Model-Based Vulnerability Assessment of Self-Adaptive Protection Systems*

Ricardo J. Rodríguez[1] and Stefano Marrone[2]

[1] Research Institute of Applied Sciences in Cybersecurity
University of León, Spain
[2] Dip. di Matematica e Fisica
Seconda Università di Napoli, Italy
rj.rodriguez@unileon.es, stefano.marrone@unina2.it

Abstract. Security mechanisms are at the base of modern computer systems, demanded to be more and more reactive to changing environments and malicious intentions. Security policies unable to change in time are destined to be exploited and thus, system security compromised. However, the ability to properly change security policies is only possible once the most effective mechanism to adopt under specific conditions is known. This paper proposes a model-based approach to accomplish this goal: a vulnerability model of the system is built by means of a model-based, layered security approach, and used to quantitatively evaluate the best protection mechanism at a given time and hence, to adapt the system to changing environments. The evaluation relies on the use of a powerful, flexible formalism such as Dynamic Bayesian Networks.

Keywords: Cybersecurity modelling, Vulnerability Evaluation, Dynamic Bayesian Networks, Adaptive Protection Systems

1 Introduction

Many modern computer systems are needed and used for our day-to-day living. For instance, e-mail, digital media news, or search engine websites are continuously visited by Internet users – as pointed out by Alexa's traffic top ten ranking websites. These systems are exposed to Internet users with malicious intents (i.e., attackers) who may disrupt its normal operation, thus leading to service unavailability and even financial losses.

Cyber-security helps protecting against these risks first by identifying assets targeted by an attacker and then applying security measures (or policies) to protect them. However, the ever-changing nature of Internet hinders the effectiveness of these security policies, normally built upon a good design and relying on manual tuning [1]. Although a system can improve security in several ways (e.g., more secure communication protocols, additional security devices,

* This work was partially supported by Spanish National Cybersecurity Institute (INCIBE) according to rule 19 of the Digital Confidence Plan (Digital Agency of Spain) and the University of León under contract X43.

strict security policies), this improvement does not come at zero cost and affects other non-functional properties, such as availability, Quality-of-Service, or performance [2].

In that sense, systems able to change their behaviour and/or structure in response to the environment and the system itself have become a recent, important trend in the research community [3]. *Self-adaptive systems* can become a great solution to critical systems where failures or malfunctions may result in catastrophic consequences [4]. For instance, a system can be secured following a self-adaptive approach where its resilience, dependability, and configuration may change depending on several external conditions (e.g., a system may redirect connections to a spare server with a different OS when detects several intrusion attempts to the original server). Of course, self-adaption to different situations does not come either at zero cost [5].

In this paper, we propose a model-based approach that allows to represent critical systems with self-adaptive capabilities. Our approach relies on security layers that relate to them, being also complementary and increasing system's complexity. A security layer comprises sensors that monitor external conditions and produce an assessment, reporting when an attack attempt is discovered. When this happens, our approach counteracts adding other security layer, thus enhancing the security of the system. Vulnerability assessment that is a prime step in the definition of such systems is performed by means of formal models – in particular, using Dynamic Bayesian Networks (DBN)[3]. An example inspired by a real network intrusion case is used to illustrate our approach.

The paper is organised as follows. Section 2 relates closest works in the literature and previous concepts. Section 3 introduces our model-based, layered security approach while Section 4 describes the DBN modelling guidelines. Section 5 shows the application to a network intrusion case. Section 6 states conclusions and future work.

2 Background and Related Works

DBN are a way to extend Bayesian Networks (BN) to model the probability distributions of a collections of random variables taking time into account [6]. In such variables, the Conditional Probability Table (CPT), the mean the probability distribution function of a (D)BN variable is defined, also considers the dependency from previous values of other variables (as well as from the variable itself). Figure 1 shows a sample DBN model where a variable Y is subject to other variable X inside a single slice time: this dependency is represented by the continuous arrow from X to Y. Variables may also influence the value of others in a future time slice. This is the case of X that influences its own future value X', represented by the dashed line from X to X'. In this sample model, Y_t depends on X_t while X_t depends on X_{t-1}. In this sense, the term "dynamic" means we are modelling a dynamic system, not that the network changes over time.

[3] In this paper, we use DBN interchangeably as singular and plural acronym

Vulnerability evaluation represents a challenging topic especially in the field of cyber-protection: the high mutability of the knowledge on both systems and protection systems causes a never-ending process where new defence mechanisms are followed by the discovery of new exploitations in a very short time. For these reasons, security of cyber-physical infrastructures is often considered a multi-faceted, multi-disciplinary problem that requires an integrated approach [4, 7].

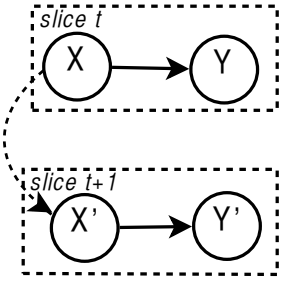


Fig. 1. A DBN Example.

In security analysis, several modelling methods integrate attack and defence aspects, at different level of abstraction: Garcia addresses both of them, but attacks are described at a high level of detail [8]; Defence trees (an extension of attack trees [9]) accounts for both attacks and countermeasures [10]; Attack Response Trees incorporate both attack and response mechanism notations [11]. Another meaningful example of integration of different aspects of modelling and sensing techniques in [12] where a hierarchical approach to intrusion detection is boosted by the usage of Knowledge Engineering techniques. Other works rely on the quantitative use of other formalisms such as Generalized Stochastic Petri Nets (GPSN) [13] and Bayesian Networks (BN) [14].

Notwithstanding their flexibility, DBN have not received the right attention from the scientific community. With respect to the cited works, the proposed approach aims at exploit all the features of the DBN: the DBN formalism has been chosen to conjugate modelling expressiveness and capability to analyse models. In synthesis, DBN are more powerful with respect to BN and more simple to use and analyse than GSPN. Hence, with respect to the works in [13] and in [14], this paper respectively improves time to analyse the model (allowing run-time usage possible for large systems) and introduces relationships between events in time (that is not possible with plain BN).

To best of our knowledge, there are few works using DBN for vulnerability and security analysis [15, 16]. With respect to these works, this paper proposes a DBN modelling approach where, respectively, the focus is not only on attack phases but can also capture the dynamic and evolving relationships between attack actions and defence mechanisms; and DBN are not used as machine learning tool to infer knowledge about the attacker habits, but as a tool for applying such knowledge in protection system design. In fact, it could be possible a synergy between these approaches.

3 The Onion Security Model

Our approach relies on security layers. We define a *Security Layer* (SL) as a UML Component Diagram (UML-CD) which comprises a *Sensor* component as depicted in Fig. 2. A UML-CD is a UML structural diagram used to illustrate

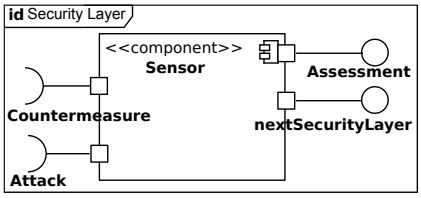


Fig. 2. A *Security Layer* model element.

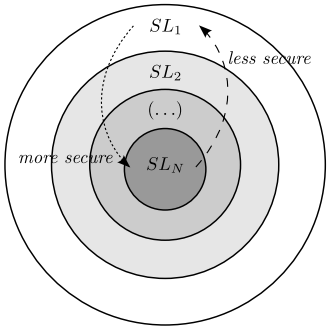


Fig. 3. The *Onion security model*.

pieces of software and/or controllers that make up a system [17]. A *Sensor* defines a model of a system aimed at capturing and interacting with real-world events (e.g., a firewall, an Intrusion Detection System (IDS), or a network load-balancer). The *Sensor* has two inputs (left hand of the component) and two outputs (right hand).

The *Attack* input port describes the sequence of steps leading to a successful attack on the system. This sequence of steps can be modelled, for instance, by means of a state-chart diagram. Output ports are *Assessment*, which expresses the quantitative assessment performed by the *Sensor*, and *nextSecurityLayer*, which allows for connecting an SL with other SL' through *Countermeasure* input port. Thus, different security layers are connected through *Countermeasure*/*nextSecurityLayer* ports. Note that these diagrams are only used for representing the static part of the system: its dynamics is expressed by the DBN model, introduced in the next section.

Figure 3 sketches our model-based, layered approach using aforementioned security layers. Several security layers are defined in a system to be protected, where each security layer SL_{i+1} is more secure (and consequently more restrictive somehow) than SL_i. The specific moment of transition among layers SL_i, SL_{i+1} is determined by $Assessment_i$, depending on the input $Attack_i$.

4 Modelling Guidelines for DBN

The objective of this section is to guide the reader to structure a DBN model according to the proposed approach. Figure 4 shows the proposed overall DBN model schema. The actions accomplished by the *Attacker* are modelled by an **Attack** submodel: this submodel communicates system *events* to a **Sensor** submodel whose aim is to represent real world sensing concerns (e.g., honeypots, monitoring systems, log analysers can all be modelled as sensors). Detected events (*detections*) are then communicated to an **Assessment** submodel. This last submodel is in charge to analyse detected events and to decide whether an attack is actually occurring, then assigning a proper SL as a countermeasure.

Here, we propose a modelling approach able to capture the adaptive feedback given by the **Assessment** model to the **Sensor** model: one of the simplest adaptive strategy may consist in (de)activating sensors when some operating conditions require a more accurate estimation of the suspected threat. The model also highlights proper *alarms* raised to the *Security Operators*. In the following, we focus on the single submodels showing how some recurrent situations can be modelled with "DBN patterns".

4.1 Attack Modelling

We start from the general hypothesis that *an attack is a sequence of attacking steps*: each of these is modelled by a DBN variable ranging $\{Unattempted, Ongoing, Succeed, Failed\}$ values. In the following, these values are expressed by their initials.

The most simple DBN pattern is constituted by a single attack step which evolves in time and does not depend on other steps. Hence, the value of this variable is a function only of its value at previous stage. Figure 5 shows the DBN submodel representing this situation and its related CPT. The parameters used in the CPT are: the probability P_{start} of an attacker that attempts to exploit the step; the probability P_{end} of the action ending in a single time slice; and the quantification of the vulnerability (i.e., the probability P_{succ} of a successful attack to the vulnerability). Finally, $P_s = P_{end} \cdot P_{succ}$, and $P_{ns} = P_{end} \cdot (1 - P_{succ})$.

A more complex situation is represented by two attack steps that are sequentially dependent, i.e., a step A can influence a step B. Figure 6(a) depicts a scenario where a dependency from A to B exists (i.e., from A at slice t to B at slice $t+1$). In this case, the CPT in Fig. 6(b) shows that B is *Ongoing* when A is *Successed* with probability P_{start}; while it remains $Unattempted$ with probability $(1 - P_{start})$. Otherwise, B remains $Unattempted$. Note that the CPT does not consider *Succeed* and $Failed$ cases, since they can be subject of other DBN submodels.

Other scenarios may involve non-deterministic choices where an independent successful event can trigger only one of two successive events; deterministic choices, used when the value of another boolean variable is used to determine which specific event occurs; or parallelism, when two actions start in parallel after the success of a first. Furthermore, more patterns can be combined to model complex attack scenarios.

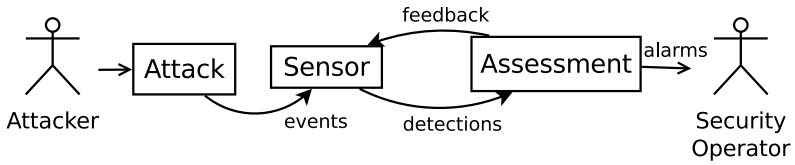


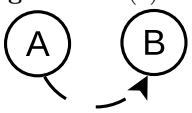
Fig. 4. DBN Model Structure.

(a)

A(t - 1)	A′(t)=U	A′(t)=O	A′(t)=S	A′(t)=F
U	$1 - P_{start}$	P_{start}	0	0
O	0	$1 - P_{end}$	P_s	P_{ns}
S	0	0	1	0
F	0	0	0	1

(b)

Fig. 5. DBN (a) model of single step and its (b) CPT.



(a)

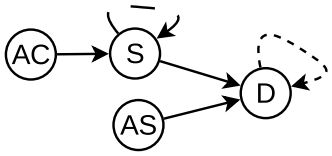
A(t - 1)	B(t)=U	B(t)=O
U	1	0
O	1	0
S	$1 - P_{start}$	P_{start}
F	1	0

(b)

Fig. 6. DBN (a) model of a sequence of steps and its (b) CPT.

4.2 Sensor Modelling

DBN modelling of sensing activity refers to a single DBN pattern as depicted in Fig. 7(a). The pattern comprises four nodes: a node S that models the sensor itself; a node D that models the detection activity; an activation node AC, which is described later; a node AS that detects an attack step. The node AC means the activation of the sensor by means of the **Assessment** submodel, i.e., it models the effects of the "adaptive feedback" actions from the *Assessment* module as in Fig. 4.



(a)

AC(t)	S(t - 1)	S(t)=On	S(t)=Off
On	On	$1 - P_f$	P_f
On	Off	P_r	$1 - P_r$
Off	On	0	1
Off	Off	0	1

(b)

Fig. 7. DBN Sensing pattern: (a) DBN submodel, (b) CPT of the S node.

The nodes have the following values: S can be $\{On, Off\}$, which represents the functioning of the sensor; AS has four values as previously described; D can be $\{True, False\}$, where $True$ means the sensor detects the threat, $False$ otherwise; AS represents an activation command to the sensor (hence, it can be represented by $\{On, Off\}$ in its simplest form). S can depend on AC since this last node is not mandatory, representing an always-on device where it is

not present. Moreover, the value of S also depends on its previous value since failures and repairing are possible for such devices. Figure 7(b) reports a typical CPT for an S node where P_f and P_r are respectively the probabilities of failure and repairing of the device in a time slice[4]; D nodes are more complex since they involve three parent nodes. The underlying logic of a CPT for D nodes is described by the following statements: when the sensor is off, an attack step is never detected; when the sensor is on and it has previously raised an alarm, the device continues to produce an alarm until it is switched off; when the sensor is on and the sensor has not previously raised an alarm and the threat is unattempted or failed, the device can erroneously detect a threat with a fpp probability; if the sensor is on and the sensor has not previously raised an alarm and the threat is ongoing or successfully brought, the device can detect a threat with a certain probability $(1 - fnp)$. The value of fpp (resp. fnp) is the false positive (resp. negative) probability, i.e, the probability that the device does (resp. does not) detect an alarm if the threat does not (resp. does) occur.

4.3 Assessment Modelling

The **Assessment** submodel represents decision mechanisms combining atomic detections of threats into complex assessment of system SLs. The combination of simple detections can be made by the algebraic operators inspired in [18]: AND, OR, NOT, and SEQ operator (i.e., an operator that is true when the inputs become true in a certain order). For the sake of space, we omit here the translation of these operators into BN models, previously reported in [19].

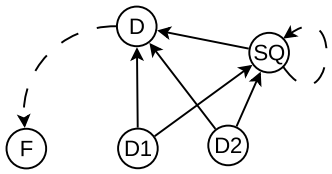
Figure 8(a) represents the most interesting case of the SEQ operator. The correlation is made between two detections $D1$ and $D2$ which are "D-nodes" of a DBN sensing pattern; the output of such correlation is the D node. D has the same $\{True, False\}$ nature of $D1$ and $D2$. F is a node representing the feedback given to the sensor layer (i.e., the AC node of the Sensing pattern). Another node, SQ, is in charge of keeping memory of the sequence of the events; the values of this node are $\{NIL, OK, KO\}$ that respectively stand for: none of the events has arrived, the sequence is correct, and the sequence is not correct. While the CPT of the D node implements an AND of the three parent nodes, the CPT of SQ recognises if $D1$ occurs before $D2$ (see Fig. 8(b)).

It is worth to note that more operators can be composed together in order to create complex logical expressions: the D nodes of a DBN submodel can be used as input node of another submodel. Moreover, the outputs of the top submodels represent the alarms raised to Security Operators.

5 Case Study

This section applies modelling approach to an example inspired by a real network intrusion case study occurred in a cancer and AIDS research organisation [20];

[4] Let T be the amount of time in a time slice, λ and μ the failure and repairing rates of the device, respectively. Hence, $P_f = \lambda \cdot T$ and $P_r = \mu \cdot T$.



(a)

SQ(t - 1)	D1	D2	SQ=NIL	SQ=OK	SQ=KO
NIL	True	True	0	0	1
NIL	True	False	0	0	1
NIL	False	True	0	1	0
NIL	False	False	1	0	0
OK	True	True	0	1	0
OK	True	False	0	1	0
OK	False	True	0	1	0
OK	False	False	1	0	0
KO	True	True	0	0	1
KO	True	False	0	0	1
KO	False	True	0	0	1
KO	False	False	1	0	0

(b)

Fig. 8. SEQ-based DBN assessment: (a) submodel, (b) CPT of the *SQ* node.

whose research laboratories became unavailable for several days resulting in financial losses.

As described in [20], an illegal user account was created in one of the organisation's servers by first accessing – using a stolen account – and then exploiting vulnerable services in such a server. Using this server as a base of operations, other computers were similarly compromised, while the user returned regularly to exfiltrate sensitive data from the network. Since the organisation only monitored on the Internet border, but not its internal subnets, these attacks launched from within the organisation's own network were totally in a blind spot and become unnoticed. Applying our approach in this scenario, we define four different security layers described textually as follows:

- SL_1**:** An Internet border firewall that monitors incoming network packets, analyses them based on predefined filtering rules, and performs an assessment providing the rate of suspicious incoming packets.
- SL_2**:** A behavioural-based network IDS focused on analysing any network packets, comparing current network flow to previous known attack-flow models. Similarly, the assessment returns a rate to express the confidence of being under attack at a given moment.
- SL_3**:** An anomaly network IDS focused on analysing packets coming from a specific network. In this case, the assessment provides a confidence level of having indeed an intrusion.
- SL_4**:** A load-balance server, which redirects incoming suspicious network traffic to an isolated machine acting as a honeypot. Thus, all suspicious traffic can be logged for forensics analysis.

Figure 9 represents the DBN model of the case study. We suppose that the attack starts from gaining unauthorised access (*Access*); it continues by intruding in the machine *M0* and using this system to attack *M1* and then *M2* by means of the gateway *G0*. The first Security Level is modelled by the *SL1* node that

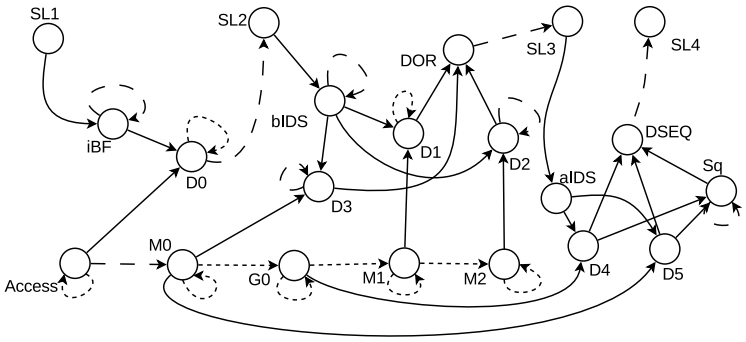


Fig. 9. DBN model of the running example.

is used as an activator of an Internet Border Firewall (modelled through the sensing pattern - *iBF*, *D0*). The feedback of this pattern (considering a trivial assessment submodel) is to trigger the *SL2*. At this level, all the networks (i.e., all the machines) are monitored by a behavioural IDS: when any anomaly is detected (modelled by a sensing pattern, *bIDS-D1-D2-D3*, and by an OR-based assessment, *DOR*) the *SL3* is activated. The last security level implements an anomaly network IDS focused on detecting a sequence of non-normal packets sent from *M0* and from *G0*. In this way, a sensing pattern and an SEQ-based operator are used: the output is the activation of the *SL4* which cut off the threat from the network to protect and isolate it.

6 Conclusions

This paper introduces an approach for the evaluation of the vulnerability of self-adaptive protection systems. This approach is based on the Onion Security Model, a high-level modelling approach where higher security levels increase the security by improving sensing and countermeasures based on lower ones. To make the approach concrete, Dynamic Bayesian Networks are used to capture probabilistic relationships as well as dependency in time among attacking and protecting events. First applications to a real intrusion scenario show that DBN model well these concerns. This paper constitutes a first step in the modelling of such systems. Future works will demonstrate the effectiveness of the approach also by comparing the efficiency of DBN with related to other formalisms (e.g., BN and GSPN). Next steps will involve the completion of the available DBN patterns also by giving some examples of countermeasure modelling.

References

1. Devanbu, P.T., Stubblebine, S.: Software Engineering for Security: a Roadmap. In: Proceedings of the Conference on The Future of Software Engineering. ICSE '00, New York, NY, USA, ACM (2000) 227–239

2. Rodríguez, R.J., Trubiani, C., Merseguer, J.: Fault-Tolerant Techniques and Security Mechanisms for Model-based Performance Prediction of Critical Systems. In: Proc. of the 3rd ISARCS, ACM (2012) 21–30
3. de Lemos et al., R.: Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In: Software Engineering for Self-Adaptive Systems II. Volume 7475 of Lecture Notes in Computer Science., Springer (2013) 1–32
4. Department of Homeland Security: NIPP 2013-Partnering for Critical Infrastructure Security and Resilience. Technical report, U.S. D.H.S. (2013)
5. Perez-Palacin, D., Mirandola, R., Merseguer, J.: On the relationships between QoS and software adaptability at the architectural level. Journal of Systems and Software **87**(0) (2014) 1–17
6. Dean, T., Kanazawa, K.: A Model for Reasoning about Persistence and Causation. Computational Intelligence **5**(2) (1989) 142–150
7. Macdonald, D., Clements, S., Patrick, S., Perkins, C., Muller, G., Lancaster, M., Hutton, W.: Cyber/physical security vulnerability assessment integration. In: Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES. (Feb 2013) 1–6
8. Garcia, M.L.: Vulnerability Assessment of Physical Protection Systems. 1st edn. Butterworth-Heinemann (November 2005)
9. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: Information Security and Cryptology - ICISC 2005, 8th International Conference, Seoul, Korea, December 1-2, 2005, Revised Selected Papers. (2005) 186–198
10. Bistarelli, S., Fioravanti, F., Peretti, P., Santini, F.: Evaluation of complex security scenarios using defense trees and economic indexes. J. Exp. Theor. Artif. Intell. **24**(2) (2012) 161–192
11. Zonouz, S.A., Khurana, H., Sanders, W.H., Yardley, T.M.: RRE: A Game-Theoretic Intrusion Response and Recovery Engine. IEEE Trans. Parallel Distrib. Syst. **25**(2) (2014) 395–406
12. Ficco, M.: Security event correlation approach for cloud computing. Int. J. High Perform. Comput. Netw. **7**(3) (September 2013) 173–185
13. Flammini, F., Marrone, S., Mazzocca, N., Vittorini, V.: Petri Net Modelling of Physical Vulnerability. In: Critical Information Infrastructure Security. Volume 6983 of LNCS., Springer (2013) 128–139
14. Xie, P., Li, J.H., Ou, X., Liu, P., Levy, R.: Using Bayesian networks for cyber security analysis. In: Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on. (June 2010) 211–220
15. Frigault, M., Wang, L., Singhal, A., Jajodia, S.: Measuring Network Security Using Dynamic Bayesian Network. In: Proceedings of the 4th ACM Workshop on Quality of Protection. QoP '08, New York, NY, USA, ACM (2008) 23–30
16. Tang, K., Zhou, M.T., Wang, W.Y.: Insider cyber threat situational awareness framwork using dynamic Bayesian networks. In: Proceedings of the 4th International Conference on Computer Science Education (ICCSE). (July 2009) 1146–1150
17. OMG: Unified Modelling Language: Superstructure. Object Management Group. (August 2011) Version 2.4, formal/11-08-05.
18. Chakravarthy, S., Mishra, D.: Snoop: An expressive event specification language for active databases. Data and Knowledge Engineering **14**(1) (1994) 1–26
19. Flammini, F., Marrone, S., Mazzocca, N., Pappalardo, A., Pragliola, C., Vittorini, V.: Trustworthiness Evaluation of Multi-sensor Situation Recognition in Transit Surveillance Scenarios. In: Security Engineering and Intelligence Informatics. Volume 8128 of Lecture Notes in Computer Science. (2013) 442–456
20. Casey, E.: Case study: Network intrusion investigation – lessons in forensic preparation. Digital Investigation **2**(4) (2005) 254–260