Using CAPEC for Risk-Based Security Testing

Fredrik Seehusen[1]

[1] Department for Networked Systems and Services, SINTEF ICT
PO Box 124 Blindern, N-0314 Oslo, Norway
{fredrik.seehusen}@sintef.no

Abstract. We present a method for risk-based security testing that takes a set of CAPEC attack patterns as input and produces a risk model which can be used for security test identification and prioritization. Since parts of the method can be automated, we believe that the method will speed up the process of constructing a risk model significantly. We also argue that the constructed risk model is suitable for security test identification and prioritization.

Keywords: Risk assessment, testing, security, risk-based testing

1 Introduction

Risk assessment and testing are two areas that are traditionally addressed in isolation and that are supported by dedicated tools and processes, e.g. ISO31000 [1] for risk assessment and ISO/IEEE 29119 [2] for testing. However, the combination of these two areas can be mutually benefiting. On the one hand, the risk assessment can be used to guide the testing. On the other hand, the testing can be used to validate the risk model estimates.

A risk-based testing process is a process obtained by adding a structured risk assessment activity to a traditional testing process. In order for this enhanced process to be an improvement over the traditional testing process, the time spent on the added risk activity must make up for the loss of time spent on the traditional testing activities. It is therefore desirable that the risk assessment should not be too time consuming and it should result in a risk picture which is useful for the testing.

In this paper, we present a method and a technique which addresses both of these issues. Firstly, the method automates much of the process of constructing a risk model, and also the process of test-identification and prioritization based on risk models. Secondly, the method produces risk models that describe CAPEC attack patterns including known vulnerabilities which provide a relevant starting point for security test identification.

The expected end user of our method is a security tester and/or a risk analyst. From the user perspective, the method has three steps. In Step I, the user selects a set of attack patterns from the CAPEC dictionary of attack patterns [9] and generates a risk model automatically using our CAPEC to risk model technique. In step II of the method, the user manually refines the risk model in order to make it system/domain specific. In Step III, the user automatically generates a prioritized list of test procedures from the risk model using our technique test procedure prioritization. The resulting test procedures are intended to be used as starting point for test design, implementation, and execution.

The main contribution of this paper is the description of the method and the technique for automated risk model generation based on CAPEC attack patterns. The technique for test procedure derivation and prioritization is described in a separate paper [14]. There are many approaches to risk-based testing, but we are not aware of any approaches that automates the construction of a risk model which is used for test identification and prioritization.

The paper is structured as follows: In Sect.2 we provide an overview of our method and describe the criteria it is intended to fulfill. In Sect.3, Sect.4, and Sect.5 we respectively describe step I, step II, and step III of our method in more detail. In Sect.6 we discuss related work and in Sect.7 we provide conclusions.

2 Overview of method and success criteria

In this section, we give and overview our method for using CAPEC attack patterns in a risk-based testing process and describe criteria it is intended to fulfill. In general, it is not given that any combination of risk assessment and testing is useful. There are many risk assessment methods. These differ w.r.t. to the way in which the risk assessment is documented, the degree of structure, and the intended target time. For instance, some check list based approaches may only take a few hours while other more rigorous approaches may take thousands of hours. In any case, in a risk-based testing process, the time spent on the risk assessment should make up for the time lost in the testing activity. Therefore it is important the risk assessment is not too time consuming.

In addition to this, the outcome of the risk assessment, which we will refer to as a *risk model*, should provide useful input to the testing activity in the sense that it will reduce time and/or increase quality of the test results. There are many different kinds of risk model documentation languages, and not all of them may be suitable. Furthermore, the risk model has to be on the right level of abstraction and have the right focus. For instance, if it is too vague or documents risks and circumstances that cannot be investigated through testing, then the risk model might not be suitable.

In summary, we believe that the fulfillment of the following criteria will increase the likelihood of a risk-based testing process being an improvement over a traditional testing process:

C1 The risk assessment should not be too time consuming.
C2 The risk assessment should help reduce the time spent on test identification/design.
C3 The risk assessment should help increase the quality of the testing activity.

The method presented in this paper is supported by two techniques/transformations (both of which have been implemented in proof-of-concept tools) which help automate parts of the process:

T1 A transformation from CAPEC attack a patterns into CORAS risk models.
T2 A transformation from CORAS risk models into a prioritized list of test procedures.

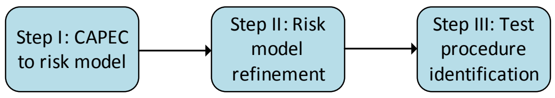
Step I: CAPEC to risk model
Step II: Risk model refinement
Step III: Test procedure identification

Fig. 1. *Method overview*

As illustrated in Fig.1, the method has three steps. In step I, the users selects a set of attack pattern from the CAPEC dictionary, and then uses **T1** to automatically generate a CORAS risk model. In step II, the user refines the resulting risk model to make it specific to the target of analysis. This step has to be performed manually. In Sect. 4, we will discuss different ways in which the risk model can be refined. In step III, the users annotates the risk model with test specific annotations, and uses **T2** to automatically generate a prioritized list of test procedures which can be used as starting point for test design.

The two techniques and the three steps of the method are meant to be used a part of a more comprehensive process for risk assessment and security testing. To illustrate this, we have in Fig.2 mapped the three steps of the method (as shown by the labeled circles) onto typical activities of a risk assessment and a testing process, in this case, corresponding to ISO31000 [1], and ISO/IEEE 29119 [2], respectively.

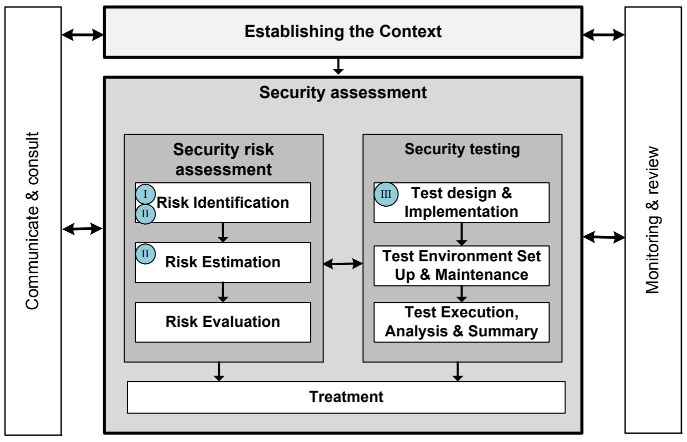

Fig. 2. *Method context*

3 Step I: From CAPEC to Generic CORAS Risk Models

In step I of our method, the user selects a set of attack patterns from the CAPEC dictionary and uses **T1** to automatically generate a CORAS risk model.

As shown in Fig. 2, we assume that an *Establishing the Context* activity is performed before entering into step I. We will not describe this activity in detail since it is not specific to the method we present. However, typical artefacts that may have been described before entering into step I are: the target of evaluation; likelihood and consequence scale definitions; asset definitions; risk evaluation criteria.

For selecting the relevant CAPEC attack patterns, the user should formulate clear attack pattern selection criteria, and then walk through the list of all attack patterns and discard those patterns which do not fulfill the critera. Examples of possible critera are:

- The attack pattern must be within the scope of the system under evaluation. For instance, if the attack pattern describes how to exploit a particular functionality such as login, and the target system does not have that functionality, the attack pattern would not satisfy the criterion.
- The attack pattern must exploit a weakness which is on the CWE (Common Weakness Enumeration) [10] list of top 25 most severe weaknesses.

The selection of the CAPEC attack patterns should be supported by a tool which allows the user to browse the attack patterns, and mark those which will be used as a basis for risk model generation. We have a developed a proof-of-concept tool for doing this. The tool allows the user to import the CAPEC dictionary (represented as an XML file), and display its contents in a tree view showing only information which is relevant for the translation into CORAS risk models. The tree view allows the user to delete and edit attack patterns, as well as to supply addition parameters to the transformation. After this editing is done, the user may use the tool to automatically export the attack patterns into a CORAS risk model.

In the following, we make precise what we mean by a *CORAS risk model* (Sect. 3.1) and a *CAPEC attack pattern* (Sect. 3.2), then we describe the technique **T1** for translating CAPEC attack patterns to CORAS risk models (Sect. 3.3).

3.1 CORAS risk models

There are many different kinds of languages for describing risk models. Our technique uses the CORAS language for model-based risk assessment [8]. CORAS risk models are used for documenting risks as well as events and circumstances that can cause risks. As illustrated by the example in Fig. 3, a CORAS risk model is a directed acyclic graph where every node is of one of the following kinds:

Threat A potential cause of an unwanted incident or threat scenario.

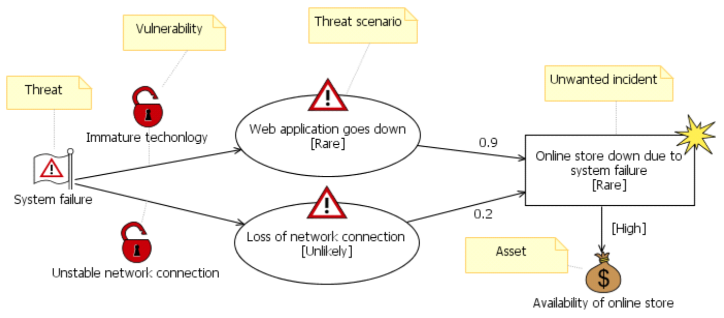
Vulnerability
Threat scenario
Threat
Unwanted incident
Immature techonlogy
Web application goes down [Rare]
0.9
Online store down due to system failure [Rare]
System failure
0.2
[High]
Loss of network connection [Unlikely]
Asset
Unstable network connection
Availability of online store

Fig. 3. *Example of a CORAS risk model*

Threat scenario A chain or series of events that is initiated by a threat and that may lead to an unwanted incident.
Unwanted incident An event that harms or reduces the value of an asset.
Asset Something to which a party assigns value and hence for which the party requires protection.

Note that risks can also be represented in a CORAS risk model. These correspond to pairs of unwanted incidents and assets. If an unwanted incident harms exactly one asset, as is the case in Fig. 3, then this unwanted incident will represent a single risk.

Relations and nodes may have the following assignments:

Likelihood values may be assigned to a threat scenario and unwanted incident A, estimating the likelihood of A occurring.
Conditional likelihood values may be assigned to relations going from A to B, estimating the conditional likelihood that B occurs given that A has occurred.
Consequence values may be assigned to relations going from A to B, estimating the consequence the occurrence of A has on B.
Vulnerabilities may be assigned to relations going from A to B, describing a weakness, flaw or deficiency that opens for A leading to B.

3.2 Common Attack Pattern Enumeration and Classification (CAPEC)

CAPEC is a comprehensive dictionary and classification taxonomy of known security attacks [9]. It contains more than 400 attack patterns, all of which are described in terms of a set of *attributes* such as attack name, method of attack, related weaknesses, typical severity etc. In order to define a transformation from a CAPEC attack pattern to a CORAS risk model, we must ask

– What attributes of an attack can be expressed in a CORAS risk model?
– Which of these attributes are described by the CAPEC attack pattern?

In answer to the first question, we believe that the following information about a security attack can be represented and would be of value in a CORAS risk model:

A The name of the attack.
B An estimate of how likely it is that the attack is initiated.
C An estimate of how likely it is that the attack will succeed given that it is initiated.
D A list of consequences/unwanted incidents which a successful attack can cause/lead to.
E An estimate of how likely it is that a successful attack will lead to the unwanted incidents.
F A list of assets that can be affected by the unwanted incidents of successful attacks.
G A description of which assets that can be harmed by an unwanted incident.
H An estimate of the consequence that an unwanted incident has on each of its assets.
I A list of vulnerabilities that can be exploited by the attack.

The attributes which can be derived from a CAPEC attack pattern are A, C, D, F, G, H, and I. Information about the attributes which are not described by a CAPEC attack pattern (B and E) can be supplied as input to the transformation to the risk model or/and as part of step II of our method.

Table 1 shows the format of the CAPEC attributes which can be expressed in a CORAS risk model. Henceforth, whenever we write CAPEC attack pattern, or attack pattern, we will mean an instance of the Table 1. Note that a CAPEC attack pattern may include many more attributes than those shown in Table 1. However, these attributes are difficult to expressed in a CORAS risk model, and are therefore ignored by our translation.

In Table 2 and Table 5, we have given examples of CAPEC attack pattern 34 and 64, respectively.

3.3 From CAPEC Instances to Generic CORAS Risk Models

Having made precise what is meant by a CORAS risk model and a CAPEC attack pattern we now describe the translation from a set of attack patterns to a CORAS risk model.

The information which cannot be derived from a CAPEC pattern can be supplied to the transformation in addition to the CAPEC instances. In particular, this information is:

- a mapping *lm* from CAPEC likelihoods to CORAS likelihoods;
- a default initiation likelihood *dil*, specifying the likelihood that an attack will be initiated;
- a default technical impact likelihood *dtil* specifying the conditional likelihood of a successful attack leading to a technical impact.

Name	A pair (ID, N) where ID denotes the identifier of the attack pattern and N denotes the name of the pattern.
Typical likelihood of exploit	A likelihood LE denoting the likelihood that the attack will succeed
Attack motivation-consequences	A list $(TI_1, S_1), (TI_2, S_2), \ldots (TI_n, S_n)$ of n pairs of the form (TI, S), where TI denotes the name of a technical impact and S denotes the scope of TI given as a subset of the set { Availability, Confidentiality, Integrity }
CIA impact	A triple (cia_c, cia_i, cia_a) denoting the impact/consequence the attack has on confidentiality, integrity, and availability, respectively.
CWE ID (Related weaknesses)	A list $v_1, v_2, \ldots, v_n$ of n elements denoting CWE vulnerabilities that can be exploited by the attack.

Table 1. Format of a CAPEC attack pattern

Name	(CAPEC-34, HTTP Response Splitting)
Typical likelihood of exploit	Medium
Attack motivation-consequences	(Execute unauthorized code or commands, Confidentiality, Integrity, Availability), (Gain privileges / assume identify, Confidentiality)
CIA impact	(High, High, Low)
CWE ID (Related weaknesses)	CWE-113 Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting'), CWE-697 Insufficient Comparison, CWE-707 Improper Enforcement of Message or Data Structure, CWE-713 OWASP Top Ten 2007 Category A2 - Injection Flaws

Table 2. Example of CAPEC attack pattern 34

Given this information, the outcome of a transformation from a CAPEC instance on the form shown in Table 1 will in the general case be a CORAS risk model on the form shown in Fig 4. To distinguish between variables and strings/constants, we have in Fig. 4 denoted all non-variables inside quotation marks. For instance, we have written "Attacker", meaning that Attacker is not a variable, but should appear as a constant string which is not dependent on the CAPEC instance being translated. The variables in the diagram such as ID, N, v_1, LE, etc. are all taken from the CAPEC instance which is assumed to be the input to the translation (see Table 1).

As illustrated in Fig. 4, each CAPEC instance is translated into two threat scenarios: one threat scenario corresponding to the initiation of the attack, and one threat scenario corresponding to a successful attack. The threat scenario describing attack initiation is given likelihood dil. The condition likelihood that the attack will be successful given that it is initiated is given by $lm(LE)$, i.e. the exploit likelihood of the CAPEC instance LE translated to the CORAS model likelihood by function lm.

Name	(CAPEC-62,Cross Site Request Forgery (aka Session Riding))
Typical likelihood of exploit	High
Attack motivation-consequences	(Read application data, Confidentiality), (Modify application data, Integrity), (Gain privileges / assume identity, Confidentiality)
CIA impact	(High, High, Low)
CWE ID (Related weaknesses)	CWE-352 Cross-Site Request Forgery (CSRF), CWE-664 Improper Control of a Resource Through its Lifetime, CWE-732 Incorrect Permission Assignment for Critical Resource, CWE-716 OWASP Top Ten 2007 Category A5 - Cross Site Request Forgery (CSRF)

Table 3. Example of CAPEC attack pattern 62

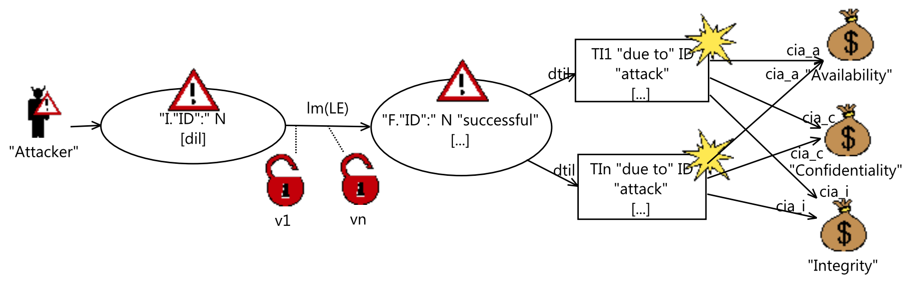

Fig. 4. *CORAS risk model showing outcome of translation function*

Given that the attack described by the CAPEC instance is successful, it can lead to one or more technical impacts with conditional likelihood *dtil*. Each technical impact of the CAPEC instance is translated to an unwanted incident in the CORAS model. These unwanted incidents may in turn be connected to one of the three assets Availability, Confidentiality, or Integrity, and the consequences of the unwanted incidents towards these is given by the CIA values of the CAPEC instance.

The assets that a technical impact is connected to are decided by the scope of the technical impact. For instance, if the scope of the technical impact includes all three assets, then the technical impact will be connected to all the three assets. If the scope only includes e.g. Confidentiality, then the technical impact will only be connected to the Confidentiality asset.

Each weakness of the CAPEC attack pattern is translated to a vulnerability in the CORAS risk model (shown as a red lock) and attached to the relation going from the threat scenario describing attack initiation to the threat scenario describing attack success.

As an example, assume that we supply the following input to the translation: the CAPEC instances 34 and 62 shown in Table 2 and Table 3, respectively; a mapping lm defined by $\{Low \mapsto eLow, Medium \mapsto eMedium, High \mapsto$

eHigh}; a default initiation likelihood *iHigh*; a default technical impact likelihood *tMedium*. Then the output of the translation will be the CORAS risk model shown in Fig. 5. Note that the likelihood values of the threat scenarios describing successful attacks and unwanted incidents describing attack consequences are undefined. However, these likelihood values can be calculated automatically from the other likelihood values in the risk model as described in e.g. [8] or [14].

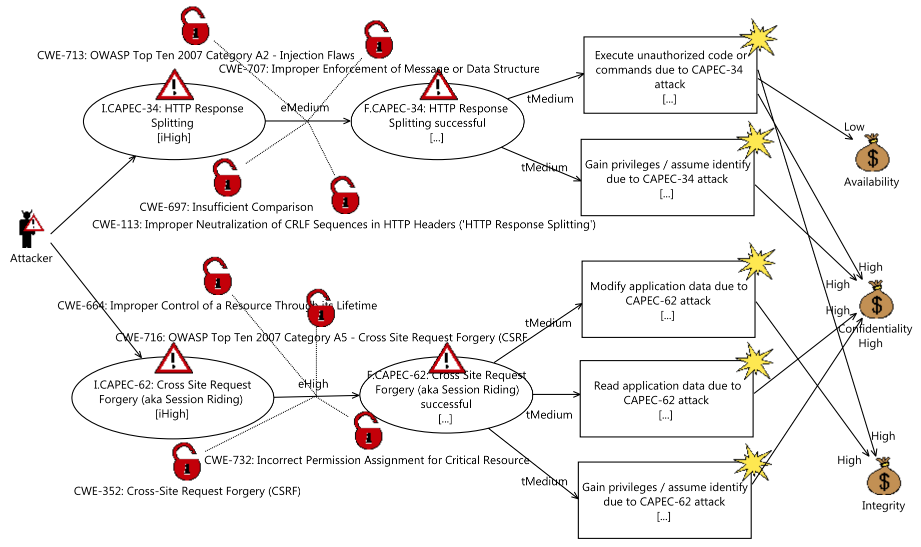
CWE-713: OWASP Top Ten 2007 Category A2 - Injection Flaws
CWE-707: Improper Enforcement of Message or Data Structure
I.CAPEC-34: HTTP Response Splitting [iHigh]
eMedium
F.CAPEC-34: HTTP Response Splitting successful [...]
tMedium
Execute unauthorized code or commands due to CAPEC-34 attack [...]
tMedium
Gain privileges / assume identify due to CAPEC-34 attack [...]
Low
Availability
CWE-697: Insufficient Comparison
CWE-113: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
Attacker
CWE-664: Improper Control of a Resource Through its Lifetime
CWE-716: OWASP Top Ten 2007 Category A5 - Cross Site Request Forgery (CSRF
I.CAPEC-62: Cross Site Request Forgery (aka Session Riding) [iHigh]
eHigh
F.CAPEC-62: Cross Site Request Forgery (aka Session Riding) successful [...]
tMedium
Modify application data due to CAPEC-62 attack [...]
tMedium
Read application data due to CAPEC-62 attack [...]
tMedium
Gain privileges / assume identify due to CAPEC-62 attack [...]
CWE-732: Incorrect Permission Assignment for Critical Resource
CWE-352: Cross-Site Request Forgery (CSRF)
High
High
High
Confidentiality
High
High
High
Integrity

Fig. 5. *Risk model obtained by translation of CAPEC-34 and CAPEC-62*

4 Step II: From Generic CORAS Risk Models to Target Specific Risk Models

The translation of CAPEC instances results in a CORAS risk model which is not specific to a particular system under test or target of evaluation. For this reason, we suggest that CORAS risk model be manually refined to make it more relevant for a particular target of evaluation. There are several different ways that the CORAS risk model can be refined for this purpose. In this section, we cover the most important ones.

4.1 Refinement of Likelihood and Consequence Values

All likelihood and consequences of the CORAS risk model obtained from a set of CAPEC instances are not specific to the target of evaluation. One way of refining the risk model is therefore to examine each likelihood and consequence estimate of the risk model, and adjust them as necessary. For instance, both threat scenarios describing attack initiation are in Fig. 5 given the likelihood *iHigh*. As previously described, this a default likelihood value which is supplied as an additional parameter to the transformation since the likelihood cannot be derived from the CAPEC patterns. The user of our approach should examine these likelihood values in particular, and adjust them if necessary.

4.2 Refinement by Element Splitting

In some cases, it may be the case that some of the attacks or technical impacts derived from the CAPEC instances are described in a too generic way. In these cases, the user should consider refining the risk model by splitting threat scenarios or unwanted incidents. For instance, if it necessary to distinguish between different features of the target of evaluation that are subject to the attack, then this can expressed in the risk model by splitting threat scenarios. For instance, we could split the threat scenario *I.CAPEC-34: HTTP response splitting* in Fig. 5 into the two threat scenarios *I.CAPEC-34A: HTTP response splitting targeting feature A* and *I.CAPEC-34B: HTTP response splitting targeting feature B*. This will allow us express the fact that the CAPEC-34 attack may be initiated with different likelihoods against feature A or feature B.

4.3 Refinement by Element Merging

The opposite of refinement of splitting is refinement by merging. If threat scenarios or unwanted incidents in the risk model describe similar phenomena, then we should consider merging them. For the risk model which is generated from the CAPEC instances, it may be particularly relevant to merge the unwanted incidents describing consequences of CAPEC attacks, since many attacks have the same kinds of consequences. For instance, in Fig. 5, there are two unwanted incidents called *Gain privileges/assume identify due to CAPEC-34 attack* and *Gain privileges/assume identify due to CAPEC-62 attack*. These unwanted incidents represent the same kind of consequence and they differ only in the manner of initiation, and we may therefore consider merging these into one unwanted incident.

4.4 Refinement by Element Addition

The final kind of refinement that we will consider is refinement by element addition. This kind of refinement may be particularly relevant for defining new risks that are specific to the target of evaluation. All unwanted incidents in a risk model derived from CAPEC instances correspond to technical impacts which are described in a quite general manner. Defining new unwanted incidents which are more specific to the target of evaluation and that can be caused by the technical impacts may therefore be relevant. In addition to this, if many CAPEC instances are transformed into a risk model, then we can potentially end up with a great number of possible risks (recall that a risk corresponds to an unwanted incident that harms an asset). Therefore focusing the risk assessment on a few risks which are specific to the target of evaluation is a good way of making the risk model more manageable.

An example of this is given in Fig. 6 where the risk model of Fig. 5 has been refined by adding three new unwanted incidents to the risk model (as shown on the far right of the diagram), and connecting these to the old unwanted incidents and the assets.

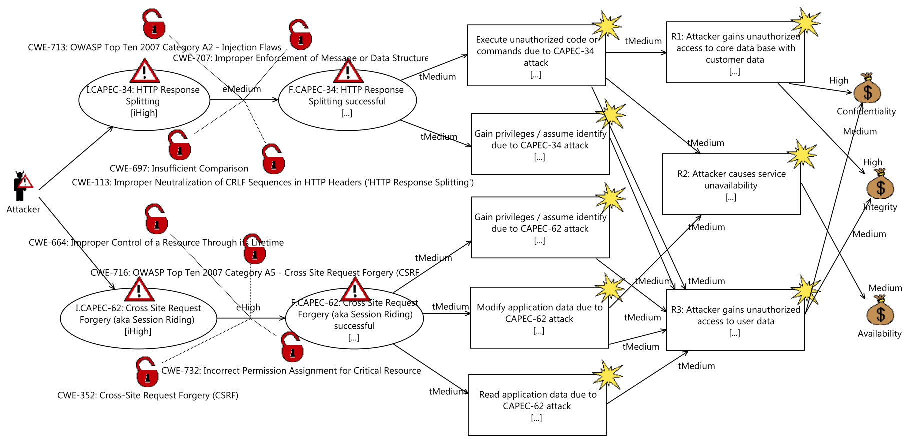
CWE-713: OWASP Top Ten 2007 Category A2 - Injection Flaws
CWE-707: Improper Enforcement of Message or Data Structure
I.CAPEC-34: HTTP Response Splitting [iHigh]
eMedium
F.CAPEC-34: HTTP Response Splitting successful [...]
CWE-697: Insufficient Comparison
CWE-113: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
Attacker
CWE-664: Improper Control of a Resource Through its Lifetime
CWE-716: OWASP Top Ten 2007 Category A5 - Cross Site Request Forgery (CSRF
I.CAPEC-62: Cross Site Request Forgery (aka Session Riding) [iHigh]
eHigh
F.CAPEC-62: Cross Site Request Forgery (aka Session Riding) successful [...]
CWE-732: Incorrect Permission Assignment for Critical Resource
CWE-352: Cross-Site Request Forgery (CSRF)
tMedium
Execute unauthorized code or commands due to CAPEC-34 attack [...]
Gain privileges / assume identify due to CAPEC-34 attack [...]
Gain privileges / assume identify due to CAPEC-62 attack [...]
Modify application data due to CAPEC-62 attack [...]
Read application data due to CAPEC-62 attack [...]
R1: Attacker gains unauthorized access to core data base with customer data [...]
R2: Attacker causes service unavailability [...]
R3: Attacker gains unauthorized access to user data [...]
High
Medium
Confidentiality
Integrity
Availability

Fig. 6. *Example of refinement by element addition*

5 Step III: From Specific Risk Models To Test Procedures

The purpose of this step is to identify and prioritize test procedures that can be used as starting point for a security test design activity. This step has two tasks

- Determine whether testing is necessary
- If testing is necessary, use technique **T2** to generate and prioritize test procedures based on the risk models.

The technique **T2** is defined elsewhere [14], however we will discuss its use here for purpose of self-containment and for arguing that the risk model obtained from CAPEC translation is a suitable starting point for test procedure identification.

To determine whether testing is necessary, it is useful to represent the risks of a CORAS risk model in a *risk matrix*. Such a risk matrix can be automatically generated from the risk model given that we have defined all likelihood values precisely and that we have a risk model that is complete in the sense that all initial threat scenarios and all edges/transitions have been given likelihood values. An example of a risk matrix with the three risks R1 - R3 (shown on the right side of Fig. 6) is given in Fig. 7.

Here the vertical axis shows the consequence scale and the horizontal axis shows the likelihood scale. The likelihoods of the risks are given as intervals, i.e. the left hand side of the boxes indicates the minimum likelihood of the intervals, and the right hand side indicates the maximum likelihoods. This should be understood as an expression of the belief that the actual likelihood of the risks lies somewhere within these intervals without knowing precisely where. In the risk matrix of Fig. 7, the diagonal line separates the area into two risk values: *Acceptable* and *Unacceptable*. We see that risk $R1$ is *Unacceptable*, risk $R3$ is *Acceptable*, and $R2$ can be either *Acceptable* or *Unacceptable* depending on what its actual likelihood is.

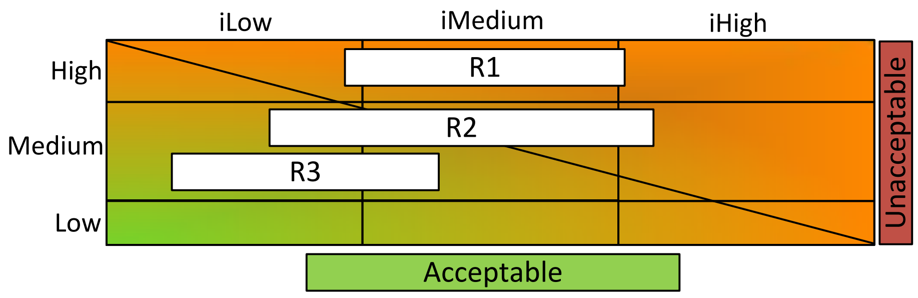

Fig. 7. *Risk matrix*

How can we use the risk matrix to determine whether testing is necessary? In our view, in the context of risk-based testing, testing can be used a means of gaining knowledge which allows us to estimate the likelihoods of risks and their causes more accurately. In our view, testing is most useful/beneficial if the knowledge acquired through testing could lead to a change in decision making based on risks.

In the current example, the knowledge obtained from testing may be expressed as a narrowing of the likelihood intervals of the risks. If we assume that the decision on how to treat the identified risks is entirely based on their risk value, then testing is necessary it could help us determine a risks risk value more accurately. This means that the decision of whether to test or not is based on the uncertainty of the estimates of the risk model as opposed to the severity of risks. For instance, even though risk $R1$ is considered *Unacceptable* there is no need to obtain new information through testing to reduce the size of the likelihood interval because this will not change the risk value of the risk. However, risk $R2$ is a different matter, it spans both the acceptable and the unacceptable area, thus the problem is not necessarily that the risk is unacceptable, but that we do not know whether it is acceptable or not. For risk $R2$, it makes sense to perform testing to gain new knowledge that allows us to determine whether $R2$ should be treated or not.

If we decide that testing is necessary, the next task is to use technique **T2** for test prioritization and selection to generate a prioritized set of test procedures from the risk model. A risk model can be seen as a set of statements about the world. Testing a risk model corresponds to checking the degree to which these statements are correct. Given a risk model, the first question we must ask is which of its kinds of statements are the most natural starting point for test identification? As discussed further in [15], we believe that the statements derived from edges of a risk model are the most natural starting point. An edges going from a node A to a node B with conditional likelihood l means that "A leads to B with conditional likelihood l", thus a test procedure corresponding to this edge is a statement of the form "Check that A leads to B with conditional likelihood l".

Potentially, every edge of risk model gives rise to a test procedure. However, in practice we do not have time to test every one of these test procedures. Thus we have to prioritize and then select the test procedures that are most important. To achieve this, we must for each edge of a risk model ask whether its corresponding test procedure is within the scope of the risk assessment, and if yes, estimate the resources/effort it would require to implement and execute the test procedure. Subsequently, given an estimate of maximum total effort available for testing, we can use the technique described in [15] to obtain a prioritized list of test procedures that should be implemented and tested. In the following, we illustrate this in an example.

Assume we want to identify test procedures on the basis of the risk model shown in Fig. 6. Our first task is to check whether the test procedure corresponding to each edge of the risk model is within the scope of the assessment, and if yes, estimate the time it will take to test it. Assuming that we are only interested in performing software security testing, then the two edges going from the threat scenarios describing attack initiation to successful attacks are the most natural starting point for testing. Assume that we estimate the time it takes to implement and execute these tests procedures to 1 day each, and that we only have 1 day in total available for doing the testing. We are thus forced to choose which one of the test procedures to test. In order to decide this, we use our technique for test procedure prioritization described in [15], and automatically obtain the test procedures and their priority values as shown in Table 4. Since the first test procedure has a higher priority than the second, we choose the first as the test procedure to test. We believe that the test procedures of Table 4 are a suitable starting point for security test design. In general, we believe that the risk models generated from CAPEC instances (other than the ones shown in the example) are on a suitable level of abstraction for test procedure identification. This is particularly the case for the test procedures derived from edges describing the likelihood of successful CAPEC attacks and the vulnerabilities that may be exploited for this purpose.

Priority	Test procedure	Severity
1	Check that Cross Site Request Forgery (aka Session Riding) leads to Cross Site Request Forgery (aka Session Riding) successful with conditional likelihood $[0.001, 0.1]$, due to vulnerabilities OWASP Top Ten 2007 Category A5 - Cross Site Request Forgery (CSRF), Incorrect Permission Assignment for Critical Resource, Cross-Site Request Forgery (CSRF) and Improper Control of a Resource Through its Lifetime.	2.138E-4
2	Check that HTTP Response Splitting leads to HTTP Response Splitting successful with conditional likelihood $[1.0E-4, 0.001]$, due to vulnerabilities Insufficient Comparison, Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting'), Improper Enforcement of Message or Data Structure and OWASP Top Ten 2007 Category A2 - Injection Flaws.	3.152E-8

Table 4. Prioritized list of test procedures

6 Related work

The Common Weakness Risk Analysis Framework (CWRAF) [11] is similar to our work. CWRAF builds on the Common Weakness Scoring System (CWSS) and provide a way of customizing the CWSS scores to specific business domains, technologies or environments. This framework is similar to ours in that it is related to CAPEC (both CAPEC and CWRAF reference the same weaknesses) and that it can be used for the purpose of prioritization. However, the main difference between our approach and CWRAF is that we take likelihood values into account, whereas CWRAF on addresses consequences/impacts only. Thus CWRAF can be considered a more lightweight approach than ours. Another difference is that CWRAF is used to prioritize *weaknesses* and not *attack patterns* as we do.

There are approaches that are similar to ours in that they address the automatic generation of risk models. However, none of the approaches we are aware of take CAPEC as input. For instance, Sheyner et. al. [16] proposes an approach for automated generation and analysis of attack graphs. However, the approach assumes a formally defined safety property as input which in our opinion may limit the applicability of the approach. Similarly, Phillips and Swiler [13] also propose an approach for automatically generating attack graphs. The approach assumes as input a network configuration file describing a network topology, a set of attacker profiles, and a set of attack templates. Based on these inputs, an attack graph can be generated. The approach differs from ours in that is addresses *network attacks*, and that it is based on a description of a network topology.

There are many approaches that combine risk assessment and testing. See [3] and [5] for a survey of the literature in the area. Almost all the approaches to risk-based testing use risk assessment in one of two ways. Either (I) the risk assessment is used to prioritize those parts/features of the system under test that are most risky, or (II) risk assessment is used to identify tests (often as part of a failure/threat identification process). Our approach fits best into category (II). Other approaches that fall into this category are Murthy et. al. [12], Zech et al. [18, 17], Casado et al. [4], Kumar et al. [7], and Gleirscher [6]. However, none of these approaches use the risk assessment results for test prioritization and none of them consider the automated generation of risk models based on CAPEC.

7 Conclusion and Future Work

We have presented a method for risk-based testing that takes a set of CAPEC attack patterns as input and produces a risk model and a prioritized list of test procedures that can be used as a starting point for security testing. We have describe a technique for automating the construction of the risk model and shown how or method can be used in combination with a technique for test identification and prioritization.

We believe our method supports criteria **C1 - C3** as described in Sect.2. In particular, criterion **C1** is supported by our technique **T1** for risk model generation which reduces the time of constructing the risk model compared to a traditional manual process. Criterion **C2** is supported since the generated risk model is suitable for test procedure identification as argued in Sect. 5. Criteria **C3** is supported since our method provides a sound basis for test-procedure prioritization based on risk model information. This enables the testing to be focused on the attacks and/or vulnerabilities that are most relevant for obtaining an accurate and correct risk model.

As part of future work, we plan to further develop our proof-of-concept tools supporting our techniques, and unify these into a single tool which will support both risk model analysis (such as consistency checking, likelihood calculation, and risk visualization), risk model generation from CAPEC, and test identification and prioritization.

Acknowledgments. This work has been conducted as a part of EU project RASEN (316853) funded by the European Commission within the 7th Framework Program.

References

1. *ISO 31000:2009(E), Risk management – Principles and guidelines*, 2009.
2. *ISO/IEEE 29119, Software and system engineering - Software Testing-Part 1-4*, 2012.
3. M. M. Alam and A. I. Khan. Risk-based testing techniques: A perspective study. *International Journal of Computer Applications*, 65(1):42–49, March 2013.
4. R. Casado, J. Tuya, , and M. Younas. Testing long-lived web services transactions using a risk-based approach. In *Proc. 10th International Conference on Quality Software (QSIC)*, pages 337–340. IEEE Computer Society, 2010.

5. G. Erdogan, Y. Li, R. K. Runde, F. Seehusen, and K. Stølen. Approaches for the combined use of risk analysis and testing: a systematic literature review. *STTT*, 16(5):627–642, 2014.
6. M. Gleirscher. Hazard-based selection of test cases. In *Proc. of the 6th international workshop on automation of software test*, pages 64–70. ACM, 2011.
7. N. Kumar, D. Sosale, S. N. Konuganti, and A. Rathi. Enabling the adoption of aspects - testing aspects: A risk model, fault model and patterns. In *Proc. of the 8th ACM International Conference on Aspect-oriented Software Development*, AOSD '09, pages 197–206. ACM, 2009.
8. M. S. Lund, B. Solhaug, and K. Stølen. *Model Driven Risk Analysis - The CORAS Approach.* Springer, 2011.
9. MITRE. Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org (Visited 30 March 2015), 2015.
10. MITRE. Common Weakness Enumeration (CWE). https://cwe.mitre.org (Visited 14 April 2015), 2015.
11. MITRE. Common Weakness Risk Analysis Framework (CWRAF). https://cwe.mitre.org/cwraf/ (Visited 30 March 2015), 2015.
12. K. K. Murthy, K. R. Thakkar, and S. Laxminarayan. Leveraging risk based testing in enterprise systems security validation. In *Proc. of the First International Conference on Emerging Network Intelligence*, pages 111–116. IEEE Computer Society, 2009.
13. C. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. In *Proc. of the 1998 Workshop on New Security Paradigms*, NSPW '98, pages 71–79, New York, NY, USA, 1998. ACM.
14. F. Seehusen. A technique for risk-based test procedure identification, prioritization and selection. In *Proc. of Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*, pages 277–291.
15. F. Seehusen. A technique for risk-based test procedure identification, prioritization and selection. In *Proc. of Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium (ISoLA 2014)*, Lecture Notes in Computer Science, pages 277–291. Springer, 2014.
16. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *Proc. of the 2002 IEEE Symposium on Security and Privacy*, SP '02, pages 273–, Washington, DC, USA, 2002. IEEE Computer Society.
17. P. Zech, M. Felderer, and R. Breu. Towards a model based security testing approach of cloud computing environments. In *Software Security and Reliability Companion (SERE-C), 2012 IEEE Sixth International Conference*, pages 47–56. IEEE, 2012.
18. P. Zech, M. Felderer, and R. Breu. Towards risk - driven security testing of service centric systems. In *QSIC*, pages 140–143. IEEE, 2012.