Quantitative Information Security Risk Estimation Using Probabilistic Attack Graphs

Pontus Johnson[1], Alexandre Vernotte[1(✉)], Dan Gorton[2], Mathias Ekstedt[1], and Robert Lagerström[1]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
{pontusj,vernotte,mekstedt,robertl}@kth.se
[2] Foreseeti AB, Stockholm, Sweden
dan.gorton@foreseeti.com

Abstract. This paper proposes an approach, called pwnPr3d, for quantitatively estimating information security risk in ICT systems. Unlike many other risk analysis approaches that rely heavily on manual work and security expertise, this approach comes with built-in security risk analysis capabilities. pwnPr3d combines a network architecture modeling language and a probabilistic inference engine to automatically generate an attack graph, making it possible to identify threats along with the likelihood of these threats exploiting a vulnerability. After defining the value of information assets to their organization with regards to confidentiality, integrity and availability breaches, pwnPr3d allows users to automatically quantify information security risk over time, depending on the possible progression of the attacker. As a result, pwnPr3d provides stakeholders in organizations with a holistic approach that both allows high-level overview and technical details.

Keywords: Quantitative risk analysis · Attack graphs · Threat modeling · Network security · Information security

1 Introduction

ICT systems have become an integral part of business and life. At the same time, these systems have become extremely complex, often hosting thousands of software applications, databases, operating systems, servers, processes, data, and more. In these complex systems-of-systems exist numerous vulnerabilities waiting to be exploited by potential threat actors [27,30]. Examples include power grids being shut down[1], smart cars taken[2], and financial institutions being hit by server side [20] and denial of service attacks. This trend has been overseen by responsible authorities who step up the minimum requirements for risk management [5], including requirements of recurring risk analysis [7,8]. However, government action is slowed down by multiple contrasting figures concerning

[1] http://www.cnn.com/2016/02/03/politics/cyberattack-ukraine-power-grid/.
[2] http://money.cnn.com/2012/09/27/technology/bank-cyberattacks/.

J. Großmann et al. (Eds.): RISK 2016, LNCS 10224, pp. 37–52, 2017.
DOI: 10.1007/978-3-319-57858-3_4

the impact of cyber attacks, which in turn makes it hard to identify new cost-effective security policies [2]. Thus, the ability to measure security is becoming a top priority in most organizations today. One example of this trend is the World Economic Forum (WEF) paper "Partnering for Cyber Resilience Towards the Quantification of Cyber Threats" published in January 2015 [9]. WEF acknowledge that cyber risk is increasingly viewed as key element of enterprise risk management and is requesting industry-specific risk models to, for example, enable cyber risk transferring.

In the individual organizations, there are many stakeholders which are interested in the management of the IT landscape and its security [11]. For some of the stakeholders, a system overview is just about enough, while others require details. So far this is also mirrored in the commonly employed tools, e.g. Visio and PowerPoint for C-level management and vulnerability scanners for network administrators. These solutions tend to focus either on providing a holistic view of the system without any connection to the actual details, or on a small part of the system, thus neglecting the bigger picture. Hence, there is a need for holistic approaches that also consider technological details [29]. However, most approaches available are driven by manual labor and require a high level of expertise, which in information security is both expensive and hard to come by [26].

pwnPr3d [18] (for *Pwn*[3] *Prediction*, pronounced [pˈəʊnprɪd]) is an attacker-centric threat modeling technique for automated threats identification and quantification based on network modeling. As opposed to most other similar approaches, pwnPr3d integrates reusable analysis capability. Instead of relying on human expertise to analyze a model and decide whether it is secure or not, pwnPr3d can automatically perform this analysis. That is, the security expertise is built into the model. In its analysis, pwnPr3d generates probability distributions over the Time To Compromise (TTC) for each asset in the system, and estimates information security risk as a probability distribution of the system-wide cost of security failure. As a result, pwnPr3d provides the various stakeholders of an organization with a cyber security evaluation of their systems that is tailored to their concerns.

This paper introduces an extension to pwnPr3d's meta-modeling architecture that allows for automated quantitative information security risk estimation. A new modeling entity, called *Information*, makes it possible for users to define the atomic cost of security breaches (namely, confidentiality, integrity and availability breaches) regarding a particular piece of information. Then, a dedicated algorithm, directly integrated into the TTC calculation, computes the global quantitative information security risk depending on the possibilities presented to the attacker. The end result is a cumulative frequency distribution of the increasing cost impact of security breaches over time. The remainder of this paper is structured as follows: Sect. 2 presents related work focusing on other modeling approaches more or less similar to pwnPr3d. Then, Sect. 3 introduces

[3] *Pwn* is originally a misspelling of the word *own*, in information security signifying the compromise of a computer system.

the two top layers of pwnPr3d's modeling architecture. Next, Sect. 4 describes the quantitative estimation calculation of information security risk. Section 5 exemplifies the use of pwnPr3d through a motivating example. Finally, Sect. 6 concludes the paper.

2 Related Work

Several methodologies center on identifying and quantifying the security risks present on a system or system-of-systems [1,6,21,23]. These methodologies typically break down risk analysis and assessment into several activities, and provide guidance on how to efficiently perform each activity. For instance, The Australian/New Zealand Standard AS/NZS 4360 [6] sets out a risk management process that consists of six stages: Establish the context, identify the risks, analyse the risks, evaluate the risks, and finally treat the risks. The NIST SP 800-30 Risk Assessment Framework [23] proposes a more detailed process composed of nine stages, typically isolating the identification of threats and vulnerabilities. OCTAVE [1] consists of a three-phase risk assessment strategy that the evaluation team must follow to extract appropriate mitigation strategies. Sometimes, a textual or graphical language is involved to provide further guidance. CORAS [21], which follows the process defined in [6], models threat scenarios as directed acyclic graphs whose nodes and edges are weighted, i.e. assigned with likelihood values (e.g., probabilities, frequencies, or intervals of these).

A common drawback of these methodologies is that they tend to consider threats as independent events and thus do not include their potential conditional dependencies in the risk estimation. Moreover, they do not provide automated analysis, and this activity remains to be done manually.

Many approaches propose to assess the cyber security of systems and networks by modeling probabilistic attack graphs. A popular approach is to exploit the output from network vulnerability scanners to model attack graphs. MulVal [14] derives logical attack-graphs by associating the vulnerabilities extracted from scans with a probability derived from their CVSS score, which express how likely an attacker is to exploit them successfully. NAVIGATOR [4] consider identified vulnerabilities as directly exploitable by the attacker (given that he has access to the vulnerable system). The TVA tool [24] models networks in terms of security conditions and uses a database of exploits as transitions between these security conditions. Another widespread solution for the representation of attack graphs and the computation of attack probabilities is Bayesian Networks [10,28,31]. In [10], the authors translate "raw" attack graphs obtained with the TVA-tool into dynamic Bayesian networks, and convert CVSS scores of vulnerabilities to probabilities. Similarly, the authors in [31] rely on CVSS to model uncertainties in the attack structure, the actions of the attacker and the triggering of alerts. In [28], the authors use Bayesian attack graphs to estimate the security risk on network systems and produce a security mitigation plan using a genetic algorithm. Similar to pwnPr3d in ambition is P2CySeMoL [13], which is a probabilistic relational model (PRM) with the purpose to estimate the cyber security of enterprise-level system architectures.

These approaches are efficient at evaluating the cyber security of systems in terms of threat and vulnerability identification, likelihood and severity. However, they mainly focus on the technical aspects of threats and vulnerabilities, while remaining business-value-neutral. Furthermore, most of them are either manual or they indirectly rely on vulnerability scanners that, as stated above, have disputable vulnerability detection rates.

Noel et al. [25] propose to measure security risk of networks using attack graphs. The analysis takes into account associated network operational costs and attack impact costs, making it possible to combine the likelihood of an attack, its projected cost and the mitigation cost. However, the attack graph modeling and the calculation remain manual.

3 pwnPr3d's Meta-modeling Architecture

pwnPr3d is an attacker-centric threat modeling approach that allows for automated threat identification and quantification based on a model of the network under analysis, by combining a network architecture modeling language and a probabilistic inference engine. The language couples the assets of a network with attack steps that define how these assets can be compromised and what the possible consequences on the other assets are. Thus, based on a network model instance, pwnPr3d automatically generates an attack graph based on the nature of its assets and their relations. The attack graph is analyzed by considering the entry point of the attacker in the network, i.e. one or several attack steps defined as successful attempts. In addition, pwnPr3d also allows probability distributions over the Time To Compromise (TTC) for attack steps by quantifying the attack step (conditional) dependencies. Such quantitative data can be collected from various sources including surveys and studies such as [12,19]. pwnPr3d enables to automatically identify and quantify a broad set of threats, covering most of the STRIDE classification [16].

Based on a network model instance, pwnPr3d automatically generates an attack graph and analyzes it by considering the entry point of the attacker in the network, i.e. one or several attack steps defined as successful attempts. The likelihood L of assets being compromised is obtained by quantifying the attack step (conditional) dependencies and deducing probability distributions over the Time To Compromise (TTC) for attack steps. Such quantitative data can be collected from various sources including surveys and studies such as [12,19]. The cost impact I of a security incident on an information asset is defined by the users. For each asset, three types of security incident are considered: confidentiality, integrity and availability breaches. As a result, pwnPr3d quantitatively estimates information security risk R over time, depending on the calculated progression of the attacker.

pwnPr3d's modeling language is designed as a closed meta-modeling architecture, similarly to MOF [22], which offers multiple benefits when it comes to system and network modeling. One the one hand, it provides separation of concerns making it possible to capture the attack graph theory in the lower layers of the meta-model, and spreads to the higher layers. The end goal is that end

users only model the assets and their relations, while all attack graph logic is encapsulated in lower layers. On the other hand, it allows a high flexibility in terms of introducing new types of assets and vulnerabilities. Components can be modeled with great level of detail for reuse as encapsulated wholes. For example, an operating system can be modeled as a composition of sub-components (applications, user accounts, network interfaces), themselves represented as a composition of sub-components. Modeling with much details enables a broad coverage of attacks, both between components and within the internals of a component. This ultimately leads to the creation of standard component libraries containing specific products (e.g., a Netgear wgr614 router).

The next sections present the terminology and modeling concepts of pwnPr3d. Only the first two layers are described, in order to keep the presentation concise.

3.1 Layer-0: Assets and Attack Graph Theory

The main purpose of *Layer-0* is to couple the components of an IT infrastructure and the attack surface of the attacker. It defines the attack graph theory, i.e. the possible progression of the attacker through attack steps, as well as TTC calculation. The metamodel of Layer-0 is depicted in Fig. 1. Its main three entities are described below.

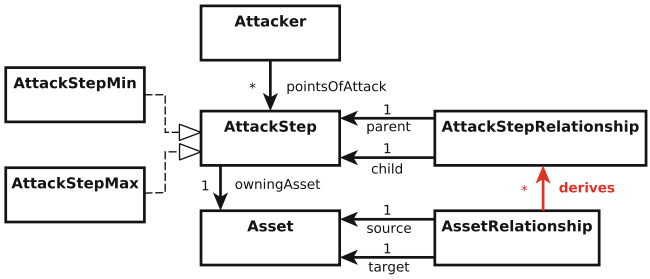


Fig. 1. Layer-0 metamodel

Asset is the class that ties together the logic of pwnPr3d. It is the class that later instantiate the core constituents of the system and the network, such as software, hardware and information. Such constituents can be related to one another through the *AssetRelationship* entity (e.g., to represent a physical connection between two computers). This is following standard object oriented modeling approaches.

Attacker represents a malicious actor that threatens the security of the system by compromising assets. In pwnPr3d, the *Attacker* entity defines the starting point of the attack. It can be connected to any *AttackStep* entity; such connections denote the source of the attack vectors. These particular attack steps thus always have a TTC that evaluates to 0.

Attack steps are actions conducted by an attacker to compromise an asset. As such, each attack step in pwnPr3d is associated to the asset it targets. Attack steps are related to one another through the *AttackStepRelationship* entity forming an attack graph. The *derives* link binds one or several *AttackStepRelationship* entities to an *AssetRelationship* entity. This is a key feature of pwnPr3d as it defines the attack graph construction theory in Layer-0, which spreads to the higher layers of the language. It thus allows for the automatic derivation of the attack graph from the behavioral relationships between assets. It is further explained in Sect. 3.2.

pwnPr3d models attack graphs as edge-weighted directed graphs where nodes represent attack steps, a subset of these nodes denotes the starting points(s) of attack, directed edges defines the possible progression of the attacker in the modeled system through the successful attempt of attack steps, and an edge weight function defines the probability distribution over time that an attacker will successfully attempt an attack step (i.e. TTC). Two kinds of attack steps are introduced in pwnPr3d: *attack step minimum* as_{min} and *attack step maximum* as_{max}, in order to specify the possible prerequisites of an attack step e.g., that the attacker needs access as well as the proper privileges in order to compromise a system. These two specializations echo the *AND* and *OR* gates that are generally used in previous works, although they have been adapted in pwnPr3d to enable the probabilistic inference of attack steps' TTC. Thereby, the attacker can attempt an as_{min} only if s/he has successfully attempted *at least one* of the attack step's parents, similarly to an OR gate. In case of several parents being compromised, the attack step's TTC will be computed with its parent's lowest TTC. If the attack step is an as_{max}, the attacker must have successfully attempted *all* of the attack step's parents before being able to attempt it, similarly to an AND gate. The attack step's TTC will be computed with its parent's highest TTC, as the approximation of true AND TTC. This approximation is a worst case, as an attacker typically will require longer time than so.

It should be noted that prerequisite relationships between attack steps should not be mistaken for direct causality. There is no guarantee that an attack will succeed as it is dependant upon a multitude of factors. The imperfect nature of exploits is one. The skill set of the attacker is another. Therefore, the edges outgoing an attack step define the possibilities that are presented to an attacker upon successful compromise of the attack step.

Calculation of TTC follows a two-steps process:

1. Each edge of the attack graph is "concretized" by getting a sample from its TTC probability distribution. The sampled value becomes the weight of the edge and represents the TTC of the edge's target attack step, given that the attacker has successfully attempted the edge's source attack step;
2. An adapted version of Dijkstra's shortest path algorithm calculates the smallest TTC value for each attack-step, depending on its ancestry. More concretely, we use Dial's Approximate Buckets implementation [3].

This process is performed N times (e.g., 500 times), and each attack step keeps track of the TTC values it has been assigned with. The end result is a frequency distribution of the successful attempt of an attack step over time.

3.2 Layer-1: Network and System-Specific Logic

Layer-1 introduces the network and system-specific logic for the attack graph generation, the various threat types that can be identified in a network, and loss calculation from CIA breaches. It uses Layer-0 as a meta-model and all the classes introduced in this layer are instances of the *Asset* entity, and each *Asset* instance contains its own set of attack steps.

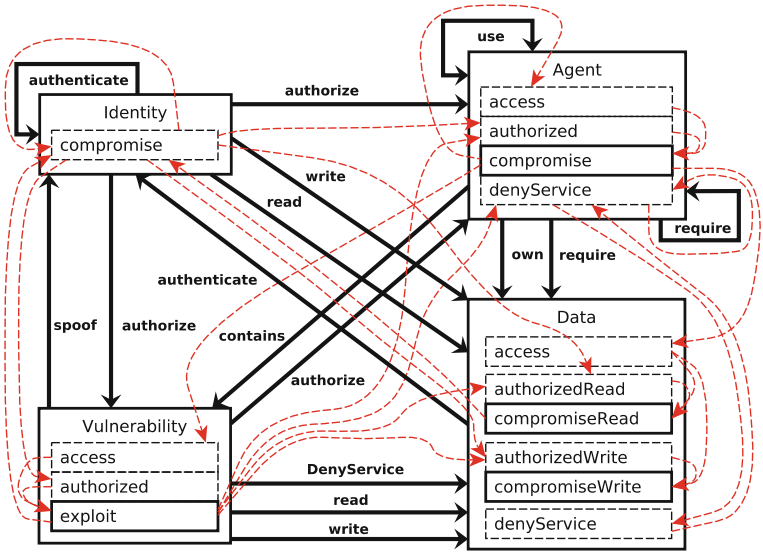


Fig. 2. The Layer-1 model including classes, class relationships, attacks steps and their dependencies. The containing entities are *Asset* instances, and the contained entities are attack steps related to their owning asset. Attack steps with dashed lines symbolize as_{min} and solid lines as_{max}. Solid edges represent behavioral associations (*AssetRelationship* instances), and dashed edges define the possible progression of the attacker from one attack step to another (*AttackStepRelationship* instances). (*Derive* associations are not represented in the figure.)

While Layer-0 encapsulates the attack graph theory, Layer-1 encapsulates the attack graph logic, i.e. how to derive the attack graph from an object model. Each Layer-1 entity owns a set of attack-steps that relate to one another, and each relationship between two entities derives a particular set of *AttackStepRelationships*. Hence, users must only instantiate the four entities (Agent, Identity, Data, Vulnerability) and their relationships, when creating a Layer-1 object model.

The model of Layer-1, depicted in Fig. 2, consists of four *Assets* instances, discussed below. For each entity, we describe its nature, its relationships with the other entities, and the attack step edges that its relationships derive.

Identities are authorization concepts that specify the restriction rules enforced in the system. Their core purpose in pwnPr3d is to specify the required privileges to read and write *Data*, control an *Agent*, exploit a *Vulnerability*. For example, only administrators are allows to read a particular file, say */etc/passwd/*.

The identity entity has one attack step: $compromised_{min}$. If the attacker compromises an identity (e.g., via credentials disclosure), s/he "assumes" this identity and gains all privileges that this identity represents on the network.

Identities have four different relationship types. First, an identity can be *authorized* to access an agent. At the attack step level this leads to the derivation of an edge from identity.*compromise* to agent.*authorize*. Second, an identity can *authenticate* as another identity (e.g., the admin of a system also has user privileges). By compromising this identity, the attacker also gains the privileges from the authenticated identity. This relationship derives an edge from administrator.*compromise* to user.*compromise*. Third, An identity can be authorized to *read* and/or *write* data. Such a relationship derives an edge between identity.*compromise* to datum.*authorizedRead* and/or datum.*authorizedWrite*. Lastly, an identity can be *authorized* to exploit a vulnerability e.g., an attacker must gain user privileges on a system to exploit a vulnerability. An *authorize* relationship leads to the derivation of an edge from identity.*compromise* to vulnerability.*authorized*.

Agents represent any active entity in the network: software, hardware, or people. An agent has four attack steps: (i) $access_{min}$ (the attacker has logical access to the agent so that it is reachable), (ii) $authorized_{min}$ (the attacker has the capability to control the agent), (iii) $compromise_{max}$: the attacker has fully assumed and taken over the agent, and (iv) $denyService_{min}$: the attacker is preventing the agent from working properly, aka a Denial-of-Service (DoS). Both *access* and *authorized* are parents of *compromise*, which specifies that in order to compromise an agent, the attacker must have logical access to it *and* the necessary privileges.

An agent may *require* another to function properly e.g., an OS requires a network interface to send data over the network. If the attacker was to perform a DoS attack on the network interface, the operating system would no longer be able to communicate. Therefore, an attack step edge is derived, from *denyService* of the required agent to *denyService* of the requiring agent. Agents may also *use* one another e.g., the network interfaces of a switch and a host use one another to exchange data. Two attack step edges are derived: one from agentA.*compromise* to agentB.*access*, and one from agentB.*compromise* to agentA.*access*.

Moreover, an agent may *require* data to function properly, e.g., in order to calculate the fastest route between two places, data about both places must be available. Hence an edge is derived from *denyService* of the datum to *denyService* of

the agent. Agents may also *own* data, e.g., a database server contains sensitive data. When the attacker compromises an agent that owns data, s/he gains logical access to the data. If s/he DoS the agent, the data can no longer be accessed. Two attack step edges are derived: (1) from agent.*compromise* to datum.*access*, and (2) from agent.*denyService* to datum.*denyService*.

Lastly, an agent may *contain* a vulnerability, denoting when an asset holds a bug. If the attacker compromises the agent, s/he gets access to the vulnerability. Therefore, this attack sequence is represented by an edge from agent.*compromise* to vulnerability.*access*.

Data represents any form of information: files, transportation messages, commands, credentials, encryption, etc.

The *Data* entity has six attack steps: (i) $access_{min}$ (the attacker has logical access to the datum but still cannot read/write), (ii) $authorizedRead_{min}$ (the attacker has authorization to read the datum), (iii) $authorizedWrite_{min}$ (the attacker has authorization to write the datum), (iv) $compromiseRead_{max}$ (the attacker can read the datum), (v) $compromiseWrite_{max}$ (the attacker can write the datum), and (vi) $denyService_{min}$ (the attacker denies access to the datum). *access* and *authorizedRead* are parents of *compromiseRead*, and *access* and *authorizedWrite* are parents of *compromiseWrite*: the attacker can read (respectively write) a datum if s/he has logical access to it and has gained read (respectively write) privileges. Such privileges can typically be obtained from the compromising of an identity (e.g. identity spoofing), or by exploiting a vulnerability that directly bypasses the access restriction.

A special kind of datum in pwnPr3d is credentials and encryption keys. These are represented through the *authenticate* relation to *Identity*. If an attacker succeeds with *compromiseRead* on a datum, s/he also compromises all the identities that the datum authenticates. Note that due to a lack of space, only a simplified representation of data is presented. An aspect that is omitted is the capability of encapsulating data to represent network messages and encrypted files.

Vulnerabilities represent flaws in the implementation or design of a system: they constitute loopholes in the rule set represented by the other assets, associations and relations. In pwnPr3d, the possible prerequisites and consequences of a vulnerability exploit are modeled rather than how the vulnerability is exploited. The fact that not all vulnerability exploits result in successful compromises is captured with the probabilities in the attack step relations. Moreover, the existence of a vulnerability may be uncertain. It may be the case for instance that the administrator has secured his system even though the manufacturer has not published a patch yet. The uncertainty of a vulnerability existence is represented as a probability distribution, which further influence the calculation of TTC.

The *Vulnerability* entity has three attack steps: (i) $accessed_{min}$ (the attacker has logical access to the vulnerability), (ii) $authorized_{min}$ (the attacker has gained the necessary privileges to exploit the vulnerability), and (iii) $exploit_{max}$ (the attacker can exploit the vulnerability).

Both *access* and *authorized* are parents of *exploit*: the attacker needs access to the vulnerability and the necessary privileges in order to exploit the vulnerability. For instance, an attacker might only be able to install a firmware rootkit if s/he is (remotely) connected to the targeted system and has user privileges.

Vulnerabilities can be exploited to *spoof* an identity for escalation of privileges, i.e. compromise it through another identity. A spoof relationship leads to the derivation of an edge from $exploit_{max}$ to the attack step $compromised_{min}$ from the spoofed identity. A vulnerability exploit can also allow an attacker to *read* (respectively *write*, i.e. data tampering) a datum (i.e. information disclosure), given logical access. Two edges are derived from this relationship: from vulnerability.*exploit* to datum.*read* and to datum.*write*. Finally, a vulnerability exploit can *authorize* access to an agent, aka bypass the restriction in place. Hence, an edge is derived from vulnerability.*exploit* to agent.*authorized*. Lastly, a vulnerability when exploited can allow an attacker to DoS the agents that contain it. An edge is derived from vulnerability.*exploit* to agent.*denyService*.

4 Extension for Quantitative Information Security Risk Estimation

Information security risk is defined in ISO/IEC 27005 as "the potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm to the organization.", that is measured "in terms of a combination of the likelihood of an event and its consequence" [17]. Formally speaking, the risk R is obtained from the product of the likelihood L of a security incident occurring times the impact I it will have on the organization ($R = L * I$).

In the previous section, we described how pwnPr3d automatically computes the likelihood of attacks in term of time to compromise: The likelihood L of assets being compromised is obtained by quantifying the attack step (conditional) dependencies and deducing probability distributions over the Time To Compromise (TTC) for attack steps. In this section, we propose an extension to pwnPr3d's class model that enables users to assign the cost value I of a security incident to information assets, reflecting the cost impact of a security incident on the corresponding asset. For each asset, three types of security incident are considered: confidentiality, integrity and availability breaches. As a result, pwnPr3d quantitatively estimates information security risk R over time, depending on the calculated progression of the attacker. This extension, as depicted in Fig. 3, consists of the introduction of a new Layer-1 element that represents information assets.

Information is considered immaterial, and as such has no direct relationship with agents, identities nor vulnerabilities. It can only indirectly relate to these through a Data entity that *represents* the information. The Data entity represents the format (e.g., XML), and the Information entity represents its meaning and its value. Hence, when an identity has read privileges on a datum, it has by extension read privileges on the information itself. Furthermore, information may be represented by multiple data stored in different places (e.g., the enterprise performs regular back-ups of a database).

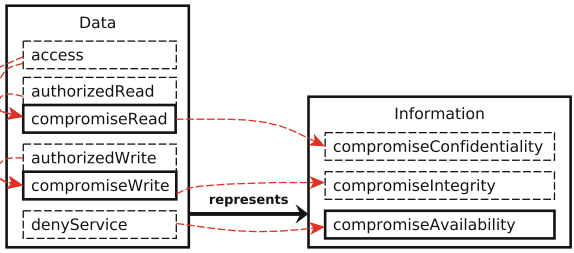


Fig. 3. Relationship between Data and Information

The Information entity can be compromised according to three attack steps, matching the CIA triad:

- $compromiseConfidentiality_{min}$: the attacker has gained logical access to one of the data that represent the information with read privileges, which gives him the possibility to access the information.
- $compromiseIntegrity_{max}$: the attacker has gained logical access to one of the data with write privileges, he can therefore compromise the integrity of the information.
- $compromiseAvailability_{max}$: the attacker has made all the data representing the information unavailable to their surroundings (e.g., through denial of service), hence compromising the information's availability. Because this is a direct technical consequence, the TTC of the attack step edge between data.*denyService* and *compromiseAvailability* is set to 0.

compromiseIntegrity and *compromiseAvailability* are as_{max}, since Information can be present in a system in multiple places. Therefore, if the attacker compromises one of the representing data, technically the information is still available/coherent. Contrariwise, $compromiseConfidentiality_{min}$ is an as_{min} because it only takes the attacker to attempt *compromiseRead* on one of the representing data to compromise the information confidentiality.

Each information instance must be valued with three attributes that express the cost impact of Confidentiality, Integrity and Availability breaches. The type of cost is reliant on the type of the attack step, i.e. *compromiseConfidentiality* relates to Confidentiality cost, *compromiseIntegrity* to Integrity cost, and *compromiseAvailability* to Availability cost. It is the users responsibility to quantify the cost impact of CIA breaches for each information instance. Indeed, evaluating such costs is an onerous and very speculative task that involves many factors [15]. One may consider immediate losses as well as delayed losses, including time sensitivity, impact on the stock market, cost of asset recovery, and so on. Deciding which factors should be considered and to what extent is a real challenge and as a result, the quantification might be quite inaccurate, regardless of the employed evaluation methodology. To palliate this inaccuracy, impact costs are defined as probability distributions. In the next section, for instance, costs are quantified using truncated normal distributions. Another option would

be to use beta distributions, to model the variable level of confidence within the distribution. Note that, in this paper, we do not provide insights on how to calculate individual impact costs and how to derive probability distributions from them (as it is well discussed in the literature). Instead, the focus is solely on how these cost are aggregated w.r.t the attack-graph analysis.

The calculation of quantitative information security risk is directly integrated into the TTC calculation algorithm. Before each TTC calculation, all the probability distributions over CIA cost impacts are sampled. After each TTC calculation, the successfully attempted attack steps that are owned by Information entities are inspected to collect tuples composed of the TTC value of the attack-step and the sampled cost impact of the owning Information entity. Tuples are then ordered based on their TTC value (from soonest to latest), and their associated cost impact are cumulated: the cumulative cost of a given tuple is the sum of its initial cost and the cumulative cost of its predecessor. Once the TTC calculation algorithm has been executed N times, the obtained N collections of tuples are merged and distributed in time bins. The end result is a cumulative frequency distribution of the increasing cost impact of CIA breaches over time, depending on the progression of the attacker in the network. Users are presented with a cumulative histogram featuring the 5, 50, and 95 percentiles.

5 Motivating Example

Applying pwnPr3d on the test enterprise network involves the design of a topology model that comprises all the components and assets of the network, how they connect to one another, what the various access restrictions (e.g. firewall rules) are, what the value of information assets is, as well as the introduction of one (or several) attacker(s). Of course, the goal of full threat analysis and security risk calculation automation is only achieved when complex classes have been defined and grouped in component libraries ultimately made available to end users (e.g., someone needs to define how Windows 10 is constructed). Once the appropriate classes are available, pwnPr3d can be used for the evaluation of different design scenarios. However, high-level components and product libraries are out of the scope of this paper, the focus being on the core layers of pwnPr3d and its extension that makes it possible to compute global information security risk. Hence, it should be noted that the motivating example presented below does not reflect how end users would model their enterprise network.

Consider a snippet of a heavily simplified software development enterprise network. It is composed of a Windows 10 client host that has two level of privileges: guest and user. To get user privileges, one must know the associated credentials. The windows 10 client host is connected to a Linux server host, which stores all the source code created within the enterprise. With user privileges on the Windows 10 client host, one has access to the home folder related to the user account, and admin privileges on the Linux server host. Finally, the Windows 10 client host has a known (fictional) vulnerability that, if exploited, gives the attacker authorzed read on the user credentials of the host. In this example, the

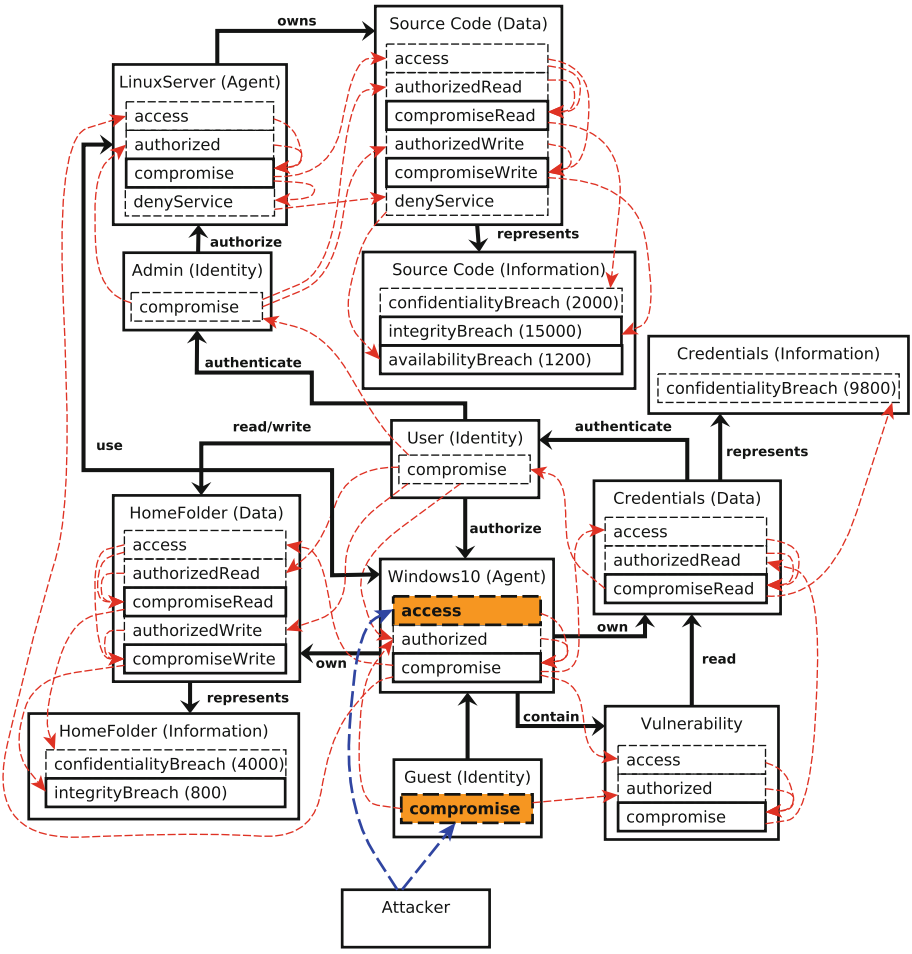
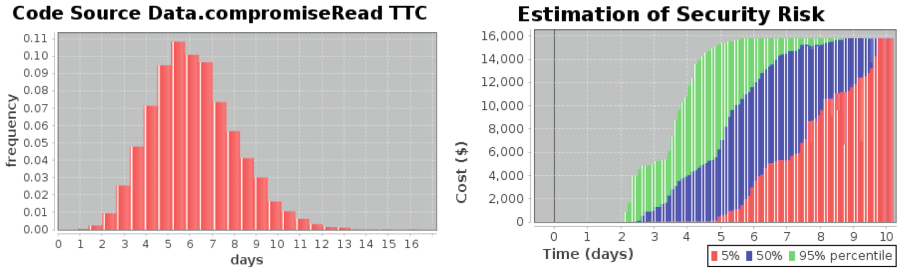


Fig. 4. pwnPr3d model of the test network, with sampled impact costs of CIA breaches

goal is to measure the possible progression of the attacker and the corresponding estimation of information security risk given that the s/he has logical access to the Windows 10 client host, with guest privileges. It is defined in the model with two *startingPoint* relations from the attacker to the concerned objects. The pwnPr3d object model for this example is depicted in Fig. 4, designed using pwnPr3d's layer 1 meta-model.

Because the attacker has logical access to the Windows host with guest privileges, the host is considered compromised. Therefore, the attacker can now exploit the vulnerability in ordeer to obtain read authorization on the user credentials. If performed, the attacker has the possibility to become *User* on the Windows host. Since credentials are data, a cost of 9800 has been set in case of a confidentiality breach, which gets marked as "reached". The attacker also gets to



(a) (b)

Fig. 5. pwnPr3d result: frequency distribution over the time to compromise code source data (a) and quantitative information security risk estimation of the test network (b)

possibility to read and write data in the user's home folder, and if performed, the associated breaches cost are also marked as "reached". Furthermore, by being *User* on the Windows host, the attacker can move laterally and compromise the Linux server with admin privileges. If so, the attacker has the possibility to get read/write permissions on the source code. Again, if performed, further security cost is marked as "reached".

Figure 5 shows the results produced by pwnPr3d for the test network. As presented in *(a)*, a frequency distribution over the time to compromise is computed for each attack step that has been successfully attempted by the attacker. In this example, the average TTC read access on the code source data is approximately 6 days. The combination of TTC from all the attack steps that are related with information assets with the impact cost of these assets is then collected and presented to users in the form of an histogram, as depicted in (b), representing the increasing security risk over time. The 5th and 95th percentiles are impact cost distributions for each time span (a time span being a tenth of day). For example, at day 6 in the figure, the impact cost of the 5% lowest cost calculations tops at around $2000. It means that, if there were 1000 calculations, ranked from lowest overall impact cost to higher overall impact cost after 6 days, the 5th percentile is the overall impact cost of the 50th lowest calculation. Similarly, the 95th percentile is the overall impact cost of the 50th highest calculation.

6 Conclusions

pwnPr3d is an attacker-centric probabilistic threat modeling technique for automated risk identification and quantification based on a topology model of the system under analysis. The components of the system, depending on their nature and how they relate to one another, are automatically coupled with attack steps that define how these assets can be compromised: the threat analysis is built-in and no security expertise is required from the users. An attack graph is calculated from the topology model and populated with probability distributions over the Time To Compromise (TTC) on each of its attack steps, thus defining the

likelihood of the identified threats to exploit a vulnerability. Once users have defined the value of information assets to their organization, pwnPr3d automatically computes a quantitative estimation of information security risk over time, depending on the calculated progression of the attacker.

Future work is directed toward two research directions: (i) the extension of the language to include complex components and products for users to simply instantiate, and (ii) the extension of risk analysis to tangible assets in order to improve its overall accuracy and precision. Furthermore, a thorough experimentation on real-life systems is ongoing to validate the approach.

Acknowledgments. The work presented in this paper has received funding from the European Unions Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 607109 as well as the Swedish Civil Contingencies Agency (MSB) through the research centre on Resilient Information and Control Systems (RICS).

References

1. Alberts, C.J., Dorofee, A.: Managing Information Security Risks: The OCTAVE Approach. Addison-Wesley Longman Publishing Co., Inc. (2002)
2. Armin, J., Thompson, B., Ariu, D., Giacinto, G., Roli, F., Kijewski, P.: 2020 cybercrime economic costs: No measure no solution. In 10th International Conference on Availability, Reliability and Security (ARES), pp. 701–710. IEEE (2015)
3. Cherkassky, B.V., Goldberg, A.V., Radzik, T.: Shortest paths algorithms: theory and experimental evaluation. Math. Program. **73**(2), 129–174 (1996)
4. Chu, M., Ingols, K., Lippmann, R., Webster, S., Boyer, S.: Visualizing attack graphs, reachability, and trust relationships with navigator. In: Proceedings of the 7th International Symposium on Visualization for Cyber Security, pp. 22–33. ACM (2010)
5. European Commission. Towards a general policy on the fight against cyber crime (2007). http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52007DC0267. Accessed 5 March 2017
6. Cooper, D.: The australian and new zealand standard on risk management, as/nzs 4360: 2004. Tutorial Notes: Broadleaf Capital International Pty Ltd, pp. 128–151 (2004)
7. ECB. Recommendations for the security of internet payments (2015). https://www.ecb.europa.eu/pub/pdf/other/recommendationssecurityinternetpaymentsoutcomeofpcfinalversionafterpc201301en.pdf, Accessed 5 March 2017
8. FFIEC. Supplement to authentication in an internet banking environment (2011). https://www.fdic.gov/news/news/financial/2011/fil11050.pdf. Accessed 5 March 2017
9. W. E. Forum. Industry agenda. partnering for cyber resilience - towards the quantification of cyber threats, January 2015. http://www3.weforum.org/docs/WEFUSA_QuantificationofCyberThreats_Report2015.pdf. Accessed 5 March 2017
10. 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, pp. 23–30. ACM (2008)

11. Goodyear, M., Goerdel, H.T., Portillo, S., Williams, L.: Cybersecurity management in the states: The emerging role of chief information security officers. Available at SSRN **2187412** (2010)
12. Holm, H.: A large-scale study of the time required to compromise a computer system. IEEE Trans. Dependable Secure Comput. **11**(1), 2–15 (2014)
13. Holm, H., Shahzad, K., Buschle, M., Ekstedt. M.: P cysemol: predictive, probabilistic cyber security modeling language. IEEE Trans. Dependable Secure Comput. **12**(6), 626–639 (2015)
14. Homer, J., Zhang, S., Ou, X., Schmidt, D., Du, Y., Rajagopalan, S.R., Singhal, A.: Aggregating vulnerability metrics in enterprise networks using attack graphs. J. Comput. Secur. **21**(4), 561–597 (2013)
15. Hoo, K.J.S.: How much is enough? A risk management approach to computer security. Stanford University Stanford, Calif (2000)
16. Howard, M., LeBlanc, D.: Writing secure code, 2nd edn. (2002)
17. E. ISO. Iec 27005: 2011 (en) information technology-security techniques-information security risk management switzerland. ISO/IEC (2011)
18. Johnson, P., Vernotte, A., Ekstedt, M., Lagerström, R.: pwnpr3d: an attack-graph-driven probabilistic threat-modeling approach. In: 11th International Conference on Availability, Reliability and Security (ARES). IEEE (2016)
19. Jonsson, E., Olovsson, T.: A quantitative model of the security intrusion process based on attacker behavior. IEEE Trans. Softw. Eng. **23**(4), 235–245 (1997)
20. Kaspersky. The great bank robbery: Carbanak cybergang steals $1bn from 100 financial institutions worldwide (2015). http://usa.kaspersky.com/about-us/press-center/press-releases/2015/great-bank-robbery-carbanak-cybergang-steals-1-billion-100-fina. Accessed 5 March 2017
21. Lund, M.S., Solhaug, B., Stølen, K.: Model-Driven Risk Analysis: The CORAS Approach. Springer Science & Business Media, Heidelberg (2010)
22. Meta object facility (MOF) 2.5 core specification (2015). http://www.omg.org/spec/MOF/2.5/
23. S. NIST. 800–30. Risk management guide for information technology systems, pp. 800–30 (2002)
24. Noel, S., Elder, M., Jajodia, S., Kalapa, P., O'Hare, S., Prole, K.: Advances in topological vulnerability analysis. In: Conference For Homeland Security, CATCH 2009. Cybersecurity Applications Technology, pp. 124–129, March 2009
25. Noel, S., Jajodia, S., Wang, L., Singhal, A.: Measuring security risk of networks using attack graphs. Int. J. Next Gener. Comput. **1**(1), 135–147 (2010)
26. Nyanchama, M.: Enterprise vulnerability management and its role in information security management. Inform. Syst. Secur. **14**(3), 29–56 (2005)
27. Ponemon Institute. Cost of cyber crime report (2013)
28. Poolsappasit, N., Dewri, R., Ray, I.: Dynamic security risk management using Bayesian attack graphs. IEEE Trans. Dependable Secure Comput. **9**(1), 61–74 (2012)
29. Soomro, Z.A., Shah, M.H., Ahmed, J.: Information security management needs more holistic approach: a literature review. Int. J. Inf. Manage. **36**(2), 215–225 (2016)
30. Verizon. Data breach investigations report (2014)
31. Xie, P., Li, J.H., Ou, X., Liu, P., Levy, R.: Using Bayesian networks for cyber security analysis. In: 2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 211–220. IEEE (2010)