2011 International Conference on Network-Based Information Systems

Threat Tree Templates to Ease Difficulties in Threat Modeling

Ikuya Morikawa and Yuji Yamaoka
Fujitsu Laboratories Limited, Kawasaki, Japan
{*ikuya,yamaoka*}*@labs.fujitsu.com*

***Abstract*—Threat trees are notable tools in the security analysis process called "threat modeling". The trees are used to identify how and under what condition threats can be realized, which will help proper estimation of risks and planning of countermeasures. However, it is difficult for an average analyst to construct adequate trees, because security expertise, particularly from an attacker's perspective, is required to find potential attack scenarios. In this paper, we propose threat tree templates to help non-expert analysts to construct threat trees. Each template is a redundant threat tree, loaded with branches representing many possible attack scenarios, as well as typical examples of corresponding vulnerabilities and countermeasures against such attacks. We also propose a keyword system for the templates, designed to filter out irrelevant scenarios.**

***Keywords*-software design; security analysis; threat modeling;**

I. Introduction

Threat modeling[1][6] is a process of security analysis, proposed and promoted as a part of Microsoft Secure Development Lifecycle (SDL)[2]. One of its notable features is use of threat trees, also known as attack trees. A threat tree represents a set of possible attack scenarios toward a given threat, and is used to identify how and under what condition the threat can be realized against the target system.

Constructing threat trees, however, is the most difficult part in the process. It is because finding a possible attack scenario requires security expertise, especially from an attacker's perspective, which average analysts are often lacking in.

In this paper, we propose use of threat tree templates to lower such difficulties. The templates are a collection of redundant threat trees, loaded with branches representing many possible attack scenarios, as well as typical examples of corresponding vulnerabilities and countermeasures against such attacks. By using these templates, an analyst can easily construct threat trees for the target system, by choosing or modifying branches provided in them. Furthermore, there are keywords assigned to the significant branches in the templates. These keywords are carefully designed to enable easy and quick filtering of branches irrelevant to the target system.

The rest of the paper is organized as follows. In Section II, we illustrate the threat modeling process, the role of threat trees in it, and difficulties to use them. In Section III, we propose the use of threat tree templates to ease such difficulties. Their structure, usage, and the keyword system for filtering are described. Then we discuss an evaluation of our templates in Section IV, related works in Section V, and finally conclude the paper in Section VI.

II. Threat Modeling

In this section, we introduce the threat modeling process, threat trees, and describe difficulties of using threat trees.

A. Threat Modeling Process

Basically, the threat modeling process[1][2][6] is performed in the following steps:

1) Decompose the target system into components, preferably into data flow diagrams (see below).
2) Identify threats for each component. The process recommends STRIDE classification to think of potential threats of every kind. STRIDE is formed with six threat classes: spoofing, tampering data, repudiation, information disclosure, denial of service, and elevation of privilege.
3) Evaluate risk of each threat. A threat tree helps to illustrate how and under what condition the threat could be realized, i.e., the system might be attacked. To estimate an amount of the risk, an analyst can utilize DREAD or other methods.
4) Prioritize the threats. And then consider mitigations and countermeasures against them if needed.

A data flow diagram (DFD) is a diagram focused on creation, transfer, and storing of data. DFD is composed of the following four types of entities (or components): process, external entity (EE; also called interactor,) data store, and data flow. Hierarchical representation is often desirable; complex process (also multiprocess) entity is used for that purpose. A simple example of DFD is shown in Figure 1.

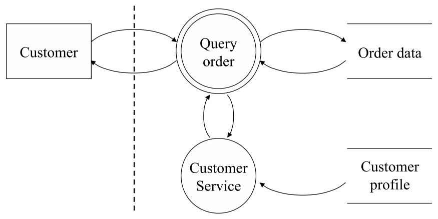

Figure 1. An example of data flow diagram (DFD).

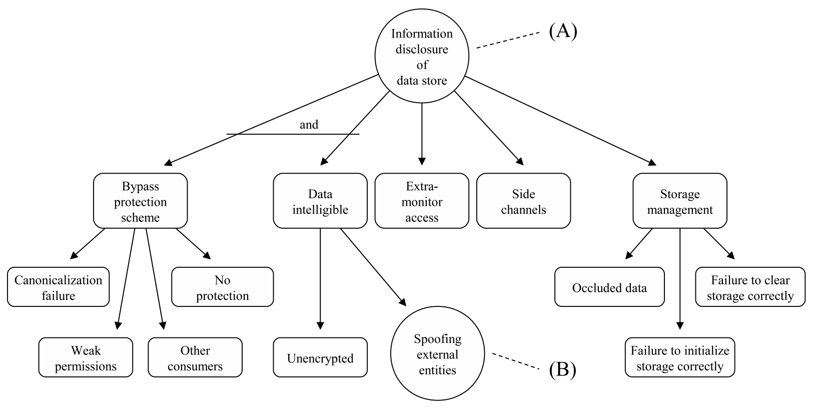

Figure 2. An example of a threat tree pattern shown in Howard and Lipner[2].

Notable features of the process can be summarized as follows.

- Focused on data. Representing the system in DFDs tends to clarify data flows and trust boundaries, which are important in security analysis.
- Semi-exhaustive enumeration of possible threats. By considering every combination of components and threat classes (STRIDE), one can somewhat thoroughly identify possible threats.
- Use of threat trees to break down threats. They provide, to some extent, a logical approach to analyze threats.
- Threats are supposed to be potential. It is not recommended to discuss threat mitigations during threat modeling. This prevents wrong prejudication.

As shown in above, threat trees play an important role in the threat modeling process. We illustrate them in the next subsection.

B. *Threat Trees*

A threat tree, also known as an attack tree[5], is a deductive method to analyze causes of a threat event. Figure 2 shows an example of a threat tree. The root node (A) represents a threat in question. Each of other descendant nodes represents a sub-threat: a condition or an event on which its parent can happen. If any node has multiple child nodes, their relation is disjunctive (OR) by default, unless the edges are marked specifically as "and", in which case the relation among them is conjunctive (AND). Sometimes a path ends with a node taking circular shape (B), representing a dependency to another threat tree. It means that its parent node can be endangered on successfully exploiting the depended threat. For example, a spoofing attack against a process protected by password authentication may be depending on an information leak of the corresponding password from a data store.

With such a structure, a threat tree as a whole represents how and under what condition the threat in question can be realized against a target system. Each path from a leaf node to the root corresponds to a possible attack scenario. If any conjunction (AND) appears on the way, the scenario is formed by fulfilling multiple paths. Although it is not shown in the figure, there can also be a mitigation as a leaf node, meaning that the corresponding attack path can be prevented by a countermeasure being addressed.

A threat tree can also be written in a structured text form as shown in Figure 3. We use this form preferably in this paper to save spaces. Instead of line numbers as shown in the figure, one can prefix each line with structured numbers such as "1.2.1."

```
1: Information disclosure of data store
2:  (AND) Bypass protection scheme
3:    Canonicalization failure
4:    Weak permissions
5:    Other consumers
6:    No protection
7:  (AND) Data intelligible
8:    Unencrypted
9:    Spoofing external entities
   ...
```

Figure 3. Text version of the same threat tree (partial) in Figure 2.

C. *Difficulties of Using Threat Trees*

Although a threat tree is a competent tool to break down a threat, it is considered as the most difficult part in the threat modeling process, because it requires security expertise to find possible attack scenarios. Thinking of a child node is equivalent to finding an attack technique or condition, which requires a lot of knowledge and experience in security, and even creativity, mostly from an attacker's perspective. Therefore, average practitioners often stumble at the task, resulting in unbearable amount of effort, insufficient analysis, or both. Even the advocates of the threat modeling, Howard and Lipner, admitted "the key weakness was building the threat trees" in their later publication[2].

It has been suggested that threat trees for the same threat class tend to be similar and therefore reusing existing ones could be helpful. In the above mentioned book, they recommended reuse of "threat tree patterns," rather than constructing trees by oneself, and devoted a whole chapter for exhibiting 12 tree patterns, each composed of 10–20 nodes and presented with a page-wide table of descriptions. The tree shown in Figure 2 is actually a threat tree pattern from the book.

Despite their claim, our experience has shown that these patterns are insufficient in actual use. We consider that the patterns have the following problems:

- Simply lacking in variations of attack scenarios.
- Too abstract. They lack concrete descriptions and examples. It is hard, especially for average analysts, to think of actual attack scenarios against a real system from the patterns.
- Mixed perspective, which may confuse analysts. For example, one leaf node denotes "Replay" that represents an attacker's behavior, where another denotes "Weak message integrity" that represents a flaw in the target system.

To cope with these problems, we propose threat tree templates, playing the role similar to the threat tree patterns, but are improved in many aspects.

III. Threat Tree Templates

A threat tree template is essentially a redundant threat tree, fully loaded with a lot of potential attack scenarios. Concrete examples of attack techniques, common flaws, countermeasures, and their descriptions are included as well, to help users to understand the threat.

The templates are handled in two phases: when a template designer prepares one, and when a template user (who is performing threat modeling) constructs a threat tree from it. It is assumed that template designers have appropriate security expertise, whereas template users are non-expert in security. In this section, we use the word "user" as a reference to a template user.

Basically, we provide six templates, one for each threat class of STRIDE. One can specialize them to suit better for a specific system or setting.

In the rest of this section, we describe the structure and the instances of the templates, their usage, and the keyword system for filtering.

A. Structure

A threat tree template is a tree composed of the following types of nodes: Threat, Dependency, Example, and Mitigation. It is essentially a tree of Threats, optionally with nodes of the other types.

Threat

Threat node represents any threat. It must be a root node or a child of another Threat node. Every Threat node should be accompanied with a description written from an attacker's perspective. Sometimes it may be described as "An attacker can {something harmful} by exploiting {some weakness}," if it is beneficial to mention a related weakness.

Dependency

Dependency means that its parent can be realized depending on another threat tree. Dependency must be a child of a Threat or Example node. Optional.

Example

Example represents an illustration of an attack or a vulnerability for the corresponding Threat. It must be a child of a Threat node or another Example node. Optional.

Mitigation

Mitigation optionally represents a common mitigation technique or countermeasure for its parent. It must be a child of a Threat or an Example node. Optional.

Note that a child threat in the templates can be an elaboration of its parent, instead of only being a cause of the parent as in usual threat trees.

With nodes of these types, a template forms a tree structure that typically looks like Figure 4.

```
1: Threat
2:   Threat
3:     Example
4:     Mitigation
5:     Dependency
6:   Threat
7:     Threat
8:       Example
9:         Mitigation
10:      Example
11:    Threat
12:      Mitigation
```

Figure 4. Typical structure of a threat tree template. (Partial)

Relations among child nodes are disjunctive (OR) by default. A conjunctive (AND) relation is represented by an And node, not mentioned in the above for simplicity.

B. Embodiment

Figure 5 shows our threat tree template for Spoofing. It is only depicting Threat and Dependency nodes; Examples and Mitigations are omitted for simplicity and space saving. Threats and Dependencies comprise the backbone of the tree, representing its abstract structure, while Examples and Mitigations provide concrete expressions of attack scenarios.

In the figure, Dependency nodes are followed by terms such as "`<- Tampering`", representing that it is caused

```
1: Threat: Spoof this process/EE by pretending as another process/EE.
2:   Depend: Tamper with an id/auth function. [system] <- Tampering
3:   Depend: Obtain an id/auth code. [system, human] <- Information leak
4:   Threat: Guess an id/auth code. [system]
5:     Threat: Simply guess, perform a brute-force and/or dictionary attack.
6:     Threat: Obtain a code in a valid way, and guess an other's from it.
7:   Threat: Exploit a flaw in id/auth technology. [system, human]
8:     Threat: Exploit a bug or misconfiguration of id/auth function. [system]
9:     Threat: Falsify id/auth code to cheat id/auth function.
10:    Threat: Obtain an encrypted id/auth code, and use it in a replay attack.
11:    Threat: Force a known id/auth code to another user, so that an attacker
               can pretend. [system]
12:  Threat: Hijack any process working for another user. [system]
```

Figure 5. Threat tree template for Spoofing. Only Threat and Dependency nodes are shown.

by another threat (e.g., Tampering) against the same or another entity. Given in square brackets (such as "`[system, human]`") are keywords assigned for that node. Keywords and their use in filtering are explained in Section III-D.

To depict the size of the templates, Table I shows the numbers of nodes by node type in the templates for all threat classes. The numbers of paths are shown as well, which can be considered rough estimates of numbers of attack scenarios.

Table I
NUMBER OF NODES AND PATHS IN THREAT TREE TEMPLATES.

Threat class	Nodes			Paths
(STRIDE)	Threat + Dep.	Example	Mitig.	
Spoofing	12	37	25	32
Tampering	11	26	15	26
Repudiation	6	12	17	13
Info Disclosure	17	37	20	39
DoS	8	13	12	13
Elevation of Priv.	7	11	10	11

C. Constructing a Threat Tree

Threat tree templates are used to construct threat trees for each entity in DFD. There is one template for each threat class of STRIDE. A user picks a corresponding template as a basis of threat tree, then browses it to find attack scenarios relevant to the target system, referring to their descriptions and examples. Irrelevant nodes can be removed. One can upgrade a relevant attack example to a regular threat node, or copy and modify an existing threat or example to be more suitable for the situation. If it is found that the threat is depending on another threat implied by a Dependency node, the corresponding threat tree should be prepared.

D. Keyword System for Filtering

Because a threat tree template contains a lot of attack scenarios, their descriptions and examples, even users with less security expertise can build a decent threat tree without missing possible scenarios. However, the sheer number of scenarios, descriptions, and examples makes the templates bulky and complex, resulting in moderate time and effort required on its use.

To cope with the problem, we designed a keyword system for quick filtering of nodes apparently irrelevant to the target system.

The keyword system is designed for simplicity, to provide ease of learning and usage:

- A child node implicitly inherits the conditions for its parent. It eliminates necessity to copy keywords from the parent.
- If a child node has its own keywords, they narrow the condition inherited from its parent.
- Keywords are simply written as a list, without using operators (AND/OR) or parenthesis.

Keywords are used in both of the design phase and the tree-construction phase. In designing a template, the designer assigns keywords to the nodes. In constructing a threat tree from a template, a user with less security expertise chooses keywords applicable to each DFD entity, which are in turn used to filter nodes in the templates. In the following subsections, we will show the details of the keywords and their usage.

1) List of Keywords: The list of the keywords designed for our threat tree templates are shown in Table II. The first, left-most column shows eleven keywords currently available: *human*, *system*, *ui*, *file*, *physical*, *db*, *web*, *webc*, *webs*, *ds*, and *df*. The second column shows more descriptive names for them. The third column, titled "Related DFD Entity," is for template users, describing what kind of DFD entities each keyword is applicable to. The right-most column, "Related Threats/Examples," is for template designer, describing types of threats that each keyword should be assigned to.

2) Keywords in Templates: When designing templates, the designer can assign appropriate keywords for Threat or

Table II
LIST OF KEYWORDS.

Keyword	Meaning	Related DFD Entity	Related Threats/Examples
human	Human	Any external entity that is a human, or contains a human factor. Antonym of *system*, but a compound entity may have both *human* and *system*.	Exploiting a human factor. The threat depends on someone being fooled or making a mistake.
system	System	An autonomous system such as computers or program, or a part of it. Applicable to virtually all processes and some of external entities. Antonym of *human*, but a compound entity may have both *human* and *system*.	Exploiting a system. The threat depends on some design flaw, bug, misconfiguration, or just ignorance of preventing mishaps.
ui	User Interface	Any process that interacts with users. Any data that works for UI (such as HTML), or any process that creates or modifies such data are also included.	Like *human*, but *ui* is more specific to user-interactive functions, such as displaying information and receiving user input, which can be abused to fool users or falsify their interaction.
file	File	Any entity that uses or contains files or file systems.	Specific to file systems, such as path manipulation or temporary file abuse.
physical	Physically accessible	Any entity that may be physically accessed by an attacker, such as portable media/devices, or terminals/consoles potentially open to an attacker.	Threats that become feasible when an attacker physically accesses or gets hold of the device.
db	Database	Any entity that is a database, or is containing or using a database.	Specific to database, such as SQL injection.
web	Web	Concerning any behavior or data related to Web, such as HTTP and HTML. *webc* or *webs* should be applied instead of *web* if possible.	Specific to Web technologies. Should also be marked with *webc*, *webs*, or both.
webc	Web client	A process performing any client-side role for Web.	Specific to Web clients. Should be marked with *web*.
webs	Web server	A process performing any server-side role for Web.	Specific to Web servers. Should be marked with *web*.
ds	Data store	A data store, or a multiprocess containing a data store.	Threats only applicable to data stores.
df	Data flow	A data flow, or a multiprocess containing a data flow.	Threats only applicable to data flows.

Example nodes according to the fourth column in Table II. Keywords are written as a list given in brackets such as "`[system, human]`" in Figure 5. Order of appearance in the list is not significant.

3) Choosing Keywords for DFD Entity: On threat modeling, a user of the templates is advised to choose appropriate keywords applicable to each DFD entity in question, according to the third column in Table II. Table III shows typical choices for common instances of DFD entities, such as a human user, a Web server, and a database server. In the table, keywords in parentheses are meant to be optional, as they are not necessarily but often applicable for the corresponding entity type.

Note that a data store such as a DB server or a file server is suggested to have the keyword *system*, which is usually chosen for processes. It is meant to avoid missing potential threats, because such a server, often represented as a data store in DFD, actually contains intermediate processes performing some tasks like interpretation or access control.

We also provide a simple program that assists users to choose appropriate keywords in accordance with Table III.

4) Filtering: On filtering, a set of keywords assigned to each node in the template is converted to a predicate, regarding each keyword as a Boolean variable. Formally, assuming a node N has a set of keywords K, the predicate of relevance R_N for the node N is

$$R_N = \begin{cases} R'_N & \text{if } N \text{ is the root,} \\ R'_N \wedge R'_{parent(N)} & \text{otherwise,} \end{cases}$$

Table III
KEYWORDS TYPICALLY APPLIED TO DFD ENTITIES.

Entity	Example	Applicable keywords
External entity	Human	*human*, (*webc*)
	Human + device	*human*, *ui*, *system*, (*webc*, *file*, ...)
	External system	*system*, ...
Process	Web server	*system*, *webs*, *ui*, (*file*, ...)
	DB server	*system*, *ds*, *db*, (*file*, *ui*, ...)
	(Part of) Software	*system*, (*file*, *ui*, *db*, ...)
	Terminal device	*system*, *ui*, (*physical*, *file*, ...)
	Compound system	*system*, (*ds*, *df*, *file*, *ui*, *web*, *webc*, *webs*, *db*, *physical*, *human*, ...)
Data store	DB server	(Same as "DB server" process)
	File server or shared filesystem	*ds*, *system*, *file*, (*ui*, ...)
	File	*ds*, *file*, (*system*, ...)
	Data on RAM	*ds*, (*system*, ...)
	Portable media or document	*ds*, *physical*, (*file*, ...)
Data flow		*df*, (*physical*, *ui*, *web*, ...)

where

$$R'_N = \begin{cases} true & \text{if } K = \phi, \\ \bigvee_{k \in K} k & \text{otherwise,} \end{cases}$$

and $parent(N)$ specifies the parent node of N.

For example, if there are three nodes (X, Y, and Z) and their keywords and relations are "X [] $>$ Y [*system*, *ui*] $>$ Z [*web*]", where X is the root node and "$>$" represents a parent-child relation, then $R_X = true$, $R_Y = system \vee ui$, and $R_Z = (system \vee ui) \wedge web$. Other examples of the

Table IV
EXAMPLES OF DETERMINING PREDICATES

Node [keywords]	Predicate of relevance	Note
A	$true$	No keyword.
B [x]	x	A keyword.
C [x, y, z]	$x \vee y \vee z$	List of keywords.
B [x] > D [y]	$x \wedge y$	Narrowing.
E [x, y] > F [x, z]	$(x \vee y) \wedge (x \vee z)$ $= x \vee (y \wedge z)$	More narrowing.

Note: "A > B" specifies a node B that is a child of A.

predicates are shown in Table IV.

For each DFD entity, Boolean variables corresponding to keywords chosen by the user are regarded as $true$. In this setting, any node N that has a predicate $R(N)$ evaluated to $true$ are considered relevant to that DFD entity.

We also provide a prototype tool to perform calculation of predicates and filtering of nodes.

IV. Evaluation

We performed a preliminary evaluation on a small scale, to know how our templates could improve a process of constructing threat trees and its output. Given a sample application system and its decomposition as a DFD, we picked two threats, of Spoofing and Tampering respectively, from 60+ threats we identified. Each of three users (U_1, U_2, and U_3) tried to build threat trees for both threats, firstly without help of the templates and then secondly with them. Numbers of paths addressed in the first and the second tree are enumerated as p_1 and p_2 respectively, and the increase $\Delta = p_2 - p_1$ and the ratio p_2/p_1 are evaluated.

Table V shows the result. We can see that more paths are found by using a template than without it in every case, by a factor of a range between 1.3 and 2.2. Although it is difficult to bring out fair insights from this kind of experiments, we can naively presume that the templates helped to find attack scenarios that had been missed without using them, and thus improved quality of outcomes from the threat tree construction process.

Table V
EVALUATION RESULT

Target threat	User	# of paths		Path increase	
		p_1	p_2	Δ	p_2/p_1
A (Spoofing)	U_1	14	19	+5	1.357
	U_2	4	8	+4	2.000
	U_3	14	20	+6	1.429
B (Tampering)	U_1	5	11	+6	2.200
	U_2	3	5	+2	1.667
	U_3	5	9	+4	1.800

V. Related Works

Although threat trees have been known among security experts[5], it is only recently that they have become popular among software designers and developers since Microsoft proposed their use in the threat modeling process[1][6]. As already mentioned, they softened recommendation for use of threat trees in their recent publication[2], presenting threat tree patterns. Our proposal is to replace the patterns with concrete templates, which help even average practitioners to effectively construct decent trees.

Research toward improving threat trees is scarce. One notable proposal in that field is Unified Threat Model[3]. It defines a formulation of threat trees and also shows algorithms to evaluate a threat modeled by a tree. Although their proposal delivers a capable basis for automatic processing of threat trees, it focuses on evaluation of trees, rather than constructing them. Constructing threat trees largely depends on human intervention, and our proposal helps better in that aspect.

Among many kinds of threat catalogs, CAPEC[4] is most notable in conjunction with our proposal. CAPEC is an enormous collection of attack patterns, and one of its facets shows an organized tree of attack techniques. Currently it is not particularly suitable for use in threat modeling compared with our method, but there shall be room for adaptation.

VI. Conclusion

In this paper, we proposed use of templates for threat trees, which can ease difficulties of using threat trees in security analysis called threat modeling. Our templates contain plenty of possible attack scenarios, focused on attacker's perspective, as well as common examples of vulnerabilities and countermeasures. There is also a keyword system designed to enable filtering, so that security analysts can easily ignore irrelevant nodes and pick appropriate scenarios quickly. Our experimental result supports that they can improve quality of threat trees constructed, by helping to find more attack scenarios, which might be missed without using them.

References

[1] Howard, Michael and LeBlanc, David, *Writing Secure Code*, 2nd ed., Microsoft Press, 2002.

[2] Howard, Michael and Lipner, Steve, *SDL: The Security Development Lifecycle*, Microsoft Press, 2006.

[3] Li, Xiaohong and He, Ke, A Unified Threat Model for Assessing Threat in Web Applications, in Proc. of 2nd International Conference of Information Security and Assurance (ISA-08), pp. 142–145, 2008.

[4] The MITRE Corporation, CAPEC: Common Attack Pattern Enumeration and Classification, available at http://capec.mitre.org/.

[5] Schneier, Bruce, Attack Trees, *Dr. Dobb's Journal*, December 1999.

[6] Swiderski, Frank and Snyder, Window, *Threat Modeling*, Microsoft Press, 2004.