Generalized Fault Trees: from reliability to security.

Daniele Codetta-Raiteri
dcr@di.unipmn.it

DiSIT, Computer Science Institute, University of Piemonte Orientale, Alessandria, Italy

Abstract. Fault Trees (FT) are widespread models in the reliability field, but they lack of modelling power. So, in the literature, several extensions have been proposed and introduced specific new modelling primitives. Attack Trees (AT) have gained acceptance in the field of security. They follow the same notation of standard FT, but they represent the combinations of actions necessary for the success of an attack to a computing system. In this paper, we extend the AT formalism by exploiting the new primitives introduced in specific FT extensions. This leads to more accurate models. The approach is applied to a case study: the AT is exploited to represent the attack mode and compute specific quantitative measures about the system security.

Keywords: Attack Trees; Fault Trees; Petri Nets; dynamic gates; repair box; parametric form; simulation; reliability; security.

1 INTRODUCTION

Fault Trees (FT) [1] are widespread models in the reliability field and represent how combinations of component failures (called *basic events*) lead to the system failure (*top event*). Basic events are Boolean variables whose value turn from $false$ to $true$ when the component fails. The intermediate events (subsystem failures) and the top event are Boolean variables as well, with the same semantics of basic events, so their value can be determined by means of *Boolean gates* (AND, OR, etc.). From a FT model, we can obtain the *minimal cut sets* which are the minimal sets of component failures (basic events) determining the system failure (top event). If a probability distribution is associated with basic events, the FT allows the computation of several probabilistic measures, such as the system *unreliability* (the probability that the system is failed at a given time), the probability of minimal cut sets, and importance (sensitivity) indices. The FT modelling power is rather limited, mainly because basic events are assumed to be independent. So, in the literature, several FT extensions have been proposed introducing new modelling capabilities, as described in Sec. 2.

Attack Trees (AT) [2] can be considered the application of FT in the field of security. In other words, an AT follows the same formalism of a FT, but the goal is representing the combinations of actions (basic events) by an attacker, in order to succeed in compromising a system (top event). AT can be used to both graphically represent the attack mode, and assess the system security: both the qualitative analysis (minimal cut sets detection) and the quantitative analysis (computation of probabilistic measures) can be performed.

AT typically exploit only Boolean gates in order to express the attack mode. So, AT and FT have the same modelling power. In this paper, we propose to include in an AT model all the modelling primitives proposed in specific FT extensions, with the goal of designing more accurate FT models expressing more complex attack modes. In particular, in Sec. 3, we model and evaluate a case study by means of an AT including *Boolean gates*, *dynamic gates*, *repair boxes*, and the *parametric form*. The AT model is evaluated by means of Petri Net [1] generation and simulation, with the goal of computing quantitative indices concerning the system security. The case study is taken from [3]; in the current paper, we present the complete model design and evaluation.

2 RELATED WORK

Fault Trees. One of the ways to improve the reliability of a system, consists of replicating its critical components or subsystems; in these cases, the construction and the analysis of the FT may become quite unpractical because the model will be composed by several identical (large) sub-trees representing the replicated parts. *Parametric Fault Trees* (PFT) [4] were proposed with the purpose of providing the compact modelling of such parts. Using PFT, identical sub-trees are folded into a single parametric sub-tree, while the identity of each replica is maintained through the possible values of the parameters. *Dynamic Fault Trees* (DFT) [5] introduced *dynamic gates* representing several kinds of dependency between events: functional dependencies, dependencies concerning the order of events, and the presence of spare components. *Repairable Fault Trees* (RFT) [6] introduced a new primitive called *repair box* representing the presence of a repair process involving a certain set of components, and activated by the occurrence of a specific failure event. In [7] the modelling primitives present in FT, PFT, DFT and RFT formalisms have been integrated into a single formalism called *Generalized Fault Tree* (GFT). So, in a GFT model, we can exploit in a combined way, the compact modelling of redundancies and symmetries, the dependencies between events, the repair of components or subsystems.

Boolean logic Driven Markov Processes (BDMP) [8] are another extension of FT. In particular, BDMP exploit traditional Boolean gates, Markov processes can be associated with basic events, and trigger arcs are used to “activate” a Markov process as a consequence of an event. The elements of the BDMP formalism can model the same dependencies set by dynamic gates in DFT, and other situations such as recovery actions or multi-state components.

Attack Trees. The methodology of AT has become popular and has been applied in several contexts, such as SCADA systems [9, 10]. *Defence Trees* (DT) are an extension of AT where defense mechanisms or countermeasures are incorporated. In particular, in [11], they can be represented by basic events, while in [12, 13], they can appear at any level in the DT; in [14], the point of view of the attacker as well as the point of view of the defender can be analysed. Another case of model adaptation from reliability to security is the application of BDMP models to represent and evaluate attacks [15]. In particular, three types of basic event represent specific types of event during the attack.

Petri Nets. Besides AT, Petri Net based models have been applied to security: *Attack nets* [16] for penetration testing, and *Stochastic Activity Networks* (SAN) [17] with several purposes [18–21]. In general, AT models are easy to build and very readable, but they lack of modelling power because they can only represent the features that gates can express. Petri Net based models can model more complex events, but they are harder to build, less readable and less intuitive to interpret. A trade-off is the generation of Petri Net models from AT. In this way, the attack can be easily represented as a familiar model like AT, and the corresponding Petri Net can be automatically generated, and possibly edited to include further aspects that AT cannot capture. This approach has been applied in several works: in [22] a standard AT is converted into a Colored Petri Net with the aim of evaluating the model. The same goal is achieved in [23], but resorting to *Generalized Stochastic Petri Nets* (GSPN) [1]. The *Petri Net attack modeling approach* (PENET) [24] extends this approach by taking into account also some of the dynamic gates introduced in DFT, such as the *Priority AND* (PAND) gate [5], and exploits *Deterministic timed transitions Petri Nets* (DTTPN).

The same approach is applied in the current work where the AT model is conform to the GFT formalism providing several advantages:

- we can model the presence of recovery, as in the case of DT.
- All the dynamic gates of DFT are available (PAND, SEQ, WSP, FDEP [5]), instead of a subset;
- The parametric form allows to model in a compact way the contemporary presence of several attackers, while AT typically consider a single attacker. This allows the computation of quantitative measures concerning contemporary attempts of attack (Sec. 3.2).
- The presence of all the modelling primitives introduced in FT, PFT, DFT and RFT makes the AT an higher-level model which is more readable with respect to other dynamic extensions of AT or FT, such as BDMP where the dynamic aspects are "hidden" in the basic events or in the trigger arcs, instead of being explicitly represented by means of specific nodes.
- Modelling primitives taken from different formalisms (FT, PFT, DFT, RFT) can be used in a combined way. For instance, a repair box can be applied to a parametric subtree containing dynamic gates (Sec. 3.1).

Event	Description	Mean time to occurrence $1/\lambda$	Occurrence rate λ
v1	occurrence of *v1*	$(24 \cdot 60)\ h$	$0.000694\ h^{-1}$
v2	occurrence of *v2*	$(24 \cdot 90)\ h$	$0.000462\ h^{-1}$
v1REP	recovery of *v1*	$(24 \cdot 10)\ h$	$0.004166\ h^{-1}$
v2REP	recovery of *v2*	$(24 \cdot 7)\ h$	$0.005952\ h^{-1}$
LOGGING_IN	attempt to log-in	$(24 \cdot 2)\ h$	$0.020833\ h^{-1}$
CRACKING	attempt to crack the root password	$24\ h$	$0.041666\ h^{-1}$
GUESSING	attempt to guess the root password	$(24 \cdot 365\ h)$	$0.000114\ h^{-1}$
DISCOVERING	removal of a user logged-in	$24\ h$	$0.041666\ h^{-1}$

Table 1. The mean time to occurrence and the corresponding rate for each event in the case study.

3 THE CASE STUDY

The case study consists of the acquisition by an attacker, of the root password of a Unix server which is periodically characterized by two vulnerabilities: ***v1*** is the possibility that a not authorized user (attacker) logs-in; ***v2*** is the possibility to crack the root password.

The attack is performed in this way: in the time interval between the occurrence of *v1*, and the detection and recovery of *v1*, one or more attackers may try to log-in (event *LOGGING_IN*). After the detection of *v1* (event *v1REP*), the system administrator may discover and remove the not authorized users logged-in (event *DISCOVERING*). In order to detect $v1$, at least one attacker has to be logged-in. The undiscovered attackers keep their presence in the system and may discover the root password in two ways: **1)** trying to crack the root password (event *CRACKING*) during the occurrence of *v2*; **2)** trying to guess the root password (event *GUESSING*); this operation does not require any vulnerability. Also *v2* may be detected and recovered (event *v2REP*). Both *v1* and *v2* may occur again after their recovery. The server becomes compromised if at least one attacker succeeds in discovering the root password.

All the events described above may occur if allowed by the current system state and after an interval of time which is a random variable ruled by the negative exponential distribution. Tab. 1 shows the mean time to occurrence of each event, with the corresponding rate λ. Actually the values of λ have been chosen in an arbitrary way. Probability distributions and rates closer to reality might be obtained by means of statistical investigations.

Some events cannot happen before other ones. For example, the attempt to crack the root password (event *CRACKING*) cannot be performed if the attacker has not succeeded in logging-in and *v2* has not occurred. In a similar way, logging-in may be attempted only after the occurrence of *v1*. These are the temporal dependencies between the events (the symbol $\prec$ specifies that an event must precede another one):

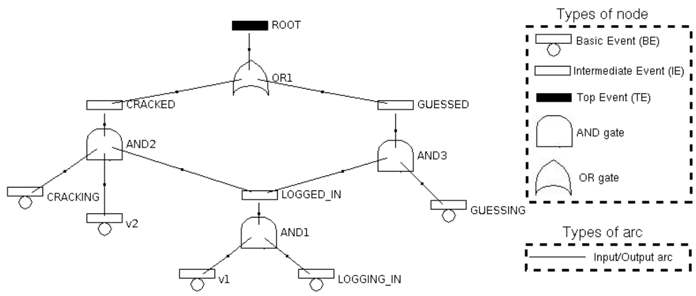

Fig. 1. Standard attack tree model of the case study.

$v1 \prec LOGGING\_IN$
$v1 \prec v1REP$
$(LOGGING\_IN \wedge v2) \prec CRACKING$
$LOGGING\_IN \prec GUESSING$
$(LOGGED\_IN \wedge v1REP) \prec DISCOVERING$
$v2 \prec v2REP$

Some events, once "enabled", may be repeatable. For instance, while *v1* is occurring, one or more attackers may log-in. In a similar way, while *v2* is occurring, one or more attackers may discover the root password. The occurrence and the recovery of a vulnerability are instead alternating events.

3.1 Model design

Standard AT. A preliminary model of the case study is the standard AT shown in Fig. 1 where only Boolean gates are present (Sec. 1). This model represents the attack mode by a single attacker: the event *LOGGED_IN* represents that the attacker has succeeded in logging-in; it is the output of an AND gate, so it occurs if both its inputs events $v1$ and *LOGGING_IN* occurs (Tab. 1). The event *ROOT* models the discovery of the root password and is the output of an OR gate, so it occurs if *CRACKED* or *GUESSED* occurs. The event *CRACKED* represents that the password has been cracked, and occurs if all the events *CRACKING*, *v2* and *LOGGED_IN* have occurred. The event *GUESSED* occurs if both *LOGGED_IN* and *GUESSING* have occurred. This model has several limits:

- it considers only a single attacker, while in the case study more attackers may act at the same time.
- It ignores the temporal dependencies between the events specified above; for instance, in the model, *LOGGING_IN* may occur at any time, before or after $v1$; *CRACKING* may occur before or after *LOGGED_IN* or $v2$.
- It does not take into account the recovery of $v1$ and $v2$.

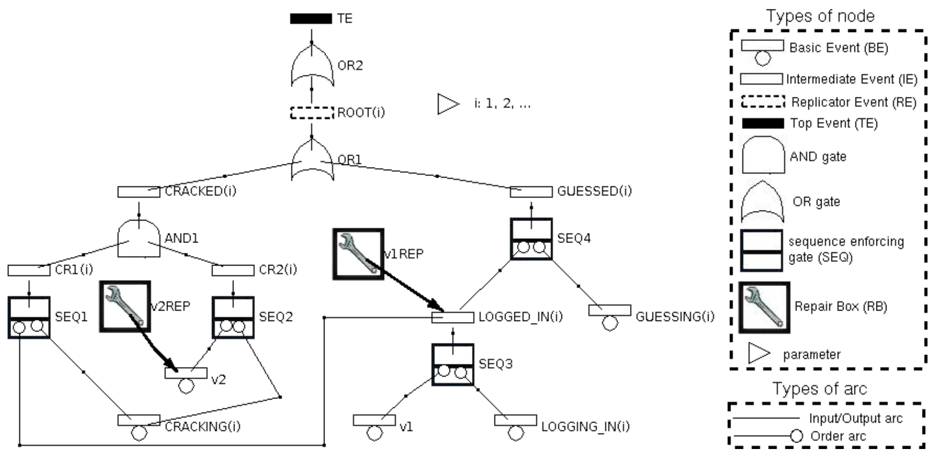

Fig. 2. Attack tree model of the case study, using GFT formalism (the labels of the nodes are explained in Tab. 1).

Such limits can be overcome by the AT shown in Fig. 2. This model contains the modelling primitives collected in the GFT formalism (Sec. 2). The top event (TE), the "root" of the AT, represents the situation where the server is compromised. This happens if at least one attacker discovers the root password.

Parametric form. TE is the output of an OR gate connected to the event *ROOT(i)*, with $i = 1, 2, \ldots$. *ROOT(i)* represents the discovery of the root password by the i-th attacker introduced inside the system. *ROOT(i)* is actually a *replicator event*. This means that the sub-tree below *ROOT(i)* is the compact representation of several sub-trees with the same structure. The identity of each sub-tree is maintained by the possible values of the parameter i which is associated with the events in the sub-tree, with the exception of $v1$ and $v2$ which are instead events shared by all the replicated sub-trees. So, each sub-tree folded in the parametric sub-tree, concerns the actions by the i-th attacker (Sec. 2). *ROOT(i)* is the output of another OR gate, so *ROOT(i)* happens if the i-th attacker succeeds in cracking (event *CRACKED(i)*) or guessing the password (event *GUESSED(i)*).

Dynamic gates. We use three *Sequence Enforcing* (SEQ) gates forcing their input events to occur in a specific order. The output event of this gate corresponds to the last input event in the sequence. The basic event *CRACKING(i)* (attempt to crack the password by the i-th attacker) is connected as second input, to two SEQ gates. Therefore this event may happen only after the success of the log-in by the i-th attacker (event *LOGGED_IN(i)*) and the vulnerability *v2* (basic event *v2*). In the same way, the attempt to log-in by the i-th attacker (event *LOGGING_IN*(i)) may happen only after the vulnerability *v1* (basic event *v1*). Also the event *GUESSING(i)* (attempt to guess the password by the i-th attacker) is connected to a SEQ gate: such event may happen only after the event *LOGGED_IN(i)*.

Repair box. In an AT, the repair box (Sec. 2) can be used to model the recovery of a vulnerability. In Fig. 2, two repair boxes (Sec. 2) are present: the repair box called *v1REP* represents the recovery of *v1* and the detection of the not authorized users logged in. For this reason, *v1REP* is connected to the event *LOGGED_IN(i)* due to the sequence of the basic events *v1* and *LOGGING_IN(i)*. The repair box *v2REP* instead, represents only the recovery of the vulnerability *v2*, so it is connected to the basic event $v2$. The rates of basic events and repair boxes are the values of λ in Tab. 1.

3.2 Model evaluation

Dependencies are present in the model, due to dynamic gates and repair boxes. Therefore it needs the state space analysis; this means generating all the possible system states and stochastic transitions between states. This can be performed by converting the AT into a *Generalized Stochastic Petri Net* (GSPN) [1]. Then, by exploiting the available GSPN solution techniques, we can generate and the analyze the underlying *Continuous Time Markov Chain* (CTMC) [1]. An alternative to analysis is the GSPN simulation. The AT in Fig. 2 is translated into the GSPN in Fig. 3. Both models have been edited by means of *Draw-Net* [25].

In FT, basic events are repeatable only in case of repair. For instance, a component may fail and then, undergo repair, fail again, and so on. In the AT, a basic event may be repeatable for an undefined number of times, even in absence of recovery. For example, while the system suffers from the vulnerability *v1*, an attempt to log-in may occur even if another attempt has already been done. As a consequence, any number of attackers may log-in. For this reason, we did not follow the conversion rules defined in [7] because they are oriented to reliability.

The repetitions of basic events leads the dimensions of the state space to become infinite, so the model cannot undergo analysis. A remedy to this problem consists of setting a limit to the number of repetitions of an event. For instance, we could assume that 10 is the highest number of attackers logged-in. This approach reduces the dimensions of the state space, but they still remain relevant, and the model may not be realistic. So, the GSPN obtained from the AT, has been evaluated using simulation instead of analysis. In this way, we avoid to impose limits to the number of event repetitions, and the simulation execution is less expensive than analysis, in terms of computing complexity. We executed 100000 simulation cycles in order to obtain the results described below.

GSPN. A GSPN contains places (appearing as circles), immediate transitions (black bars) and timed transitions (white bars). Places contain tokens which are moved by transitions in immediate way or after a random period of time.

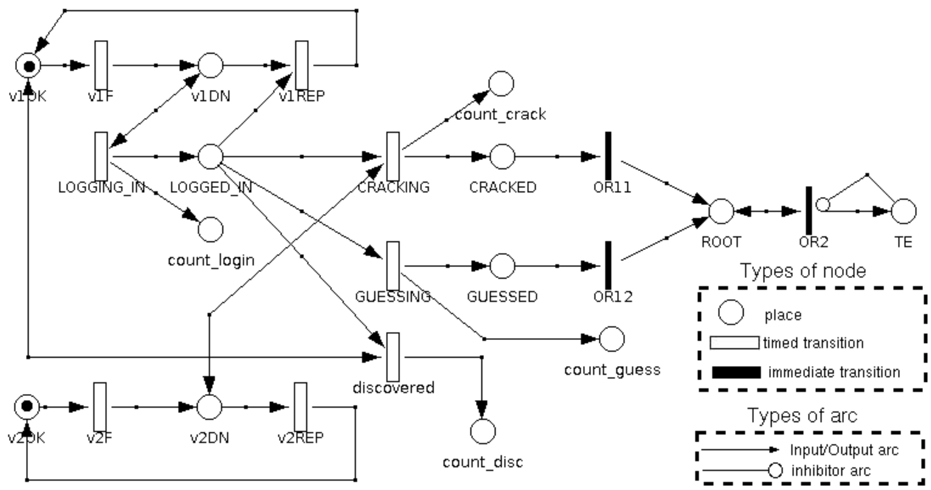

Fig. 3. GSPN model of the case study.

In the GSPN in Fig. 3, the vulnerability $v1$ is modelled by the places $v1OK$ which contains one token in case of absence of $v1$, and $v1DN$ containing one token in case of presence of $v1$. The occurrence of $v1$ is modelled by the transition $v1F$, while its recovery is modelled by the transition $v1REP$. The vulnerability $v2$ is modelled in a similar way. The transition *LOGGING_IN* is enabled by the presence of one token inside the place $v1DN$, and produces the tokens inside the place *LOGGED_IN* corresponding to the number of attackers logged-in. This enables the transition *GUESSING* moving tokens from the place *LOGGED_IN* to the place *GUESSED*, in order to model the success of password guessing. If marked, the place $v1OK$ enables the transition *discovered* removing tokens from the place *LOGGED_IN*, with the purpose of modelling the removal of attackers from the system. The transition *CRACKING* is enabled by the contemporary presence of tokens inside the places *LOGGED_IN* and *v2DN*, and moves tokens from *LOGGED_IN* to *CRACKED*, in order to model the success of password cracking. The tokens inside *GUESSED* or *CRACKED* are moved into the place $ROOT$ by the transition $OR1$ or $OR2$, with the aim of representing that an attacker has obtained the root password. The presence of any quantity of tokens inside $ROOT$ determines the place TE to be marked by one token. This represents that the system is compromised (the place TE corresponds to the top event of the AT).

The rates of timed transitions are the values of λ reported in Tab. 1. With the goal of computing specific indices, further places have been added to the GSPN: *count_login*, *count_disc*, *count_crack*, *count_guess*. They count the number of: successful attempts to log-in, attackers removed from the system, successful attempts to crack the password, successful attempts to guess the password, respectively.

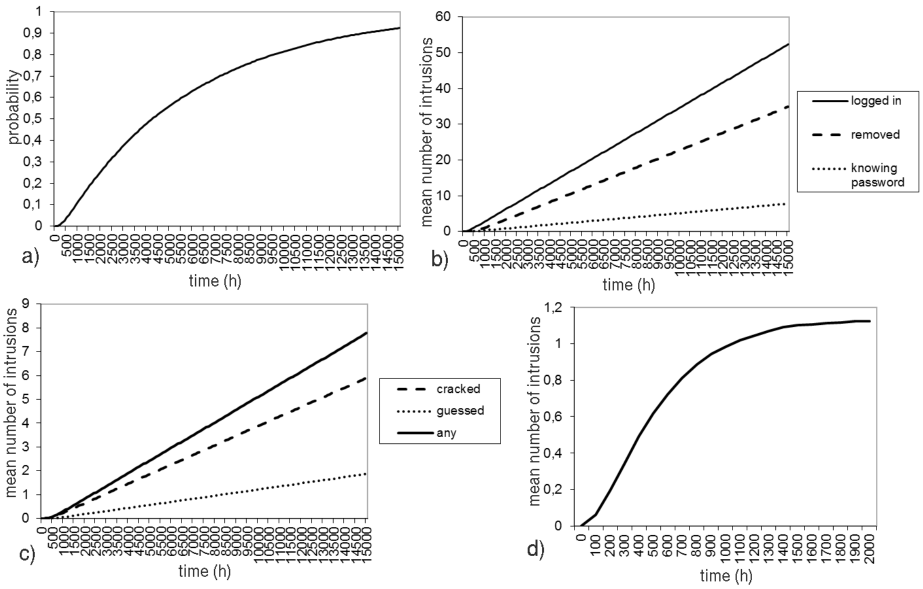

Fig. 4. a) Probability that the system is compromised.
b) The mean number of: intrusions in the system, discovered intrusions, attackers that have discovered the root password.
c) The mean number of: attackers that have discovered the root password, by means of password cracking, password guessing, or any of them.
d) The mean number of undetected attackers that have not discovered yet the root password.

Results. Several measures concerning the system security have been computed, as a function of the time varying between 0 and 15000 hours. Fig. 4.a shows the probability that the system has been compromised. This means that at least one attacker has discovered the root password. This measure may be interpreted as the unreliability of the system (Sec. 1), and has been computed as the mean number of tokens present inside the place TE. Since this number can be 0 or 1, its mean provides a probability value. Fig. 4.b shows the mean numbers of intrusions in the system, discovered intrusions, attackers that have discovered the root password by the performance of password cracking or guessing. These measures have been computed as the mean number of tokens inside the places *count_login*, *count_disc*, $ROOT$, respectively. Fig. 4.c shows the mean number of attackers that have discovered the root password, by means of password cracking, password guessing, or any method. These measures have been computed as the mean number of tokens inside the places *count_crack*, *count_guess*, $ROOT$, respectively. Fig. 4.d shows the mean number of undetected attackers that have not discovered yet the root password (mean number of tokens inside $LOGGED\_IN$). Such measure reaches a steady value equal to 1.13, after about 3000 hours.

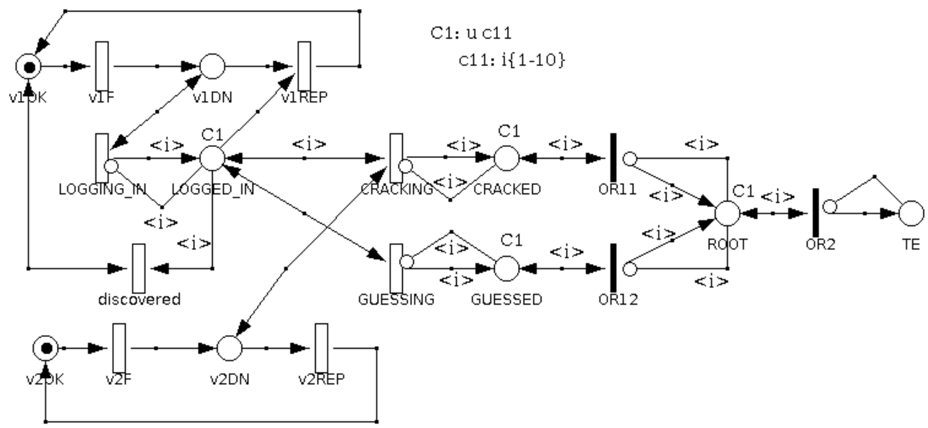

Fig. 5. SWN model of the case study.

SWN. An alternative way to solve the AT model in Fig. 2 is the *Stochastic Well-formed Net* (SWN) [26] shown in Fig. 5, instead of GSPN. In the SWN, tokens can be coloured, so the attackers can be distinguished by means of a colour class ($C1$) associated with the places representing the state of the attack (*LOGGED_IN*, *CRACKED*, *GUESSED*, *ROOT*). A SWN can be analyzed by generating the corresponding symbolic state space whose size is reduced with respect to the ordinary state space. However, the number of colours in a colour class has to be limited, so the number of attackers has to be limited to a certain amount. SWN can undergo simulation as well.

4 CONCLUSIONS

The aim of this paper is transferring our experience about FT extensions, from reliability to security. The GFT formalism defined for reliability evaluation purposes, has been adopted for AT, so that we can model the attack mode by resorting to a single generalized formalism including and integrating Boolean gates, dynamic gates, the parametric form and repair boxes. In this way, the modelling power of AT is improved in a relevant way, so that more accurate models can be designed. The approach has been applied to a case study characterized by recoveries, symmetries and dependencies between events. The current work is a first attempt to use the GFT formalism for AT, so the case study is rather preliminary; however, it serves as proof-of-concept to demonstrate the feasibility of using the GFT formalism, with the consequent improvement of the modelling possibilities. The AT of the case study has been evaluated by conversion into a Petri Net and in particular, a GSPN undergoing simulation. The goal is to avoid the problem of state space explosion, due to the repeatable events.

We believe that the formalism needs further improvements in order to be suitable for the security field. For example, using GFT formalism, the AT model takes into account both the attack mode and the recovery mode. Actually, repair boxes can represent reactive recovery processes. This means that the recovery can be performed only as a consequence of a partial or complete intrusion. The formalism may be extended by taking into account the preventive recovery as well. In this way, preventive countermeasures could be included in the AT model. This was already done in [12, 14], but using only Boolean gates. Moreover, we plan to compute indices which are more security-oriented, with respect to the measures computed in this paper. Importance measures for security are defined in [11, 12], such as *Return on Attack* (ROA) and *Return on Investment* (ROI).

Our intention in the future is solving AT models by means of *Dynamic Bayesian Networks* (DBN) [27], already exploited for DFT and RFT analysis. The advantage is the possibility of easily modelling multi-state components and computing predictive, diagnostic, or importance measures conditioned by observations about the system or components state. In the security field, observations may concern the action by intruders, the presence of vulnerabilities or countermeasures. We plan to use AT as an high-level model to represent the attack mode and generate the corresponding DBN.

References

1. Sahner, R., Trivedi, K., Puliafito, A.: Performance and Reliability Analysis of Computer Systems; An Example-based Approach Using the SHARPE Software Package. Kluwer Academic Publisher (1996)
2. Schneier, B.: Attack trees. Dr. Dobb Journal of Software Tools **24**(12) (1999) 21–29
3. Codetta-Raiteri, D.: A preliminary application of generalized fault trees to security. In: International Conference on Security and Cryptography, SCITEPRESS (July 2013) 609–614
4. Bobbio, A., Franceschinis, G., Gaeta, R., Portinale, L.: Parametric fault tree for the dependability analysis of redundant systems and its high-level Petri net semantics. IEEE Transactions on Software Engineering **29**(3) (March 2003) 270–287
5. Dugan, J.B., Bavuso, S.J., Boyd, M.A.: Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems. IEEE Transactions on Reliability **41** (1992) 363–377
6. Codetta-Raiteri, D., Franceschinis, G., Iacono, M., Vittorini, V.: Repairable Fault Tree for the automatic evaluation of repair policies. In: International Conference on Dependable Systems and Networks, Florence, Italy, IEEE Computer Society (June 2004) 659–668
7. Codetta, D.: Extended Fault Trees Analysis supported by Stochastic Petri Nets. PhD thesis, Dipartimento di Informatica, Università di Torino (November 2005)
8. Bouissou, M., Bon, J.L.: A new formalism that combines advantages of fault-trees and Markov models: Boolean logic Driven Markov Processes. Reliability Engineering and System Safety **82**(2) (November 2003) 149–163
9. Byres, J., Franz, M., Miller, D.: The use of attack trees in assessing vulnerabilities in SCADA systems. In: International Infrastructure Survivability Workshop, Lisbon (2004)

10. Ten, P.C.W., Liu, C.C., Govindarasu, M.: Vulnerability assessment of cybersecurity for SCADA systems using attack trees. In: Power Engineering Society General Meeting. (2007) 1–8
11. Bistarelli, S., Fioravanti, F., Peretti, P.: Defense trees for economic evaluation of security investments. In: International Conference on Availability, Reliability and Security, IEEE Computer Society (2006)
12. Roy, A., Kim, D.S., Trivedi, K.S.: Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees. Security and Communication Networks **5**(8) (2012) 929–943
13. Zonouz, S.A., Khurana, H., Sanders, W.H., Yardley, T.M.: RRE: A game-theoretic intrusion Response and Recovery Engine. In: International Conference on Dependable Systems & Networks, IEEE Computer Society (2009) 439–448
14. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of attack–defense trees. Formal Aspects of Security and Trust (2011) 80–95
15. Piètre-Cambacédès, L., Bouissou, M.: Attack and defense modeling with BDMP. In: Computer Network Security. Springer (2010) 86–101
16. McDermott, J.P.: Attack Net Penetration Testing. In: Workshop on New security paradigms. (2000)
17. Sanders, W., Meyer, J.: Stochastic activity networks: Formal definitions and concepts. Lecture Notes in Computer Science **2090** (2001) 315–343
18. Gupta, V., Lam, V., Ramasamy, H.V., Sanders, W.H., Singh, S.: Dependability and performance evaluation of intrusion-tolerant server architectures. In: Dependable Computing. Springer (2003) 81–101
19. Van Ruitenbeek, E., Sanders, W.H.: Modeling peer-to-peer botnets. In: International Conference on Quantitative Evaluation of Systems, IEEE (2008) 307–316
20. Singh, S., Cukier, M., Sanders, W.: Probabilistic validation of an intrusion-tolerant replication system. In: International Conference on Dependable Systems and Networks, IEEE Computer Society (2003) 615–624
21. Codetta-Raiteri, D., Nai, R.: Evaluation of communication scenarios inside the Electrical Power System. International Journal of Modelling and Simulation **30** (September 2010) 345–352
22. Helmer, G., Wong, J., Slagell, M., Honavar, V., Miller, L., Wang, Y., Wang, X., Stakhanova, N.: Software fault tree and coloured petri net–based specification, design and implementation of agent-based intrusion detection systems. International Journal of Information and Computer Security **1**(1) (2007) 109–142
23. Dalton, G., Mills, R.F., Colombi, J.M., Raines, R.A.: Analyzing attack trees using generalized stochastic Petri nets. In: Information Assurance Workshop, IEEE (2006) 116–123
24. Pudar, S., Manimaran, G., Liu, C.C.: PENET: A practical method and tool for integrated modeling of security attacks and countermeasures. Computers & Security **28**(8) (2009) 754–771
25. Codetta-Raiteri, D., Franceschinis, G., Gribaudo, M.: Defining formalisms and models in the Draw-Net Modeling System. In: International Workshop on Modelling of Objects, Components and Agents, Turku, Finland (June 2006) 123–144
26. Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications. IEEE Transactions on Computers **42** (1993) 1343–1360
27. Portinale, L., Bobbio, A., Codetta-Raiteri, D., Montani, S.: Compiling dynamic fault trees into dynamic Bayesian nets for reliability analysis: The Radyban tool. CEUR Workshop Proceedings **268** (2007)