Computer Standards & Interfaces 36 (2014) 734–747

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

An extensible pattern-based library and taxonomy of security threats for distributed systems

Anton V. Uzunov [a,*], Eduardo B. Fernandez [b,1]

[a] School of Computer Science, The University of Adelaide, Adelaide, South Australia 5005, Australia

[b] Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, 777 Glades Rd., Boca Raton, FL 33431, United States

ARTICLE INFO

Available online 30 December 2013

Keywords:
Distributed systems security attacks
Threat patterns
Threat modeling
Pattern-based security threat taxonomy
Peer-to-peer system-specific threats

ABSTRACT

Security is one of the most essential quality attributes of distributed systems, which often operate over untrusted networks such as the Internet. To incorporate security features during the development of a distributed system requires a sound analysis of potential attacks or threats in various contexts, a process that is often termed "threat modeling". To reduce the level of security expertise required, threat modeling can be supported by threat libraries (structured or unstructured lists of threats), which have been found particularly effective in industry scenarios; or attack taxonomies, which offer a classification scheme to help developers find relevant attacks more easily. In this paper we combine the values of threat libraries and taxonomies, and propose an extensible, two-level "pattern-based taxonomy" for (general) distributed systems. The taxonomy is based on the novel concept of a threat pattern, which can be customized and instantiated in different architectural contexts to define specific threats to a system. This allows developers to quickly consider a range of relevant threats in various architectural contexts as befits a threat library, increasing the efficacy of, and reducing the expertise required for, threat modeling. The taxonomy aims to classify a wide variety of more abstract, system- and technology-independent threats, which keeps the number of threats requiring consideration manageable, increases the taxonomy's applicability, and makes it both more practical and more useful for security novices and experts alike. After describing the taxonomy which applies to distributed systems generally, we propose a simple and effective method to construct pattern-based threat taxonomies for more specific system types and/or technology contexts by specializing one or more threat patterns. This allows for the creation of a single application-specific taxonomy. We demonstrate our approach to specialization by constructing a threat taxonomy for peer-to-peer systems.

1. Introduction

Over the last decade distribution has become one of the main characteristic features of software systems, prompted in large measure by the expanding needs of businesses, scientific organizations and individuals who wish to collaborate across geographical distances, share data and resources or simply perform computations remotely. To support such features, however, the corresponding systems must often span untrusted networks – with the Internet being a prime example – making them susceptible to a wide range of attacks both at the individual host and network levels. Security attributes, therefore, are among the most important quality attributes for distributed systems operating in untrusted environments, and have consequently received much attention over the years [1–4]. To incorporate these attributes during the development of a distributed system, whether using a systematic approach (i.e. a methodology [5]) or in some ad-hoc fashion, requires the introduction of a number of security measures, which, in turn, are the result of analyzing the potential attacks or threats to a system in a given context. This analysis process is often termed *threat modeling* [6,7], and is performed during the requirements analysis stage, the design stage, or both. In all cases the process generally requires developers to conjecture possible attacks to different assets or parts of a system, to assess their risk and likelihood, and to determine at a high level how they could potentially be mitigated.

Conducting threat modeling usually requires a sound knowledge of a system's technical domain and sufficient security expertise to consider both generic and specific attacks for various system- and/or technology-specific contexts. These security knowledge requirements can leave most "off-the-street" developers estranged (cf. [8]), with the net result that threat modeling is not performed or, when performed, is performed sub-optimally or with significant effort involved (cf. [9]). As Dhillon [9] points out, a *threat library* that collects common threats to a given system-/technology-specific context can greatly enhance the efficacy of the threat modeling process and hence put it back, so to speak, on the project map. A threat library such as the one used at EMC [9], or even by

* Corresponding author.
E-mail addresses: anton.uzunov@adelaide.edu.au (A.V. Uzunov), ed@cse.fau.edu (E.B. Fernandez).
[1] Currently visiting professor at Universidad Técnica Federico Santa María, Valparaíso, Chile.

0920-5489/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.csi.2013.12.008

Microsoft for web applications [10], can also go a long way in educating developers about common threats, rendering future threat modeling tasks easier.

Despite their value, threat libraries encompass only a set of specific, pre-defined threats, making the discovery of new threats or the same threats in different architectural contexts more difficult. In this respect the use of *threat* or *attack taxonomies* such as Microsoft's STRIDE [7] (an acronym for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Escalation of privileges), can be more useful, since they allow an arbitrary number of threats to be considered that fall within one or more categories. However, most taxonomies are either at a very high level of abstraction and hence require significant security expertise to identify appropriate threats (cf. [9] for STRIDE), or, in general, are simply not appropriate for threat modeling or indeed any form of security assessment in the first place [11]. Those that are appropriate for security assessment and at the right level of abstraction are not necessarily useful during the earlier stages of the SDLC (e.g. they consider post-design vulnerabilities), do not provide appropriate categories for threat modeling, or are relevant only to specific contexts (see [11] for a broad overview and references). Finally, the taxonomies referred to above – excepting STRIDE – are for vulnerabilities and attacks, not threats, which is a subtle but important difference (e.g. unsafe code execution is a threat realized by multiple attacks in different contexts).

In this paper we combine the values of threat libraries and threat taxonomies and propose what we term (with some risk of using terminology loosely) a *pattern-based threat taxonomy* for (general) distributed systems. In our approach, each threat is encapsulated in a new type of pattern (see [12]) called a *threat pattern*, which can be customized and instantiated in particular architectural contexts to define specific threats to a system. This allows developers to quickly consider a range of relevant threats in various architectural contexts as befits a threat library, increasing the efficacy of, and reducing the expertise required for, threat modeling. Threat patterns can also be related to corresponding misuse patterns [13], which can detail the attacks realizing a particular threat and educate developers. The (base) taxonomy aims to classify a wide variety of more abstract, system- and technology-independent threats, which keeps the number of threats requiring consideration during a threat modeling process manageable, increases the taxonomy's applicability, and makes it both more practical and more useful for security novices and experts alike. Employing patterns also helps to establish a common domain vocabulary, promoting the use of consistent threat names and concepts by developers in their everyday security-related work.

Despite the breadth of our taxonomy, each distributed system type, and even the technologies employed to realize a system, can create a variety of specific threats, which may not be explicitly present among our proposed threat patterns, or, more precisely, may not be present at the base level of abstraction. To solve this problem, we propose a simple and effective method to extend threat taxonomies by specializing one or more threat patterns to new system-/technology-specific contexts. This allows for the construction of application-specific taxonomies by taking the union of the set of relevant system-/technology-specific taxonomies, which in turn allows developers to consider the widest range of applicable threats in any given architectural context. We demonstrate our approach to specialization by constructing a threat taxonomy for peer-to-peer systems.

The latter example also demonstrates the purely "taxonomic" feature of our proposal, where each pattern in the taxonomy for distributed systems acts as a "category" for more specialized patterns and pattern instances. This feature allows known threats to be classified in a way that has value during system development, i.e. specific attacks such as CodeRed worm, SQL slammer exploit and, indeed, thousands of others, can be seen more abstractly as collections of individual threats (scanning, probing, injection, etc.), which require mitigation irrelevant of whether they are launched against a system automatically (malicious software) or manually (malicious hackers).

The rest of this paper is structured as follows. In Section 2 we introduce the concept of a threat pattern, relate it to the existing misuse patterns of Fernandez and colleagues [13], and discuss threat taxonomies (Section 2.1); we also define the architectural contexts of the threat patterns (Section 2.2). In Section 3 we present our (base) threat taxonomy for distributed systems and discuss pattern specialization and instantiation. In Section 4 we specialize a number of (base) threat patterns to construct a taxonomy for peer-to-peer systems. In Section 5 we consider related work; and in Section 6 we conclude and discuss future research directions.

2. Background and definitions

In this section we provide the necessary background for the rest of the paper by defining threat patterns and pattern-based threat taxonomies (Section 2.1) as well as architectural contexts (Section 2.2).

2.1. Threat patterns and pattern-based threat taxonomies

The concept of a pattern in software engineering has received much attention over the last 15 years, both in academia and the industry, owed in large measure to the pioneering work of Gamma and colleagues in the field of object-oriented design [14]. Patterns have been found useful in diverse areas such as software architecture, fault-tolerance, parallel programming and security, with each area boasting a sizable catalog of patterns available for developers to use. Within security in particular, solutions in the form of *security patterns* have appeared steadily in the literature to cover most major security-related concerns (see [15]). The pattern concept has also been found useful for the reverse side of the security solution landscape, namely, for security attacks, in the form of *attack patterns* [16] and *misuse patterns* [13,17,18]. Attack patterns capture the steps required to perform a specific security attack (exploit) in a generic fashion; misuse patterns, on the other hand, detail a complex attack on a system related to particular architectural components, capturing the structure and dynamics of the attack, forensic information and much else besides. Misuse patterns in particular, like security patterns, are full *software patterns*, which capture a set of principal design decisions [19] or define constraints determining a family of architectures that satisfy them [20]. In the case of security patterns this implies architectural impact [15], and in the case of misuse patterns, a strong architectural relation.

Not all software patterns, however, have *direct* architectural impact, or *concrete* architectural relation. From a more abstract point of view, a pattern can be seen simply as "the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts" [12]. One can thus define a type of software pattern that has a more generic architectural relation conforming to the latter characterization, which we term an *abstract software pattern*. The security *solution* analog of this general type of pattern was proposed by Fernandez et al. [21] in the form of abstract security patterns. In this section, we define the *threat* analog: threat patterns, which are the core constituents of our threat taxonomy.

A conceptual model relating the various patterns for security mentioned thus far, as well as many of the main concepts contained in the rest of this paper, can be seen in Fig. 1, with the more important elements appearing in different colors or, if viewed without color, different shades of gray (purely to differentiate them from other elements and from each other). For associations that appear as being (solely) vertically aligned in the figure, an arrow to the left implies they should be read "upwards" (e.g. each security patterns addresses one or several specific security policies), and an arrow to the right implies they should be read "downwards" (e.g. each general threat pattern is for a given general architectural context). In what follows, words appearing in *italic* font refer to (class) elements in the model.