4th IFAC Workshop on Dependable Control of Discrete Systems
The International Federation of Automatic Control
September 4-6, 2013. University of York, York, UK

A methodology for qualitative/quantitative analysis of weighted attack trees

Andrea Bobbio, Lavinia Egidi, Roberta Terruggia

DiSIT - Computer Science Institute, Università del Piemonte Orientale "A. Avogadro", Alessandria, Italy
(e-mail: {andrea.bobbio,lavinia.egidi,roberta.terruggia}@unipmn.it)

Abstract: Attack and Defense Trees (ADT) constitute a formal modeling technique that has become dominant in recent years in the area of qualitative and quantitative cybersecurity analysis of ICT and digital control systems. A Weighted-ADT (WADT) is augmented with cost or impact attributes to evidence the most convenient attack sequence in terms of investment budget and provoked damage and to provide an indication on how to mitigate the located breaches by means of suitable countermeasures. The original analysis technique proposed in this paper is based on the representation of a WADT by means of an extension of Binary Decision Diagrams (BDD), called Multi Terminal Binary Decision Diagrams (MTBDD). MTBDDs allow the modeler to evaluate the probability distribution function of the cost and impact related to any possible attack scenario. A running example illustrates the methodology.

Keywords: System security, Weighted Attack and Defence Trees, Multi-Terminal Binary Decision Diagrams.

1. INTRODUCTION

The vulnerability of digital systems to cyber attacks is one of the major threats in the operability of ICT and control systems. There is a need to understand all the different ways in which a system can be attacked, so that appropriate countermeasures can likely be designed to thwart those attacks. The characterization of the attack and the choice of the countermeasures require new conceptual approach and extended analytical tools as described in Kroeger (2008) and Shaw (2012).

Attack Trees (AT) provide a formal way of describing the security of systems subject to various kinds of cyber attacks. Basically, attacks against a system can be represented in a tree structure, with the root node as the final goal and different ways of achieving that goal as hierarchies of events and leaf nodes. Attack trees were introduced by Schneier (1999), as a visual and systematic methodology for security assessment. Since then, they have been widely used both in industrial and academic environments, also for the analysis of Industrial Control and SCADA systems (Byres et al. (2004)).

Each basic leaf of the AT represents an atomic exploit. When each leaf is labelled with the probability that the corresponding exploit is successful, then the model can be used to evaluate the probability that the attacker attains the final goal. Attack trees can be also weighted by adding cost and impact at the level of atomic exploit, or at any level of the AT, thus modelling the fact that different attack strategies can require different budgets and cause different economic damages.

The model can be further enriched by allowing to contrast the system vulnerabilities with countermeasures so that an exploit can propagate only if the countermeasure designed to block it fails. Such structures are called Attack and Defense Trees (ADT) after Roy et al. (2011). Also countermeasures are likely to have a failure probability and a deployment cost. Weighted Attack and Defense Trees (WADT) account for all this.

Previous work Roy et al. (2011) was devoted to derive the minimum value of the cost and the maximum value of the impact. Since in a probabilistic WADT both the *cost* and the *impact* of the attack are discrete random variables, we propose in this paper to enlarge the view by evaluating their distribution, i.e. to find which is the probability of reaching a successful attack at a given cost and with a given impact (cf. Bobbio and Terruggia (2009)).

In order to do this, we propose a new representation and analysis technique for WADTs based on an extension of Binary Decision Diagram, called Multi-Terminal Binary Decision Diagrams (MTBDD). MTBDDs provide a more general and efficient evaluation tool for the weight functions associated to a WADT and allow the modeler to evaluate the probability distribution function of the cost and impact related to any possible attack scenario. The numerical results provided for the example have been obtained from an original software tool already developed by the authors (Bobbio and Terruggia (2009)).

2. MTBDD FOR WEIGHTED ATTACK TREES

An Attack Tree (AT) is a multi-level hierarchical structure based on logical AND and OR operators. The top node is the ultimate goal (we also call it *top event* borrowing this terminology from Fault Tree Analysis). It is reached via intermediate subgoals, represented by the internal gates of the AT. The leaves of the AT represent the atomic attack exploits through which the overall attack can be carried out.

There is no standard way to represent ATs, as witnessed by Byres et al. (2003), Ten et al. (2007) and Kordy et al. (2012). So we choose to use the standardized IEC notation for Fault Tree analysis (IEC-10125 (1990)).

We present our methodology with the aid of a simple example. The goal of the attack we describe is to gain control of a computer. The attack leverages some weaknesses in the system and in the security policy. The attacker can use different strategies: either she logs in as a user in the system and then escalates privileges, or she exploits a buffer-overflow vulnerability to install a back-door. In the latter case the buffer-overflow exploit is successful if the relevant software runs with root privileges. In the former scenario, the attacker gains the log-in credentials either discovering that a default user password to the system has never been changed (O'Harrow (2012)), or guessing a user password that is poorly protected. After she has logged in, she either exploits the software that runs with root privileges to obtain a root shell and then install a back-door to come back at will, or she attempts to read the password file; if there are no read restrictions on this file, she will derive the root password and, again, have full control of the system.

The attack is modelled by the AT of Figure 1. The top event (CTR) represents full success of the attacker, who gains control of the computer. The leaves represent the atomic exploits, in detail:

- *dpwd:* the attacker acquires a default user password;
- *gpwd:* the attacker guesses a user password;
- *login:* the attacker writes username and password at the login prompt;
- *ssw:* the attacker exploits a software that runs as superuser;
- *bo:* the attacker exploits a buffer-overflow vulnerability;
- *pwdf:* the attacker reads the file of the passwords.

The intermediate goals, labelled in capital letters, are

- UC: the attacker obtains user credentials
- UL: the attacker is logged in as a user;
- RC: the attacker runs specifically crafted code;
- RS: the attacker gets a shell as root;
- BD: the attacker installs a backdoor;
- RP: the attacker gets a root password.

It can be seen, for instance, that the attacker needs to obtain user credentials (in either one of the two possible ways), or else her login procedure fails. The exploit *dpwd* takes advantage of a default account on the system that has been forgotten there, and whose password has not been changed as mentioned in O'Harrow (2012); we assume here that the account has only user privileges. Of course exploit *login* is not really an exploit, but it can become an attack tool in this setting.

2.1 Qualitative analysis

The qualitative analysis is intended to find the combinations of elementary events that lead to the Top event (CRT). If the basic leaves are assumed to be Boolean variables with two possible states - $true = 1$ (the exploit is present) and $false = 0$ (the exploit is not present) - the goal of the attack, CTR in Figure 1, is a Boolean function generated as follows:

$$\begin{aligned} CTR &= BD \vee RP \\ &= (RC \wedge RS) \vee (UL \wedge pwdf) \\ &= (bo \wedge ssw \wedge UL) \vee (UL \wedge pwdf) \\ &= (bo \wedge ssw \wedge UC \wedge login) \vee (UC \wedge login \wedge pwdf) \\ &= (bo \wedge ssw \wedge (dpwd \vee gpwd) \wedge login) \vee \\ &\qquad ((dpwd \vee gpwd) \wedge login \wedge pwdf) \end{aligned} \tag{1}$$

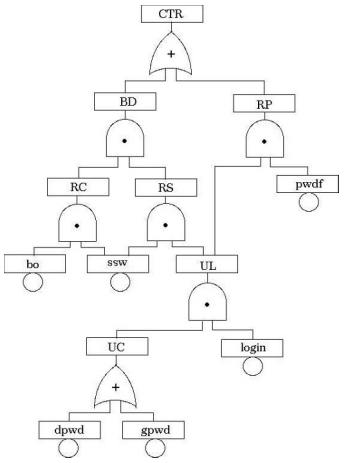

Fig. 1. Cracking a Unix server

Borrowing the terminology from Fault Tree Analysis, we can identify the list of the minimal combinations of elementary attack exploits that lead to the final goal as the minimal cut sets *mcs* of the AT. The order of an *mcs* is the number of its conjuncts. Then, each *mcs* represents an attack strategy and its order is the number of atomic exploits that must be simultaneously successful in order to reach the final goal. Analysis of Equation 1 shows that there are 4 minimal combinations of events that lead to the final attack goal and they are expressed as the following combinations of basic attack leaves:

$$\begin{array}{ll} mcs_1 & bo \wedge ssw \wedge dpwd \wedge login \\ mcs_2 & bo \wedge ssw \wedge gpwd \wedge login \\ mcs_3 & dpwd \wedge login \wedge pwdf \\ mcs_4 & gpwd \wedge login \wedge pwdf \end{array} \tag{2}$$

The first two, mcs_1 and mcs_2, have order 4, whereas mcs_3 and mcs_4 have order 3.

The Boolean function in (1) can be suitably represented and analysed by means of a Binary Decision Diagram (BDD). A BDD is a binary tree used to represent a Boolean function (see Bryant (1986)). Each node of the BDD represents a Boolean variable, corresponding to a leaf of the AT, that can have two successors generated by assigning the value 0 and 1 to that variable. In a graphical representation of a BDD the left branch corresponds to the value 1 and is represented with a solid line, and the right branch (represented by a dotted line) corresponds to the value 0. A BDD has only two terminal leaves, labelled 0 and 1; the value of the Boolean function for a specific assignment to the variables is represented by the leaf at which the corresponding path from the root ends. The BDD of the function of Equation 1 is reported in Figure 2. The variable that is used as pivot at each level of the decomposition is reported on the left of the figure.

All the paths on the BDD of Figure 2 that go from the initial node *login* to the terminal node T, indicate the sequence of actions that can be followed to launch a successful attack and include all the *mcs* of the AT obtained in (2).

Structural Importance Measure. The atomic attack exploits do not have the same criticality in determining the success of the attack. Hence, it is important to rank the exploits according to some structural importance index, that is based on the knowledge of the Boolean function associated to the root (Equation 1) (cf. Barlow and Proschan (1975) and Fricks and Trivedi (2003)).

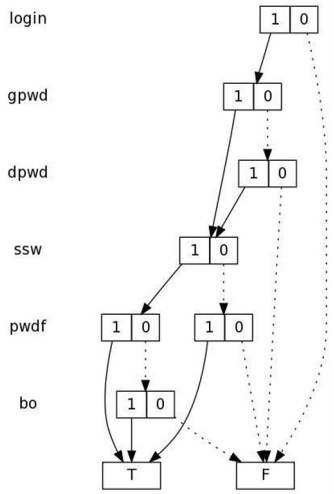

Fig. 2. BDD of the AT of Figure 1

Given an attack tree with n attack leaves, we can build the Boolean function $B(\underline{X})$ where $\underline{X}$ is the n-dimensional vector representing the status (0 or 1) of the leaves. $\underline{X}$ has 2^n possible values by combining the 0 and 1 of the n variables. By definition, $B(\underline{X}) = 1$ when the attack is successful and $B(\underline{X}) = 0$ when the attack is unsuccessful. For each attack leaf $i \in n$ we can apply the so called Shannon decomposition by defining two values of the function $B(\underline{X})$: $B^1_{x_i=1}(\underline{X})$ when the value of x_i is stuck to 1 and $B^0_{x_i=0}(\underline{X})$ when the value of x_i is stuck to 0.

$$\begin{aligned} B^1_{x_i=1}(\underline{X}) &= B(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \\ B^0_{x_i=0}(\underline{X}) &= B(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \end{aligned} \quad (3)$$

Based on Equations (3) the Shannon decomposition, which is also the basic rule for the construction of the BDD following Bryant (1986), becomes:

$$B(\underline{X}) = (x_i \wedge B^1_{x_i=1}(\underline{X})) \vee (\overline{x_i} \wedge B^0_{x_i=0}(\underline{X})) \quad (4)$$

Notice that $B^1_{x_i=1}(\underline{X})$ can assume either values 1 or 0 when the final goal is reached or not reached, being x_i active. Similarly, $B^0_{x_i=0}(\underline{X})$ can assume either values 1 or 0 when the final goal is reached or not reached, being x_i non active. Hence the difference $B^1_{x_i=1}(\underline{X}) - B^0_{x_i=0}(\underline{X})$ is different from 0 (and equal to 1) only in the frontier states for variable x_i that are the states in which the change of value from 1 to 0 of the only variable x_i changes the value of the Boolean function $B(\underline{X})$ from 1 to 0. Based on this observation, we compute $B^1_{x_i=1}(\underline{X})$ and $B^0_{x_i=0}(\underline{X})$ for all the 2^n combinations of the variables and we define the structural importance coefficient of variable x_i as:

$$I^{St}_{x_i} = \frac{\sum_{X \in 2^n} B^1_{x_i=1}(\underline{X}) - B^0_{x_i=0}(\underline{X})}{2^n} \quad (5)$$

where the numerator counts the frontier states for variable x_i, and $I^{St}_{x_i}$ is the percentage of frontier states for x_i.

2.2 Quantitative analysis

Different attack exploits may be carried out with a different probability. If we are able to assign a probability to all the leaves of the attack tree we can calculate the probability of reaching the top event as well as any intermediate event and the probability of the minimal combination of events that lead to the final goal (the *mcs*). If p_i is the probability associated to the presence of the i-th exploit $x_i = 1$, $(1-p_i)$ is the probability associated to the event $x_i = 0$. Combining the theorem of total probability with the Shannon decomposition (4), we can write:

$$P(B(\underline{X})) = p_i \cdot P(B^1_{x_i=1}(\underline{X})) + (1-p_i) \cdot P(B^0_{x_i=0}(\underline{X})) \quad (6)$$

In the present example, we have assumed that the atomic exploits have the probabilities listed in the second column of Table 1. The probabilities have been chosen arbitrarily, since they depend on the adopted security policy. We assume here that it is always possible to access the login prompt, that the file of passwords is protected by default, that default passwords are normally changed but that users are not very reliable in their way of protecting them. We also assume that software patches are not diligently applied and that there is a not too small probability that relevant software runs with root privileges, denoting a limited awareness of security risks.

exploit	*proba bility*	*cost c*	*impact i*	*structural coeff* $I^{St}_{x_i}$	*Birnbaum coeff* $I^{B}_{x_i}$
dpwd	0.3	10	13000	0.08	6.56E-02
gpwd	0.6	60	16000	0.08	1.15E-01
login	1	0	8000	0.23	1.18E-01
ssw	0.3	130	30000	0.05	2.74E-01
bo	0.4	100	10000	0.05	2.05E-01
pwdf	0.05	10	35000	0.14	6.34E-01

Table 1. Probability, cost and impact assigned to the attack exploits of Fig. 1 and, on the right, their importance coefficients

Propagating Equation (6) along the AT of Figure 1 we can compute the probability that the final goal $P(CTR)$ is reached as well as the probability of any *mcs* (i.e. that any strategy of attack is carried out).

Probabilistic Importance Measure. When probabilities of atomic attack exploits are known a new criticality index can be defined to rank the importance of the various leaves in determining the achievement of the final goal. This new measure is called Birnbaum coefficient after Birnbaum (1969), and it is defined as:

$$I^B_{x_i} = P(CTR(x_i = 1)) - P(CTR(x_i = 0)) \quad (7)$$

where:

$P(CTR(x_i = 1))$ is the probability of the root of the tree when leaf x_i is stuck to 1;
$P(CTR(x_i = 0))$ is the probability of the root of the tree when leaf x_i is stuck to 0.

The Birnbaum importance measure of an attack event represents the change in the probability that the final goal is reached caused by the probability difference when the attack exploit is used $(x_i = 1)$ or not $(x_i = 0)$.

Table 1 shows the computed structural importance coefficient $I^{St}_{x_i}$ and the Birnbaum coefficient $I^B_{x_i}$ in columns 5 and 6, respectively. While the structural coefficient reaches its maximum value for the exploit *login*, as expected, since *login* is an element of all the *mcs* (2), the Birnbaum coefficient depends on the probability values assigned to the exploits and has its maximum for the exploit *pwdf*.

3. WEIGHTED ATTACK TREE - WAT

A more effective and detailed analysis of an attack sequence should also include the cost of implementing the attack and the impact (in terms of monetary value) that the attack may have on the attacked system. We call *attack cost* or simply *cost* the cost

of implementing a specific attack exploit, and *impact cost* or simply *impact* the monetary damage caused by the attack. In the present example we assign costs and impacts only to the basic exploits. While for the costs the choice is reasonable since the exploits represent the entry points from which an attack can be initiated, a measure of the economic damage should be assigned to each level of the AT.

The propagation of the cost in the AT occurs with the following rules, as in Roy et al. (2011):

(1) the *cost* (resp. *impact*) in output to an AND gate is the sum of the costs (resp. impacts) of its inputs elements. The rationale behind this propagation rule is that all the inputs must be true for an AND gate to be true and hence their costs sum up.
(2) the *cost* in output to an OR gate is the minimum cost among its inputs while the *impact* in output to an OR gate is, on the contrary, the maximum impact among its inputs. The rationale behind this propagation rule is that in front of a single choice represented by an OR gate, the most convenient strategy for the attacker (corresponding to the worst scenario for the defender) is to select the alternative with the minimum cost and the maximum impact.

A global optimization strategy is more complex and requires a more articulated multi-objective strategy that trades off between costs and impacts as well as takes into account additional measures, such as probability of detection of single exploits and their criticality indices.

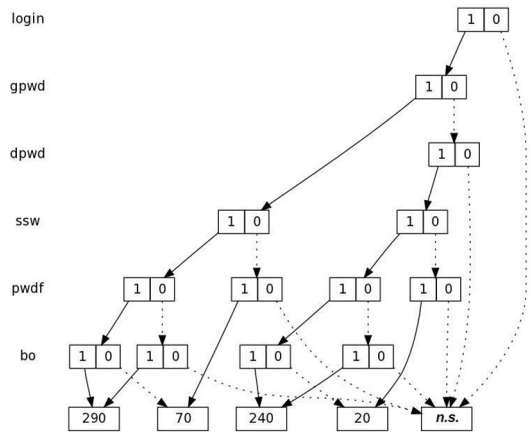

Fig. 3. MTBDD of the cost function computed on the BDD of Figure 2

Columns 3 and 4 of Table 1 report the costs and the impacts assigned as input parameters to the different atomic exploits. We set to 0 the cost of logging in into the system, since if the login prompt is always available, then there is no cost in using it. But it can have a non null impact, as shown by the attacks we are considering. Similarly, the cost of exploiting a default account or of reading the file of the passwords is negligible, if such exploits are at all possible. But they have high impact, especially the latter that exposes the root password. Even if user passwords are not strong and are ill protected, the effort to crack them is more costly. We assign a higher impact to the exploit *gpwd* than to *dpwd* since in the former case the attacker gains access to user data, which could be sensitive; on the other hand, we have assumed that the default user account has been forgotten on the system and therefore the attacker's access to it is dangerous only in that she is now logged in the system.

cost	*probability of successful attack*	
c	*of cost* c	*of cost* $\leq c$
20	1.500E-02	1.500E-02
70	2.100E-02	3.600E-02
240	3.420E-02	7.020E-02
290	4.788E-02	1.181E-01
$n.s.$	8.819E-01	-

Table 2. Attack cost vs mass and cumulative probability

The cost values can be included in the analysis by resorting to an extension of the BDD called Multi Terminal Binary Decision Diagrams (MTBDDs) or Algebraic BDDs (cf., e.g., Clarke et al. (1995) and Bahar et al. (1997)). An MTBDD allows one to represent a real function of Boolean variables as a binary tree. While BDDs have only two terminal leaves 0 and 1, MTBDDs can have more than two terminal leaves that identify all the possible values taken by the Boolean function along the paths from the root to the leaves. Therefore MTBDDs provide a compact representation of weighted Boolean functions by means of the Shannon's decomposition principle. By adopting for the *cost* and for the *impact* the aforementioned propagation rules, (1) (for the AND gates) and (2) (for the OR gates), we can build the MTBDD whose terminal leaves represent the minimum *cost* along that path or the maximum *impact* along that path.

With the cost data of Table 1, the MTBDD of the cost function is reported in Figure 3. The terminal leaves of the MTBDD give the possible minimal costs versus the attack strategy that an attacker should invest to reach the goal; the label $n.s.$ indicates that the attack is not successful. Since the exploits have a probability of being successful, we can evaluate the probability of reaching the final leaves by means of Equation (6). The results are summarized in Table 2. Column 1 reports the values that the total cost c can assume in any possible scenario obtained from the MTBDD of Figure 3. Column 2 is the probability mass of reaching a successful attack goal with the corresponding cost c; the last row $n.s.$ gives the probability that the attack is not successful. Column 3 is the cumulative distribution function i.e. the probability that an attack is successful with a cost $\leq c$. Note that the cumulative distribution is defective, since there is a non null probability that the attack is not successful.

impact	*probability of successful attack*	
i	*of impact* i	*of impact* $> i$
n.s.	8.819E-01	-
56000	5.280E-03	1.128E-01
59000	2.640E-02	8.640E-02
61000	1.440E-02	7.200E-02
64000	7.200E-02	0.0

Table 3. Attack impact vs mass and survival probability

A similar reasoning can be carried on with respect to the exploit impact values of Table 1. Figure 4 represents the MTBDD for the impact function whose distribution is summarized in Table 3. Column 1 reports the possible impacts i computed from the MTBDD of Figure 4, Column 2 the corresponding probability mass and Column 3 the survivor distribution function i.e. the probability that an attack is successful with an impact $> i$. Note that the survivor distribution is defective at the origin with a mass equal to the probability that the attack is non successful (row $n.s.$).

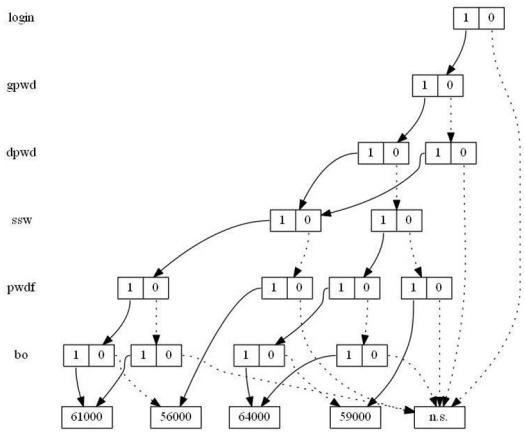

Fig. 4. MTBDD of the impact function computed on the BDD of Figure 2

Notice that the granularity of exploits and the levels of the WAT at which allocate costs and impacts is arbitrarily chosen by the analyst.

4. WEIGHTED ATTACK DEFENSE TREES - WADT

Attack and Defense Trees (ADTs) incorporate defense mechanisms or countermeasures, as described in Ten et al. (2007). This allows to evaluate the effectiveness of a security plan or policy adopted to thwart the identified weaknesses.

The idea behind ADTs is that a countermeasure hinders or mitigates an attack event by reducing the probability that the event occurs (see, e.g., Ten et al. (2007) and Roy et al. (2011)). Therefore, in terms of Boolean logic, the attack attempt propagates if the exploit is successful AND, at the same time, the countermeasure fails to work. So an ADT is an AT that has additional leaves that represent failing defenses, and these leaves are input to an AND gate together with the attack exploit(s) they counter (Roy et al. (2011)). Hence, the probability value associated with a countermeasure is the probability that the countermeasure *fails*.

More than one countermeasure could be activated to hinder a single exploit or, conversely, a single countermeasure could block more than one exploit on different paths to the root event. Moreover, countermeasures can be designed to hinder the achievement of an intermediate attack goal, as opposed to a single exploit.

In our simple case study, we propose the following countermeasures that hinder each a specific exploit:

- *chg:* change default passwords (to counter *dpwd*);
- *spwd:* enforce a strong password policy (to counter *gpwd*);
- *fw:* install a firewall to limit remote access (to counter *login*);
- *lp:* apply least privilege principle in configuring the services (to counter *ssw*);
- *ptch:* apply regularly security patches to the system (to counter *bo*);
- *lim:* limit read privileges to the password file (to counter *pwdf*).

This set of countermeasures encompasses technical measures and policy requirements. For instance to enforce a strong password policy, one can implement ageing passwords together with a check on the strength of the chosen passwords, but it is also necessary to create awareness in users so that they do not paste a note with their password to their monitor. The *fw* countermeasure assumes that remote access is not required for the system at issue.

We assume that the countermeasure least likely to fail is limiting of access privileges to the password file, since that must be done once for all; for the same reason, it has negligible cost. Tasks that must be carried out by the technical staff each time a new service is installed (like *lp* or *chg*) are subject to a higher chance of being disregarded. We consider that changing default passwords has a limited cost, still more than limiting the visibility of the password file, since the former must be done routinely. Applying the least privilege principle might sometimes be complicated and therefore might be willingly overlooked; since it is more complicated, it is more costly in terms of work required and possibly technical training. Enforcing strong passwords requires the cooperation of users and is therefore significantly more prone to failure. It requires both technical tools and an extensive information campaign, which explain its rather high cost.

exploit	*counter measure*	*proba bility*	*cost*
dpwf	*chg*	0.25	30
gpwf	*spwd*	0.4	180
login	*fw*	0.2	300
ssw	*lp*	0.3	60
bo	*ptch*	0.1	50
pwdf	*lim*	0.01	10

Table 4. Probability of failure and cost for installing the countermeasures

Applying patches must be a regular activity that we assume is undertaken with little effort. The cost accounts for the planning necessary in order to make sure that the patching doesn't disrupt regular activities. If the overall system is complex there might be dependencies between the various services run by the system, which might result in the impossibility to upgrade the software timely or even less diligent execution on the part of the technical staff (and therefore higher failure probability). A similar situation might cause higher costs in terms of both technical work and planning. But we have chosen an easier setting.

Finally, installing a firewall limits access to the login prompt. There are ways to bypass a firewall (which accounts for an albeit small probability of failure). The costs include hardware expenses, licensing and technical training of the staff.

For easy reference, Table 4 reports the basic exploits in Column 1 and the corresponding countermeasures in Column 2. Column 3 provides the probability that the countermeasure fails in blocking the attack exploit, and in Column 5 the cost of implementing the countermeasure.

4.1 Balance analysis with countermeasures

Countermeasures have a beneficial effect in reducing the probability that an attack is successful but at a given cost. In order to balance the beneficial effect with the implementation cost new measures and new coefficients need to be defined and evaluated. Since the *mcs* constitute the minimal attack sequences that lead to the final attack, they provide a natural way to initiate the balance analysis, whose main results are reported in Table 5.

mcs	*prob no counterm*	*prob with counterm*	*impact* Im_{mcs}	*cost* Cm_{mcs}	*ROI*
mcs_1	3.60E-02	5.40E-05	61000	440	3.98
mcs_2	7.20E-02	1.73E-04	64000	590	6.79
mcs_3	1.50E-02	7.50E-06	56000	340	1.47
mcs_4	3.00E-02	2.40E-05	59000	490	2.61

Table 5. Probability vs economical balance for installing the countermeasures

Column 1 lists the four *mcs* of the AT of Figure 1, in the same order of Equation (2) that gives the detailed list of exploits pertaining to each *mcs*. Column 2 and 3 give the probability that the final attack is reached along the corresponding *mcs* before and after installing the countermeasures, respectively. A quick comparison shows that the activation of the countermeasures is able to reduce the probability that the attack arrives at the final goal by several orders of magnitude. Column 4 reports the impact caused by a succesful attack along the *mcs*, and is derived from the MTBDD of Figure 4 (see also Table 3). Column 5 reports the cost of implementing the countermeasures for all the exploits forming the *mcs* and it is computed from the values of Table 4.

On the basis of the previous data we are now in a position to evaluate an index from the field of economics that has been adapted by Roy et al. (2011) to the security scenario in order to quantify the nature of the balance between the attacker and the defender. This index is called the *Return on Investment (ROI)* and measures the profit obtained by the implementation of the countermeasures. In synthesis, ROI quantifies the reduction in the impact due to reduction in probability of the attack with respect to the cost of implementing the countermeasures. We evaluate a ROI index for each *mcs* according the following expression:

$$ROI_{mcs_i} = \frac{Im_{mcs_i} \Delta Pr_{mcs_i} - Cm_{mcs_i}}{Cm_{mcs_i}} \qquad (8)$$

With reference to Table 5, Im_{mcs_i} is the impact of the *mcs* of Column 4, ΔPr_{mcs_i} is the difference between Column 2 and 3 and provides the probability reduction without and with countermeasures, and Cm_{mcs_i} is the cost of implementing the countermeasures in Column 5. The numerical values obtained from (8) are reported in Column 6. The ROI data on the table indicate that the most effective investment is in protecting the system from an attack along mcs_2.

5. CONCLUSIONS

The paper has presented a methodology, with an illustrative example, for the analysis of WADT based on an extension of BDD that allows one to efficiently compute various quantitative and probabilistic measures. The advantage of a quantitative analysis of the security of a system is that it gives a rational base to organize the system's protection and to implement countermeasures that optimize the installation investment. The numerical results have been obtained from a software tool called WNRA previously developed by the authors (Bobbio and Terruggia (2009)). For the methodology to be really effective, the security problem must be preceded by a careful analysis of the costs that an attacker is likely to afford, the damage that an attack can produce, and the cost of implementing the countermeasures.

The methodology we propose in this paper is static in nature and therefore doesn't take into account dynamic aspects of an attack. Future work will address dynamic models initiating from Bayesian networks that appear to be very suited for this purpose—see Bobbio et al. (2008).

REFERENCES

Bahar, I., Frohm, E., Gaona, C., Hachtel, G., Macii, E., Padro, A., and Somenzi, F. (1997). Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2-3), 171–206.

Barlow, R. and Proschan, F. (1975). *Statistical Theory of Reliability and Life Testing*. Holt, Rinehart and Winston.

Birnbaum, Z. (1969). On the importance of different components in a multicomponent systems. In P. Krishnaiah (ed.), *Multivariate Analysis - II*, 581–592. Academic Press.

Bobbio, A., Codetta-Raiteri, D., Montani, S., and Portinale, L. (2008). Reliability analysis of systems with dynamic dependencies. In *Bayesian Networks: A Practical Guide to Applications*, 225–238. Wiley.

Bobbio, A. and Terruggia, R. (2009). Reliability and quality of service in weighted probabilistic networks using algebraic decision diagrams. In *Proceedings IEEE Annual Reliability and Maintainability Symposium*, 19–24. Fort Worth, TX.

Bryant, R. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35, 677–691.

Byres, J., Carter, J., Elramly, A., and Hoffman, D. (2003). Worlds in collision-ethernet and the factory floor. In *ISA 2002 Emerging Technologies Conference*. Chicago.

Byres, J., Franz, M., and Miller, D. (2004). The use of attack trees in assessing vulnerabilities in SCADA systems. In *IISW'04*. Lisbon.

Clarke, E., Fujita, M., and Zhao, X. (1995). Applications of multi-terminal binary decision diagrams. Tech.Rep. CMU-CS-95-160.

Fricks, R. and Trivedi, K. (2003). Importance analysis with markov chains. In *Proceedings IEEE Annual Reliability and Maintainability Symposium*, 89–95.

IEC-10125 (1990). *Fault Tree Analysis*. IEC-Standard-No. 10125.

Kordy, B., Pouly, M., and Schweitzer, P. (2012). Computational aspects of attack & defense trees. In *Security and Intelligent Information Systems*, volume 7053 of *LNCS*, 103–116. Springer.

Kroeger, W. (2008). Critical infrastructures at risk: a need for a new conceptual approach and extended analytical tools. *Reliab. Eng. Syst. Safety*, 93, 1781–1787.

O'Harrow, R. (2012). Search engine exposes industrial-sized dangers. http://www.theage.com.au/digital-life/consumer-security/search-engine-exposes-industrialsized-dangers-20120604-1zrnw.html.

Roy, A., Kim, D.S., and Trivedi, S. (2011). Act : Towards unifying the constructs of attack and defense trees. *Security and Communication Networks*, 3, 1–15.

Schneier, B. (1999). Attack trees. *Dr. Dobb Journal of Software Tools*, 24(12), 21–29.

Shaw, W. (2012). SCADA system vulnerabilities to cyber attack. In *Electric Energy online.com*. Available at http://www.electricenergyonline.com/?page=show_article&mag=23&article=181.

Ten, C.W., Liu, C.C., and Manimaran, G. (2007). Vulnerability assessment of cybersecurity for scada systems using attack trees. In *Procs. IEEE Power Engineering Society General Meeting*, 1–8.