Project Title	Wide – Impact cyber Security Risk framework
Project Acronym	WISER
Grant Agreement No	653321
Instrument	Innovation Action
Thematic Priority	Cybersecurity, Privacy & Trust, Risk Management, Assurance Models
Start Date of Project	01.06.2015
Duration of Project	30 Months
Project Website	www.cyberwiser.eu

D3.1 - CYBER RISK PATTERNS

Work Package	WP 3.1, WISER Modelling
Lead Author (Org)	Gencer Erdogan and Atle Refsdal (SINTEF)
Contributing Author(s) (Org)	Alberto Luca Biasibetti (AON), Susana Gonzalez Zarzosa (ATOS), Antonio Alvarez Romero (ATOS), Ales Cernivec (XLAB), Giorgio Aprile (AON).
Due Date	31.05.2016
Date	31.05.2016
Version	1.0

Dissemination Level

X	PU: Public
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

The work described in this document has been conducted within the project WISER. This project has received funding from the European Union's Horizon 2020 (H2020) research and innovation programme under the Grant Agreement no 653321. This document does not represent the opinion of the European Union, and the European Union is not responsible for any use that might be made of its content.

Versioning and contribution history

Version	Date	Author	Notes
0.1	28.09.2015	Gencer Erdogan and Atle Refsdal (SINTEF)	Initial version of risk patterns and related indicators.
0.2	29.09.2015	Gencer Erdogan and Atle Refsdal (SINTEF)	The CORAS language and its terminology.
0.3	26.04.2016	Gencer Erdogan and Atle Refsdal (SINTEF)	Introduction and inclusion of risk pattern Client-Server Protocol Manipulation.
0.4	29.04.2016	Gencer Erdogan (SINTEF)	Inclusion of indicators for pattern in Section 5.2.1 and synchronized indicators in all patterns with the list of indicators in Section 6.
0.5	09.05.2016	Gencer Erdogan (SINTEF)	Included Section 2, method for pattern and indicator identification.
0.6	10.05.2016	Gencer Erdogan (SINTEF)	Section 4. Minor update of Section 2.
0.7	12.05.2016	Gencer Erdogan (SINTEF)	Updated all patterns in Sections 5.1-5.6: Updated descriptions and graphical models with additional threat scenarios and vulnerabilities. Added structure of Sections 5.7-5.10.
0.8	17.05.2016	Alberto Luca Biasibetti (AON), Gencer Erdogan (SINTEF)	AON included initial version of risk pattern Denial of Service. SINTEF restructured Section 5 to separate patterns supported by Cyber Wiser Essentials and Cyber Wiser Plus.
0.9	18.05.2016	Alberto Luca Biasibetti (AON), Susana Gonzalez Zarzosa (ATOS), Antonio Alvarez Romero (ATOS), Ales Cernivec (XLAB), Gencer Erdogan (SINTEF)	Update of patterns in Sections 5.1.1 and 5.1.2. Inclusion of indicators from D4.1. Correction and update of all patterns with respect to indicators. Correction of some indicators (Section 6).

0.10	19.05.2016	Gencer Erdogan (SINTEF)	Inluded two additional risk patterns for Cyber Wiser Essential (Sections 5.1.3 and 5.1.4).
0.11	23.05.2016	Gencer Erdogan and Atle Refsdal (SINTEF)	Executive summary
0.12	23.05.2016	Gencer Erdogan (SINTEF)	Removed all predefined estimates from the risk models (Section 5). Conclusions (Section 7).
0.13	24.05.2016	Susana Gonzalez Zarzosa (ATOS)	Included indicators specific to Denial of Service attacks
0.14	24.05.2016	Gencer Erdogan (SINTEF)	Updated DoS attack pattern w.r.t. new indicators in v0.13. Described why CORAS risk models are used as basis for risk estimation and why the models do not cover impact estimation.
0.15	25.05.2016	Gencer Erdogan and Atle Refsdal (SINTEF)	Final touches before first internal review.
0.16	27.05.2016	Antonio Álvarez (ATOS), Giorgio Aprile (AON), Alberto Luca Biasibetti (AON)	Review with comments.
0.17	27.05.2016	Gencer Erdogan (SINTEF)	Updated document based on feedback from review.
0.18	29.05.2016	Gencer Erdogan (SINTEF)	Clean version for GA acceptance.
0.19	31.05.2016	Gencer Erdogan (SINTEF)	Update based on feedback from GA.
1.0	31.05.2016	Antonio Álvarez (ATOS)	Delivery to EC

Disclaimer

Table of Contents

List of Tables

List of Figures

Executive Summary

This document reports on cyber risk patterns provided by the WISER framework to support cyber risk modelling. The purpose is to support clients of WISER with a set of predefined risk patterns they may instantiate without going through an extensive risk analysis process. The risk patterns capture typical cyber risks in a generic way and are available to the client in CyberWISER Essential, as well as CyberWISER Plus. This document also provides guidelines for how to instantiate WISER risk patterns.

In order to address the most common attack scenarios, the WISER risk patterns are based on well-known and widely used libraries such as the Common Attack Pattern Enumeration and Classification (CAPEC) [19] and the Open Web Application Security Project (OWASP) [20]. The following points list the risk patterns considered by WISER and described in this document, which are also the risk patterns currently provided by the WISER framework. The WISER framework is designed in a fully scalable way, thus further patterns can clearly be added and the framework allows doing that in a simple and effective way (as explained in D2.3).

- Denial of service attack
- Invalidated redirects and forwards
- Bypass login by brute force or DNS login attack
- Compromise security via Trojan-malware
- Client-server protocol manipulation
- Session hijacking
- Cross site request forgery
- SQL injection
- Buffer overflow
- Relative path traversal

Each pattern is described textually and graphically. The textual representation provides a systematic description of the risk pattern (using tables), while the graphical representation illustrates the risk pattern in terms of CORAS risk models [17]. As explained in D3.2, the risk patterns are used as basis to develop machine-readable risk assessment algorithms, which are in turn fed into the WISER Risk Assessment Engine to assess cyber risks in real-time.

The WISER framework collects information, through what we refer to as *indicators,* used by the algorithms executed in the Risk Assessment Engine to assess the risk exposure of an organization. In other words, indicators are an important part of WISER risk patterns as they provide the input used to assess the risk exposure. This document describes all indicators in WISER. We distinguish among four different types of indicators:

- *Business configuration indicators* are obtained manually through single/multiple-choice questions asked to the user when configuring WISER.
- *Vulnerability test result indicators* are obtained through non-intrusive vulnerability scans initiated by the user.
- *Network monitoring indicators* are obtained from network-layer sensors deployed in the running target infrastructure.
- *Application monitoring indicators* are obtained from application-layer sensors deployed in the running target infrastructure.

1 Introduction

1.1 Purpose and Scope

This report documents the 10 initial risk patterns developed for the WISER framework. A risk pattern in WISER is a generic description of a cyber-attack against cyber systems. The risk patterns are generic in the sense that they apply for systems or aspects of systems with similar characteristics, thus, not specified for one particular system. For example, a risk pattern addressing attacks against client-server protocols applies for systems based on the client-server architecture.

The risk patterns described in this document capture attacks of high consequence commonly carried out on cyber systems. The risk patterns also capture pieces of information obtained via the WISER infrastructure, referred to as indicators, used as basis to assess risk. The role of risk patterns in WISER is to provide users with predefined common attack scenarios they can select to assess risk, without the need to carry out an extensive risk analysis. The risk patterns are available to the client in Cyber Wiser Essential, as well as Cyber Wiser Plus.

The patterns are represented as CORAS risk models. A CORAS risk model describes risk patterns in terms of threats, threat scenarios initiated by threats, risks caused by threat scenarios, vulnerabilities exploited by threats in order to cause risks, and finally, security assets harmed. The risk patterns also represent indicators provided by the WISER infrastructure used as basis for assessing the risk exposure of an organization. We group indicators into the following categories: (1) indicators based on business configuration questionnaire, (2) indicators based on test results, (3) indicators based on network layer monitoring, and (4) indicators based on application layer monitoring. For each risk pattern described in this document, we also provide detailed description of its indicators. Deliverable D4.1 (design of the WISER monitoring infrastructure) also provides description of some of the indicators described in this document.

As mentioned above, the risk patterns are generic. This means that users of WISER may encounter situations where they have to adjust a pattern to their system. To support this, this document also provides guidelines for risk-pattern instantiation. Instantiating a risk pattern, in the context of WISER, means to modify the pattern such that it is adjusted specifically for the target system.

Cyber risks may be assessed at different abstraction levels. The risk patterns in this document are described at different abstraction levels. As listed below, this has several advantages.

- Different clients may have different preferences with respect to abstraction level. While some may be interested in assessments of detailed risks, others may prefer risks expressed in terms of more high-level business incidents.
- Offering patterns at different abstraction levels opens for risk aggregation, for example, by combining a set of low-level risk patterns with a more abstract pattern.
- We will exploit the three Full Scale Pilots in WISER to try out risk patterns. Having patterns at different levels of abstraction thus allows us to explore under what circumstances the different abstraction levels are best suited.

1.2 Structure of the document

In Section 2 we describe the approach used to identify risk patterns and their indicators. In Section 3 we describe the presentation format of WISER risk patterns, and in Section 4, we provide guidelines to instantiate the risk patterns. In Section 5, we describe in detail the 10 initial risk patterns developed for the WISER framework, while in Section 6 we describe all indicators used in the patterns. Finally, in Section 7 we conclude the report.

2 Method for pattern and indicator identification

Figure 1 illustrates the method used to identify, describe and model cyber-risk patterns. As mentioned in Section 1, this report documents the 10 initial risk patterns developed for the WISER framework. The process in Figure 1 was carried out to document each of the 10 risk patterns.

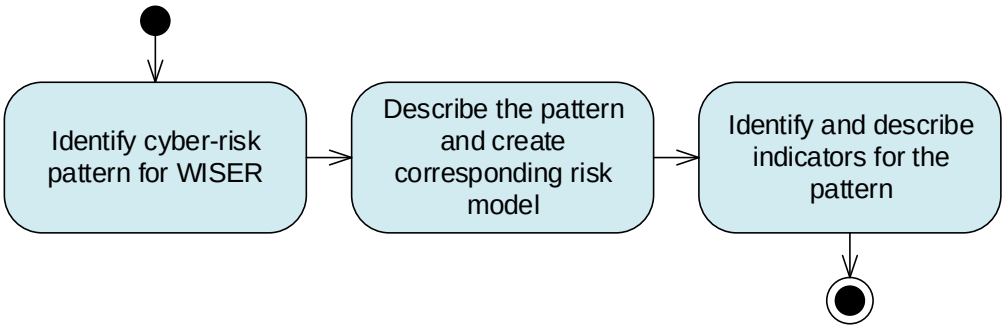

Figure 1: Method for pattern and indicator identification.

In the first step, we identified cyber-risk patterns by taking into consideration well-known cyber-attacks with high impact. In particular, we identified risk patterns by making use of the library Common Attack Pattern Enumeration and Classification (CAPEC) [19], and the top 10 security risks provided by Open Web Application Security Project (OWASP) [20], which are widely used libraries containing cyber-risk patterns. Most of the well-known cyber-attacks are in fact web-based [19][20], thus most of the patterns in this document are also web-based.

Having identified the pattern, next we described the pattern in a systematic way by making use of tables designed to structure the necessary information. In addition to a unique ID, name, description, and source of the pattern, the table includes information related to general characteristics of the target affected by the pattern, security assets affected, and a list of vulnerabilities exploited by the pattern. Based on the information in the table, we created a graphical model of the risk pattern by making use of the CORAS language [17].

Finally, having created a graphical risk model of the risk pattern, we identified and described relevant indicators. We considered the elements in the risk model, such as threat scenario, vulnerability, and unwanted incident, and for each element, we identified relevant indicators in terms of short textual descriptions. These indicators were added to the risk model, and attached to the considered element of the risk model. However, to link the indicators to the WISER infrastructure, we additionally described the indicators in detail. We did this in a systematic way by making use of tables designed to capture the necessary information. For each indicator, we described a unique ID, motivation for the indicator, the type of indicator (business configuration, test results, network-layer monitoring, or application-layer monitoring), as well as means of obtaining indicator value in the WISER infrastructure.

3 Presentation format for risk patterns

The risk patterns are presented textually and graphically. The textual representation provides a systematic description of the risk pattern, while the graphical representation illustrates the risk pattern in terms of CORAS risk models.

In the textual representation, we use tables to systematically organize and describe the patterns. Section 3.1 provides the table format used to describe the risk patterns. In the graphical representation, we use the CORAS language to model the risk patterns. Section 3.2 presents the CORAS language and describes the graphical elements in the language.

3.1 Risk pattern table format

Id:	<Unique ID>	**Name:**	<Pattern name>
Pattern source:	<The source in which the pattern is described>		
Target characteristics:	<Characteristics of the target affected by the risk pattern>		
Description:	<Pattern description>		
Affected security assets:	<List of security assets affected by the pattern>		
Exploited vulnerabilities:	<List of vulnerabilities exploited by the pattern>		
Related indicators:	<List of indicators relevant for the pattern>		

3.1.1 Id

Every risk pattern is uniquely identified. All the ID's have the naming convention WRP-<Number> where WRP stands for WISER Risk Pattern.

3.1.2 Name

Each pattern is described by a short name.

3.1.3 Pattern source

This field provides the source(s) in which the pattern is described such as online risk pattern repositories, scientific papers, white papers, etc.

3.1.4 Target characteristics

This field describes characteristics of the types of target affected by the risk pattern. For example, web-based applications, network-layer components, databases, and so on.

3.1.5 Description

This field gives a description of the risk pattern.

3.1.6 Affected security assets

This field provides a list of security assets affected by the risk pattern, for example confidentiality, integrity, and availability of data.

3.1.7 Exploited vulnerabilities

This field provides a list of vulnerabilities that are exploited by the risk pattern in order to achieve the unwanted incident, that is, the risk.

3.1.8 Related indicators

This field provides a list of indicators relevant for the risk pattern. The indicators are listed in terms of indicator identifiers (ID's). The complete description for each indicator is found in Section 6.

3.2 The CORAS language

Figure 2 shows an example of a CORAS risk model. The dashed arrows in the figure are not part of the model and are only used to point out the various constructs in the CORAS language. As illustrated, 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.

- *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. Notice that this means that the term asset is used in a wide sense; it can include any tangible or intangible entity of value for the party in question. In the context of cyber security, some typical examples are confidentiality, availability and integrity of information, as well as the reputation.

Risks correspond to pairs of unwanted incidents and assets. If an unwanted incident harms exactly one asset, as illustrated in Figure 2, then the unwanted incident represents a single risk. If an unwanted incident harms two assets, then the unwanted incident represents two risks, etc. Vulnerabilities are also represented in a CORAS risk model. Before explaining what vulnerabilities are, we consider the three kinds of relations in a CORAS risk model.

- *Initiates relation*: A relation that goes from a threat *A* to a threat scenario or an unwanted incident *B*, meaning that *A* initiates *B*.
- *Leads to relation*: A relation that goes from a threat scenario or an unwanted incident *A* to a threat scenario or an unwanted incident *B*, meaning that *A* leads to *B*.
- *Impacts relation*: A relation that goes from an unwanted incident *A* to an asset *B*, meaning that *A* impacts *B* with some consequence.
- *Vulnerability*: A weakness, flaw or deficiency that opens for *A* leading to *B*. Vulnerabilities are modelled as open locks, and are attached on the initiates relations or the leads-to relations.

To support risk estimation, the CORAS language uses the following three risk measures.

- *Likelihood values*: May be assigned to a threat scenario or an unwanted incident *A*, estimating the likelihood of *A* occurring.
- *Conditional probabilities*: May be assigned to the leads-to relations going from *A* to *B*, estimating the probability that *B* occurs given that *A* has occurred.
- *Consequence values*: May be assigned on the impacts relations going from *A* to *B*, estimating the consequence that the occurrence of *A* has on *B*.

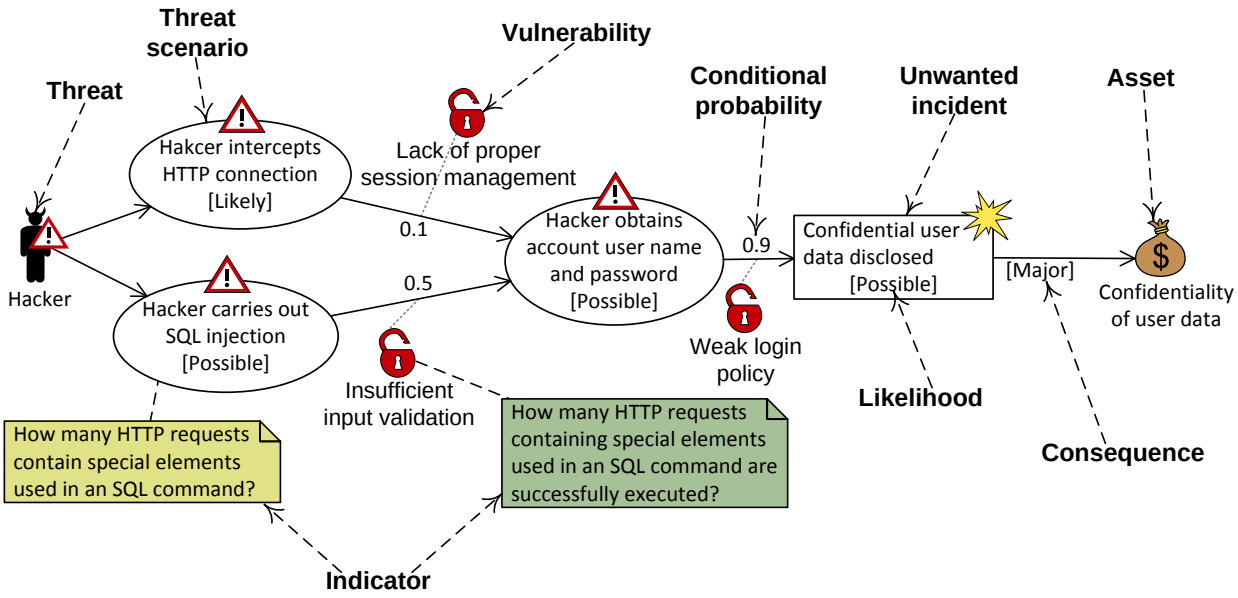

Figure 2: Example of a CORAS risk model.

What has been described to this point is part of the CORAS language. The reader is referred to Lund et al. [17] for a further explanation of the CORAS approach and the various constructs in the CORAS language. However, in the context of WISER it is necessary to extend the CORAS language with additional constructs referred to as *indicators*.

- *Indicator*: By indicator we mean a piece of information that is relevant for assessing the risk level. An indicator may be assigned to any risk-model element. Let us consider a simple example. Assume we have modelled a vulnerability *Weak password*. A potential indicator for this vulnerability could be the number of users that do not follow the password policy. If we gather information indicating a large number of users not complying with the password policy, we may argue that the target under analysis is most likely exposed to the vulnerability *Weak password*. The indicators are defined in terms of questions, for example, "How many users do not comply with the password policy?"

Indicator values may be obtained by different means. For example, in some situations it is sufficient to base the indicator value on expert knowledge, while in other situations it may be necessary to implement sensors at the network layer in order to derive indicator values based on continuous network monitoring. Moreover, some indicator values may be obtained by a combination of, for example, expert knowledge *and* continuous network monitoring (obtaining the same indicator value by different means). In the context of WISER we have identified four types of indicators.

- *Business configuration*: Indicator values are obtained by asking business related questions. The indicator values are thus based on expert knowledge. This type of indicator is non-intrusive in the sense that it does not require the implementation of sensors in the target under analysis.
- *Test results*: Indicator values are obtained by carrying out tests. The indicator values are thus based on test results. This type of indicator is non-intrusive in the sense that it does not require the deployment of sensors in the target under analysis.
- *Network-layer monitoring*: Indicator values are obtained by monitoring the network layer. This type of indicator is intrusive in the sense that sensors need to be deployed in the network-layer of the target under analysis.

- *Application-layer monitoring*: Indicator values are obtained by monitoring the application layer. This type of indicator is intrusive in the sense that sensors need to be deployed in the application-layer of the target under analysis.

A colouring convention is used to differentiate graphically between the four indicator types. As shown in Table 1, the indicator type business configuration is represented by the colour blue, test results are represented by the colour green, network-layer monitoring is represented by the colour yellow, and application-layer monitoring is represented by the colour red.

Table 1: The four indicator types and their associated colouring convention

Indicator type:	Business configuration (non-intrusive)
	Test results (non-intrusive)
	Network-layer monitoring (intrusive)
	Application-layer monitoring (intrusive)

Figure 2 illustrates two indicator types: network-layer monitoring and test results. The indicator "How many HTTP requests contain special elements used in an SQL command" is of type network-layer monitoring and is assigned to the threat scenario "Hacker carries out SQL injection". The indicator "How many HTTP requests containing special elements used in an SQL command are successfully executed?" is of type test results and is assigned to the vulnerability "Insufficient input validation".

As mentioned previously, the indicator values for one particular indicator may be gathered by different means. This means that one indicator may belong to several types. However, each indicator is assigned a main/default type and if necessary supported by the other types. For example, the indicator "How many HTTP requests containing special elements used in an SQL command are successfully executed?" is mainly of type test results, but it can also be supported by application-layer monitoring if relevant sensors are deployed in the target under analysis.

4 Guidelines for risk-pattern instantiation

A risk pattern in WISER is basically a generic risk model assumed to be of general relevance and therefore offered to all organizations adopting WISER. Clients can start from a selection of existing risk patterns and instantiate these for their particular system instead of creating their own risk models from scratch. Instantiating a risk pattern, in the context of WISER, means to modify the pattern such that it is adjusted specifically for the target system. Figure 3 illustrates the process for instantiating risk patterns in WISER. This figure is based on the overall method for cyber risk modelling documented in D3.2. The only difference is that the process depicted in Figure 3 takes as input an existing risk pattern, while the process in D3.2 is carried out to create a risk model from scratch.

4.1 Modifying CORAS diagram (Step 1)

The process for risk-pattern instantiation depicted in Figure 3 starts with the assumption that the client has already selected a risk pattern to instantiate. This includes the CORAS diagram representing the risk pattern as well as its corresponding DEXi or R model. Having selected the risk pattern, the client needs to answer a question in order to proceed with the process. At the first decision-point in Figure 3, the client needs to decide whether the CORAS diagram capturing the selected risk pattern is valid for the client's target system. If the answer to this question is no, then the client proceeds to Step 1. In Step 1.1, the client modifies the CORAS diagram in order to adjust the pattern specifically for the client's target system. The client may modify the CORAS diagram in terms of *editing*, *adding*, or *deleting* risk-model elements:

- *Edit* the textual description in the risk-model elements. For example, consider the risk model in Figure 2. The client may edit the threat scenario "Hacker carries out SQL injection" to

"Hacker carries out SQL injection on the database containing customer data". Another example is to rewrite the vulnerability "Insufficient input validation" as "Insufficient input validation of transactions carried out on customer database", etc.

- *Add* risk-model elements such as additional threat scenarios or vulnerabilities.
- *Delete* risk-model elements the client perceives as unnecessary for the target system.

Having adjusted the CORAS diagram, the client may proceed to Step 1.2. The purpose of Step 1.2 is to ensure that the CORAS diagram reflects, as far as possible, the actual reality with respect to potential threats, vulnerabilities, threat scenarios and risks. This is because the CORAS diagram serves as the basis for the machine-readable risk-assessment algorithm, that is, the DEXi or R model as explained in D3.2. Step 1.2 in Figure 3 is identical to Step 1.2 in the overall method for cyber risk modelling documented in D3.2. The reader is therefore referred to D3.2 for further explanation on validating the CORAS diagram. The output of Step 1 is a validated CORAS diagram with indicators.

However, if the answer to the initial question ("Is the CORAS diagram valid for your target system?") is yes, then the client may skip Step 1, because then there is no need to modify and validate the CORAS diagram. Moreover, if the answer to the aforementioned question is yes, then the client must answer a second question: "Is the assessment algorithm valid for your target system?" – in other words, whether the corresponding DEXi or R model is valid. If the answer to this question is yes, then the client may skip Step 2, because then there is no need to update and validate the assessment algorithms. In summary, if the answer to both questions in the process depicted in Figure 3 is yes, then that means the client choose to use the selected risk pattern as provided by WISER, without any modifications. However, if the answer to the first or second question is no, then the client needs to carry out Step 2, which is discussed in Section 4.2.

4.2 Modifying assessment algorithm (Step 2)

As depicted in Figure 3, there are two possible ways to enter Step 2. The first one is after carrying out Step 1 and the second one is if the answer to the second question is no.

In Step 2, the purpose is to update the assessment algorithms such that they correctly reflect the structure of the underlying CORAS diagram, and that it is valid for the target system. Depending on whether the assessment algorithm is defined in DEXi or **R** the client needs to carry out Step 2.1a (update assessment algorithm using DEXi) or 2.1b (update assessment algorithm using **R**), respectively. D3.2 explains how to define assessment algorithms based on the underlying CORAS model.

Having carried out Step 2.1a or 2.1b, the client needs to validate the assessment algorithms. The purpose is to establish its consistency and overall soundness in order to obtain user acceptance and confidence that the outputs from the algorithm provide useful information that reflect reality reasonably well. Step 2.2 in Figure 3 is in line with Step 2.2 in the overall method for cyber risk modelling documented in D3.2. The reader is therefore referred to D3.2 for further explanations on validating the assessment algorithm.

Finally, the output of Step 2 is a set of validated cyber-risk assessment algorithms. These assessment algorithms, defined in terms of DEXi or R models, are in turn used by the Risk Assessment Engine in WISER to assess cyber-risk exposure.

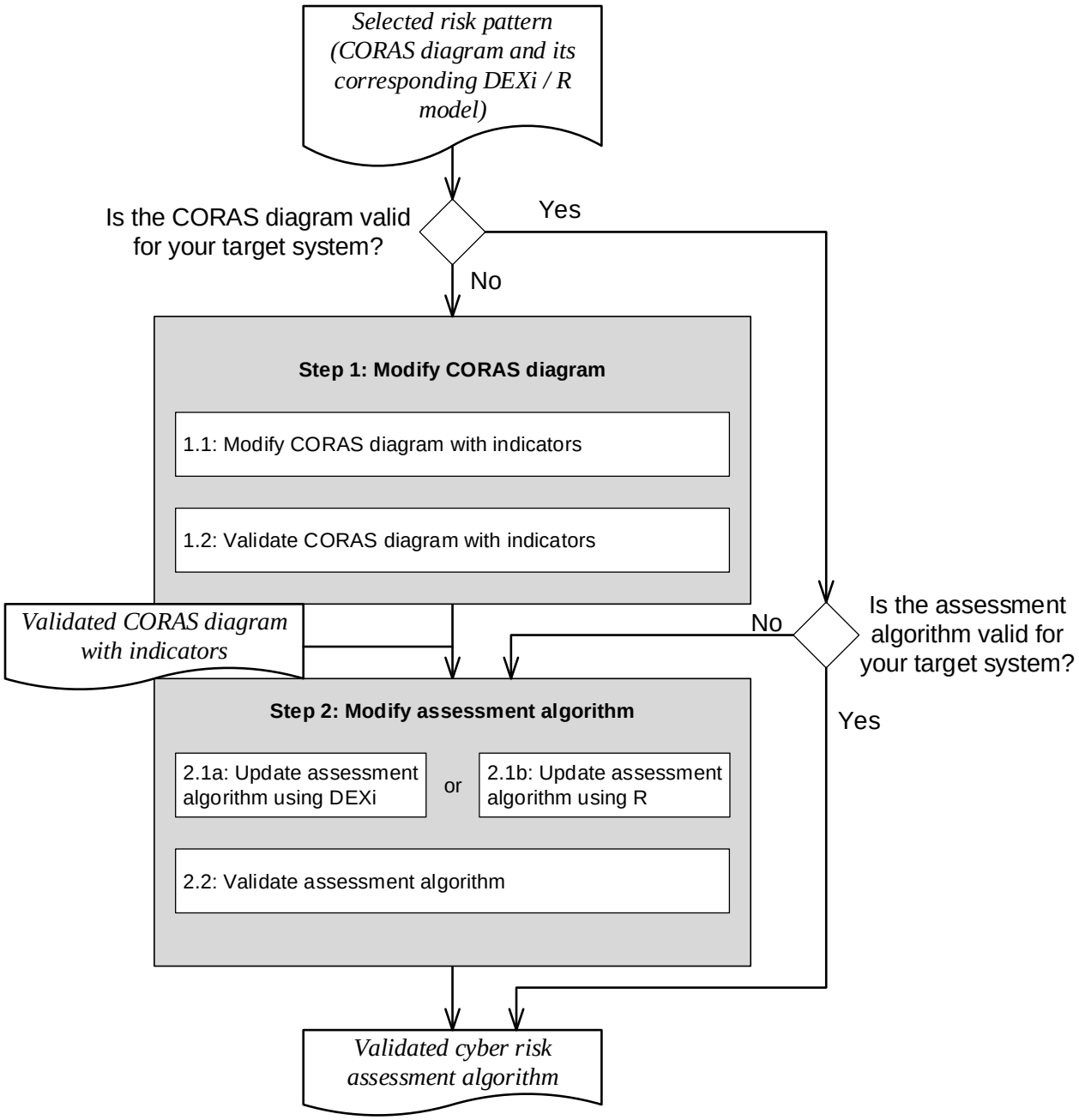

Figure 3: Process for risk-pattern instantiation

5 Risk patterns

This section presents risk patterns currently supported by the WISER framework. The WISER framework is designed in a fully scalable way, thus further patterns can clearly be added and the framework allows doing that in a simple and effective way (as explained in D2.3). The patterns are presented in two groups: patterns not including application-layer indicators (Section 5.1), and patterns including application-layer indicators (Section 5.2). The reason to why they are grouped in this manner is because application layer indicators are client-specific and they need to be adjusted

specifically for each target system. All risk patterns are first described textually and then graphically using CORAS risk models, as explained in Section 3.

However, before we present the patterns, we need to clarify the difference between likelihood estimation and consequence estimation of a cyber risk in context of WISER. As pointed out in Section 3.2, the likelihood estimate is an estimate for how often a risk may occur, while the consequence estimate is an estimate of the impact of a risk given that it occurs. In WISER we distinguish between economic impact and societal impact of risk.

As mentioned in Section 3.2, and illustrated in Figure 4, a risk corresponds to a pair of an unwanted incident and a security asset. The risk level of a risk is determined by the likelihood of the incident and its consequence for the asset.

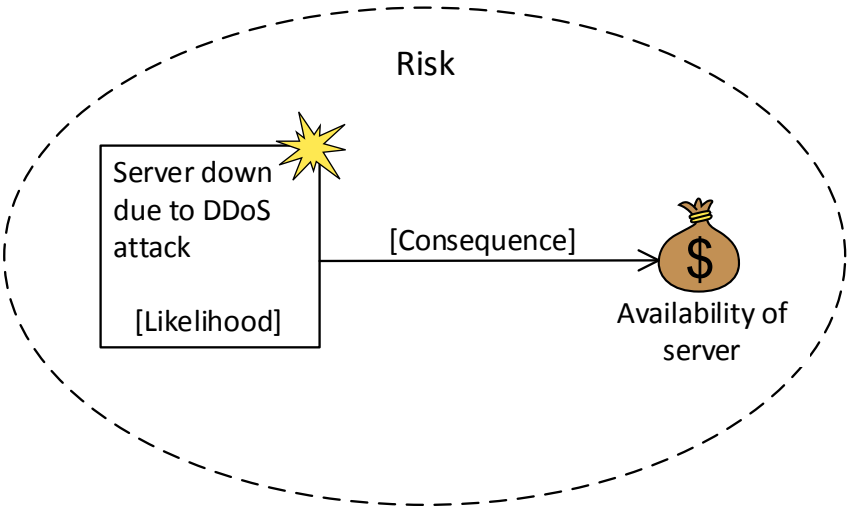

Figure 4: The relationship of risk, likelihood, and consequence

In this document, we use CORAS risk models with indicators to capture cyber-risk patterns. This has two main purposes. First, CORAS risk models are used as a basis to schematically define algorithms estimating the likelihood of risks (for example, the likelihood of "Server down due to DDoS attack" illustrated in Figure 4). That is, there is a direct link between the structure of a CORAS risk model and the resulting algorithm calculating the likelihood of the risks described in the model. Second, the indicators in CORAS risk models are used to capture sources from which data is collected to feed the algorithms calculating the likelihood. In other words, CORAS risk models capture cyber-risk patterns and are used as basis to estimate the likelihood of risks. This document covers the foundation for likelihood estimation. Detailed guidelines for how to translate a CORAS risk model into algorithms (in DEXi or **R**) calculating risk likelihood are provided in D3.2.

From a methodological perspective, the consequence estimation of a given incident is carried out in the same manner independently of the structure of the risk pattern. How to estimate the consequence of a risk is therefore covered in D3.2 (and later in D3.4). In particular, Section 9 in D3.2 explains how to assess the economic impact of cyber risks, and Section 10 in D3.2 explains how to assess the societal impact of cyber risks.

5.1 Patterns not including application-layer indicators

Although the patterns described in this section are not supported by application-layer indicators, they may be extended (following guidelines to instantiate patterns in Section 4) to support indicators at the application layer.

5.1.1 Denial of service attack

Id: WRP-1	**Name:** Denial of Service Attack
Pattern source:	This risk pattern is based on OWASP Denial of Service (DoS) attack [23] and OWASP Application Denial of Service [24].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications.
Description:	Denial of Service (DoS) is an attack technique with the intent of preventing a web site from serving normal user activity. DoS attacks, which commonly targets the network layer, are also possible at the application layer. These malicious attacks can succeed by starving a system of critical resources, vulnerability exploit, or abuse of functionality. DoS attacks will often attempt to consume all of a web site's available system resources such as: CPU, memory, disk space etc. When any of these critical resources reach full utilization, the web site will normally be inaccessible.
Affected security assets:	• Availability of service. • Availability compromise for consume resources, as: ○ Bandwidth ○ Database connections ○ Disk storage ○ CPU ○ Memory ○ Application specific resources
Exploited vulnerabilities:	• (CWE-400) Uncontrolled Resource Consumption ('Resource Exhaustion') [29]
Related indicators:	IN-9, IN-21, IN-60, IN-61, IN-62, IN-63, IN-64.

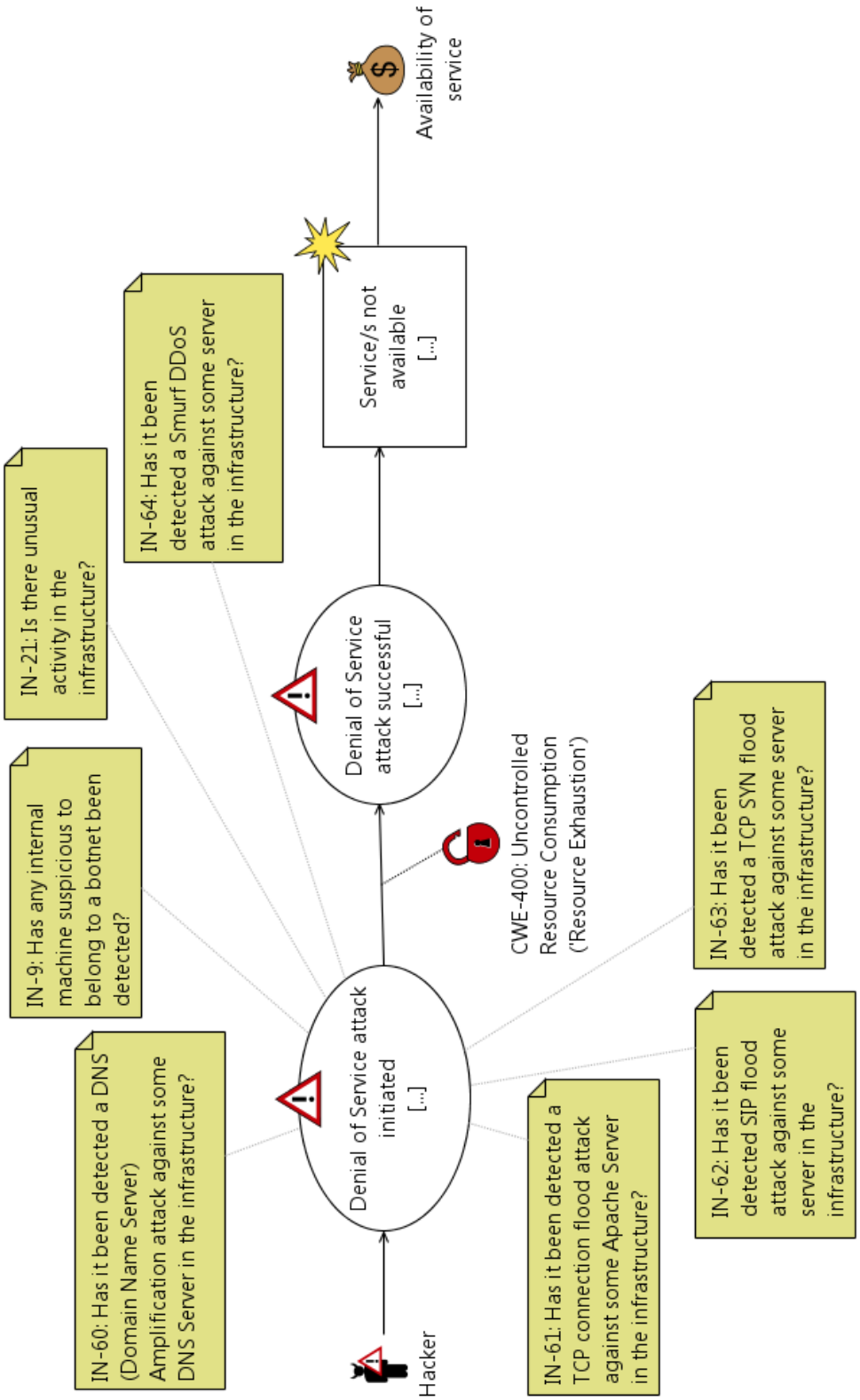

Figure 5: Denial of service

5.1.2 Invalidated redirects and forwards

Id: WRP-2	**Name:** Invalidated Redirects and Forwards
Pattern source:	This risk pattern is based on OWASP Invalidated Redirects and Forwards (OWASP TOP 10 Vulnerabilities 2013) [25].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications.
Description:	An open redirect is an application that takes a parameter and redirects a user to the parameter value without any validation. This is a vulnerability often used in phishing attacks to get users to visit malicious sites without the victim realizing it. Applications frequently redirect users to other pages, or use internal forwards in a similar manner. Sometimes the target page is specified in an invalidated parameter, allowing attackers to choose the destination page. Detecting unchecked redirects is easy. Look for redirects where you can set the full URL. Unchecked forwards are harder, because they target internal pages.
Affected security assets:	• Integrity of system • Confidentiality of user data
Exploited vulnerabilities:	• (CWE-601) URL Redirection to Untrusted Site ('Open Redirect') [30]: For example, Site allows invalidated code triggering redirect or forward (HTTP response codes 300-307).
Related indicators:	IN-32, IN-8, IN-11, IN-23.

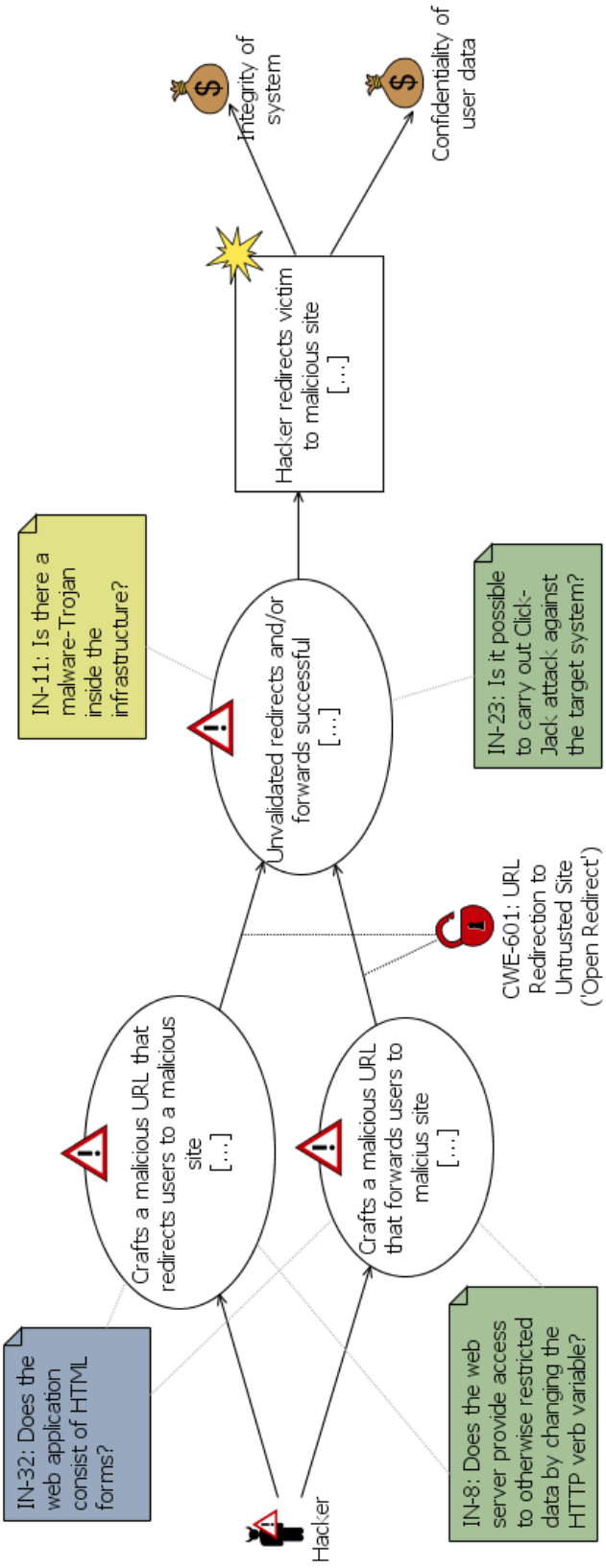

Figure 6: Invalidated redirects and forwards

5.1.3 Bypass login by brute force or DNS login attack

Id: WRP-3	**Name:** Bypass Login
Pattern source:	This risk pattern is based on CAPEC-112 [26].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications.
Description:	Security assets related to information, functionalities of a website, the identity of a user, etc. are protected by a secret value. In this attack, the attacker attempts to gain access to the asset under protection by using trial-and-error to exhaustively explore all the possible secret values to (hopefully) guess the correct value that will unlock the asset. Examples are passwords, encryption keys, database lookup keys etc. [26]. In particular, this pattern explores an attack to bypass authentication mechanisms by brute force or DNS login attack.
Affected security assets:	• Confidentiality of data • Access control • Authorization
Exploited vulnerabilities:	• (CWE-326) Inadequate encryption strength [31] • (CWE-330) Use of insufficiently random values [32] • (CWE-521) Weak password requirements [33]
Related indicators:	IN-20, IN-21.

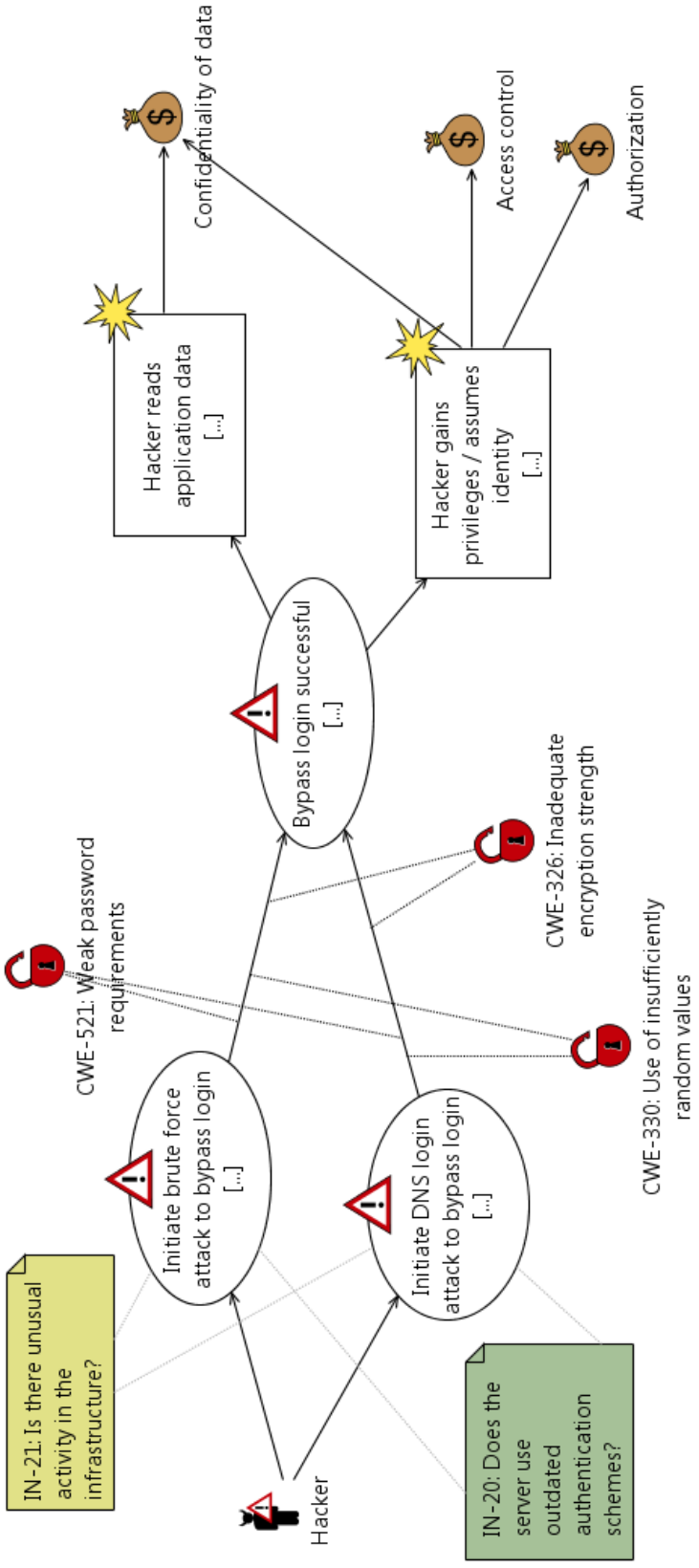

Figure 7: Bypass login by brute-force attack or DNS attack

5.1.4 Compromise security via Trojan-malware

Id: WRP-4	**Name:** Compromise security via Trojan-malware
Pattern source:	This risk pattern is based on CAPEC-542 [27].
Target characteristics:	Computer systems organization: Distributed architectures. Networks: Application layer protocols. Information systems: Web applications.
Description:	According to CAPEC-542: "An adversary develops targeted malware that takes advantage of a known vulnerability in an organizational information technology environment. The malware crafted for these attacks is based specifically on information gathered about the technology environment. Successfully executing the malware enables an adversary to achieve a wide variety of negative technical impacts."
Affected security assets:	• Confidentiality • Integrity • Availability
Exploited vulnerabilities:	• (CWE-507) Trojan Horse [34] ○ Lack of malware detection and protection
Related indicators:	IN-11.

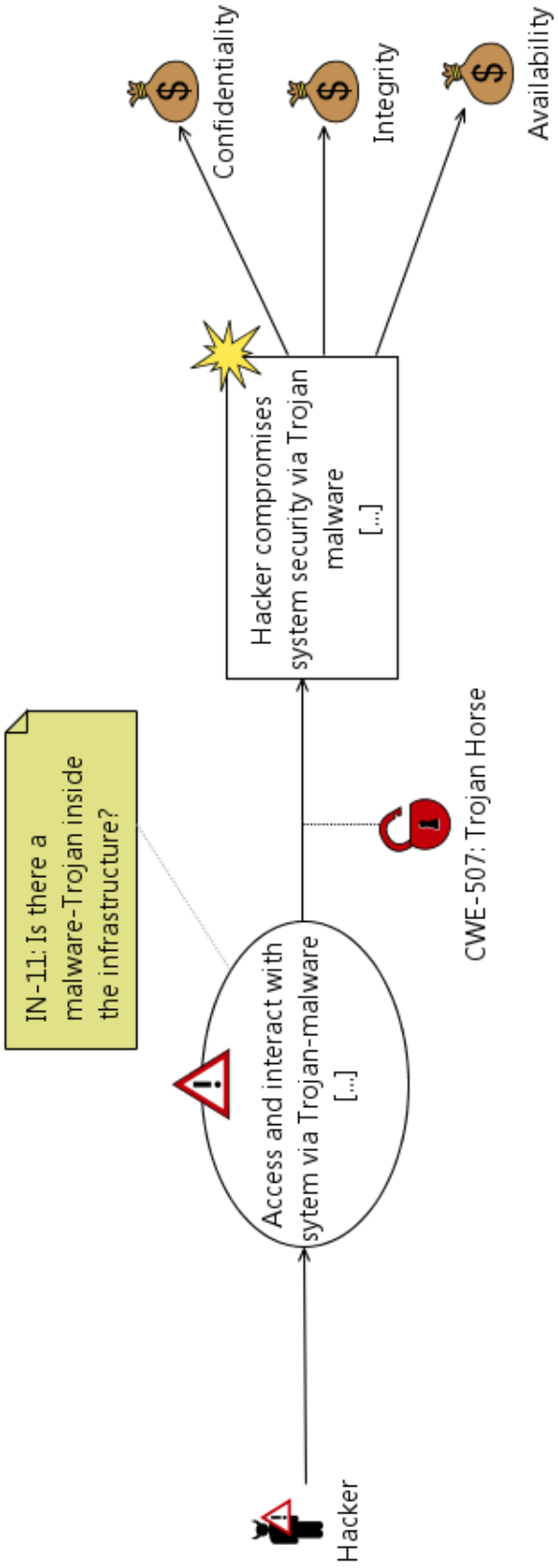

Figure 8: Compromise security via Trojan-malware

5.2 Patterns including application-layer indicators

The patterns described in this section are advanced patterns that are supported by all four kinds of indicators. In particular, the application-layer indicators in these patterns need to be adjusted and implemented based on the specific target-system owned by the client. Because of this, most indicators in the patterns in this section must be developed for each instantiation.

However, wherever possible, WISER provides indicators that may be reused with minimum need for manual adjustment and implementation when instantiated. In the following patterns, indicators that may be reused with minimum need for manual adjustment are shown with a solid frame as illustrated on the left-hand side of Figure 9. Indicators that need to be implemented on case basis are represented with a dashed frame as shown on the right-hand side in Figure 9. As explained in Section 3.2, the indicators have a background colour reflecting that an indicator is based on either business configuration, test results, network-layer monitoring, or application-layer monitoring.

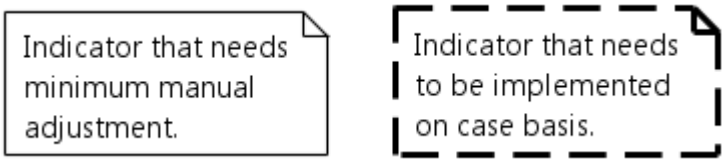

Figure 9: Graphical representation of indicators that need minimum manual adjustment and indicators that need to be implemented on case basis

5.2.1 Client-server protocol manipulation

Id: WRP-5	**Name:** Client-Server Protocol Manipulation
Pattern source:	This risk pattern is based on CAPEC-220 [6].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications.
Description:	An adversary takes advantage of weaknesses in the protocol by which a client and server are communicating to perform unexpected actions. Communication protocols are necessary to transfer messages between client and server applications. Moreover, different protocols may be used for different types of interactions. For example, an authentication protocol might be used to establish the identities of the server and client while a separate messaging protocol might be used to exchange data. If there is a weakness in a protocol used by the client and server, an attacker might take advantage of this to perform various types of attacks. For example, if the attacker is able to manipulate an authentication protocol, the attacker may be able spoof other clients or servers. If the attacker is able to manipulate a messaging protocol, the attacker may be able to read sensitive information or modify message contents. This attack is often made easier by the fact that many clients and servers support multiple protocols to perform similar roles. For example, a server might support several different authentication protocols in order to support a wide range of clients, including legacy clients. Some of the older protocols may have vulnerabilities that allow an attacker to manipulate client-server interactions [6].
Affected security assets:	• Confidentiality of server data in storage or in transit • Integrity of server data in storage or in transit • Availability of server data in storage or in transit
Exploited vulnerabilities:	• (CWE-113) Improper Neutralization of CRLF Sequences in HTTP

	Headers [35] • (CWE-20) Improper input validation [36] • (CWE-302) Authentication bypass by assumed-immutable data (In this pattern: HTTP verbs are used as factors in a security decision) [37] • (CWE-303) Incorrect implementation of authentication algorithm (outdated authentication schemes/mechanisms) [38]
Related indicators:	IN-1, IN-2, IN-3, IN-4, IN-5, IN-6, IN-7, IN-8, IN-10, IN-12, IN-13, IN-14, IN-15, IN-16, IN-17, IN-18, IN-19, IN-20.

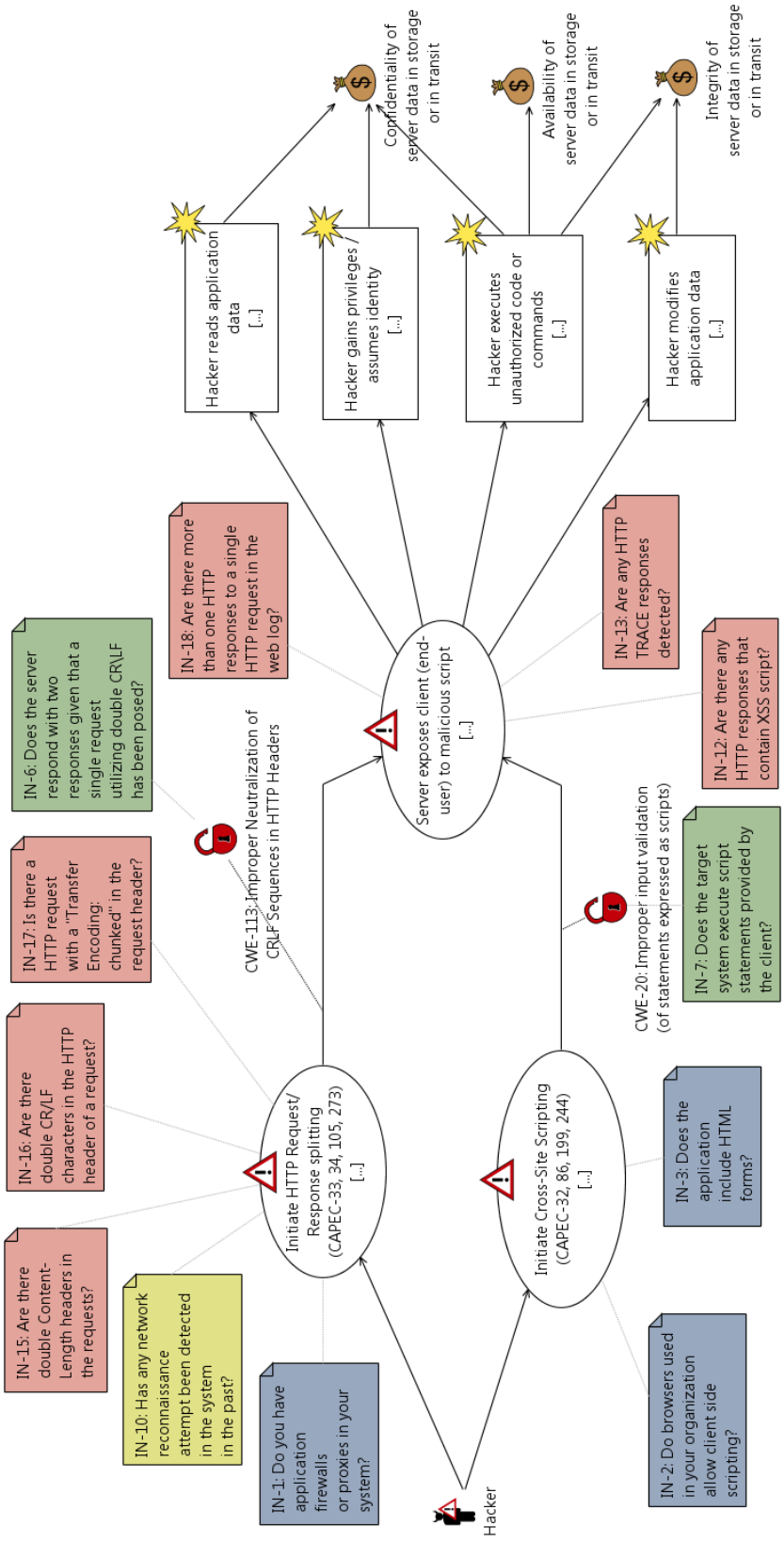

Figure 10: Client-server protocol manipulation (part 1)

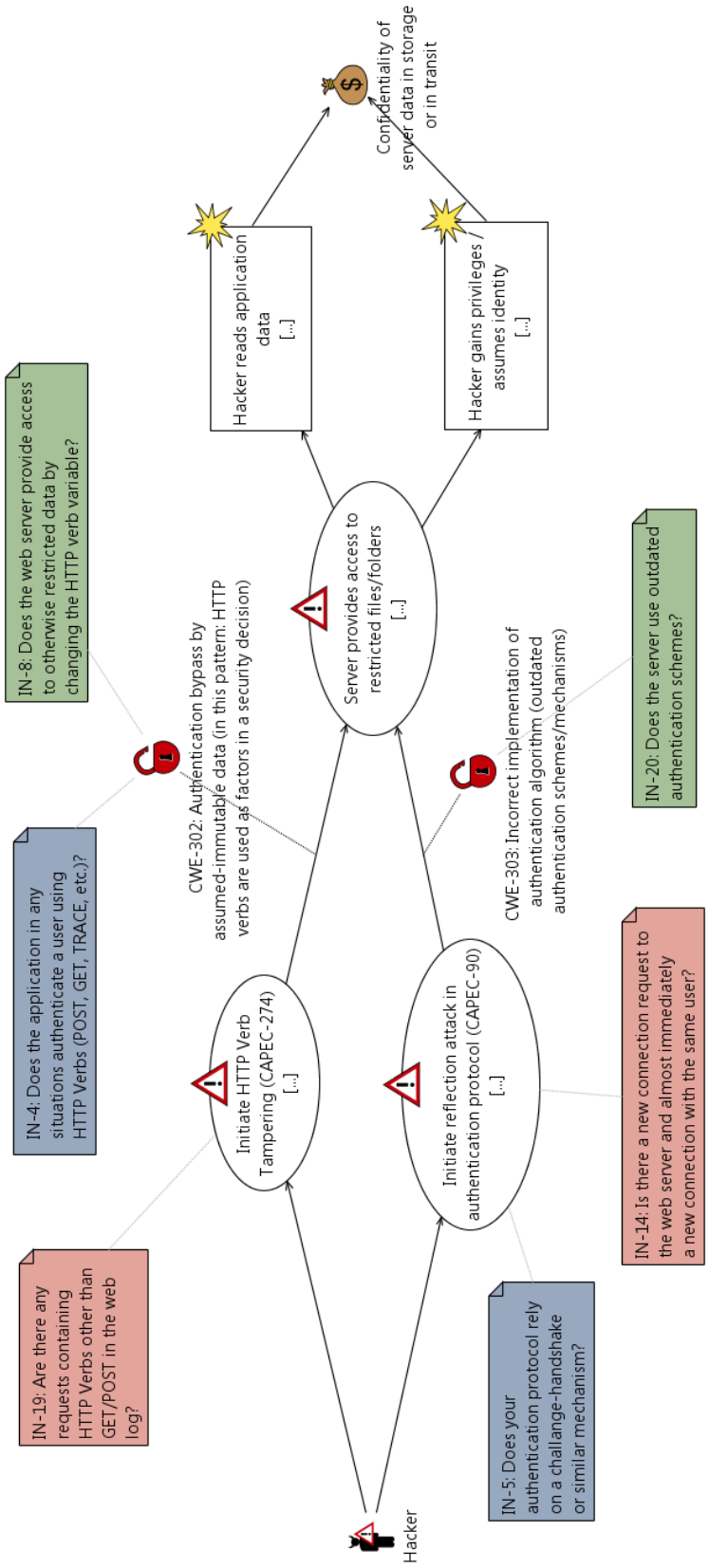

Figure 11: Client-server protocol manipulation (part 2)

5.2.2 Session hijacking

Id: WRP-6	**Name:** Session Fixation
Pattern source:	This risk pattern is based on CAPEC-61 [1], CAPEC-31 [21], and OWASP Session Fixation [11].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications. Session fixation typically targets sessions in web applications: session token in a URL argument, session token in a hidden form field, and session ID in a cookie.
Description:	Session hijacking is an attack with the objective to obtain a valid user-session established between a client and a web server. This is also known as session fixation. The motivation behind this attack is to gain access to a web application with the privileges of a valid user, which in turn allows the attacker to freely act on behalf of the victim user. If the hijacked session belongs to an administrator the attacker may in principle carry out all administrative actions on the underlying web application.
Affected security assets:	• Authorization of features provided by web application • Access Control of web application • Confidentiality of web application data
Exploited vulnerabilities:	• (CWE-361) Improper management of session time and state [39] • (CWE-732) Incorrect permission assignment for critical resource [40] • (CWE-664) Improper control of a session resource through its lifetime [41] • (CWE-311) Missing encryption of sensitive data [42] • (CWE-565) Reliance on Cookies without validation and integrity checking [43]
Related indicators:	IN-30, IN-34, IN-35, IN-41, IN-42, IN-51, IN-52.

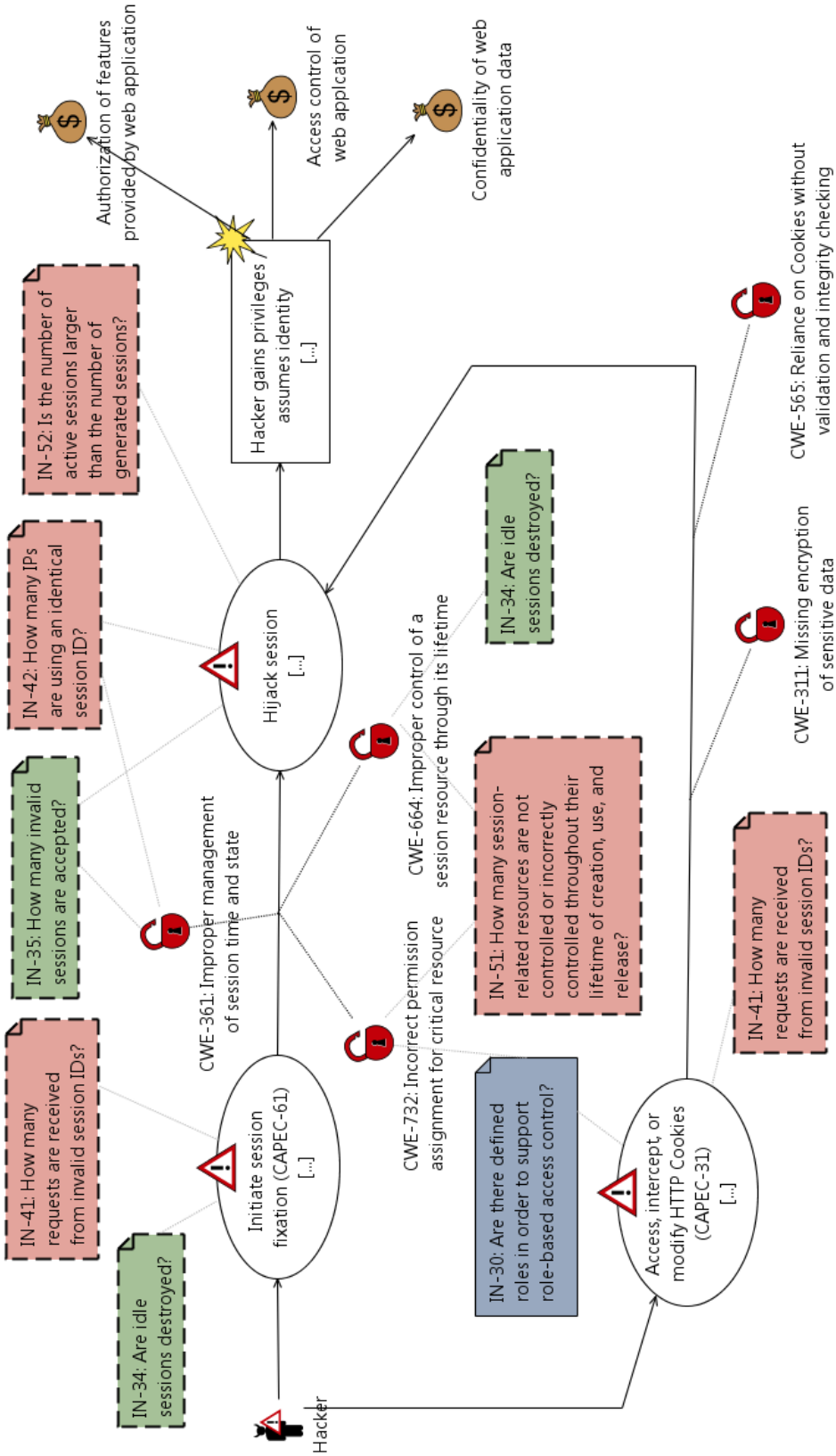

Figure 12: Session hijacking

5.2.3 Cross site request forgery

Id: WRP-7	**Name:** Cross Site Request Forgery
Pattern source:	This risk pattern is based on CAPEC-62 [2] and OWASP Cross-Site Request Forgery [12].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications. Cross-site request forgery typically targets critical functions of a web application. This includes the database of a web application.
Description:	Cross-site request forgery (CSRF) is an attack with the objective to trick a user currently authenticated on a web application to execute an action on behalf of the attacker. A typical scenario is that an attacker forges a request to a web application in terms of a HTML link, sends this link to the victim user, and waits for the victim user to execute the request by clicking on the link. If the victim is someone who uses online banking, a successful CSRF attack could for example force the user to transfer funds. If the victim is an administrative account, CSRF could compromise the entire web application.
Affected security assets:	• Authorization of features provided by web application • Access Control of web application • Confidentiality of web application data • Integrity of web application data
Exploited vulnerabilities:	• (CWE-306) Missing authentication for critical function [44] • (CWE-732) Incorrect permission assignment for critical resource [40] • (CWE-664) Improper control of a session resource through its lifetime [41] • (CWE-20) Improper input validation (of statements expressed as scripts) [36]
Related indicators:	IN-30, IN-31, IN-34, IN-36, IN-43, IN-51, IN-53.

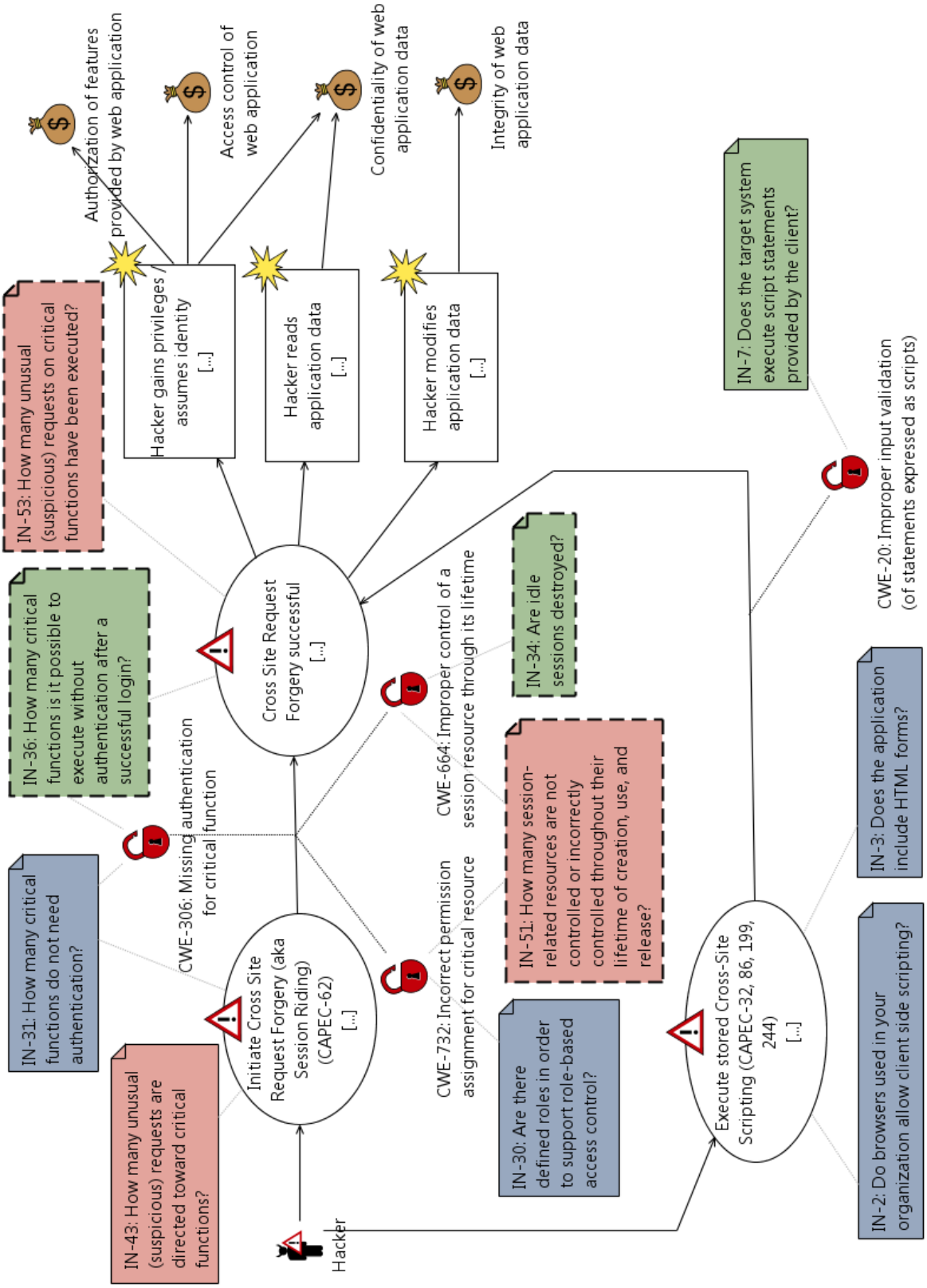

Figure 13: Cross site request forgery

5.2.4 SQL injection

Id: WRP-8	**Name:** SQL Injection
Pattern source:	This risk pattern is based on CAPEC-66 [3] and OWASP SQL Injection [13].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications. SQL injections target the database of a web application.
Description:	SQL injection is an attack with the objective to execute SQL queries on the database of a web application via input fields available on the web application (for example, HTML forms). A successful SQL injection can enable the attacker to read sensitive data from the database, modify data in the database (for example inserting, updating, or deleting data), as well as execute administrative operations on the database such as shutting down the database management system.
Affected security assets:	• Authorization of web-application database • Access Control of web-application database • Availability of data in web-application database • Confidentiality of data in web-application database • Integrity of data in web-application database
Exploited vulnerabilities:	• (CWE-89) Improper neutralization of special elements used in an SQL command [45] • (CWE-74) Improper neutralization of special elements in output used by a downstream component [46] • (CWE-390) Detection of SQL-related error conditions without action [47]
Related indicators:	IN-32, IN-37, IN-38, IN-44, IN-45, IN-54, IN-55, IN-56.

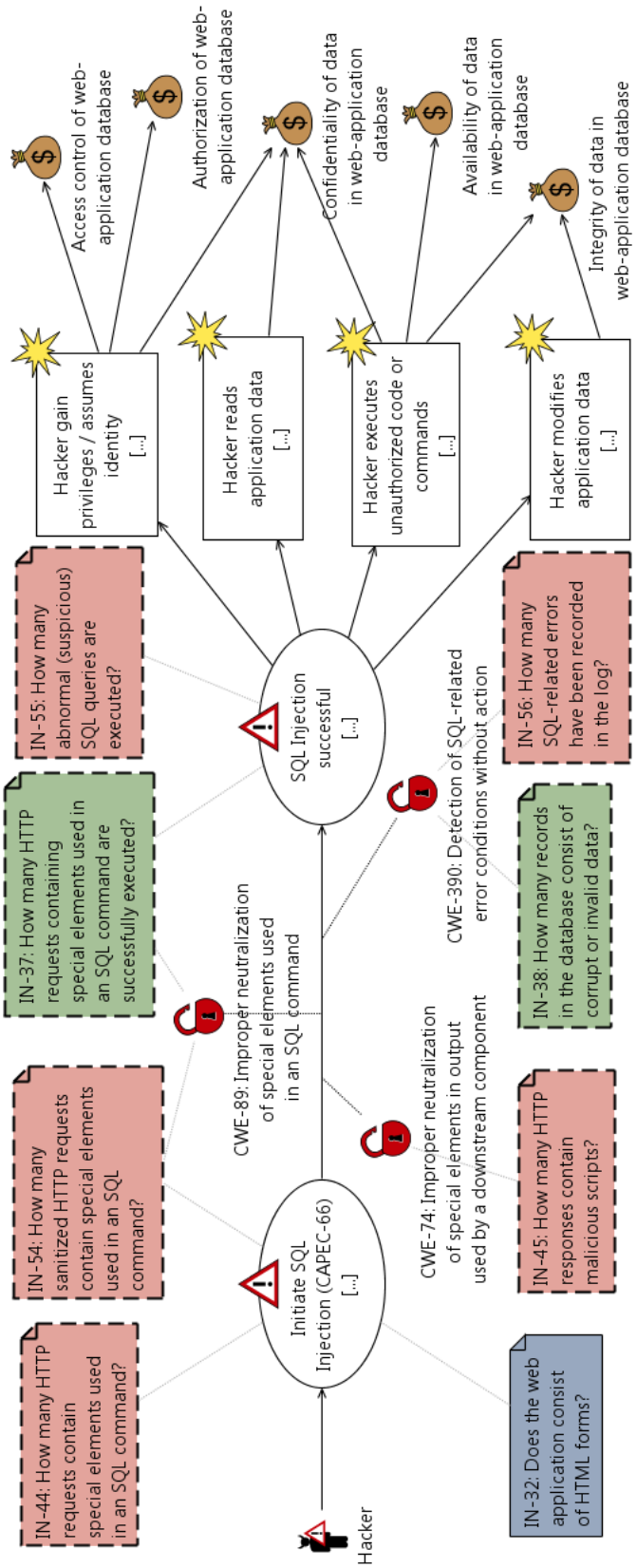

Figure 14: SQL injection

5.2.5 *Buffer overflow*

Id: WRP-9	**Name:** Buffer Overflow
Pattern source:	This risk pattern is based on CAPEC-100 [4] and OWASP Buffer Overflow [14], [15].
Target characteristics:	Computer systems organization: All architectures. Buffer overflow flaws can be present in web servers as well as application server products that serve the static and dynamic portions of a site, or in the web application itself. Buffer overflows can also be found in custom web application code.
Description:	Buffer overflow is an attack with the objective to corrupt data, crash software, or cause the execution of malicious code by writing outside the bounds of a block of allocated memory. In a classic buffer overflow exploit, the attacker sends data to a program, which it stores in an undersized stack buffer. The result is that information on the call stack is overwritten, including the function's return pointer. The data sets the value of the return pointer so that when the function returns, it transfers control to malicious code contained in the attacker's data. There are a variety of buffer overflow types, including Heap buffer overflow and Off-by-one Error.
Affected security assets:	• Authorization • Access Control • Availability • Confidentiality • Integrity
Exploited vulnerabilities:	• (CWE-120) Buffer copy without checking size of input [48] • (CWE-680) Integer overflow to buffer overflow [49] • (CWE-805) Buffer access with incorrect length value [50] • (CWE-119) Improper restriction of operations within the bounds of a memory buffer [51] • (CWE-131) Incorrect calculation of buffer size [52] • (CWE-129) Improper validation of array index [53]
Related indicators:	IN-46, IN-57, IN-58.

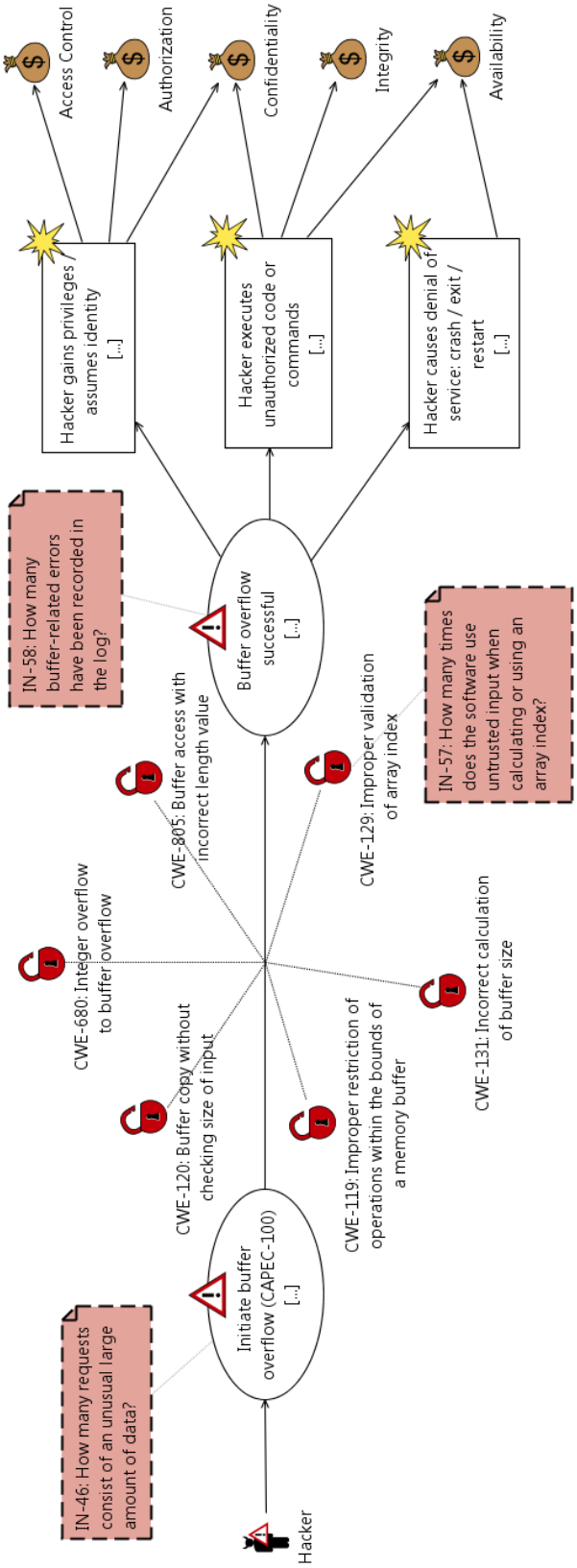

Figure 15: Buffer overflow

5.2.6 Relative path traversal

Id: WRP-10	**Name:** Relative Path Traversal
Pattern source:	This risk pattern is based on CAPEC-139 [5] and OWASP Path Traversal [16].
Target characteristics:	Computer systems organization: Client-server architecture. Networks: Application layer protocols. Information systems: Web applications. Path traversal flaws can be present in web servers and web applications.
Description:	Path traversal is an attack with the objective to access files and directories that are stored outside the web root folder. This is typically carried out by manipulating URL variables that reference files with “dot-dot-slash (../)” sequences and its variations. By using "../" sequences an attacker may move up to root directory, thus navigating through the file system.
Affected security assets:	• Authorization of web-server or web-application data • Availability of web-server or web-application data • Confidentiality of web-server or web-application data • Integrity of web-server or web-application data
Exploited vulnerabilities:	• (CWE-20) Improper input validation of URL pathnames [36] • (CWE-22) Improper limitation of a pathname to a restricted directory [54] • Lack of mechanisms to monitor the velocity of requests (typically spaces apart regularly, e.g. 0.8 seconds between each path-request)
Related indicators:	IN-33, IN-39, IN-40, IN-47, IN-48, IN-49, IN-50.

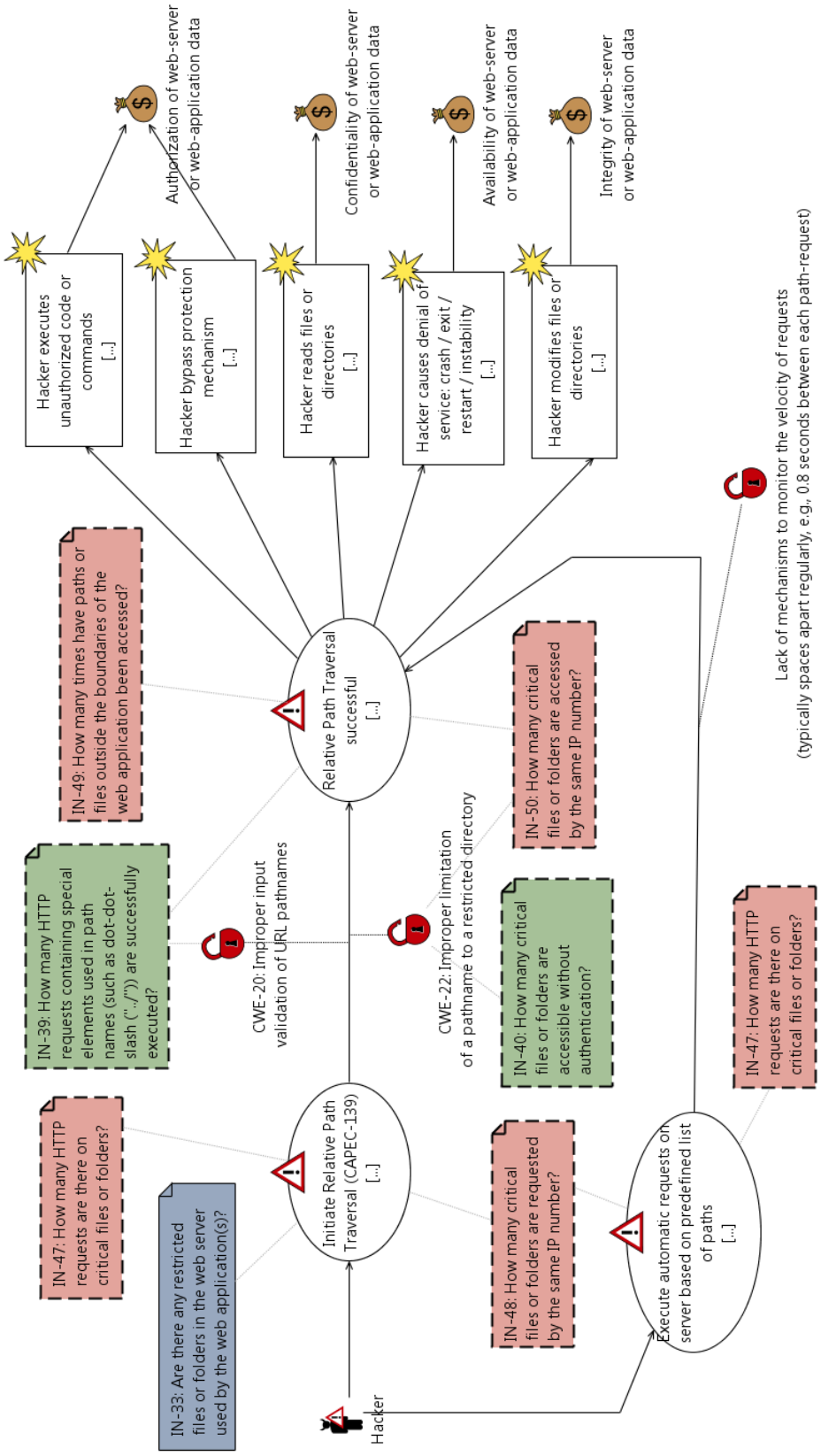

Figure 16: Relative path traversal

6 Indicators

6.1 Business configuration (non-intrusive)

Id:	IN-1	
Question:	Do you have application firewalls or proxies in your system?	
Data type:	Boolean: Yes/No	
Motivation:	HTTP Request/Response Splitting may result from the discrepancies in parsing HTTP requests between HTTP entities such as web caching proxies or application firewalls. Having such entities in the HTTP-parsing chain may expose the system to HTTP Request/Response Splitting attacks [7].	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target.	

Id:	IN-2	
Question:	Do browsers used in your organization allow client side scripting?	
Data type:	Boolean: Yes/No	
Motivation:	Cross-Site Scripting attacks exploit the ability of most browsers to interpret "data", "javascript" or other URI schemes as client-side executable content placeholders. These attacks consist of passing a malicious URI in an anchor tag HREF attribute or any other similar attributes in other HTML tags. The attack is executed when the browser interprets the malicious content, for example, when the victim clicks on a malicious link [8]. Employees in an organization may be exposed to such attacks if they use browsers that allow the execution of scripts, such as javascript.	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target.	

Id:	IN-4
Question:	Does the application in any situations authenticate a user using HTTP Verbs (POST, GET, TRACE, etc.)?
Data type:	Boolean: Yes/No
Motivation:	If a web environment allows administrators to restrict access based on the HTTP Verb it may be vulnerable to HTTP Verb Tampering attacks. Attackers can often provide a different HTTP Verb, or even provide a random string as a verb in order to

	bypass certain restrictions. This allows the attacker to access data that should otherwise be protected [9].	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target (administrators).	

Id:	IN-5	
Question:	Does your authentication protocol rely on a challenge-handshake or similar mechanism?	
Data type:	Boolean: Yes/No	
Motivation:	Reflection attacks are of great concern to authentication protocols that rely on a challenge-handshake or similar mechanism. An attacker can impersonate a legitimate user and can gain illegitimate access to the system by successfully mounting a reflection attack during authentication [10].	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target.	

Id:	IN-30	
Question:	Are there defined roles in order to support role-based access control?	
Data type:	Boolean: Yes/No	
Motivation:	This indicator can be used to check if the target has clearly defined roles. Roles are used to provide the necessary privileges to certain user groups. If the target does not have clearly defined roles then certain user groups may have access to critical resources they should not have access to. The target may thereby be vulnerable to, for example, incorrect permission assignment for critical resource.	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	(X)
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target. • Application-layer monitoring (intrusive): Indicator value may also be obtained by monitoring the number of different roles	

	accessing the target.

Id:	IN-31	
Question:	How many critical functions do not need authentication?	
Data type:	Integer	
Motivation:	This indicator can be used to get an estimation of the number of critical functions in the target, and how many of these do need authentication. If there are a large number of critical functions, but only a few of these need authentication, then this may indicate that the target is vulnerable to, for example, missing authentication for critical functions.	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target.	

Id:	IN-32	
Question:	Does the web application consist of HTML forms?	
Data type:	Boolean: Yes/No	
Motivation:	This indicator can be used to identify whether the target (web application) consists of HTML forms. Most injection attacks are carried out via HTML forms. If the target provides HTML forms to the end-user, then it may be vulnerable to, for example, SQL injections.	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target.	

Id:	IN-33	
Question:	Are there any restricted files or folders in the web server used by the web application(s)?	
Data type:	Boolean: Yes/No	
Motivation:	This indicator can be used to identify whether the target (web application) makes use of files or folders located in a restricted area of the web server on which it is hosted. Attacks such as path traversals are typically carried out on such targets.	
Indicator type:	Business configuration (non-intrusive)	X
	Test results (non-intrusive)	

	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Business configuration (non-intrusive): Indicator value may be obtained by asking the question to experts of the target.	

6.2 Test results (non-intrusive)

Id:	IN-6 (this indicator corresponds to indicator WI-TR-05-CRLF in Deliverable D4.1)	
Question:	Does the server respond with two responses given that a single request utilizing double CR\LF has been posed?	
Data type:	Boolean: Yes/No	
Motivation:	An HTTP Request/Response splitting attack may be carried out successfully if the target system improperly neutralizes CR\LF sequences in HTTP headers. One way to check if the target system may be exposed to HTTP Request/Response splitting attacks is to test for this vulnerability by checking if the server responds with two responses given that a single request utilizing double CR\LF has been posed.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, a manual test: (1) capture the HTTP response using a proxy tool such as OWASP ZAP, (2) check if the response contains parameters expecting value from the client, for example "http://.../main.jsp?interface=advance", (3) insert into the 'interface' parameter the sequence: %0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2035%0d%0a%0d%0a<html>Sorry,%20System%20Down</html> , (4) capture the HTTP response again and check if the response consists of two headers. If the response consists of two headers, then the target system is vulnerable to the vulnerability Improper Neutralization of CRLF Sequences in HTTP Headers.	

Id:	IN-7 (this indicator corresponds to indicator WI-TR-01-XSS in Deliverable D4.1)	
Question:	Does the target system execute malicious script statements provided by the client?	
Data type:	Boolean: Yes/No	
Motivation:	A Cross-Site Scripting attack may be carried out successfully if the target system improperly validates input expressed as scripts (for example javascript).	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	

	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, a manual test. There are many different ways to test for cross-site scripting attacks, but one simple way is to test the so-called reflected cross-site scripting attack: (1) Detect input vectors such as HTTP parameters, POST data, hidden form field variables, etc., using a web proxy tool, (2) Analyse each input vector to detect potential vulnerabilities using specially crafted input data with each input vector (such as <script>alert(123)</script>), (3) For each test input attempted in the previous phase, check whether the input script is executed. If it is executed, then the target system is not validating script statements and is thus vulnerable to improper input validation. • W3af has an XSS plugin capable of detecting reflected XSS vulnerabilities on the target web server.	

Id:	IN-8 (this indicator corresponds to indicator WI-TR-02-ACCESS in Deliverable D4.1)	
Question:	Does the web server provide access to otherwise restricted data by changing the HTTP verb variable?	
Data type:	Boolean: Yes/No	
Motivation:	An HTTP Verb Tampering attack may be carried out successfully if the target system filters access based on HTTP verbs.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, a manual test: (1) capture the HTTP request to a restricted URL you wish to access using a proxy tool such as OWASP ZAP, (2) change the HTTP verb parameter to any other parameter than GET/POST, such as OPTIONS or DELETE, (3) If you gain access to restricted URL then you have bypassed an access restriction. In that case, the target system is vulnerable to authentication bypass by assumed-immutable data. • W3af tool has a plugin htaccessMethods that is capable of detecting vulnerabilities related to misconfiguration of .htaccess files limiting the access to specific verbs [18].	

Id:	IN-20 (this indicator corresponds to indicator WI-TR-03-AUTH in Deliverable D4.1)
Question:	Does the server use outdated authentication schemes?
Data type:	Boolean: Yes/No
Motivation:	Using outdated authentication schemes or mechanisms may expose the target system to authentication attacks such as reflection attacks in authentication protocol

	and HTTP verb tampering.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	X
		Network-layer monitoring (intrusive)	
		Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by for example, carrying out brute force attack tests, such as those supported by the w3af plugins formAuthBrute and basicAuthBrute.		

Id:	IN-23 (this indicator corresponds to indicator WI-TR-04-CLICKJ in Deliverable D4.1)		
Question:	Is it possible to carry out Click-Jack attack against the target system?		
Data type:	Boolean: Yes/No		
Motivation:	Hijacking user actions on web pages are likely to occur if the target system is vulnerable to clickjacking. In a clickjacking attack, the victim is tricked into unknowingly initiating some action in one system while interacting with the user interface from seemingly completely different system [22].		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	X
		Network-layer monitoring (intrusive)	
		Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value can be obtained by carrying out tests using W3af.		

Id:	IN-34		
Question:	Are idle sessions destroyed?		
Data type:	Boolean: Yes/No		
Motivation:	A typical flaw related to session management in web applications is to keep sessions active and "alive" even if the end user is not active for a long time, and in some cases kept alive even if the web browser is terminated. Such flaws are commonly exploited in session fixation and cross site request forgery attacks. This indicator may be used to identify whether the target properly destroys idle sessions.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	X
		Network-layer monitoring (intrusive)	
		Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, a manual test: (1) Sign in to a web application using a web browser. (2) Close the web browser used to sign in. (3) Start the web browser again. (4) Point the web browser to the web page again and check if you are still		

	logged in. These kinds of tests are also possible using free security testing tools such as OWASP ZAP, or an off-the-shelf web vulnerability scanner.

Id:	IN-35	
Question:	How many invalid sessions are accepted?	
Data type:	Integer	
Motivation:	As IN-34, this indicator is also related to improper session management. In particular, this indicator can be used to check if the target (web application) accepts invalid sessions, for example, sessions that should have been expired, or sessions that are guessed correctly by a malicious user.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, a manual test: (1) Sign in to a web application using a web browser. (2) Capture the HTTP request from the web browser to the web application using a proxy tool such as the one in OWASP ZAP. (3) Copy the session data. (4) Proceed to the web application and log out. (5) Point the web browser to the web application again. (6) Do not log in this time, but instead paste the previous session data in the HTTP request using the proxy tool and check if you are able to use the web application as if you have logged in correctly. These kinds of tests are also possible using off-the-shelf web vulnerability scanners.	

Id:	IN-36	
Question:	How many critical functions is it possible to execute without authentication after a successful login?	
Data type:	Integer	
Motivation:	The target (web application) may consist of a number of critical functions. A user is typically allowed to execute all functions in a web application after a successful login. However, in some web applications, such as internet banking, it is necessary to make additional authentications when carrying out critical functions. For example, if a user logs in to his/her bank account and wants to transfer money to someone, then there should be two authentication checkpoints: once when the user logs in to the internet bank account, and once when the user places a money transfer order. If all critical functions of a web application may be executed after one single login, then a successful cross site request forgery is more likely to be successful.	
Indicator type:	Business configuration (non-intrusive)	(X)
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	

Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, manually mapping all critical functions, and then executing the functions to check if a second authentication is required. This may also be executed manually by "recording" the tests manually using a tool such as Selenium [28], and then execute the test again using Selenium. • Business configuration (non-intrusive): Indicator value may also be collected by asking the question to experts of the target web application.

Id:	IN-37	
Question:	How many HTTP requests containing special elements used in an SQL command are successfully executed?	
Data type:	Integer	
Motivation:	SQL injections are carried out via HTML forms, which are sent to the target (web application) via the HTTP request. Thus, such HTTP requests that are successfully executed indicate successful SQL injections.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, automatically carrying out *N* number of SQL injection tests, and recording the number of successful SQL injections.	

Id:	IN-38	
Question:	How many records in the database consist of corrupt or invalid data?	
Data type:	Integer	
Motivation:	One of the goals of SQL injections is to corrupt existing data in a database, or to inject invalid (malicious) data. This indicator can be used to check the number of database-records that contain corrupt or invalid data, which may be an indication of successful SQL injections.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by, for example, carrying out tests directly on the database in terms of SQL transactions to check for records containing corrupt or invalid data. The tests may be automated by creating scripts for this purpose.	

Id:	IN-39	
Question:	How many HTTP requests containing special elements used in path names (such as dot-dot-slash ("../")) are successfully executed?	
Data type:	Integer	
Motivation:	Path traversal attacks are typically carried out by including special elements in path names (such as dot-dot-slash "../") inside the URL request sent to a web application. This indicator can be used to identify whether URL requests containing path-name elements are parsed by the web application. If such URL requests are accepted and parsed by the web application then it may be vulnerable to improper input validation of URL names, and thus vulnerable to path traversal attacks.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by manually exploring several combinations of "../" on a URL and check if some files are accessed. This can also be tested automatically using web vulnerability scanners.	

Id:	IN-40	
Question:	How many critical files or folders are accessible without authentication?	
Data type:	Integer	
Motivation:	This indicator can be used to check if it is possible to access critical files or folders outside the boundaries of the web application without authentication. That is, files or folders in the web server that should not be accessible by an end-user. If it is possible, then this is an indication that the target (web application) may be vulnerable to improper limitation of path names to a restricted directory, and thus vulnerable to path traversal attacks.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	X
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Test results (non-intrusive): Indicator value may be obtained by manually attempting to access restricted files. This test can also be carried out automatically using a web vulnerability scanner. The scanners typically "crawl" the web application, and give a list of all accessible files and folders.	

6.3 Network-layer monitoring (intrusive)

Id:	IN-9 (this indicator corresponds to indicator WI-NL-02-Botnet in Deliverable D4.1)	
Question:	Has any internal machine suspicious to belong to a botnet been detected?	
Data type:	Boolean: Yes/No	
Motivation:	It has been detected IPs belonging to client´s infrastructure suspicious of belonging to a botnet	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	X
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value may be obtained by means of DNS Traffic Sensor.	

Id:	IN-10 (this indicator corresponds to indicator WI-NL-01-NetworkScan generated by the Monitoring Engine in Deliverable D4.1)	
Question:	Has any network reconnaissance attempt been detected in the system in the past?	
Data type:	Boolean: Yes/No	
Motivation:	Prior to an HTTP Request/Response splitting attack, an attacker may explore the network path in order to identify various entities in the HTTP-parsing chain. The attacker may identify entities such as proxies and firewalls in order to prepare an HTTP Request/Response splitting attack.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	X
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value may be obtained by analysing web/network log, or by placing honeypot inside the client's architecture.	

Id:	IN-11 (this indicator corresponds to indicator WI-NL-03-Malware in Deliverable D4.1)	
Question:	Is there a malware-Trojan inside the infrastructure?	
Data type:	Boolean: Yes/No	
Motivation:	It has been detected malware signatures in the client's infrastructure network traffic.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	X
	Application-layer monitoring (intrusive)	
Means of obtaining	• Network-layer monitoring (intrusive): Indicator value can be	

indicator value:	detected by the Monitoring Engine.

Id:	IN-21 (this indicator corresponds to indicator WI-NL-04-UnusualActivity in Deliverable D4.1)		
Question:	Is there unusual activity in the infrastructure?		
Data type:	Boolean: Yes/No		
Motivation:	It has been detected unusual activity against IPs in client's infrastructure.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	
		Network-layer monitoring (intrusive)	X
		Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value can be obtained via the detection of Brute Force attacks by means of the Monitoring Engine or the detection of the DNS Login attack by means of the DNS Traffic sensor.		

Id:	IN-60 (this indicator is included in the indicator WI-NL-05-DenialOfService in Deliverable D4.1)		
Question:	Has it been detected a DNS (Domain Name Server) Amplification attack against some DNS Server in the infrastructure?		
Data type:	Boolean: Yes/No		
Motivation:	It has been detected a DNS Amplification attack, which is a reflection-based distributed denial of service attempt, against some DNS Server in client's infrastructure.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	
		Network-layer monitoring (intrusive)	X
		Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value can be obtained via DNS Traffic Sensor.		

Id:	IN-61 (this indicator is included in the indicator WI-NL-05-DenialOfService in Deliverable D4.1)		
Question:	Has it been detected a TCP connection flood attack against some Apache Server in the infrastructure?		
Data type:	Boolean: Yes/No		
Motivation:	It has been detected a denial of service attempt using HTTP GET flooding against IPs in client's infrastructure.		
Indicator type:		Business configuration (non-intrusive)	

	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	X
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value can be obtained via Snort Sensor and alarm in the monitoring engine.	

Id:	IN-62 (this indicator is included in the indicator WI-NL-05-DenialOfService in Deliverable D4.1)	
Question:	Has it been detected SIP flood attack against some server in the infrastructure?	
Data type:	Boolean: Yes/No	
Motivation:	It has been detected a denial of service attempt using SIP flooding against IPs in client's infrastructure.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	X
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value can be obtained via Snort Sensor and alarm in the monitoring engine.	

Id:	IN-63 (this indicator is included in the indicator WI-NL-05-DenialOfService in Deliverable D4.1)	
Question:	Has it been detected a TCP SYN flood attack against some server in the infrastructure?	
Data type:	Boolean: Yes/No	
Motivation:	It has been detected a potential Tsunami SYN flood denial of service attempt against IPs in client's infrastructure.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	X
	Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value can be obtained via Snort Sensor and alarm in the monitoring engine.	

Id:	IN-64 (this indicator is included in the indicator WI-NL-05-DenialOfService in

	Deliverable D4.1)		
Question:	Has it been detected a Smurf DDoS attack against some server in the infrastructure?		
Data type:	Boolean: Yes/No		
Motivation:	It has been detected a distributed denial of service attempt in which large numbers of ICMP packets with the intended victim's spoofed source IP in client's infrastructure are broadcast using an IP broadcast address.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	
		Network-layer monitoring (intrusive)	X
		Application-layer monitoring (intrusive)	
Means of obtaining indicator value:	• Network-layer monitoring (intrusive): Indicator value can be obtained via Snort Sensor and alarm in the monitoring engine.		

6.4 Application-layer monitoring (intrusive)

Id:	IN-12 (this indicator corresponds to indicator WI-AL-04-XSS response detected by Snort in Deliverable D4.1)		
Question:	Are there any HTTP responses that contain XSS script?		
Data type:	Boolean: Yes/No		
Motivation:	One way to detect whether the target system is vulnerable to XSS attacks is to monitor its responses to the client. If the responses contain XSS scripts then this indicates that the target system is vulnerable to XSS attacks and may therefore expose the client to malicious scripts.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	
		Network-layer monitoring (intrusive)	
		Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained using IDS sensors to check whether responses contain XSS in HTTP headers.		

Id:	IN-13 (this indicator corresponds to indicator WI-AL-05-HTTPTRACEResponse generated by OSSEC in Deliverable D4.1)
Question:	Are any HTTP TRACE responses detected?
Data type:	Boolean: Yes/No
Motivation:	An attacker may trick a client to carry out an HTTP TRACE requests to the target system. This is referred to as Cross-Site Tracing (XST). This kind of attack enables an attacker to steal the victim's session cookie and possible other authentication credentials transmitted in the header of the HTTP request when the victim's browser communicates with target system. HTTP TRACE requests are not harmful, but may be an indication of malicious activity (possible XST attacks).

Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained using IDS sensors to check whether there are any HTTP TRACE responses.	

Id:	IN-14 (this indicator corresponds to indicator WI-AL-06-Reconnections generated by Honeypot in Deliverable D4.1)	
Question:	Is there a new connection request to the web server and almost immediately a new connection from the same user?	
Data type:	Boolean: Yes/No	
Motivation:	In a Reflection attack in authentication protocols, an attacker needs to open two subsequent connections to the target server. Such an activity should be considered suspicious and may indicate a potential reflection attack in the authentication protocol.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by analysing web/network log.	

Id:	IN-15 (this indicator corresponds to indicator WI-AL-01-DuplicatedHTTPHeader detected by SNORT in Deliverable D4.1)	
Question:	Are there double Content-Length headers in the requests?	
Data type:	Boolean: Yes/No	
Motivation:	An attacker may post a malicious HTTP Request utilizing double Content-Length headers to cause request splitting.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring web log.	

Id:	IN-16 (this indicator corresponds to indicator WI-AL-07-DoubleCRLF detected by

	OSSEC in Deliverable D4.1)		
Question:	Are there double CR/LF characters in the HTTP header of a request?		
Data type:	Boolean: Yes/No		
Motivation:	Post a malicious HTTP Request utilizing double CR/LF characters in HTTP header to cause request splitting.		
Indicator type:	Business configuration (non-intrusive)		
	Test results (non-intrusive)		
	Network-layer monitoring (intrusive)		
	Application-layer monitoring (intrusive)	X	
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring web log.		

Id:	IN-17 (this indicator corresponds to indicator WI-AL-08-ChunkedEncoding detected by OSSEC in Deliverable D4.1)		
Question:	Is there a HTTP request with a "Transfer Encoding: chunked" in the request header?		
Data type:	Boolean: Yes/No		
Motivation:	An attacker may post a malicious HTTP Request utilizing "Transfer Encoding: chunked" in the request header to cause request splitting.		
Indicator type:	Business configuration (non-intrusive)		
	Test results (non-intrusive)		
	Network-layer monitoring (intrusive)		
	Application-layer monitoring (intrusive)	X	
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring web log.		

Id:	IN-18 (this indicator corresponds to indicator WI-AL-09-MultipleHTTPResponses detected by OSSEC in Deliverable D4.1)		
Question:	Are there more than one HTTP responses to a single HTTP request in the web log?		
Data type:	Boolean: Yes/No		
Motivation:	A strong indication of a successful HTTP Response/Request splitting is if there are more than one HTTP responses to a single HTTP request. This indicates that the target system is vulnerable to HTTP Request/Response splitting and may therefore expose the client to malicious scripts.		
Indicator type:	Business configuration (non-intrusive)		
	Test results (non-intrusive)		
	Network-layer monitoring (intrusive)		
	Application-layer monitoring (intrusive)	X	
Means of obtaining	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring web log.		

indicator value:	

Id:	IN-19 (this indicator corresponds to indicator WI-AL-02-UnknownHTTPMethod detected by SNORT in Deliverable D4.1)		
Question:	Are there any requests containing HTTP Verbs other than GET/POST in the web log?		
Data type:	Boolean: Yes/No		
Motivation:	HTTP Verbs other than GET/POST are not harmful, but they are not commonly used. If HTTP Verbs other than GET/POST are discovered it may be an indication of HTTP Verb Tampering attacks – especially if the target system does not use HTTP Verbs other than GET/POST.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	
		Network-layer monitoring (intrusive)	
		Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring web log.		

Id:	IN-41		
Question:	How many requests are received from invalid session IDs?		
Data type:	Integer		
Motivation:	Session fixation attacks are often carried out by manually or automatically guessing correct session IDs. A web application generates session IDs for each user. If there are a large number of requests from invalid session IDs, that is, session IDs not recognized by the web application, then this may indicate attempts of session fixation attacks.		
Indicator type:		Business configuration (non-intrusive)	
		Test results (non-intrusive)	
		Network-layer monitoring (intrusive)	
		Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring incoming requests at network layer and filter out invalid session IDs.		

Id:	IN-42
Question:	How many IPs are using an identical session ID?
Data type:	Integer
Motivation:	Different IPs that makes use of an identical session ID indicates that the web application may be vulnerable to improper session management. Best practice shows that a web application must assign a unique session ID per host. That is, the web application should also take the IP into consideration when assigning a session ID. In

	principle, if the same session ID is used by more than one IP, then that may indicate a successful session fixation attack.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	(X)
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring the number of IPs making use of the same session ID. • Test result (non-intrusive): It is also possible to test this manually by logging in to a web application from different IPs using the same session ID. This test is carried out by two users located on different IPs: (1) User A logs in to the web application and captures the session ID using a proxy tool, such as the one available in OWASP ZAP. (2) User A sends the complete session ID to user B who accesses the web application from a different IP. (3) User B accesses the web application and makes use of the session ID retrieved by User A. This is done by copy-pasting the session ID in the HTTP request using a proxy tool. (4) If successfully authenticated, then User B has managed to log in as User A using the session ID of User A.	

Id:	IN-43 (this indicator corresponds to indicator WI-AL-03-UnusualWebActivity in Deliverable D4.1)	
Question:	How many unusual (suspicious) requests are directed toward critical functions?	
Data type:	Integer	
Motivation:	One of the main objectives of cross site request forgeries is to force a legitimate user of a web application to execute a critical function on behalf of the attacker. A large number of requests on critical functions may indicate an attempt of cross site request forgery.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring the number of requests on critical functions.	

Id:	IN-44
Question:	How many HTTP requests contain special elements used in an SQL command?
Data type:	Integer
Motivation:	Special elements used in SQL commands have no place in HTTP requests. The detection of such HTTP requests indicates an attempt of SQL injection.

Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring the HTTP requests and checking if they contain special elements used in SQL commands.	

Id:	IN-45	
Question:	How many HTTP responses contain malicious scripts?	
Data type:	Integer	
Motivation:	What is typically overlooked in the context of SQL injection (and other injection types) is the sensitization of the HTTP responses a web application generates which is sent to the end-user. SQL commands or other scripts that have been successfully injected (stored) in a database may in turn generate various alterations in the HTTP response. This can cause devastating consequences because every legitimate user of the web application may be exposed to the attack.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may also be monitored at the application layer by scanning the generated HTTP response before it is sent to the end-user. Indicator value may also be obtained by monitoring the HTTP responses sent to the end-user.	

Id:	IN-46	
Question:	How many requests consist of an unusual large amount of data?	
Data type:	Integer	
Motivation:	A request to a web application or a database may be intercepted and injected with a large amount of data for the purpose of causing buffer overflows at the target.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring incoming requests and detecting an unusual large amount of data in the request.	

Id:	IN-47		
Question:	How many HTTP requests are there on critical files or folders?		
Data type:	Integer		
Motivation:	A large number of HTTP requests on critical files or folders may indicate an attempt of path traversal attack.		
Indicator type:	Business configuration (non-intrusive)		
	Test results (non-intrusive)		
	Network-layer monitoring (intrusive)		
	Application-layer monitoring (intrusive)	X	
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring the number of HTTP requests directed toward critical files and folders.		

Id:	IN-48		
Question:	How many critical files or folders are requested by the same IP number?		
Data type:	Integer		
Motivation:	While IN-47 explores the number of HTTP requests directed toward critical files and folders in general, IN-48 explores the number of critical files or folders requested by the *same IP number*. As IN-47, IN-48 may also indicate a path traversal attack.		
Indicator type:	Business configuration (non-intrusive)		
	Test results (non-intrusive)		
	Network-layer monitoring (intrusive)		
	Application-layer monitoring (intrusive)	X	
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring the number of HTTP requests directed toward critical files and folders by the same IP number.		

Id:	IN-49		
Question:	How many times have paths or files outside the boundaries of the web application been accessed?		
Data type:	Integer		
Motivation:	While IN-47 and IN-48 explores the number of *HTTP requests* on critical files and folders, IN-49 explores the number of critical files and folders *that have been successfully accessed*. This indicator can help identify the likelihood of the occurrence of successful path traversals.		
Indicator type:	Business configuration (non-intrusive)		
	Test results (non-intrusive)		
	Network-layer monitoring (intrusive)		
	Application-layer monitoring (intrusive)	X	

Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring the log of network activity.

Id:	IN-50	
Question:	How many critical files or folders are accessed by the same IP number?	
Data type:	Integer	
Motivation:	This indicator explores the number of critical files and folders successfully accessed by the same IP number. This may be an indication of successful path traversals carried out by a particular IP number. This indicator can help identify the likelihood of successful path traversals carried out by certain IP numbers (to map possible malicious users).	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by monitoring the log of network activity.	

Id:	IN-51	
Question:	How many session-related resources are not controlled or incorrectly controlled throughout their lifetime of creation, use, and release?	
Data type:	Integer	
Motivation:	This indicator can be used to check if session-related resources are correctly controlled throughout their lifetime of creation, use, and release. An Incorrect control of session-related variables may indicate that the target (web application) is vulnerable to incorrect permission assignment for critical resources, and improper session management.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions as part of a session-creation function that check whether session-related resources are correctly controlled during creation, use, and release. For example, whether a session ID is correctly terminated.	

Id:	IN-52
Question:	Is the number of active sessions larger than the number of generated sessions?
Data type:	Boolean: Yes/No

Motivation:	This indicator may be used to check if there are any active sessions that are not supposed to exist in the first place. If a web application keeps a record of sessions it has generated as well as active sessions, then it should be easy to identify sessions that are active but not generated by the web application, which in turn is an indication of a successful session fixation attack.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions that checks whether an active session is in fact a session generated/validated by the web application.	

Id:	IN-53	
Question:	How many unusual (suspicious) requests on critical functions have been executed?	
Data type:	Integer	
Motivation:	While IN-43 explores the number of *HTTP requests* attempting to carry out a cross site request forgery, this indicator explores the number of unusual (suspicious) requests on critical functions that have actually been executed. This may indicate the likelihood of a successful cross site request forgery attack. By unusual/suspicious requests we mean, for example in the context of internet banking, an unusual money transfer request carried out by a user that normally do not carry out such requests.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions that analyses user behaviour and checks for unusual user behaviour based on user history.	

Id:	IN-54	
Question:	How many sanitized HTTP requests contain special elements used in an SQL command?	
Data type:	Integer	
Motivation:	While IN-44 explores the number of HTTP requests containing special elements used in an SQL command, this indicator explored the number of HTTP requests that still contain special elements used in an SQL command after a possible sanitation. This double checking is necessary in some cases where bypassing sanitations may be possible. For example, in the case for PHP, if SQL command parameterization is not correctly implemented it may be possible to carry out SQL injections.	
Indicator type:	Business configuration (non-intrusive)	

	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions that analyses a SQL query after it has been sanitized. It may be regarded as a test of the sensitization function.	

Id:	IN-55	
Question:	How many abnormal (suspicious) SQL queries are executed?	
Data type:	Integer	
Motivation:	This indicator explores the number of abnormal (suspicious) SQL queries successfully executed. This, in turn, may indicate successfully executed SQL injections. Abnormal/suspicious SQL queries are for example SQL queries that have deleted something in the database (data in databases are often kept inactive instead of actually deleting the data).	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions that identify SQL queries not normally executed.	

Id:	IN-56	
Question:	How many SQL-related errors have been recorded in the log?	
Data type:	Integer	
Motivation:	A high number of SQL-related errors may be an indication of SQL injections.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions that analyse the database transaction log.	

Id:	IN-57
Question:	How many times does the software use untrusted input when calculating or using an array index?
Data type:	Integer

Motivation:	Using untrusted input for calculating an array index indicates that the target is vulnerable to improper validation of array index. This in turn may cause buffer overflows.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions that analyse the number of times untrusted input is used for calculating an array index.	

Id:	IN-58	
Question:	How many buffer-related errors have been recorded in the log?	
Data type:	Integer	
Motivation:	A large number of buffer related errors in the log may indicate successfully executed buffer overflow attacks.	
Indicator type:	Business configuration (non-intrusive)	
	Test results (non-intrusive)	
	Network-layer monitoring (intrusive)	
	Application-layer monitoring (intrusive)	X
Means of obtaining indicator value:	• Application-layer monitoring (intrusive): Indicator value may be obtained by implementing functions that analyse the system error log.	

7 Conclusions

In this report, we have presented 10 cyber risk patterns currently provided by the WISER framework. We started by explaining the approach used to identify and describe the risk patterns. The approach used consisted of three main steps: (1) identify cyber-risk pattern for WISER based on well-known and common attack scenarios, (2) describe the pattern and create corresponding risk model, and (3) identify and describe indicators for the pattern. These steps were carried out iteratively to identify and document the following ten important risk patterns that capture common cyber-risks with high impact.

- Denial of service attack
- Invalidated redirects and forwards
- Bypass login by brute force or DNS login attack
- Compromise security via Trojan-malware
- Client-server protocol manipulation
- Session hijacking
- Cross site request forgery
- SQL injection

- Buffer overflow
- Relative path traversal

These patterns are described textually and graphically. The textual representation provides a systematic description of the risk pattern (using tables), while the graphical description illustrates the risk pattern in terms of CORAS risk models [17]. The risk models also represent indicators used to collect necessary data to support real-time assessment of risks captured by the pattern. We also provided a detailed description of all indicators used in the above risk patterns. We distinguish between four kinds of indicators:

- *Business configuration indicators* are obtained manually through single/multiple-choice questions asked to the user when configuring WISER.
- *Vulnerability test result indicators* are obtained through non-intrusive vulnerability scans initiated by the user.
- *Network monitoring indicators* are obtained from network-layer sensors deployed in the running target infrastructure.
- *Application monitoring indicators* are obtained from application-layer sensors deployed in the running target infrastructure.

We have also offered guidelines clients may use to instantiate the above patterns provided by the WISER framework.

Besides capturing common cyber-risks with high impact, the risk patterns are used as basis to schematically define algorithms used by the Risk Assessment Engine in WISER to estimate the likelihood of risks.

The patterns, as presented in this document, provide the foundation to further develop corresponding assessment algorithms (sometimes referred to as modelling rules in earlier deliverables) in the context of the Full Scale Pilots in the upcoming months. The assessment algorithms will be developed according to the guidelines documented in D3.2.

8 References

[1] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-61: Session Fixation. https://capec.mitre.org/data/definitions/61.html Accessed: 24.09.2015.

[2] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-62: Cross Site Request Forgery (aka Session Riding). https://capec.mitre.org/data/definitions/62.html Accessed: 24.09.2015.

[3] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-66: SQL Injection. https://capec.mitre.org/data/definitions/66.html Accessed: 24.09.2015.

[4] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-100: Overflow Buffers. https://capec.mitre.org/data/definitions/100.html Accessed: 24.09.2015.

[5] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-139: Relative Path Traversal. https://capec.mitre.org/data/definitions/139.html Accessed: 24.09.2015.

[6] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-220: Client-Server Protocol Manipulation. https://capec.mitre.org/data/definitions/220.html Accessed: 26.04.2016.

[7] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-33: HTTP Request Smuggling. https://capec.mitre.org/data/definitions/33.html Accessed: 29.04.2016.

[8] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-244: Cross-Site Scripting via Encoded URI Schemes. https://capec.mitre.org/data/definitions/244.html Accessed: 29.04.2016.

[9] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-274: HTTP Verb Tampering. https://capec.mitre.org/data/definitions/274.html Accessed: 29.04.2016.

[10] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-90: Reflection Attack in Authentication Protocol. https://capec.mitre.org/data/definitions/90.html Accessed: 29.04.2016.

[11] The Open Web Application Security Project (OWASP). Session fixation. https://www.owasp.org/index.php/Session_fixation Accessed: 24.09.2015.

[12] The Open Web Application Security Project (OWASP). Cross-Site Request Forgery (CSRF). https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF) Accessed: 24.09.2015.

[13] The Open Web Application Security Project (OWASP). SQL Injection. https://www.owasp.org/index.php/SQL_Injection Accessed: 24.09.2015.

[14] The Open Web Application Security Project (OWASP). Buffer Overflow. https://www.owasp.org/index.php/Buffer_Overflow Accessed: 24.09.2015.

[15] The Open Web Application Security Project (OWASP). Buffer Overflows. https://www.owasp.org/index.php/Buffer_Overflows Accessed: 24.09.2015.

[16] The Open Web Application Security Project (OWASP). Path Traversal. https://www.owasp.org/index.php/Path_Traversal Accessed: 24.09.2015.

[17] M. Lund, B. Solhaug, K. Stølen. Model-Driven Risk Analysis: The CORAS Approach. Springer. 2011.

[18] W3af – Web Application Attack and Audit Framework. W3af plugins. http://w3af.org/ Accessed: 29.04.2016.

[19] Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org/about/index.html Accessed: 09.05.2016.

[20] The Open Web Application Security Project (OWASP). OWASP Top Ten Project. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project Accessed: 09.05.2016.

[21] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-31: Accessing/Intercepting/Modifying HTTP Cookies. https://capec.mitre.org/data/definitions/31.html Accessed: 12.05.2016.

[22] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-103: Clickjacking. https://capec.mitre.org/data/definitions/103.html Accessed: 18.05.2016.

[23] The Open Web Application Security Project (OWASP). Denial of Service. https://www.owasp.org/index.php/Denial_of_Service Accessed: 18.05.2016.

[24] The Open Web Application Security Project (OWASP). Application Denial of Service. https://www.owasp.org/index.php/Application_Denial_of_Service Accessed: 18.05.2016.

[25] The Open Web Application Security Project (OWASP). Invalidated Redirects and Forwards (OWASP Top 10 2013). https://www.owasp.org/index.php/Top_10_2013-A10-Unvalidated_Redirects_and_Forwards Accessed: 18.05.2016.

[26] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-112: Brute Force. https://capec.mitre.org/data/definitions/112.html Accessed 19.05.2016.

[27] Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC-542: Targeted Malware. https://capec.mitre.org/data/definitions/542.html Accessed: 19.05.2016.

[28] Selenium HQ – Browser Automation. http://www.seleniumhq.org/ Accessed 27.05.2016.

[29] Common Weakness Enumeration (CWE). CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion'). https://cwe.mitre.org/data/definitions/400.html Accessed 30.05.2016.

[30] Common Weakness Enumeration (CWE). CWE-601: URL Redirection to Untrusted Site ('Open Redirect'). https://cwe.mitre.org/data/definitions/601.html Accessed 30.05.2016.

[31] Common Weakness Enumeration (CWE). CWE-326: Inadequate Encryption Strength. https://cwe.mitre.org/data/definitions/326.html Accessed 30.05.2016.

[32] Common Weakness Enumeration (CWE). CWE-330: Use of Insufficiently Random Values. https://cwe.mitre.org/data/definitions/330.html Accessed 30.05.2016.

[33] Common Weakness Enumeration (CWE). CWE-521: Weak Password Requirements. https://cwe.mitre.org/data/definitions/521.html Accessed 30.05.2016.

[34] Common Weakness Enumeration (CWE). CWE-507: Trojan Horse. https://cwe.mitre.org/data/definitions/507.html Accessed 30.05.2016.

[35] Common Weakness Enumeration (CWE). CWE-113: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting'). https://cwe.mitre.org/data/definitions/113.html Accessed 30.05.2016.

[36] Common Weakness Enumeration (CWE). CWE-20: Improper Input Validation. https://cwe.mitre.org/data/definitions/20.html Accessed 30.05.2016.

[37] Common Weakness Enumeration (CWE). CWE-302: Authentication Bypass by Assumed-Immutable Data. https://cwe.mitre.org/data/definitions/302.html Accessed 30.05.2016.

[38] Common Weakness Enumeration (CWE). CWE-303: Incorrect Implementation of Authentication Algorithm. https://cwe.mitre.org/data/definitions/303.html Accessed 30.05.2016.

[39] Common Weakness Enumeration (CWE). CWE-361: Time and State. https://cwe.mitre.org/data/definitions/361.html Accessed 30.05.2016.

[40] Common Weakness Enumeration (CWE). CWE-732: Incorrect Permission Assignment for Critical Resource. https://cwe.mitre.org/data/definitions/732.html Accessed 30.05.2016.

[41] Common Weakness Enumeration (CWE). CWE-664: Improper Control of a Resource Through its Lifetime. https://cwe.mitre.org/data/definitions/664.html Accessed 30.05.2016.

[42] Common Weakness Enumeration (CWE). CWE-311: Missing Encryption of Sensitive Data. https://cwe.mitre.org/data/definitions/311.html Accessed 30.05.2016.

[43] Common Weakness Enumeration (CWE). CWE-565: Reliance on Cookies without Validation and Integrity Checking. https://cwe.mitre.org/data/definitions/565.html Accessed 30.05.2016.

[44] Common Weakness Enumeration (CWE). CWE-306: Missing Authentication for Critical Function. https://cwe.mitre.org/data/definitions/306.html Accessed 30.05.2016.

[45] Common Weakness Enumeration (CWE). CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'). https://cwe.mitre.org/data/definitions/89.html Accessed 30.05.2016.

[46] Common Weakness Enumeration (CWE). CWE-74: Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection'). https://cwe.mitre.org/data/definitions/74.html Accessed 30.05.2016.

[47] Common Weakness Enumeration (CWE). CWE-390: Detection of Error Condition Without Action. https://cwe.mitre.org/data/definitions/390.html Accessed 30.05.2016.

[48] Common Weakness Enumeration (CWE). CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow'). https://cwe.mitre.org/data/definitions/120.html Accessed 30.05.2016.

[49] Common Weakness Enumeration (CWE). CWE-680: Integer Overflow to Buffer Overflow. https://cwe.mitre.org/data/definitions/680.html Accessed 30.05.2016.

[50] Common Weakness Enumeration (CWE). CWE-805: Buffer Access with Incorrect Length Value. https://cwe.mitre.org/data/definitions/805.html Accessed 30.05.2016.

[51] Common Weakness Enumeration (CWE). CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer. https://cwe.mitre.org/data/definitions/119.html Accessed 30.05.2016.

[52] Common Weakness Enumeration (CWE). CWE-131: Incorrect Calculation of Buffer Size. https://cwe.mitre.org/data/definitions/131.html Accessed 30.05.2016.

[53] Common Weakness Enumeration (CWE). CWE-129: Improper Validation of Array Index. https://cwe.mitre.org/data/definitions/129.html Accessed 30.05.2016.

[54] Common Weakness Enumeration (CWE). CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal'). https://cwe.mitre.org/data/definitions/22.html Accessed 30.05.2016.