accounts by attackers and then abusing the received privileges (focusing mostly on injection attacks and attacks on users: social-engineering attacks, session-hijacking, cross-site scripting, etc.). There are many attacks which allow the attacker to simply receive some information. Such attacks can also be added to the model. We need to assume, that an attacker is able to get information from any account reachable from the current account. However, it is difficult to know how the received information may help the attacker to progress in her attack. For example, an attacker may force a password storage software to write the stored passwords in a visible file. Thus, there is a need for a model for the information an attacker may acquire. In this work we also do not consider availability attacks, others than ones that can be performed by abusing the received rights (e.g., executing many instances of "heavy" software or deleting configuration/data files).

Our approach shows how attack surface affects risk in qualitative and quantitative way. Quantitative computation requires low level values, e.g., probabilities and cost. It is widely accepted that such data are hard to find. On the other hand, in our work we have shown, that an analyst may re-use the system-independent information for his/her computations.

VI. Acknowledgements

This research was partially supported by SESAMO (n. 295354) and PRIN Security Horizons (funded by MIUR with D.D. 23.10.2012 n. 719) projects.

References

[1] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis*. Springer, 2011.

[2] C. J. Alberts, A. J. Dorofee, and J. H. Allen, "Octave catalog of practices," Software Engineering Institute, Carnegie Mellon University, Tech. Rep., 2001

[3] G. Stoneburner, A. Goguen, and A. Feringa, "Risk management guide for information technology systems," National Institute of Standards and Technology, Tech. Rep. 800-30, 2001

[4] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2002, p. 273

[5] C. Sarraute, G. Richarte, and J. L. Obes, "An algorithm to find optimal attack paths in nondeterministic scenarios," in *Proceedings of the AISec*. ACM Press, 2011

[6] J. P. McDermott, "Attack net penetration testing," in *Proceedings of the 2000 Workshop on New security paradigms*. New York, NY, USA: ACM Press, 2000, pp. 15–21.

[7] B. B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, and K. S. Trivedi, "A method for modeling and quantifying the security attributes of intrusion tolerant systems," *Performance evaluatin journal*, vol. 1-4, no. 56, pp. 167–186, 2004.

[8] M. Howard, "Fending off future attacks by reducing attack surface," February 4 2003, available via http://msdn.microsoft.com/en-us/library/ms972812.aspx on 22/08/2014.

[9] P. K. Manadhata, K. M. C. Tan, R. A. Maxion, and J. M. Wing, "An approach to measuring a systems attack surface," School of Computer Science. Carnegie Mellon University, Tech. Rep. CMU-CS-07-146, 2007

[10] ISO/IEC, *Common Criteria for Information Technology Security Evaluation*, 2nd ed., Common Criteria Project Sponsoring Organisations, January 2004.

[11] A. Jaquith, *Security metrics: replacing fear, uncertainty, and doubt*. Addison-Wesley, 2007.

[12] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," in *Proceedings of DARPA Information Survivability Conference & Exposition*, vol. 2. IEEE Computer Society Press, June 2001, pp. 307 – 321

[13] S. Jha, O. Sheyner, and J. Wing, "Two formal analys s of attack graphs," in *Proceedings of the 2002 IEEE Computer Society Security Foundations Workshop*. Washington, DC, USA: IEEE Computer Society, 2002, p. 49

[14] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 633–650, 1999.

[15] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup, "A weakest-adversary security metric for network configuration security analysis," in *QoP '06: Proceedings of the 2nd ACM workshop on Quality of protection*. New York, NY, USA: ACM Press, 2006, pp. 31–38

[16] K. Beckers, L. Krautsevich, and A. Yautsiukhin, "Analysis of social engineering threats with attack graphs," in *Proceedings of the 3rd International Workshop on Quantitative Aspects in Security Assurance. To appear.*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2014.

[17] L. Wang, S. Jajodia, A. Singhal, and S. Noel, "k-zero day safety: Measuring the security risk of networks against unknown attacks," in *Proceedings of the European Symposium on Research in Computer Security*, 2010, pp. 573–587

[18] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371 – 386, 2010

[19] K. Beckers, M. Heisel, L. Krautsevich, F. Maritnelli, and A. Yautsiukhin, "Considering attacker motivation in attack graphs analysis in a smart grid scenario," in *Proceedings of the second open EIT ICT Labs workshop on Smart Grid Security. To Appear.*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2014.

[20] L. Krautsevich, F. Martinelli, and A. Yautsiukhin, "Formal analysis of security metrics and risk," in *Proceedings of the IFIP Workshop on Information Security Theory and Practice*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2011, vol. 6633, pp. 304–319

[21] O. Sheyner and J. Wing, "Tools for generating and analysing attack graphs," in *Proceedings of Formal Methods for Components and Objects*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2005

[22] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke, "Model-based security metrics using adversary view security evaluation (advise)." in *Procedings of the 8th International Conference on Quantitative Evaluation of SysTems*, 2011, pp. 191–200

[23] L. Wang, S. Noel, and S. Jajodia, "Minimum-cost network hardening using attack graphs," *Computer Communications*, vol. 29, no. 18, pp. 3812–3824, 2006

[24] L. Wang, A. Liu, and S. Jajodia, "Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts." *Computer Communications*, vol. 29, no. 15, pp. 2917–2933, 2006

[25] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation," in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. New York, NY, USA: ACM Press, 2004, pp. 109–118

[26] L. A. Gordon and M. P. Loeb, *Managing Cybersecurity Resources: a Cost-Benefit Analysis*. McGraw Hill, 2006.

[27] L. Krautsevich, F. Martinelli, and A. Yautsiukhin, "Towards modelling adaptive attacker's behaviour," in *In Proceedings of 5th International Symposium on Foundations & Practice of Security*, ser. Lecture Notes on Computer Science. Springer-Verlag, 2012, vol. 7743, pp. 357–364