Project no: 269317

nSHIELD

new embedded Systems arcHItecturE for multi-Layer Dependable solutions

Instrument type: Collaborative Project, JTI-CP-ARTEMIS

Priority name: Embedded Systems

REVISION A D2.8: Final SPD metric specification

Due date of deliverable: M26

Actual submission date: 30/10/2013

Start date of project: 01/09/2011 Duration: 36 months

Organisation name of lead contractor for this deliverable:

Fundación Tecnalia Research & Innovation, TECNALIA

[Issue 4]

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2012)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	X

Document Authors and Approvals

Authors		Date	Signature
Name	**Company**		
Baldelli Renato	Selex ES	10.05.2013	
Andrea Morgagni	Selex ES	01.10.2013	
Iñaki Eguia	Tecnalia	01.10.2013	
Arkaitz Gamino	Tecnalia	01.10.2013	
Reviewed by			
Name	**Company**		
Approved by			
Name	**Company**		

Applicable Documents		
ID	**Document**	**Description**
[01]	TA	nSHIELD Technical Annex
[02]	2.5 Preliminary SPD metrics	First SPD metrics approach

Modification History		
Issue	**Date**	**Description**
Creation	01/15/04	Creation of ToC
Surface metrics	9/2013	Surface metric first approach
Multi metrics	10/2013	Multimetric first approach

Executive Summary

This document addresses 2 styles for facing security measurement and its smart aggregation. First approach tackles aggregation through attack surface metric identifying three relevant factors: Porosity, Controls and Limitations. 2nd approach fuzzy-evolutionary multi metric approach strives on different and heterogeneous metrics that are composed though an expert system and optimized with genetic algorithms.

Both of them are top innovative approaches but realistic ones. Indeed, this document faces both the theoretical challenge and practical and feasible one in the real scenarios. We will intent to implement in the selected nSHIELD scenarios: avionics, railway, voice/facial and social.

The document is completely structured in two main sections, each one corresponding to each of the approaches.

Contents

Figures

Tables

Glossary

Please refer to the Glossary document, which is common for all the deliverables in nSHIELD.

This Page is intentionally left blank

1 Introduction

The definition we can find in the New Oxford American Dictionary for the word "metric" is: “system or standard of measurement”. In mathematics and physics it is “a binary function of a topological space that gives, for any two point of the space, a value equal to the distance between them, or a to a value treated as analogous to distance for the purpose of analysis”.

In relation to IT security, metrics should help in quantifying the “impact of an attack”. We have to admit that even this definition for security metrics is not commonly accepted.

The term, as used in practice, appears to represent several different notions: metric (in the sense of quantitative standard function based on process and/or product measures), measurement, score, rating, rank, or assessment result. The metrics of security of ICT systems aim at measuring their security by means of some indicators that could be a sign of relevant security characteristics [Freiling]. With the scope of evaluating security at the system level, security metrics are therefore gauges that facilitate the decision making process during the whole life cycle of the system, while supporting their accountability. The metrics are produced by a dedicated process of collection, analysis, and reporting of relevant data. Based on the security goals and objectives, ICT security metrics should quantifiable, feasible to be objectively computed, and repeatable. By their continuous computation, metrics can provide relevant trends over time. In this way they can be applied for evaluating the fulfillment of objectives and requirements, and for directing resources to any needed prevention, mitigation or reaction procedure. Due to the variety that characterizes the ICT world, it is very hard to define a single metric that is able to capture in deep, with a “magic number”, all the aspects and impacts of a cyber-security.

Regarding the dependability, in the present technical literature, the term is used for the general description of a system characteristic but not an attribute that can be expressed using a single metric. There are several metrics, which form the foundation of dependability, such as Reliability, Availability, Safety, MTTF, Coverage, and Fault Latency. These dependability-related metrics are often measured through the life testing. However, the time needed to obtain a statistically significant number of failures makes the life testing impractical for most dependable computers. Fault injection techniques are thoroughly showed as an effective approach to testing and evaluating the systems with high dependability requirements.

The failures of critical computer-driven systems have serious consequences, in terms of monetary loss and/or human sufferings. However, for decades it has been obvious that the Reliability, Availability, and Safety of computer systems cannot be obtained solely by the careful design, the quality assurance, or other fault avoidance techniques. Proper testing mechanisms must be applied to these systems in order to achieve certain dependability requirements.

2 Terms and definitions

Please refer to the Terms and definitions in document D2.5 Preliminary Metrics Specification, which is common for all the deliverables in nSHIELD

3 nSHIELD Attack Surface Metric

In this section we want introduce the notion of “nSHIELD's attack surface” and present a way to measure it. This approach is an integration of three different works: "An attack surface metric" [01], "The Open Source Testing Methodology Manual (OSSTMM) 3" [02] and "Common Criteria Evaluation Methodology (CEM)" [03].

Even if it derives from these three different approaches, we aim to propose a method totally defined and implemented into nSHIELD application scenarios, which takes inspiration from Common Criteria Evaluation Methodology (CEM) standard and Open Source Testing Methodology Manual (OSSTMM), merging their concepts to ensure a consistently measured and expressed as a cardinal number SPD metric for embedded systems attack surface in order to obtain a systematic way to measure it.

In the next paragraph there is a brief description of how this three different works, that have inspired our approach, have been used to define it.

Intuitively, a system’s attack surface is the set of ways in which an adversary can enter the system and potentially cause damage. Hence the “smaller” the attack surface, the more secure the system.

Traditionally there are two different communities separately working on the issue of dependability and security. Community of dependability is more concerned with non-malicious faults, the security community is more concerned with malicious attacks or faults.

We propose a definition that can generically integrate dependability and security. In few words we consider the threat as the origin of the fault chain (fault -> errors -> failures) for the dependability concerns and as the potential for abuse of protected assets by the system for security concerns.

The attacker is the threat agent; it is a malicious human activity or non-malicious event.

An attacker uses nSHIELD's entry and exit points to attack the system. We introduce an entry and exit point framework to identify three relevant factors: Porosity, Controls, and Limitations.

An entry and exit point contribution to the attack surface reflects factors' likelihood of being used in attacks. For example an entry point running a method with root privilege is more likely to be used in attacks than a method running with non-root privilege. We introduce the notion of a damage potential-effort ratio (der) to estimate porosity contribution. A system’s attack surface measurement is the total contribution of the system’s factors along the porosity, controls, and limitation.

3.1 Formal model

We chose I/O automata as our model, because the notion of entry and exit points maps naturally to the input actions and output actions. An I/O automaton, $A = \langle sig(A);\ states(A);\ start(A);\ steps(A)\rangle$, is a four tuple consisting of:

1. an action signature, $sig(A)$, that partitions a set, $acts(A)$, of actions into three disjoint sets, $in(A)$; $out(A)$, and $int(A)$, of input, output and internal actions, respectively;
2. a set, $states(A)$, of states;
3. a non-empty set, $start(A) \subseteq states(A)$, of start states;
4. and a transition relation, $steps(A) \subseteq states(A) \times acts(A) \times states(A)$.

An I/O automaton’s environment generates input and transmits the input to the automaton using input actions. Conversely, the automaton generates output actions and internal actions autonomously and transmits output to its environment. We construct an I/O automaton modeling a complex system by composing the I/O automata modeling the system’s simpler components. The composition of a set of I/O automata results in an I/O automaton.

Let S be a set of systems, for each system $s \in S$, we define its environment E_s, as follow: $E_s = \langle U, T \rangle$ where U is a user, and $T = S \setminus \{s\}$ is a set of system excluding s. Clearly s interacts with its environment E_s by means its entry and exit points, called *accesses*, note that they act as the basis for attacks on the system. the I/O automata model permit to map the notion of entry points and exit points to the input actions and output actions of an I/O automaton. We detail the actions, using pre and post conditions: for an action, a, we denote $p = a.pre(a.post)$ the pre(post) condition of a. An action's pre(post) condition p is a first order predicates on the state variables. A state of system s, $st \in states(s)$, ia a mapping of the state variables to their values: $st: Var \rightarrow Val$. A state transition, $tr = \langle st, a, st' \rangle$, is the invocation of an action a in state st, resulting in the state st'.

Note that ,a system, s, can receive data from its environment if s has an input action, a, and an entity, s', in the environment has a same-named output action, a. Furthermore a system's internal actions that are not visible to other systems in the environment can be used to formalize indirect entry points. In particular we formalize data transmission using actions' pre and post conditions.

If an input action, a, of a system, s, receives a data item, d, directly from the environment, then s's subsequent behavior depends on d; hence d appears in the post condition of a and we write $d \in Res(a.post)$ where $Res : predicate \rightarrow 2^{Var}$ is a function such that for each post condition (or pre condition), p, $Res(p)$ is the set of resources appearing in p. Similarly, if an action, a, receives a data item d from another action, a', then d appears in a''s post condition and in a's pre condition.

Thus we can provide the following definitions:

- A *direct entry point* of a system, s, is an input action, a, of s, such that either (i) U or (ii) a system $s' \in T$ has the output action a.
- An *indirect entry point* of a system, s, is an internal action, a, of s, such that either (i) ∃ direct entry point, a', of s such that $a'.post \Rightarrow a.pre$ and ∃ a data item, d, such that $d \in Res(a'.post) \wedge d \in Res(a.pre)$, or (ii) ∃ indirect entry point, a'', of s, such that $a''.post \Rightarrow a.pre$ and ∃ a data item, d, such that $d \in Res(a''.post) \wedge d \in Res(a.pre)$

Analogously we can define direct and indirect exit point:

- A *direct exit point* of a system, s, is an output action, a, of s, such that either (i) U or (ii) a system $s' \in T$ has the input action a.
- An *indirect exit point* of a system, s, is an internal action, a, of s, such that either (i) ∃ direct exit point, a', of s such that $a.post \Rightarrow a'.pre$ and ∃ a data item, d, such that $d \in Res(a.post) \wedge d \in Res(a'.pre)$, or (ii) ∃ indirect exit point, a'', of s, such that $a.post \Rightarrow a''.pre$ and ∃ a data item, d, such that $d \in Res(a.post) \wedge d \in Res(a''.pre)$.

We model the user U as I/O automata, U are global with respect to the systems in S and represent the adversary who attacks the systems in S. For simplicity, we assume only one user U present in the environment.

A system's access is the subset of its resources that an attacker can use to attack the system. An attacker can use a system's entry points and exit points to attack the system. Hence the set of entry points and exit points are the relevant subset of resources that are part of the attack surface porosity.

As explained in the next session, each resource contributes in a different manner to system's porosity measurement because not all resources are equally likely to be used by an attacker. Each contribution depends on the resource damage potential, i.e., the level of harm the attacker can cause to the system in using the resource in an attack, and the effort the attacker spends to acquire the necessary access rights in order to be able to use the resource in an attack. The higher the damage potential or the lower the effort, the higher the access contribution to the porosity. In this section, we use our I/O automata model to formalize the notions of damage potential and effort. Considering a resource, r, we model the damage potential and effort as the state variables $r.dp$ and $r.ef$, respectively.

In practice, we estimate a resource damage potential and effort in terms of the resource attributes, in particular, we estimate the damage potential of a point in terms of the resource privilege, and the attacker effort in terms of the resource access rights. For example, an attacker can cause more damage to a system by using a method with `root` privilege than a method running with `non-root` privilege;

Furthermore the attacker can use a resource in an attack if he/she has the required access right, and, certainly the attacker spends effort to acquire these access rights.

In the I/O automata model, the pre and post conditions are used to formalize effort and damage potential, respectively. In fact, the effort corresponds to the pre conditions the attacker needs to satisfy to invoke an action and the damage potential corresponds to the action invocation's damaging effect stated in the action's post condition.

Considering two resources $r_1$and r_2, the notation $r_1 \succ r_2$ indicates that the resource, r_1, makes a larger contribution than a resource, r_2. In details, given two resources, r_1 and r_2, $r_1 \succ r_2$ iff either:

i. $r_1.dp \succ r_2.dp \wedge r_2.ef \succ r_1.ef$,
ii. or $r_1.dp = r_2.dp \wedge r_2.ef \succ r_1.ef$,
iii. or $r_1.dp \succ r_2.dp \wedge r_2.ef = r_1.ef$.

Similarly given two resources, , r_1 and r_2, $r_1 \succcurlyeq r_2$ iff either:

i. $r_1 \succ r_2$
ii. or $r_1.dp = r_2.dp \wedge r_2.ef = r_1.ef$.

Considering an action a, of system s, the following parametric definition can be provided:

$$a(A;B)$$

$$pre : P_{pre} \wedge A \succcurlyeq a.ef$$

$$post : P_{post} \wedge B \succcurlyeq a.dp$$

The parameters A and B represent, respectively, the highest method access rights acquired by an attacker so far, and the benefit to the attacker in using a. P_{pre} is the part of pre condition of a that does not involve access rights. Hence, the clause, $A \succcurlyeq a.ef$, captures the condition that the attacker has the required access rights to invoke a. Similarly, P_{post} is the part of a's post condition that does not involve benefit, and the clause, $B \succcurlyeq a.dp$, captures the condition that the attacker gets the expected benefit after the execution of a.

Given a system, s, and its environment, E_s, the *access* of s is its set of entry points and exit points, $\langle P^{E_s} \rangle$.

Given an environment, $E_s = \langle U, T \rangle$, and two systems, s and s', the *access* $\langle P_s^{E_s} \rangle$ of system s, is larger than the *access* $\langle P_{s'}^{E_s} \rangle$ of system s' iff $P_s^{E_s} \supset P_{s'}^{E_s}$

The qualitative comparison introduced above is not useful to determine a measurement of *access*, it is necessary a quantitative measure we define this measure in terms of the resources damage potential-effort ratios.

From an attacker's point of view damage potential and effort are related. This ratio is similar to a cost-benefit ratio; the damage potential is the benefit to the attacker in using a resource in an attack and the effort is the cost to the attacker in using the resource.

For example, if consider a method a, its damage potential is determined by the potential number of methods that a can call and hence the potential number of methods that can follow a in a schedule[1].

[1] An execution of s is an alternating sequence of actions and states beginning with a start state, and a schedule is a subsequence of the execution consisting only of the actions appearing in the execution.

similarly, the effort of a is determined by the potential number of methods that can call a and hence the potential number of methods that a can follow in a schedule;

Hence damage potential-effort ratio of a, $der(a)$, determines the potential number of schedules in which

a can appear.

We assume a function, $der : \text{point} \to \mathbb{Q}$, that maps each entry/exit point to its damage potential-effort ratio belonging to the set, $\mathbb{Q}$, of rational numbers.

Thus, to quantify a system access measurement we consider the access and the damage potential-effort ratio, in particular given a system s, and its access $\langle P^{E_s} \rangle$, the *access measurement* of s can be defined as follow:

$$\langle \sum_{a \in P^{E_s}} der(a) \rangle$$

3.2 SPD level

We consider the threat as the origin of malicious attacks and non malicious events able to subvert the security or the dependability of nSHIELD system.

A threat, to be effective, must interact either directly or indirectly with the asset. To separate the threat from the asset we need to avoid a possible interaction. Therefore it is possible to have total (100) SPD level if the threat and the asset are completely separated from each other. Otherwise what you have is an assurance protection of the asset which is provided by the controls you put on the asset or the degree to which you lessen the impact of the threat.

To have true protection of the assets different types of controls are required. However, controls also may increase the number of interactions within the scope which means more controls are not necessarily better. Therefore it is recommended to use different types of operational controls rather than just more controls. More controls of the same type of operational controls do not provide a defense in depth as access through one is often access through all of that type. This is why it is so important to be able to categorize controls by what they do in operations to be certain of the level of protection provided by them.

To better understand how our approach can work with an operational environment, we have to analyze its single elements that permit to quantify the “Attack Surface” and define the terminology that characterize these elements.

3.2.1 Porosity

As defined in the previous paragraph, SPD is a function of separation. Either the separation between an asset and any threats exists or it does not. There are 3 logical and proactive ways to create this separation:

1. Create a physical or logical barrier between the asset and the threats
2. Change the threat to a harmless state
3. Remove the threat.

When analysing the state of security we can see where there is the possibility for interaction and where there is not. We don't care the justification for all interactive points, we refer to this as the “**Porosity”**[02]**.** The porosity reduces the separation between a threat and an access. It is further categorized as one of 3 elements, complexity, access and trust which describes its function in operations which further allows the proper controls to be added during the remediation phase of improving protection.

So consider that if the separation exists properly from the threats, such as a man inside a mountain avoiding lightning, then that security is true; it is 100. For every hole in the mountain, every means for

lightning to cause harm to that man, the porosity increases as an Access. Each point of interaction reduces the security below 100, where 100 represent a full separation. Therefore, the increase in porosity is the decrease in security, privacy and dependability and each *pore* is a Complexity, Access, or Trust.

Complexity: with this term we refer to number of components critical for the dependability of the nSHIELD system, which failure might not be tolerated by system architecture for a given usage profile.

Access: since the SPD level is the separation of a threat and an asset then the ability to interact with the asset directly is to access it. Access is calculated by the number of different places where the interaction can occur. Removing direct interaction with an asset will halve the number of ways it can be taken away.

Trust: We measure trust as each relationship that exists where the system accepts interaction freely from its component or another system within the scope. While a trust may be a security hole, it is a common replacement for authentication and a means for evaluating relationships in a rational and repeatable manner. Therefore, the use of trust metrics is encouraged which will allow for one to measure how valid a trust is by calculating the amount of reliability in the trust.

For each access pore identified, it's necessary to introduce the concept of damage potential-effort ratio to have a consistent measure of the lack of separation that introduces. Not all access pores contribute equally to system's porosity measurement because not all access pores are equally likely to be used by an attacker. A pore's contribution to a system's attack surface depends on the access pore 's damage potential, i.e., the level of harm the attacker can cause to the system in using the access pore in an attack and the effort the attacker spends to acquire the necessary access rights in order to be able to use it in an attack. The higher the damage potential or the lower the effort, the higher access pore's contribution to the porosity.

We consider damage potential and effort in isolation while estimating a resource's contribution to the attack surface [01]. From an attacker's point of view, however, damage potential and effort are related; if the attacker gains higher privilege by using a method in an attack, then the attacker also gains the access rights of a larger set of methods. For example, the attacker can access only the methods with authenticated user access rights by gaining authenticated privilege, whereas the attacker can access methods with authenticated user and root access rights by gaining root privilege. The attacker might be willing to spend more effort to gain a higher privilege level that then enables the attacker to cause damage as well as gain more access rights. Hence we consider damage potential and effort in tandem and quantify a resource's contribution as a damage potential effort ratio. The ratio is similar to a cost-benefit ratio; the damage potential is the benefit to the attacker in using a resource in an attack and the effort is the cost to the attacker in using the resource.

3.2.2 Controls

Controls are a means to influence the impact of threats and their effects when interaction is required.

While there are many different names and types of operation controls, there are only 12 main categories which contain all possible controls. Two of the categories however, Identification, the verification of an existing identity, and Authorization, the granting of permissions from the rightful authority, cannot stand alone in an operational environment and instead, in operations, combine and are added to the Authentication control. This leaves Operational Security (OpSec) with ten possible controls an Analyst will need to identify and understand.

The reason why Identification and Authorization cannot be expressed operationally is because neither

can be transferred. Identity exists as is and while the means of identification, as a process, is an operational aspect, the actual process is to verify a previously provided identity from another source or from the latest in a chain of sources. Even under circumstances where a government agency officially changes the identity of a person, they are still the same person from identifying marks to their DNA and only their documentation changes. Therefore, a security process can attempt to identify someone by verifying their identity but nothing in this case is granted or provided. There is no true "granting" of identity just as there can be no true "theft" of identity. Furthermore, identity is a collection of thoughts, emotions,

experiences, relationships, and intentions, as well as physical shape or marks. You are who you are because you exist not because someone granted that to you. A perfect duplicate or clone of you is still not you because from origin your experiences will differ. While this may sound more like philosophy than security, it is very important that Analysts understand this. Identification processes only verify against a former identification process. If that process has been corrupted or can be circumvented, then the entire security foundation that requires proper identification is flawed.

Authorization, like Identification, is another operations control which cannot be transferred. It is the control to grant permissions. An employee authorized to enter a room may hold the door open for another person to enter. This does not authorize the new person. Authorization did not get transferred. This new person is trespassing in a restricted area and the employee who held open the door actually was part of a limitation in the Authentication process to grant Access.

The Authentication control combines both identification and authorization to map Access. The process is simply knowing who (or what) it is and what, where, when, and how they can access before they are granted access. Because authentication is a control for interactivity, it is one of the five Class A controls, also known as the “Interactive Controls”.

3.2.2.1 Interactive Controls

The Class A Interactive Controls make up exactly half of all the operation controls. These controls directly influence complexity, access, or trust interactions. The Class A categories are Authentication, Indemnification, Subjugation, Continuity, and Resilience.

1. **Authentication** is a control through the challenge of credentials based on identification and authorization.
2. **Indemnification** is a control through a contract between the asset owner and the interacting party. This contract may be in the form of a visible warning as a precursor to legal action if posted rules are not followed, specific, public legislative protection, or with a third-party assurance provider in case of damages like an insurance company.
3. **Resilience** is a control over all interactions to maintain the protection of assets in the event of corruption or failure.
4. **Subjugation** is a control assuring that interactions occur only according to defined processes. The asset owner defines how the interaction occurs which removes the freedom of choice but also the liability of loss from the interacting party.
5. **Continuity** is a control over all interactions to maintain interactivity with assets in the event of corruption or failure.

3.2.2.2 Process Controls

The other half of operation controls are the Class B controls which are used to create defensive processes.

These controls do not directly influence interactions rather they protect the assets once the threat is present. These are also known as Process Controls and include Non-repudiation, Confidentiality, Privacy, Integrity, and Alarm.

1. **Non-repudiation** is a control which prevents the interacting party from denying its role in any interactivity.
2. **Confidentiality** is a control for assuring an asset displayed or exchanged between interacting parties cannot be known outside of those parties.
3. **Privacy** is a control for assuring the means of how an asset is accessed, displayed, or exchanged between parties cannot be known outside of those parties.
4. **Integrity** is a control to assure that interacting parties know when assets and processes have changed.
5. **Alarm** is a control to notify that an interaction is occurring or has occurred.

While controls are a positive influence in OpSec, minimizing the attack surface, they can themselves add to the attack surface if they themselves have limitations. Often times this effect is not noticed and if the protection mechanisms aren't tested thoroughly as to how they work under all conditions, this may not become apparent. Therefore the use of controls must assure that they do not insinuate new attack vectors into the target. Therefore, sometimes no controls are better than bad controls.

To facilitate understanding of operation controls, they can be matched back to the three Information Assurance Objectives of Confidentiality, Availability, and Integrity.

Table 3-1: Information Assurance Objectives/Operation Controls mapping

Information Assurance Objectives	Operation Controls
Confidentiality	Confidentiality Privacy Authentication Resilience
Integrity	Integrity Non-repudiation Subjugation
Availability	Continuity Indemnification Alarm

3.2.3 Limitations

The inability of protection mechanisms to work is their limitations. Therefore the state of security in regard to known flaws and restrictions within the operations scope is called Limitation. It is the holes, vulnerabilities, weaknesses, and problems in keeping that separation between an asset and a threat or in assuring controls continue working correctly.

Limitations have been classified into five categories and these categories define the type of vulnerability, mistake, misconfiguration, or deficiency by operation. This is different from how limitations are classified under most security management frameworks and best practices which is why we use the term Limitation rather than more common terms to avoid confusion. Those other terms refer to vulnerabilities or deficiencies because they are categorized by the type of attack or often the threat itself. There is a focus on the risk from the attack. However, to remove bias from security metrics and provide a more fair assessment we removed the use of risk. Risk itself is heavily biased and often highly variable depending on the environment, assets, threats, and many more factors. Therefore, under OpSec, we use the term Limitations to express the difference of categorizing how OpSec fails rather than by the type of threat.

Since the number and type of threats cannot be known it makes more sense to understand a security or safety mechanism based on when it will fail. This allows the Analyst to test for the conditions in which it will no longer sustain the necessary level of protection. Only once we have this knowledge can we begin to play the what-if game of threats and risks. Then we can also invest in the appropriate type of separation or controls required and create precise plans for disasters and contingencies.

Although the Limitations are categorized here as 1 through 5 this does not mean they are in a hierarchical format of severity. Rather they are numbered only to differentiate them both for operational planning and metrics. This also means it is possible that more than one type of Limitation can be applied to a single problem. Furthermore, the weight (value) of a particular Limitation is based on the other available and corresponding controls and interactive areas to the scope, there can be no specific hierarchy since the value of each is specific to the protective measures in the scope being audited.

The five Limitation classifications are:

1. **Vulnerability** is the flaw or error that: (a) denies access to assets for authorized people or processes, (b) allows for privileged access to assets to unauthorized people or processes, or (c) allows unauthorized people or processes to hide assets or themselves within the scope.
2. **Weakness** is the flaw or error that disrupts, reduces, abuses, or nullifies specifically the effects of the five interactivity controls: authentication, indemnification, resilience, subjugation, and continuity.
3. **Concern** is the flaw or error that disrupts, reduces, abuses, or nullifies the effects of the flow or execution of the five process controls: non-repudiation, confidentiality, privacy, integrity, and alarm.
4. **Exposure** is an unjustifiable action, flaw, or error that provides direct or indirect complexity of targets or assets within the chosen scope channel.
5. **Anomaly** is any unidentifiable or unknown element which has not been controlled and cannot be accounted for in normal operations.

To better understand how Limitations fit into the OpSec framework, it can be seen mapping back to security and safety:

Table 3-2: OpSec framework/Limitations mapping

<table>
<tr><th colspan="2">Category</th><th>OpSec</th><th>Limitations</th></tr>
<tr><td colspan="2" rowspan="3">Operations</td><td>Complexity</td><td>Exposure</td></tr>
<tr><td>Access</td><td rowspan="2">Vulnerability</td></tr>
<tr><td>Trust</td></tr>
<tr><td rowspan="10">Controls</td><td rowspan="5">Class A - Interactive</td><td>Authentication</td><td rowspan="5">Weakness</td></tr>
<tr><td>Indemnification</td></tr>
<tr><td>Resilience</td></tr>
<tr><td>Subjugation</td></tr>
<tr><td>Continuity</td></tr>
<tr><td rowspan="5">Class B - Process</td><td>Non-Repudiation</td><td rowspan="5">Concern</td></tr>
<tr><td>Confidentiality</td></tr>
<tr><td>Privacy</td></tr>
<tr><td>Integrity</td></tr>
<tr><td>Alarm</td></tr>
<tr><td colspan="4">Anomalies</td></tr>
</table>

This mapping shows how Limitations effect security and how their values are determined.

A vulnerability is the flaw or error that: (a) denies access to assets for authorized people or processes (b) allows for privileged access to assets to unauthorized people or processes, or (c) allows unauthorized people or processes to hide assets or themselves within the scope. This means that Vulnerability must be mapped to all points of interaction or OpSec and because Vulnerability can circumnavigate or nullify the Controls, these must also be considered in the weighting of Vulnerability.

A weakness is a flaw in Class A Controls however can impact OpSec therefore it is mapped to all OpSec parameters as well as being mapped to this interactive class of controls.

A concern can only be found in Class B Controls however can impact OpSec therefore it is mapped to all OpSec parameters as well as being mapped to this process class of controls.

An exposure gives us intelligence about the interaction with a target and thus maps directly to Complexity and Access. This intelligence can also help an attacker navigate around some or all controls and so Exposure is also mapped to both Control classes. Finally, Exposure has no value itself unless there is a way to use this intelligence to exploit the asset or a Control and so Vulnerabilities, Weaknesses and Concerns also play a role in the weighting of Exposure's value.

An anomaly is any unidentifiable or unknown element which has not been controlled and cannot be accounted for in normal operations. The fact that it has not been controlled and cannot be accounted for signifies a direct link with Trust. This Limitation can also cause anomalies in the way Controls function and so they are also included in the weighting. Finally, as with an Exposure, an Anomaly alone does not affect OpSec without the existence of either a Vulnerability, Weakness or Concern which can exploit this unusual behavior.

Additionally, more than one category can apply to a limitation when the flaw breaks OpSec in more than one place. For example, an Authentication control which allows a person to hijack another person's credentials has a Weakness and should the credentials allow Access then it also has a Vulnerability.

In another example, an Authentication control uses a common list of names corresponding to e-mail addresses. Every address which can be found or guessed and used as a log-in is an Exposure while the control itself has a Weakness for its inability to identify the correct user of the Authentication mechanism of the log-in. If any of those credentials allow Access then we include this as a Vulnerability as well.

3.2.3.1 Justification for Limitations

The concept that limitations are only limitations if they have no business justification is false. A limitation is a limitation if it behaves in one of the limiting factors as described here. A justification for a limitation is a risk decision that is met with either a control of some kind or merely acceptance of the limitation. Risk decisions that accept the limitations as they are often come down to: the damage a limitation can cause does not justify the cost to fix or control the limitation, the limitation must remain according to legislation, contracts, or policy, or a conclusion that the threat does not exist or is unlikely for this particular limitation. Since risk justifications are not a part of calculating an attack surface, all limitations discovered must still be counted within the attack surface regardless if best practice, common practice, or legal practice denotes it as not a risk. If it is not then the audit will not show a true representation of the operational security of the scope.

3.2.3.2 Managing Limitations

Another concept that must be taken into consideration is one of managing flaws and errors in an audit.

The three most straightforward ways to manage limitations is to remove the problem area providing the interactive point altogether, fix them, or accept them as part of doing business known as the business justification.

An audit will often uncover more than one problem per target. The Analyst is to report the limitations per target and not just which are the weak targets. These limitations may be in the protection measures and controls themselves, thus diminishing OpSec. Each limitation is to be rated as to what occurs when the problem is invoked, even if that invocation is theoretical or the verification is of limited execution to restrict actual damages. Theoretical categorization, where no verification could be made, is a slippery slope and should be limited to cases where verification would reduce the quality of operations. Then, when categorizing the problems, each limitation should be examined and calculated in specific terms of operation at its most basic components. However, the Analyst should be sure never to report a "flaw within a flaw" where the flaws share the same component and same operational effect. An example of this would be a door broken open with a broken window. The door opening is an Access even if the broken window is also but both are for the same component, the door way, and same operational effect, an opening. An example from Data Networks would be a computer system which sends a kernel reply, such as an ICMP "closed port" T03C03 packet for a particular port. This interaction is not counted for all such

ports since the Access comes from the same component, the kernel, and has the same operational effect, sending a T03C03 packet per port queried.

3.2.3.3 Limitation weight definition

In the calculation of "SPD level" is important the weight of the identified limitations.

In particular we'll consider the weight of a particular limitation (vulnerabilities) based on the concept of "attack potential" described in the Common Criteria standard [03] and used in pSHIELD SPD metrics.

Attack potential is a function of expertise, resources and motivation. There are multiple methods of representing and quantifying these factors.

Motivation is an attack potential factor that can be used to describe several aspects related to the attacker and the assets the attacker desires. Firstly, motivation can imply the likelihood of an attack - one can infer from a threat described as highly motivated that an attack is imminent, or that no attack is anticipated from an un-motivated threat. However, except for the two extreme levels of motivation, it is difficult to derive a probability of an attack occurring from motivation.

Secondly, motivation can imply the value of the asset, monetarily or otherwise, to either the attacker or the asset holder. An asset of very high value is more likely to motivate an attack compared to an asset of little value. However, other than in a very general way, it is difficult to relate asset value to motivation because the value of an asset is subjective - it depends largely upon the value an asset holder places on it.

Thirdly, motivation can imply the expertise and resources with which an attacker is willing to effect an attack. One can infer that a highly motivated attacker is likely to acquire sufficient expertise and resources to defeat the measures protecting an asset. Conversely, one can infer that an attacker with significant expertise and resources is not willing to effect an attack using them if the attacker's motivation is low.

Considering the second aspect, an asset holder may believe that the value of the assets (however measured) is sufficient to motivate attack against them. Once an evaluation is deemed necessary, the attacker's motivation is considered to determine the methods of attack that may be attempted, as well as the expertise and resources used in those attacks. Once examined, the developer is able to choose the appropriate assurance level, in particular the AVA[2] requirement components, commensurate with the attack potential for the threats. During the course of the evaluation, and in particular as a result of completing the vulnerability assessment activity, the evaluator determines whether or not the Target of Evaluation (TOE), operating in its operational environment, is sufficient to thwart attackers with the identified expertise and resources.

3.2.3.3.1 Characterizing attack potential

The determination of the attack potential for an attack corresponds to the identification of the effort required to create the attack, and to demonstrate that it can be successfully applied to the TOE (including setting up or building any necessary test equipment), thereby exploiting the vulnerability in the TOE. The demonstration that the attack can be successfully applied needs to consider any difficulties in expanding a result shown in the laboratory to create a useful attack. For example, where an experiment reveals some bits or bytes of a confidential data item (such as a key), it is necessary to consider how the remainder of the data item would be obtained (in this example some bits might be measured directly by further experiments, while others might be found by a different technique such as exhaustive search). It may not be necessary to carry out all of the experiments to identify the full attack, provided it is clear that the attack actually proves that access has been gained to a TOE asset, and that the complete attack could

[2] Vulnerability assessment class (of assurance requirements) addresses the possibility of exploitable vulnerabilities introduced in the development or the operation of a TOE.

realistically be carried out in exploitation according to the AVA_VAN[3] component targeted. In some cases the only way to prove that an attack can realistically be carried out in exploitation according to the

AVA_VAN component targeted is to perform completely the attack and then rate it based upon the resources actually required. One of the outputs from the identification of a potential vulnerability is assumed to be a script that gives a step-by-step description of how to carry out the attack that can be

used in the exploitation of the vulnerability on another instance of the TOE. In many cases, the evaluators will estimate the parameters for exploitation, rather than carry out the full exploitation. The estimates and their rationale will be documented in the ETR.

3.2.3.3.2 Factors to be considered

The following factors should be considered during analysis of the attack potential required to exploit vulnerability:

a) Time taken to identify and exploit (Elapsed Time);
b) Specialist technical expertise required (Specialist Expertise);
c) Knowledge of the TOE design and operation (Knowledge of the TOE);
d) Window of opportunity;
e) IT hardware/software or other equipment required for exploitation.

In many cases these factors are not independent, but may be substituted for each other in varying degrees. For example, expertise or hardware/software may be a substitute for time. A discussion of these factors follows. (The levels of each factor are discussed in increasing order of magnitude.) When it is the case, the less “expensive” combination is considered in the exploitation phase.

Elapsed time is the total amount of time taken by an attacker to identify that a particular potential vulnerability may exist in the TOE, to develop an attack method and to sustain effort required to mount the attack against the TOE. When considering this factor, the worst case scenario is used to estimate the amount of time required. The identified amount of time is as follows:

a) less than one day;
b) between one day and one week;
c) between one week and two weeks;
d) between two weeks and one month;
e) each additional month up to 6 months leads to an increased value;
f) more than 6 months.

Specialist expertise refers to the level of generic knowledge of the underlying principles, product type or attack methods (e.g. Internet protocols, Unix operating systems, buffer overflows). The identified levels are as follows:

a) Laymen are unknowledgeable compared to experts or proficient persons, with no particular expertise;
b) Proficient persons are knowledgeable in that they are familiar with the security behavior of the product or system type;
c) Experts are familiar with the underlying algorithms, protocols, hardware, structures, security behavior, principles and concepts of security employed, techniques and tools for the definition of

[3] Vulnerability Analysis family allows application of a wide range of assessment methodologies being unspecific to the kind of an attack scenario. These unspecific assessment methodologies comprise, among other, also the specific methodologies for those TSF where covert channels are to be considered or can be overcome by the use of sufficient resources in the form of a direct attack.

new attacks, cryptography, classical attacks for the product type, attack methods, etc. implemented in the product or system type.

d) The level “Multiple Expert” is introduced to allow for a situation, where different fields of expertise are required at an Expert level for distinct steps of an attack.

It may occur that several types of expertise are required. By default, the higher of the different expertise factors is chosen. In very specific cases, the “multiple expert” level could be used but it should be noted that the expertise must concern fields that are strictly different like for example HW manipulation and cryptography. Knowledge of the TOE refers to specific expertise in relation to the TOE.

This is distinct from generic expertise, but not unrelated to it. Identified levels are as follows:

a) Public information concerning the TOE (e.g. as gained from the Internet);
b) Restricted information concerning the TOE (e.g. knowledge that is controlled within the developer organization and shared with other organizations under a non-disclosure agreement)
c) Sensitive information about the TOE (e.g. knowledge that is shared between discreet teams within the developer organization, access to which is constrained only to members of the specified teams);
d) Critical information about the TOE (e.g. knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need to know basis and individual undertaking).

The knowledge of the TOE may graduate according to design abstraction, although this can only be done on a TOE by TOE basis. Some TOE designs may be public source (or heavily based on public source) and therefore even the design representation would be classified as public or at most restricted, while the implementation representation for other TOEs is very closely controlled as it would give an attacker information that would aid an attack and is therefore considered to be sensitive or even critical.

It may occur that several types of knowledge are required. In such cases, the higher of the different knowledge factors is chosen.

Window of opportunity (Opportunity) is also an important consideration, and has a relationship to the Elapsed Time factor. Identification or exploitation of a vulnerability may require considerable amounts of access to a TOE that may increase the likelihood of detection. Some attack methods may require considerable effort off-line, and only brief access to the TOE to exploit. Access may also need to be continuous, or over a number of sessions.

For some TOEs the Window of opportunity may equate to the number of samples of the TOE that the attacker can obtain. This is particularly relevant where attempts to penetrate the TOE and undermine the SFRs (Security Functional Requirements) may result in the destruction of the TOE preventing use of that TOE sample for further testing, e.g. hardware devices. Often in these cases distribution of the TOE is controlled and so the attacker must apply effort to obtain further samples of the TOE.

For the purposes of this discussion:

a) unnecessary/unlimited access means that the attack doesn't need any kind of opportunity to be realised because there is no risk of being detected during access to the TOE and it is no problem to access the number of TOE samples for the attack;
b) easy means that access is required for less than a day and that the number of TOE samples required to perform the attack is less than ten;
c) moderate means that access is required for less than a month and that the number of TOE samples required to perform the attack is less than one hundred;
d) difficult means that access is required for at least a month or that the number of TOE samples required to perform the attack is at least one hundred;
e) none means that the opportunity window is not sufficient to perform the attack (the length for which the asset to be exploited is available or is sensitive is less than the opportunity length needed to perform the attack - for example, if the asset key is changed each week and the attack

needs two weeks); another case is, that a sufficient number of TOE samples needed to perform the attack is not accessible to the attacker - for example if the TOE is a hardware and the probability to destroy the TOE during the attack instead of being successful is very high and the attacker has only access to one sample of the TOE.

Consideration of this factor may result in determining that it is not possible to complete the exploit, due to requirements for time availability that are greater than the opportunity time.

IT hardware/software or other equipment refers to the equipment required to identify or exploit a vulnerability.

a) Standard equipment is readily available to the attacker, either for the identification of a vulnerability or for an attack. This equipment may be a part of the TOE itself (e.g. a debugger in an operating system), or can be readily obtained (e.g. Internet downloads, protocol analyser or simple attack scripts).
b) Specialised equipment is not readily available to the attacker, but could be acquired without undue effort. This could include purchase of moderate amounts of equipment (e.g. power analysis tools, use of hundreds of PCs linked across the Internet would fall into this category), or development of more extensive attack scripts or programs. If clearly different test benches consisting of specialized equipment are required for distinct steps of an attack this would be rated as bespoke.
c) Bespoke equipment is not readily available to the public as it may need to be specially produced (e.g. very sophisticated software), or because the equipment is so specialised that its distribution is controlled, possibly even restricted. Alternatively, the equipment may be very expensive.
d) The level “Multiple Bespoke” is introduced to allow for a situation, where different types of bespoke equipment are required for distinct steps of an attack.

Specialist expertise and Knowledge of the TOE are concerned with the information required for persons to be able to attack a TOE. There is an implicit relationship between an attacker's expertise (where the attacker may be one or more persons with complementary areas of knowledge) and the ability to effectively make use of equipment in an attack. The weaker the attacker's expertise, the lower the potential to use equipment (IT hardware/software or other equipment). Likewise, the greater the expertise, the greater the potential for equipment to be used in the attack. Although implicit, this relationship between expertise and the use of equipment does not always apply, for instance, when environmental measures prevent an expert attacker's use of equipment, or when, through the efforts of others, attack tools requiring little expertise to be effectively used are created and freely distributed (e.g. via the Internet).

3.2.3.3.3 Calculation of attack potential

Table 4.3 identifies the factors discussed in the previous section and associates numeric values with the total value of each factor.

Where a factor falls close to the boundary of a range the evaluator should consider use of an intermediate value to those in the table. For example, if twenty samples are required to perform the attack then a value between one and four may be selected for that factor, or if the design is based on a publicly available design but the developer has made some alterations then a value between zero and three should be selected according to the evaluator's view of the impact of those design changes. The table is intended as a guide.

The “**” specification in the table in considering Window of Opportunity is not to be seen as a natural progression from the timescales specified in the preceding ranges associated with this factor. This specification identifies that for a particular reason the potential vulnerability cannot be exploited in the TOE in its intended operational environment. For example, access to the TOE may be detected after a certain amount of time in a TOE with a known environment (i.e. in the case of a system) where regular patrols are completed, and the attacker could not gain access to the TOE for the required two weeks

undetected. However, this would not be applicable to a TOE connected to the network where remote access is possible, or where the physical environment of the TOE is unknown.

Table 3-3: Calculation of attack potential

Factor	Value
Elapsed Time	
<= one day	0
<= one week	1
<= two weeks	2
<= one month	4
<= two months	7
<= three months	10
<= four months	13
<= five months	15
<= six months	17
> six months	19
Expertise	
Layman	0
Proficient	3*[4]
Expert	6
Multiple experts	8
Knowledge of TOE	
Public	0
Restricted	3
Sensitive	7
Critical	11
Window of Opportunity	
Unnecessary / unlimited access	0
Easy	1
Moderate	4
Difficult	10
None	**[5]
Equipment	
Standard	0
Specialized	4[6]
Bespoke	7
Multiple bespoke	9

To determine the resistance of the TOE to the potential vulnerabilities identified the following steps should be applied:

a) Define the possible attack scenarios {AS1, AS2... ASn} for the TOE in the operational environment.

[4] When several proficient persons are required to complete the attack path, the resulting level of expertise still remains “proficient” (which leads to a 3 rating).

[5] Indicates that the attack path is not exploitable due to other measures in the intended operational environment of the TOE.

[6] If clearly different test benches consisting of specialised equipment are required for distinct steps of an attack, this should be rated as bespoke.

b) For each attack scenario, perform a theoretical analysis and calculate the relevant attack potential using Table 4-3.
c) For each attack scenario, if necessary, perform penetration tests in order to confirm or to disprove the theoretical analysis.
d) Divide all attack scenarios {AS1, AS2, ..., ASn} into two groups:
   1) the attack scenarios having been successful (i.e. those that have been used to successfully undermine the SFRs), and
   2) the attack scenarios that have been demonstrated to be unsuccessful.
e) For each successful attack scenario, apply Table 4-4 and determine, whether there is a contradiction between the resistance of the TOE and the chosen AVA_VAN assurance component, see the last column of Table 4.4.
f) f) Should one contradiction be found, the vulnerability assessment will fail, e.g. the author of the ST chose the component AVA_VAN.5 and an attack scenario with an attack potential of 21 points (high) has broken the security of the TOE. In this case the TOE is resistant to attacker with attack potential 'Moderate', this contradicts to AVA_VAN.5, hence, the vulnerability assessment fails.

The “Values” column of Table 4.4 indicates the range of attack potential values (calculated using Table 4.3) of an attack scenario that results in the SFRs being undermined.

Table 3-4: Rating of vulnerabilities and TOE resistance

Values	Attack potential Required to exploit scenario	TOE resistant to attackers with attack potential of	Meets assurance components	Failure of components
0-9	Basic	No rating	-	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3, AVA_VAN.4, AVA_VAN.5
10-13	Enhanced-Basic	Basic	AVA_VAN.1, AVA_VAN.2	AVA_VAN.3, AVA_VAN.4, AVA_VAN.5
14-19	Moderate	Enhanced-Basic	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3	AVA_VAN.4, AVA_VAN.5
20-24	High	Moderate	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3, AVA_VAN.4	AVA_VAN.5
=>25	Beyond High	High	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3, AVA_VAN.4, AVA_VAN.5	-

An approach such as this cannot take account of every circumstance or factor, but should give a better indication of the level of resistance to attack required to achieve the standard ratings. Other factors, such as the reliance on unlikely chance occurrences are not included in the basic model, but can be used by an evaluator as justification for a rating other than those that the basic model might indicate.

It should be noted that whereas a number of vulnerabilities rated individually may indicate high resistance to attack, collectively the combination of vulnerabilities may indicate that overall a lower rating is applicable. The presence of one vulnerability may make another easier to exploit.

In the SPD level computation, the value assigned to vulnerabilities is calculated by performing a weighted sum of all identified vulnerabilities categorized according to the table 4.4. The weight for each category of

vulnerability (basic, enhanced basic, moderate, high and beyond high) is assigned by constant values (empirical parameters derived from experience) that are characterized by being inversely proportional to the attack potential required to exploit the scenario; this is because the lower the attack potential required to exploit the scenario the greater the impact of the vulnerability and then the attack surface on which is based the whole theory.

If a PP/ST author wants to use the attack potential table for the determination of the level of attack the TOE should withstand (selection of Vulnerability analysis (AVA_VAN) component), he should proceed as follows: For all different attack scenarios (i.e. for all different types of attacker and/or different types of attack the author has in mind) which must not violate the SFRs, several passes through Table 4-3 should be made to determine the different values of attack potential assumed for each such unsuccessful attack scenario. The PP/ST author then chooses the highest value of them in order to determine the level of the TOE resistance to be claimed from Table 4-4: the TOE resistance must be at least equal to this highest value determined. For example, the highest value of attack potentials of all attack scenarios, which must not undermine the TOE security policy, determined in such a way is Moderate; hence, the TOE resistance shall be at least Moderate (i.e. Moderate or High); therefore, the PP/ST author can choose either AVA_VAN.4 (for Moderate) or AVA_VAN.5 (for High) as the appropriate assurance component.

3.2.4 Actual SPD Level

The role of the Controls is to control the porosity in OpSec. It's like having ten ways of controlling threats that come through a hole in a wall. For each hole, a maximum of ten different controls can be applied which bring security back up towards and sometimes above 100. Limitations then reduce the effectiveness of OpSec and Controls. The result of an audit which discovers and shows the Controls, and Limitations is effectively demonstrating Actual SPD level.

Actual SPD level is a term for a snapshot of an attack surface in an operational environment. It is a logarithmic representation of the Controls, Limitations, and OpSec at a particular moment in time. It is logarithmic because it represents the reality of size where a larger scope will have a larger attack surface even if mathematically the Controls will balance the OpSec. Using this as building blocks to better understand how security works, the visualization that we create from this is the effective balance created between where an attack can occur, where the Controls are in place to manage an attack, and the limitations of the protective measures.

Another benefit of mathematical representation of an attack surface as Actual SPD level is that besides just showing where protection measures are lacking it can also show the opposite. Since it is possible to have more controls than one need this can be mathematically represented as more than 100. Whether a risk assessment may make this point seem impossible, the mathematical representation is useful for showing waste. It can be used to prove when money is being overspent on the wrong types of controls or redundant controls.

3.3 nSHIELD Metrics system deployment

Operational security metrics are the metrics we are most familiar with in our lives. When we measure the height, width, or length of an object we are using an operational metric. When we write the date, have a birthday, or ask the score of a game we are using operational metrics. An operational metric is a constant measurement that informs us of a factual count in relation to the physical world we live in. They are operational because they are numbers we can work with consistently from day to day and person to person.

The only real problem with operational metrics is the requirement for knowing how to properly apply the metric for it to be useful. The completion of a thorough security test has the advantage of providing accurate metrics on the state of security. As with the use of any metric. the less thorough the test, the less accurate the overall metric.

Less skilled or less experienced Analysts will also adversely affect the quality of the metric Therefore a successful security metric requires a test which can be described as measuring the appropriate vectors while accounting for inaccuracies and misrepresentations in the collected test data as well as the skills or

experience of the security professionals performing the test. Faults in these requirements result in lower quality measurements and false security determinations therefore the metric must also be simple enough to use without making it so simple that it tells nothing. Furthermore, a proper security metric must avoid the biases inherent in risk assessments by assuring measurements have integrity. These qualities have been combined to create the SPD level.

The SPD level is a scale measurement of the attack surface, the amount of uncontrolled interactions with a target, which is calculated by the quantitative balance between operations, limitations, and controls.

Having the SPD level is to understand how much of the attack surface is exposed. In this scale, 100 is perfect balance and anything less is too few controls and therefore a greater attack surface. More than 100 shows more controls than are necessary which itself may be a problem as controls often add interactions within a scope as well as complexity and maintenance issues.

The SPD level does not measure risk for an attack surface, rather it enables the measurement of it. It cannot say if a particular target will be attacked however it can say where on a target it will be attacked, what types of attacks the target can successfully defend against, how deep an attacker can get, and how much damage can be done. With that information it is then possible to assess the trusts (and risks) much more accurately.

The SPD level is actually multiple separate calculations of Porosity, Controls, and Limitations, that when combined will show the size of an attack surface in two practical ways. The first way is in a straight calculation. It is the calculation of the Delta (Δ), a number that describes the specific exposure of that target. This is useful for determining how a new person, thing, or process will change the operational security of a new scope or as a comparison between multiple, single targets. This is also the easiest way to see Perfect Security, the perfect balance between Porosity, Controls, and Limitations. The SPD level is displayed as a positive or negative number which shows how far away the target is from a perfect security balance. A positive delta shows too much is spent on controls in general or even if the overspending is on too much of one type of control. A negative delta shows a lack of controls or controls themselves with limitations which cannot protect the target adequately. This is a powerful tool for knowing exactly where and how resources are being spent to protect a particular target. However this is not how the SPD level is most useful; that is done best the second way.

The second practical way to display the attack surface is for understanding the big picture. This is represented as Actual SPD level. Where the Delta calculation is based on perfect balance, the Actual SPD level calculation uses the Delta but also includes additional and redundant controls to provide a metric more people friendly and familiar. Here the SPD level representation is similar to how people use percentages. The SPD level is calculated with a base 10 logarithm, which makes a more comprehensible representation. While the SPD level is still a balance, perfect balance is set at 100 and calculations are made in respect to that. This will allow most people to have a quick and easy overview of all the targets in a scope or of just a single target in relation to other targets. It is extremely flexible so multiple attack surfaces can be compared by Actual SPD level even if the scope or the targets are very different: a 95% SPD level of a scope with 1000 computer systems is comparable to a 95% SPD level of a scope with just 10 computer systems, which can be again compared to a building with a 95% SPD level. All three will provide the same information to a person that the protection of the target is 5% deficient and therefore exposed to attack. With this knowledge, one can begin to assess risk and determine what is exposed, what is left uncontrolled, and if that 5% is acceptable.

3.3.1 How to obtain SPD level

The minimum SPD level is made by the calculation of porosity which are the holes in the scope.

In this kind of approach we get what we know from what is there for a particular vector and you make no assumptions surrounding what is not there. In other words we count all that which is visible and interactive outside of the scope and allows for unauthenticated interaction between other targets in the scope. That becomes the porosity. This porosity value makes the first of 3 parts of the final SPD level value. The next part is to account for the controls in place per target. This means going target by target and determining where any of the 10 controls are in place such as Authentication, Subjugation, Non-repudiation, etc. Each

control is valued as 10% of a pore since each provides 1/10th of the total controls needed to prevent all attack types. This is because having all 10 controls for each pore is functionally the same as closing the pore provided the controls have no limitations. The third part of the SPD level is accounting for the limitations found in the protection and the controls. These are also known as “vulnerabilities”. The value of these limitations comes from the porosity and established controls themselves. With all counts completed, the SPD level is basically subtracting porosity and limitations from the controls.

Unfortunately, an unskilled analyst can provide the wrong information which will translate into a bad SPD level. This is a possibility, just like it’s possible a carpenter doesn’t measure a board right or a mechanic fails to read the gauges right. The world is full of what-if scenarios. Therefore the SPD level is designed to be minimally influenced by bad auditing or cheating by eliminating the direct scope size from the metric calculation.

However, no metric can be immune from fudging and the only way to assure the most accurate SPD level is to have multiple tests over time to make the counts and to be sure the auditor will take responsibility over the accuracy of the test.

It is possible to take a short-cut in testing and still make a representative SPD level. If you don’t mind the error margin because you only want to make a quick comparison, you can just calculate the Porosity which means counting the visible and accessible targets. For example, those who run vulnerability scanners can count porosity and limitations relatively easily and assign default controls for discovered services. Analysts can also create a checklist which offers default controls for different common solutions found. These are all shortcuts to reduce the time to calculation but will affect the overall SPD level with an unknown, but perhaps acceptable, error margin.

The end result is a calculation for Actual SPD level. It applies multiple controls of the same type to satisfy double-enforcement requirements like 2-factor Authentication. It also uses Log10 to reduce large numbers into human-manageable form. People generally like to work with smaller numbers and especially as percentages which are easier to visualize. For a small scope, the accuracy of using Log10 as a reduction technique is negligible. However, if you have a very large scope with many targets you may want to work with the very large numbers for greater accuracy. Additionally if you want to see the true balance where multiple controls of the same type are not measured, that calculation can be found under the heading of True Protection.

One important requirement in applying this approach is that Actual SPD level can only be calculated per scope. A change in channel, vector, or index is a new scope and a new calculation for Actual SPD level.

However, once calculated, multiple scopes can be combined together in aggregate to create one Actual SPD level that represents a fuller vision of the operational security all scopes. For example, a test can be made of Internet-facing servers from both the Internet side and from within the perimeter network where the servers reside. That is 2 vectors. Assume that, the Internet vector is indexed by IP address and contains 50 targets. The intranet vector is indexed by MAC address and is made of 100 targets because

less controls exist internally to allow for more collaborative interaction between systems. Once each test is completed and the SPD level is counted they can be combined into one calculation of 150 targets as well as the sums of each limitations and controls. This will give a final Actual SPD metric which is more complete for that perimeter network than either test would provide alone. It would also be possible to add the analysis from physical security, wireless, telecommunications, and human security tests in the same way. Such combinations are possible to create a better understanding of the total security in a holistic way.

We proposed as an easy implementation of this approach a SPD level spreadsheet calculator.

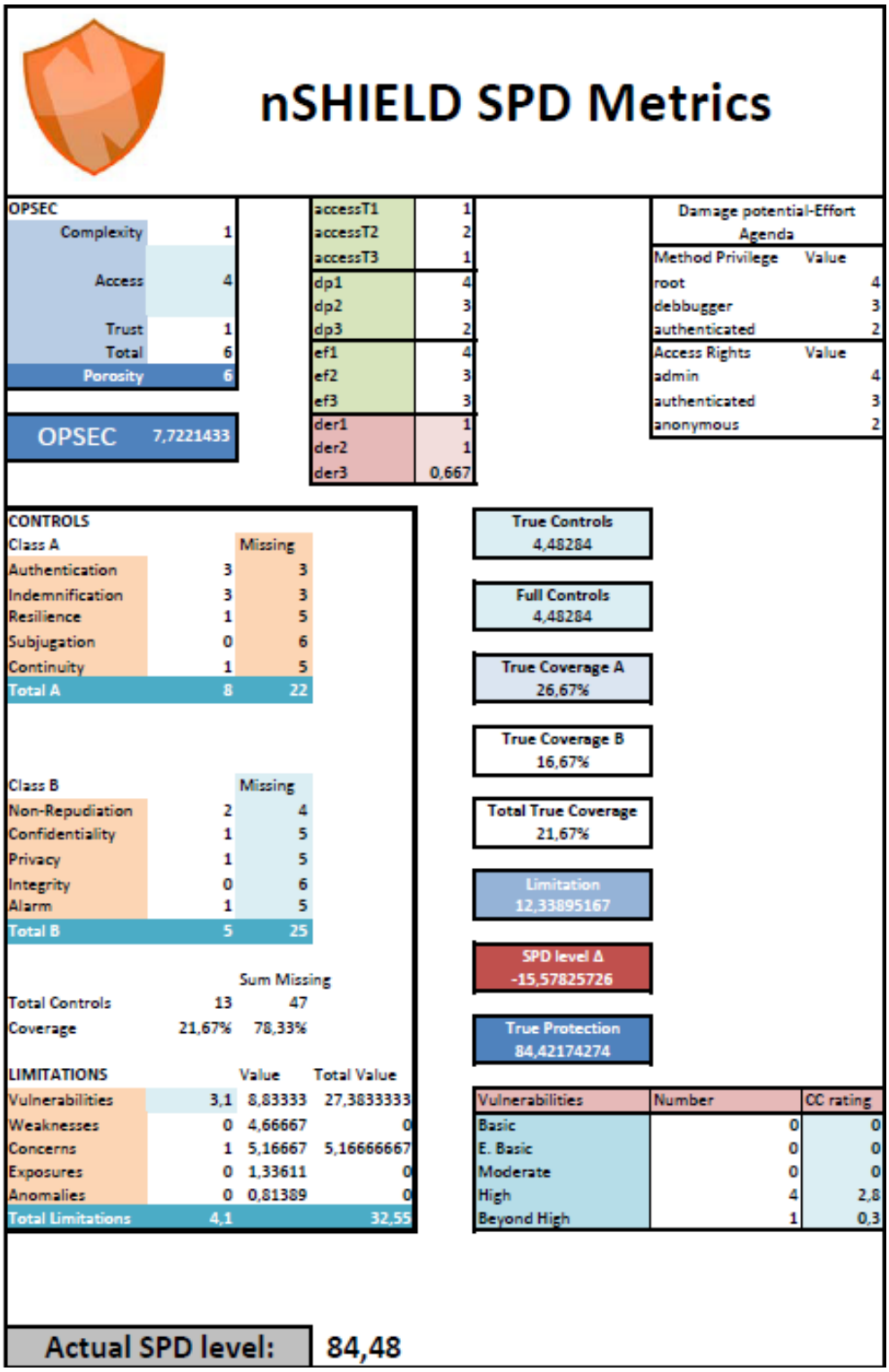


Figure 3-1: SPD level spread sheet calculator

The Analyst need only enter the values (obtained from the test data) into the empty, white boxes and the rest of the calculations will be handled automatically.

In the following paragraphs it is described the formulas and concepts use in the Actual SPD level spreadsheet implementation.

3.3.2 Turning Test Results into an Attack Surface Measurement

3.3.2.1 Operational Security

The measurement of the Attack Surface requires the measurements of Complexity, Trust, and Access relative to the scope. The number of targets in the scope that can be determined to exist by direct interaction, indirect interaction, or passive emanations is its complexity. As complexity is determined, its value represents the number of targets in the scope. Trust is any non-authenticated interaction to any of the targets. Access is the number of interaction points with each target.

Table 3-5: Operational Security measurement

Category		Description
1	**Complexity**	The number of targets in the scope. Count all targets by index only once and maintain the index consistently for all targets. It is generally unrealistic to have more targets visible than there are targets in the defined scope; however, it may be possible due to vector bleeds where a target which is normally not visible from one vector is visible due to a misconfiguration or anomaly. A HUMSEC audit employs 50 people; however, only 38 of them are interactive from the test vector and channel. This would make a complexity of 38.
2	**Access**	This differs from complexity where one is determining the number of existing targets. Here, the auditor must count each Access per unique interaction point per unique probe. In a PHYSSEC audit, a building with 2 doors and 5 windows which all open has an Access of 7. If all the doors and windows are sealed, then it is an Access of 0 as these are not points where one can gain entry. For a COMSEC audit of data networks, the auditor counts each port response as an Access regardless of how many different ways the auditor can probe that port. However, if a service is not hosted at that port (daemon or an application), then all replies instead come from the IP Stack. Therefore, a server that responds with a SYN/ACK and service interactivity to only one of the TCP ports scanned and with a RST to the rest does not have an Access count of 65536 (including port 0) since 66535 of the ports respond with the same response of RST from the kernel. To simplify, count only ports with service responses and only one IP Stack response regardless of the number of ports which can initiate this kind of interactivity. With HUMSEC audits, this is much more simplified. A person who responds to a query counts as an Access with all types of queries (all the different questions you may ask or statements made count as the same type of response on the same channel). Therefore, a person can only be an Access of 1 per channel and vector. Only a person who completely ignores the request by not acknowledging the channel is not counted.
3	**Trust**	This differs from complexity where one is determining the number of existing targets. Here, the auditor must count each Trust per unique interaction point per unique probe. In a PHYSSEC audit, a building with 2 internal doors separating rooms which open has a Trust of 2. If those doors are sealed then it is a Trust of 0 as these are not points where one can pass. For a COMSEC audit of data networks, the auditor counts each type of service forward or port forward as a Trust. With HUMSEC audits, a person who acts as a gateway to interact with other people or to access property is a trust per channel. Therefore, a person can only be a Trust of 1 per channel and vector. Only a person who does not comply with the Trust request is not counted.

3.3.2.2 Controls

The next step in calculating the SPD level is to define the Controls; the security mechanisms put in place to provide assurance and protection during interactions.

Table 3-6: Controls measurement

Category		Description
1	**Authentication**	Count each instance of authentication required to gain access. This requires that authorization and identification make up the process for the proper use of the authentication mechanism. In a PHYSSEC audit, if both a special ID card and a thumb print scan is required to gain access, then add two for authentication. However, if Access just requires one or the other, then only count one.
2	**Indemnification**	Count each instance of methods used to exact liability and insure compensation for all assets within the scope. A basic PHYSSEC example is a warning sign threatening to prosecute trespassers. Another common example is property insurance. In a scope of 200 computers, a blanket insurance policy against theft applies to all 200 and therefore is a count of 200. However, do not confuse the method with the flaw in the method. A threat to prosecute without the ability or will to prosecute is still an indemnification method-- however, it is with a limitation.
3	**Subjugation**	Count each instance for Access or Trust in the scope which strictly does not allow for controls to follow user discretion or originate outside of it. This differs from being a security limitation in the target since it applies to the design or implementation of controls. In a COMSEC data networks audit, if a log-in can be made in HTTP as well as HTTPS but requires the user to make that distinction, then it fails to count toward Subjugation. However, if the implementation requires the secured mode by default, such as a PKI internal messaging system, then it does meet the requirement of the Subjugation control for that scope. More simply, in HUMSEC, a non-repudiation process where the person must sign a register and provide an identification number to receive a document is under Subjugation controls when the provider of the document records the identification number, rather than having the receiver do so, to eliminate the recording of a false number with a false name.
4	**Continuity**	Count each instance for Access or Trust in the scope which assures that no interruption in interaction over the channel and vector can be caused, even under situations of total failure. Continuity is the umbrella term for characteristics such as survivability, load balancing, and redundancy. In a PHYSSEC audit, if it is discovered that an entry way into a store becomes blocked such that no alternate entry way is possible and customers cannot enter, that Access does not have Continuity. In a COMSEC data networks audit, if a web server service fails from high load and an alternate web server provides redundancy so no interactions are lost, this Access has Continuity.
5	**Resilience**	Count each instance for Access or Trust in the scope that does not fail open or provide new accesses upon security failure. In common language, to “fail securely”. In a PHYSSEC audit where 2 guards control Access to a door, if one is removed and the door cannot be opened by the remaining guard, then it has resilience.

Category		Description
		In a COMSEC data networks audit, if a web service requiring a log-in or password loses communication with its authentication database, then all Access should be denied rather than permitted in order to have resilience.
6	**Non-repudiation**	Count each instance for the Access or Trust that provides a nonrepudiation mechanism for each interaction to provide assurance that the particular interaction did occur at a particular time between the identified parties. Non-repudiation depends upon identification and authorization to be properly established for it to be properly applied without limitations. In a PHYSSEC audit, the Non-repudiation control exists if the entrance to a building requires a camera with a biometric face scan to gain entry and each time it is used, the time of entry is recorded with the ID. However, if a key-card is used instead, the Non-repudiation control requires a synchronized, time-coded camera to assure the record of the card-user's identity to avoid being a flawed implementation. If the door is tried without the key card, not having the synchronized camera monitoring the door would mean that not all interactions with the entryway have the Non-repudiation control and therefore does not count for this control. In a COMSEC data networks audit, there may be multiple log files for nonrepudiation. A port scan interacting at the IP Stack is recorded in one log while interaction with the web service is recorded to another log file. However, as the web service may not log the interactions from the POST method, the control is still counted; however, so is the security limitation.
7	**Confidentiality**	Count each instance for Access or Trust in the scope that provides the means to maintain the content of undisclosed interactions between the interacting parties. A typical tool for Confidentiality is encryption. Additionally, obfuscation of the content of an interaction is also a type of confidentiality, albeit a flawed one. In HUMSEC, however, a method of Confidentiality may include whispering or using hand signals.
8	**Privacy**	Count each instance for Access or Trust in the scope that provides the means to maintain the method of undisclosed interactions between the interacting parties. While "being private" is a common expression, the phrase is a bad example of privacy as a loss control because it includes elements of confidentiality. As a loss control, when something is done "in private" it means that only "the doing" is private but the content of the interaction may not be. A typical tool for Privacy is obscuring the interaction that is, having the interaction take place outside of the visibility of third parties. Confusion of the means of interaction as obfuscation is another method of applying the Privacy control. In HUMSEC, a method of Privacy may be simply taking the interaction into a closed room away from other people. In movies, we see techniques to create the Privacy control by setting two identical suitcases side by side, some type of incident to create confusion takes place, and the two people switch the suitcases in seemingly plain view.
9	**Integrity**	Count each instance for Access or Trust in the scope which can assure that the interaction process and Access to assets has finality and cannot be corrupted, stopped, continued, redirected, or reversed without it being known to the parties involved. Integrity is a change control process. In COMSEC data networks, encryption or a file hash can provide the Integrity control over the change of the file in transit. In HUMSEC, segregation of duties and other corruption-reduction mechanisms provide Integrity control. Assuring integrity in personnel requires that two or

Category		Description
		more people are required for a single process to assure oversight of that process. This includes that no master Access to the whole process exists. There can be no person with full access and no master key to all doors.
10	**Alarm**	Count each instance for Access or Trust which has a record or makes a notification when unauthorized and unintended porosity increases for the vector or restrictions and controls are compromised or corrupted. In COMSEC data networks, count each server and service which a network-based intrusion detection system monitors. Or, count each service that maintains a monitored log of interaction. Access logs count, even if they are not used to send a notification alert immediately, unless they are never monitored. However, logs which are not designed to be used for such notifications, such as a counter of packets sent and received, do not classify as an alarm as there is too little data stored.

3.3.2.3 Limitations

Finally, the limitations are verified where possible. The values of each Limitation are dependent on Porosity and Controls. This is different from the more common risk perspective where a vulnerability may be assigned a risk level based on what damage it can do, how easy it is to do, and the distance in range for the attack. Therefore the Limitation values are calculated based on the Porosity and Controls of the target they can be found on.

Table 3-7: Controls measurement

Category		Description
1	**Vulnerability**	Count separately each flaw or error that defies protections whereby a person or process can access, deny access to others, or hide itself or assets within the scope. In PHYSSEC, a vulnerability can be as simple as a glass door, a metal gate corroded by the weather, a door that can be sealed by wedging coins into the gap between it and its frame, electronic equipment not sealed from pests such as ants or mice, a bootable CD drive on a PC, or a process that allows an employee to take a trashcan large enough to hide or transport assets out of the scope. In HUMSEC, a vulnerability can be a cultural bias that does not allow an employee to question others who look out of place or a lack of training which leaves a new secretary to give out business information classified for internal use only. In COMSEC data security, a vulnerability can be a flaw in software that allows an attacker to overwrite memory space to gain access, a computation flaw that allows an attacker to lock the CPU into 100% usage, or an operating system that allows enough data to be copied onto the disk until it cannot operate anymore. In COMSEC telecommunications, a vulnerability can be a flaw in the pay phone system that allows sounds through the receiver to mimic coin drops, a telephone box that allows anyone to access anyone else's phone line, a voice mail system that provides messages from any phone anywhere, or a FAX machine that can be polled remotely to resend the last thing in memory to the caller's number. In SPECSEC, a vulnerability can be hardware which can be overloaded and burnt out by higher powered versions of the same frequency or a near

		frequency, a standard receiver without special configurations which can access the data in the signal, a receiver which can be forced to accept a third-party signal in place of the intended one, or a wireless access point dropping connections near a microwave oven.
2	**Weakness**	Count each flaw or error in the controls for interactivity: authentication, indemnification, resilience, subjugation, and continuity. In PHYSSEC, a weakness can be a door lock that opens when a card is wedged between it and the door frame, a back-up generator with no fuel, or insurance that doesn't cover flood damage in a flood zone. In HUMSEC, a weakness can be a process failure of a second guard to take the post of the guard who runs after an intruder or a cultural climate within a company for allowing friends into posted restricted spaces. In COMSEC data security, a weakness can be a log-in that allows unlimited attempts or a web farm with round-robin DNS for load balancing yet each system also has a unique name for direct linking. In COMSEC telecommunications, a weakness can be a PBX that still has the default administration passwords or a modem bank for remote access dial-in which does not log the caller numbers, time, and duration. In SPECSEC, a weakness can be a wireless access point authenticating users based on MAC addresses (which can be spoofed) or an RFID security tag that no longer receives signals and therefore fails "open" after receiving a signal from a high power source.
3	**Concern**	Count each flaw or error in process controls: non-repudiation, confidentiality, privacy, integrity, and alarm. In PHYSSEC, a concern can be a door lock mechanism whose operation controls and key types are public, a back-up generator with no power meter or fuel gauge, an equipment process that does not require the employee to sign-out materials when received, or a fire alarm not loud enough to be heard by machine workers with ear plugs. In HUMSEC, a concern can be a process failure of a guard who maintains the same schedule and routine or a cultural climate within a company that allows employees to use public meeting rooms for internal business. In COMSEC data security, a concern can be the use of locally generated web server certificates for HTTPS or log files which record only the transaction participants and not the correct date and time of the transaction. In COMSEC telecommunications, a concern can be the use of a FAX machine for sending private information or a voice mail system that uses touch tones for entering a PIN or password. In SPECSEC, a concern can be a wireless access point using weak data encryption or an infrared door opener that cannot read the sender in the rain.
4	**Exposure**	Count each unjustifiable action, flaw, or error that provides direct or indirect visibility of targets or assets within the chosen scope channel. In PHYSSEC, an exposure can be a window which allows one to view assets and processes or a power meter that shows how much energy a building uses and its fluctuation over time. In HUMSEC, an exposure can be a guard who allows all visitors to view the list of names on the sign-in sheet or a company operator who informs callers that a particular person is out sick or on vacation. In COMSEC data security, an exposure can be a descriptive and valid banner about a service (disinformation banners are not exposures) or an ICMP echo reply from a host. In COMSEC telecommunications, an exposure can be an automated company

		directory sorted alphabetically, allowing anyone to cycle through all persons and numbers, or a FAX machine that stores the last dialed numbers. In SPECSEC, an exposure can be a signal that disrupts other machinery or an infrared device whose operation is visible by standard video cameras with night capability.
5	**Anomaly**	Count each unidentifiable or unknown element which cannot be accounted for in normal operations, generally when the source or destination of the element cannot be understood. An anomaly may be an early sign of a security problem. Since unknowns are elements which cannot be controlled, a proper audit requires noting any and all anomalies. In PHYSSEC, an anomaly can be dead birds discovered on the roof a building around communications equipment. In HUMSEC, an anomaly can be questions a guard asks which may seem irrelevant to either the job or standard small talk. In COMSEC data security, an anomaly can be correct responses to a probe from a different IP address than was probed or expected. In COMSEC telecommunications, an anomaly can be a modem response from a number that has no modem. In SPECSEC, an anomaly can be a local signal that cannot be properly located nor does it do any known harm.

3.3.3 The Operational Security Formula

The SPD level is derived from three categories defined within the scope: Operational Security, Controls and Limitations. In order to begin, we must first aggregate and associate all of our input information into the appropriate categories for each input variable.

The SPD level equation requires that each of the categories be assigned a logarithmic base value to scale the three factors of Actual SPD level in accordance with the scope.

3.3.3.1 Porosity

Operational Security, also known as the scope's Porosity, is the first of the three factors of Actual SPD level that should be determined. It is initially measured as the sum of the scope's complexity (P_C), access (P_A), and trust (P_T).

$$OpSec_{sum} = P_C + P_A + P_T$$

When calculating the SPD level it is however necessary to determine the Operational Security base value, $OpSec_{base}$. The Operational Security base value is given by the equation

$$OpSec_{base} = \log^2(1 + 100 \times OpSec_{sum}).$$

Since the logarithm of 0 is not defined in the calculation we needed to include the 1+100 here. The log of 1 is 0. So if we have 0 Porosity and want to express this lack of interaction as perfect SPD level of 100 then we needed to add +1 to the equation. Without the 1+100 we would have undefined numbers in the case that the sums of any of those factors are 0. This is required by the methodology because the absence of interactions represents perfect security and therefore the logarithm should equal 0 to provide the 100.

3.3.4 The Controls Formula

The next step in calculating the SPD level is to define the Loss Controls; the security mechanisms put in place to protect the operations. First the sum of the Loss Controls, sum LCsum , must be determined by adding together the 10 Loss Control categories.

Table 3-8: Loss Control categories

Controls	**Class A**	Authentication	LC_{Au}
		Indemnification	LC_{Id}
		Resilience	LC_{Re}
		Subjugation	LC_{Su}
		Continuity	LC_{Ct}
	Class B	Non-Repudiation	LC_{NR}
		Confidentiality	LC_{Cf}
		Privacy	LC_{Pr}
		Integrity	LC_{It}
		Alarm	LC_{Al}

Thus the Loss Control sum $LCsum$ is given as

$$LCsum = LCAu + LCId + LCRe + LCSu + LCCt + LC\ NR + LCCf + LCPr + LCIt + LCAl$$

3.3.4.1 Missing controls

Given that the combination of each of the 10 Loss Controls balance the value of 1 OpSec loss (complexity, access, trust) it is necessary to determine the amount of Missing Controls, MC_{sum}, in order to assess the value of the Security Limitations. This must be done individually for each of the 10 Loss Control categories. For example, to determine the Missing Controls for Authentication (MC_{Au}) we must subtract the sum of Authentication Controls (LC_{Au}) of the scope from the $OpSec_{sum}$. The Missing Controls can never be less than zero however.

The equation for determining the Missing Controls for Authentication (MC_{Au}) is given by

$$\text{IF } OpSec_{sum} - LC_{Au} = 0$$

$$\text{THEN } MC_{Au} = 0$$

$$\text{ELSE } MC_{Au} = OpSec_{sum} - LC_{Au}$$

The resulting Missing Control totals for each of the 10 Loss Controls must then be added to arrive at the total Missing Control value ($MCsum$) as seen below.

$$MCsum = MCAu + MCId + MCRe + MCSu + MCCt + MC\ NR + MCIt + MCPr + MCCf + MCAl$$

3.3.4.2 True Controls

True Controls (*TCsum*) is the inverse of Missing Controls which means the True Controls for each individual control also need to be calculated before the results can be tallied into *TCsum*.

The equation for determining the True Controls for Authentication ($TCAu$) is given by

$$TCAu = OpSecsum - MCAu$$

The resulting True Control totals for each of the 10 Loss Controls must then be added to arrive at the total True Control value (*TCsum*) as seen below.

$$TCsum = TCAu + TCId + TCRe + TCSu + TCCt + TC\ NR + TCIt + TCPr + TCCf + TCAl$$

True Controls are used to measure the ideal placement of controls. The base value also helps to eliminate the influence of a disproportionate placement of controls on security. The True Controls base (TC_{base}) value is given as:

$$TCbase = log2(1 + 100 \times (OpSecsum - MCsum\ x\ 0.1))$$

Based on the same idea as True Controls, True Coverage (*TCvg*) can be used to measure the percentage of controls in place regarding the optimal amount and placement of controls. True Coverage is then derived using the Missing Control totals and the following equation:

$$\text{IF } OpSec_{sum} \leq 0$$

$$\text{THEN } TCvg = 0$$

$$\text{ELSE } TCvg = 1 - \frac{MC_{sum}}{10 \times OpSec_{sum}}$$

3.3.4.3 Full Controls

Full Controls, on the other hand, take into account all controls in place regardless of a balanced distribution. This value is important for measuring the worth of two-factor authentication, for example, and other instances of defense in depth for the same complexity, access or trust. The Full Controls base (FC_{base}) value is given as:

$$FC_{base} = \log^2(1 + 10 \times LC_{sum})$$

3.3.5 The Limitations Formula

Next, the Limitations are individually weighted. The weighting of the Vulnerabilities, Weaknesses and Concerns are based on a relationship between the Porosity or $OpSec_{sum}$, the Loss Controls and in the case of Exposures and Anomaly the existence of other Limitations also plays a role. An Exposure or Anomaly poses no problems alone unless a Vulnerability, Weakness or Concern is also present. Think of an Exposure like a pointer. If there is a pointer that goes nowhere, or in this case doesn't lead to anything exploitable (Vulnerability, Weakness, Concern) and all Controls are accounted for, then at the time of the test the Exposure has no effect on security and thus has no value in the SPD level.

The following value table is used to calculate the $SecLim_{sum}$ variable, as an intermediate step between the Security Limitation inputs and the $SecLim_{base}$ variable, which is the Security Limitations basic input for the rav equation.

$$\text{IF } OpSec_{sum} \leq 0$$

$$\text{THEN } MCvg = 0$$

$$\text{ELSE } MCvg = \frac{MC_{sum} \times 0.1}{OpSec_{sum}}$$

Table 3-9: SecLimsum variable calculating table

Input	Weighted Value	Variables
Vulnerability L_V	$\frac{OpSec_{sum} + MC_{sum}}{OpSec_{sum}}$	MC_{sum}: sum of Missing Controls
Weakness L_W	$\frac{OpSec_{sum} + MC_A}{OpSec_{sum}}$	MC_A: sum of Missing Controls Class A
Concern L_C	$\frac{OpSec_{sum} + MC_B}{OpSec_{sum}}$	MC_B: sum of Missing Controls Class B
Exposure L_E	$\frac{((P_C + P_A) \times MCvg + L_V + L_W + L_C)}{OpSec_{sum}}$	P_C *: sum of Complexity* P_A *: sum of Accesses* *MCvg: Percent Missing Coverage*
Anomaly L_A	$\frac{(P_C \times MCvg + L_V + L_W + L_C)}{OpSec_{sum}}$	P_C*: sum of Complexity* *MCvg:Percent Missing Coverage*

3.3.5.1 Security Limitations Base

$SecLim_{sum}$ is then calculated as the aggregated total of each input multiplied by its corresponding weighted value as defined in the table above.

$$SecLim_{sum} = \left(L_V \times \frac{(OpSec_{sum} + MC_{sum})}{OpSec_{sum}}\right) + \left(L_W \times \frac{(OpSec_{sum} + MC_A)}{OpSec_{sum}}\right) + \left(L_C \times \frac{(OpSec_{sum} + MC_B)}{OpSec_{sum}}\right) + \left(L_E \times \frac{((P_C + P_A) \times MCvg + L_V + L_W + L_C)}{OpSec_{sum}}\right) + \left(L_A \times \frac{(P_C \times MCvg + L_V + L_W + L_C)}{OpSec_{sum}}\right)$$

The Security Limitations base equation is given as:

$$SecLim_{base} = \log^2(1 + 100 \times SecLim_{sum})$$

3.3.6 The Actual SPD level Formula

This is the final part for using all previous calculations in three different ways.

3.3.6.1 Actual SPD level Delta

The Actual SPD level Delta is useful for comparing products and solutions by previously estimating the change (delta) the product or solution would make in the scope. We can find the Actual SPD level Delta, *ActSPDL*Δ , with the formula:

$$ActSPDL\Delta = FC_{base} - OpSec_{base} - SecLim_{base}$$

3.3.6.2 True Protection

Can be used as a simplified expression for the optimal coverage of a given scope where 100 signifies an optimal relationship between the Porosity, True Controls and Security Limitations. True Protection is given as:

$$TruPro = 100 + TC_{base} - OpSec_{base} - SecLim_{base}$$

3.3.6.3 Actual SPD level

To measure the current state of operations with applied controls and discovered limitations, a final calculation is required to define Actual SPD level. As implied by its name this is the whole security value which combines the three values of operational security, controls, and limitations to show the actual state of security.

Actual SPD level (total), *ActSPDL*, is the true state of security provided as a hash of all three sections. A SPD level of 100 signifies a perfect balance of security however the Actual SPD level is not a true percentage value.

Scores above 100 are also possible which signifies that the tested scope has more controls implemented than necessary which could also be proof of overspending. The final SPD level equation for Actual SPD level is given as:

$$ActSPDL = 100 + ActSPDL\Delta - \frac{1}{100} \times (OpSec_{base} \times FC_{base} - OpSec_{base} \times SecLim_{base} + FC_{base} \times SecLim_{base})$$

4 nSHIELD Security Multi Metric Approach – Towards A Security Metrics Heterogeneity Solution

4.1 Introduction

Security Measurement of complex systems is a challenging task since devices deposited across System of Systems (SoS) environments are heterogeneous and imply an security interoperability effort in order to enable a common security and resilient measurement language. Moreover, systems require more features beyond security concept and demand to preserve privacy and claim for dependable structures in order to seek for a holistic and aggregated security and safety view. This paper faces this trial by composing metrics through an evolutionary-fuzzy approach taking into account multi heterogeneous metrics and system engineers' expertise as inputs and SPD (Security, Privacy and Dependability) Matrix as dashboard-based outcome.

4.2 An Evolutionary-Fuzzy Approach towards Multi-Metric Security Risk Assessment in Heterogeneous System of Systems

In the security field one of the most sounding paradigms resides in the security interoperability within a heterogeneous device landscape. In this scenario each of such devices performs with different protocols and scales. Interoperability is by itself a challenge for engaging the security paradigm as a built-in approach. Many efforts have been devoted to define security patterns [4], [5], [6], [7] so as to develop and implement a understandable security backdrop.

Unfortunately, this background is usually quite complex. In this context, the setup elaborated by the nSHIELD initiative[7] proposes a System of Systems representing a set of assets: sub-systems, software components, protocols and devices and boards. These elements are configured along node, network and middleware-overlay layers. On the other hand, risks are denoted as the probability that a threat can become real impacting in a vulnerability of one or more components from those listed above. Threats are numerous and can be predictable or unpredictable. Those which can be predicted may be guarded by metrics, whereas those which are unpredictable could be challenged by composable, heuristics techniques. Therefore, metrics will be mapped to threats and problems to be measured. For example, if a problem of a given system is the network latency, the network latency must be measured. This obvious affirmation is the key for starting the process for selecting the correct metrics for any heterogeneous system. Consequently, risk analysis is the first process for creating a set of security metrics in scenarios with a diverse device spectrum.

This paper joins this research trend by outlining and sketching a novel multi-metric approach for heterogeneous systems capable of measuring risks impacting on different vulnerabilities in an inter-operable and normalized fashion. This practical approach requires understanding and identifying beforehand security needs of the system at hand, as well as those of its constituent components. In other words, current vulnerabilities and threats must be identified so as to discriminate those vulnerabilities and threats more likely to take place in the future. To this end, this paper elaborates in advance of several evident yet relevant observations: 1) systems' complexity yields an accordingly complex and also non-interoperable security management; and 2) without loss of generality, metrics extracted in this approach are restricted to operations (but could be extended to organizational or business indicators) and aggregate SPD metrics to the operational ones for consolidating an holistic view of Security integrated from engineering process to operational one. 3) Optimization through an expert system adjusted by an evolutionary approach for showing to system expert an easy and understandable mechanism for fine-tuning metrics measurements and systems status outputs.

[7] www.newshield.eu

4.2.1 nSHIELD SPD Metrics

The meaning of metric is understood by a business understandable concept, this is that metrics will be similar to human understanding because metric need to be understood by business and technical engineers operating systems.

Information SPD metrics are seen as an important factor in making sound decisions about various aspects of security (but also dependability), ranging from the design of SPD architectures and controls to the effectiveness and efficiency of SPD operations. SPD **metrics strive to offer a quantitative and objective** basis for security assurance.

This paper will analyse and mix both traditional operational metrics and SPD metrics since it is applied in an Industrial Systems of Systems (SoS) environment. Metrics in industrial operation (metrology) have always been regulated and certified by a higher authority. In the IT area this has been applied less rigorously since safety has always prevailed over security. However, SoS environment requires both security and safety (part of dependability) requisites since threats could proceed from both virtual and real (unpredicted/failure threats) worlds.

NSHIELD SPD metrics is the first attempt to correlate operational and SPD metrics in order to develop business continuity approach for industrial sectors. It is important to highlight that main control systems such as SCADAS or ICSs (Industrial Control Systems) depend not only on the operational process (which is linked directly to business) but also it is getting more and more dependant to robustness, resilience and security factors that preserve operation from malicious attacks and large failures. Dependability concept guarantees this fact: dependability mechanisms tackle availability (threats against DDoS) which is the most important feature for industrial operation. Security threat also can attempt against availability but also against integrity (imagine man-in-the-middle attack for value modification from smartmeters to smart-concentrators that would cause a poorer profit to electric vendors and an unequal operation for Electric DSOs (Distributed System Operators) when balancing energy offering & demand.) Privacy notion involves confidentiality and anonymity and it is getting more and more importance as big data gets involved within industrial and large organisation settings. Therefore, NHSIELD metrics are business continuity oriented, heterogeneous but measurable, understandable by the human being due to its fuzziness toward comprehensible lexica and composable since inputs are aggregated through an expert system.

This makes NSHIELD system be compliant to Security by Design (SbD) principles[8] nSHIELD SPD Metrics apply to security, privacy and dependability built-in concepts and functionalities within the whole engineering process. Furthermore, SDP functionalities are not seen as a final patch but as an intrinsic conceptualisation of the whole and holistic engineering and operation procedure. SPD metrics are extracted from a set of SPD requirements and risks statements within SoS scenarios.

nSHIELD multi-metrics approach follows a quantitative focus. This approach provides a metric template for metrics identification and gathering process. In nSHIELD project metrics first specification approach more than 60 metrics were identified and structured in layers {SPD, Lx}. Metrics that were identified are heterogeneous and sometimes overlap in different layers. The result of measurements according these metrics has to be described quantitatively.

Table 4-1: Extract of Metrics in D2.5

Nº	Layer	Name	Description	Formula/ Implementation
1	Node	Code execution	Verification that only authorized code (booting, kernel, application) runs on the system	if (SE or TPM) then run TPM; Verify corruption;

[8] http://www.ipc.on.ca/images/resources/pbd-convergenceofparadigms.pdf

2	Node	Data freshness	Ensure that received data follows a monotonically increasing timeline and that the data freshness of each packet lies within acceptable, application-specific limits	1) Strong session generators at node level 2) if (slot>lapse_time) refresh_data ();

3	Node	Digital Signatures	Ability of the node to verify digital signatures even in cases where a trusted third party is not available	if (certificate) then
				if (root_certificate) then return 2
				else return 1
				return 0

4	Node	Policy updates	Security policy updates only accept by authorized (white-listed) sources	if (slot>lapse_time) then if authorised (UserID) policyUpdate();
5	Node	Low power mode	Node able to enter into low power state without compromising its security.	
6	Node	TPM Context based encryption keys	Boolean value depending on whether context-based encryption keys functionality is implemented on the node.	if (Context) then return 1 else return 0
7	Node	Key parameterization	Key size parameterization options mapping application requirements (SHORT, MEDIUM or LONG-TERM security)	if key_parameterisation then return 1 else return 0;
8	Node	Secure Key Distribution	Secure low-cost key distribution mechanism with parameters (e.g. algorithm, key length, usage etc.) defined by the security policy requirements, taking into consideration any restrictions that participating parties impose	Compose_parameters
9	Node	Third key management	Support for third-party key management services	Compose_parameters
10	Node	Security context establishment	Node should allow security context establishment and sharing, allowing more efficient keys or key material to be exchanged, thereby increasing the overall performance and security of subsequent communications	if new_SC>currentSC then forceSC(new_SC) else efficientSC(new_SC);

11	Node	Tamper resistant	Resilience to tampering, micro-probing and reverse-engineering. Aim of this specific metric is to “detect” and report the presence of tamper-resistance provisions on the node.	if (SE is certified by FIPS 140) then return 1 else return 0;
12	Node	TPM Remote Attestation	TPM Remote Attestation functionality to ensure integrity of node prior to resource allocation	if integrity_check(TPM_RA) then return 1; else return 0;
13	Node	Virtualisation	Virtualized hardware redundancy mechanism	if harfware_virtualised then return 1; return 0;
14	Node	Runtime Reconfiguration	Runtime modification of the device’s functionality during both normal operation or fault conditions, through either hardware or software changes	if runtime_re then else return 0; return 1
15	Node	Alternative power supply sources	Support for alternative power modes, depending on the specific application and environmental conditions (e.g. vibration generator, micro-solar cells)	if alternative_PSM then return 1 else return 0;
16	Node	Dependable key distribution mechanism	Support of secure and dependable low-cost key distribution mechanisms for initialization or re-keying	Compose_parameters
17	Node	Privacy-centric physical/tamper resilience	No compromise in the privacy of contained sensitive information in the case of a malicious user gaining physical possession of the device.	if (SE is certified by FIPS 140) then return 1 else return 0;
18	Node	ECC Authentication	Optimized hardware implementation of an ECC-based public-key authentication algorithm	if (SE is optimised) then return 1; else return 0;
19	Node	storage of private information	Provisions to ensure the long term storage of private information, guaranteeing confidentiality of said information even under fault conditions	if (data_stored is encrypted && backup_process) then return 1; else return 0;
20	Node	Anonimity and location privacy	Privacy-aware management of location and other sensitive personal information, utilizing secure storage and sanitization mechanisms to be applied to such information prior to	if anonamity_on then return 1; else returno 0;

			transmission	
21	Node	Privacy accross trust domains	Security token exchange to enable the issuance and dissemination of credentials within different trust domains.	if crediential comliant to num_domain && exchange=valid then return 1; else return 0;
22	Node	eNetwork/Hybrid Network Compatibility	Support for switching between infrastructure-centric and ad-hoc networks on demand, in order to adapt continually to changes in the physical environment	if (infrastructure_topology_changeable) then return 1; else return 0;
23	Node	Flexible key distribution mechanism	Flexible and secure low-cost key distribution mechanism for initialization or re-keying, allowing the distribution of authentic public keys via insecure channels, either according to a pre-defined schedule or ad-hoc, while adhering to policy requirements as well as requirements participating parties impose	Compose_parameters
24	Node	Conflict resolution between policy domains	In case of nodes' communication between different policy domains, mechanisms facilitate the communication and resolve this conflict with the minimum processing and communication overhead	if !policy_matching (policy1, policy2,…) then result 0; else result 1;
25	Node	Dynamic security behaviour	Support for security behaviour changes based on the dynamic change of policy requirements without requiring to reprogram or shut down the node	if policy_changed then return dynamic(policy1, policy2, ..))
26	Node	Lightweight embedded operating system	Low resource requirements Operating System. Refers to the presence/installation of a lightweight O/S running on the device.	if (OSweight < lightweight) return 1; else return 0;
27	Node	SCA protection based on EM emissions	Inclusion of interfaces to monitor the nodes own EM emissions and modify its functionality accordingly, to protect its assets against SCA.	if (SCAProtection) then return 1; else return 0;

28	Node	Location awareness	Inclusion of interfaces that enable location-based functionality	if (location_enabled) return 1 else return 0;
29	Node	Situation and context awareness	Inclusion of interfaces that enable the provision of situational-aware and context-aware services and SPD.	if Enable_situational_context(current_context, num_context) then return 1; else return 0;
30	Network	Data confidentiality	Checks whether data exchange confidentiality is enforced and not only provided, so that to be in line with the CC requirements	if (data_encryption_enforcement is enabled) then return 1; else return 0;
31	Network	Data integrity	Checks whether data exchange integrity is enforced	if (data_signature_enforcement is enabled) then return 1; else return 0;
32	Network	Secure Routing	Checks whether secure routing is in operation (as required by the system) and verifies routing information on each packet	if (secure_routing is enabled) then verify_route(packet);
33	Network	Dependable authentic key distribution mechanisms	Support of secure and dependable low-cost key distribution mechanisms for initialization or re-keying	Compose_parameters
34	Network	k-anonymity	The goal of this metric is to make a user's identify information in-distinguish with other k-1 users	recursive_anonym_identification (ID, K);
35	Network	L-diversity	This metric is used in cooperation with k-anonymity. The metric adds another dimension L to incorporate with k. L is a set of distinct locations. Thus, we consider the k users in L distinct locations.	recursive_anonym_identification (ID, K, L);
36	Network	Time of confusion	Measures how long it will take until an attacker will become confused about a subjects track as the subject seeks to obfuscate its location by omitting measured samples	confusion_time(K, L);

37	Network	Network Delay	This is a performance metric used for measuring the delay induced by a node in retransmitting incoming data. This metric is important in cases where the node acts as a relaying node, where retransmission of incoming packets without extra delays is important for the overall network performance. Message relaying increases network latency and therefore network topology, e.g. a mesh or cluster-tree topology of a WSN, and the number of nodes significantly affects it.	$DR_{acb} = (D_{ac} + D_{bc}) / D_{ab}$
38	Network	Network Latency	This is a performance metric used for measuring the network latency which shall be kept low. This metric applies to the nSHIELD architecture as opposed to a single node where relaying delays due to routing decisions (static or dynamic) are induced. It quantifies the mean time needed for reaching a node.	scenario specific;
39	Network	Network Information Capacity	This is a performance metric used for measuring the network's capacity, which shall be large enough to allow the necessary traffic to go through. As a rule of thumb, at normal operation, the traffic should be about 60-70% of the network's capacity, so as to avoid bottlenecks when there will be traffic peaks.	C(P,T,I) = min {1..n} {C(Ln,T,I)},
40	Middleware	Discovery frequency	Amount of discovery events per protocol per unit time	N_{disc} / sec
41	Middleware	Metric – Discovery statistics	Discovery service related availability statistics: - Queue length (momentarily count of outstanding discovery requests) - Thread count (momentarily count of threads serving discovery requests)\ - Used memory (amount of heap memory reserved by the service - Wait time (time from arrival to processing, averaged for all requests per interval) - Processing time (time from	Five integer fields: QL: queue length [-] TC: thread count [-] UM: used memory [bytes] WT: wait time [us] = sum(wait time(i))/count(i) for i in requests PT: processing time [us] = sum(processing time(i))/count(i) for i in requests

			start of processing to sending response, averaged for all requests per interval)	
42	Middleware	Metric – Composition statistics	Composition service related availability statistics: • Queue length (momentarily count of outstanding composition requests) • Thread count (momentarily count of threads serving composition requests) • Used memory (amount of heap memory reserved by the service • Wait time (time from arrival to sending, averaged for all requests per interval) • Response time (time from start of sending to receiving response, averaged for all requests per interval)	• QL: queue length [-] • TC: thread count [-] • UM: used memory [bytes] • WT: wait time [us] = sum(wait time(i))/count(i) for i in requests • RT: response time [us] = sum(response time(i))/count(i) for i in requests
43	Middleware	Failed Discovery Request	Accumulated count of failed discovery requests per protocol	N_{fail} (accumulated per protocol)
44	Middleware	Rejected Discovery Requests	Accumulated count of rejected discovery requests per protocol	N_{reject}
45	Middleware	Composition response time	Average delay of response from an entity when answering composition requests	$Sum(T_{comp}) / N_{comp}$
46	Middleware	Failed Composition Requests	Accumulated count of failed composition requests per protocol	N_{fail}
47	Middleware	Blacklist/whitelist additions and removals	Accumulated count of the following attributes of the DOS / DDOS protection scheme: • Blacklist additions • Blacklist removals • Whitelist additions • Whitelist removals	• Blacklist additions • Blacklist removals • Whitelist additions • Whitelist removals
48	Overlay	Average Reputation	The mean value of all reputations of agents from a specific type. It is the correct rating that an agent should receive for each transaction	MEAN R_i
49	Overlay	Reputation bias	The difference between the average reputation and the actual ratings	| Average_reputation – Ri |
50	Overlay→ Network	Transaction rate	Is the portion of transactions completed successfully compared with all initiated	Successful completed transactions/initiated transactions

			transactions	
51	Overlay→ Network	The profit of each agent type	The types of agents include: • Evil or malicious • Disturbing • Selfish • Honest It is the profit that each agent type can gain according to its behaviour. The profit is higher if an agent cheats on another agent.	$p = \frac{1}{o+2} v$
52	Overlay	Uptime	This metric provides uninterrupted system availability	Uptime = (Monitoring_Time - DownTime) / Monitoring_Time
53	Overlay	Attack surface	Used to count the number of data inputs to an overlay node	Attack surface = Number of incoming connection / Number of monitoring nodes
54	Overlay → Middleware	Failed authenticati on	Measure the percentage of fail authentication attempts	Failed authentication = Authentication failures / Total number of authentications
55	Overlay	Detection accuracy	Measure of the detection accuracy	Accuracy = Undetected attacks / Total number of detected attacks
56	Overlay	Total attack impact	Measure attacks' impact	Total Attack Impact = Nodes attacked / Total number of nodes

Therefore a sub-set of quantitative oriented metrics have been selected from this main group and some other quantitative oriented ones have been added taking into account similarities related to problems and requirements.

Table 4-2: Second quantitative round Metrics

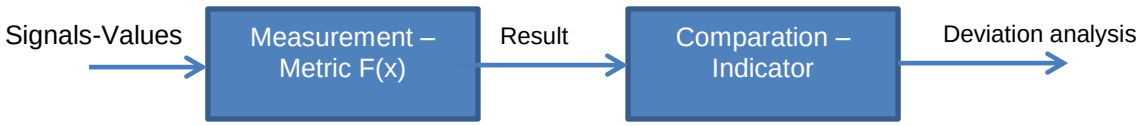
Nº	Layer	Name	Description	Formula/implementation
1	Overlay	Balanced attack surface	It is the minimal impact on the total attack surface that provides valuable information	$BATS = \frac{\sum_{n=1}^{k} R_{k*} \lvert V\_nodes_k \rvert}{\lvert Tot\_nodes \rvert}$
2	Overlay → Network	Attack Deepness	This metric indicates how deeply the attack can affect the e ntire system once one node is attacked and start	$ATD = \frac{\lvert Impacted\_nodes \rvert}{\lvert Tot\_nodes \rvert}$.

			affecting other nodes.	
3	Overlay → Network	Balanced Attack deepness	Fine-tuned ATD balanced	$BATD = \frac{\sum_{n=1}^{n=k} Rel_k * \lvert Impacted\_nodes_k \rvert}{\lvert Tot\_nodes \rvert}$
4	Overlay	System Immunity Level	Measure of level of protection of the system against a given direct target attack. Propagation attack have other metric related to attack deepness	SIL = 1 -"Atack Surface"
5	Overlay → Network	Attack Escalation Speed	Measures the speed (how fast) the attack is consolidated	AES = impacted_nodes/DifT
6	Overlay → Network → Node	Core Nodes Atacks Escalation Speed	measure of the speed depending of the relevance of different nodes	CNATES = Core_nodes_impact/difT
7	Overlay	Node Unplanned Downtime impact	Economic damage of downtown	NUDI = (node_Relevance_factor * K€)*Iterativityfactor
8	Overlay	Total Unplanned Downtime Impact	it is the total NUDI	$TUDI = \sum_{i=1}^{i=n} NUDI_i$
9	Node	Support Response Time	Mean time taken by the security team to detect the attack and start the mitigation procedure	SRT = Detecting mean time for mitigation. Scenario dependent
10	Node	Node Mean Time Recovery	Mean time needed to restore the attacked node	NMTR scenario dependent
11	Node	System Mean Time to Recovery	It is the mean time needed to completely sanitise the entire system	SMTR = scenario dependent
12	Network → Node	Vulnerability Density	Incidence rate of security breached in the nodes	VD = n_vulnerability/tot_nodes
13	Network → Node	Weighted Vulnerability Density	VD weighted for providing relevance	WVD= W*VD

This table, which reflects quantitative solutions (each metric has each own mathematical formula for a cardinal number) can be extended with new metrics, derived from each of the use cases. The porpoise of this paper is not to identify all the existing and future metrics but to manage thos identified in the correct and optimised way.

Scenario: SoS (System of Systems) and operation experience based- human factor

Identified metric will be applied in real scenarios described in WP7: avionics and railway. These scenarios are composed of systems of systems where human factor is still important in order to determine main indicators. The challenge to face in these scenarios for setting up multi metric approach is to determine indicators values.



Indicators values at first stage will be set by operation/security manager. However, these indicators will be improved by maximising function that maps the compared result with respect to the indicator and the real and experienced status of the systems taking into account the opinion of the operation and security engineers.

4.2.2 PROCEDURE

The following section describes the procedure for the multi-metric approach adoption.

Step 0 – Select correct metrics for a particular scenario and analyse its domain and regression function

Scenario owners will select metrics according to scenario requirements and risks. NSHIELD approach covers more than 60 types of SPD metrics structured in 4 layers: node, network, middleware and overlay. Scenario experts must conclude which metrics refer better to their business operation. Some might prioritise integrity to availability, some other reliability to privacy, etc.

Step 1 - Normalisation

Each of the metrics identified previously have different measures units and values range. This means that it needs to be normalised somewhat in order to have a common value range domain. This is important in order to define a common understanding value range for comparing them. Metric measure different things and therefore outcomes can be raised in different units (i.e. seconds, KW, º, etc…) and different range of numbers (i.e. from some discrete numbers such as key pars length – 64, 128, 256, 512 to more analogical ones such as time lapse in nanoseconds.)

For instance, let's suppose that network latency has a positive region (correct values) between [4..10] milliseconds. SPD values will be correct if all values are inside this range. This axiom shall only be valid in a particular domain concerning a concrete business process.

Let´s define the valid range for this problem: if normal range is between 4 and 10 millisecond we can assume that the valid range for network latency is of [0..10] ms.

Summarising:

Time	0	1	2	3	4	5	6	7	8	9	10	11	15	25
S-Level	0	2	4	6	8	11	14	17	20	23	26	32	44	58

At this stage we must say that Time is an objective and measurable concept and S Level (Security Threat Level) is a subjective but experience-earned concept that nSHIELD system operator will set up. The same will happen for P-Level and D-Level (related to privacy and dependability. S-Level represents the security threat level that will be limited by its valid boundaries and by its normal performance region.

These indicators (S-Level, P-Level, D-Level) must be determined by expert engineers and system operators who have the operational experience. This approach will take into account engineer's experience but it also will learn in order to set up the best indicators for a metric and groups of metrics.

S-Level (Security Threat Level) concerning first time values [0...3] are improbable to happen (due to systems administrator experience). Correct values occur between [4...10] and within those numbers security threat level maintains linear and systems can handle this security context in this interval. However Security context level arises exponentially when going beyond 10 ms and logarithmically when trespassing 15 milliseconds.

For this case, metric normalisation and value analysis only have been implemented for Security functionality but it could also be developed for privacy and dependability concepts referring to Network Latency Metric.

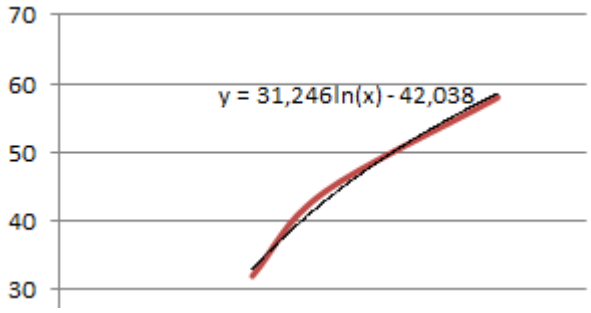
Step 2 - Regression function analysis

Once metrics are selected and normalised, values behaving within different regions must be analysed in formal mathematical manner. SPD Metric values should be evaluated against the expected risk for each of the value. This should describe a function behaviour that might follow either linear, exponential, quadratic or logarithmic curve (or some others like a disperse function.)

For the latter example, note that valid values (those that are between 4 and 10) follow a linear function F(X) = 2X + b. Furthermore the whole system follows the following function approach:

$$F(x) = \begin{cases} 2x & 0<x<4 \\ 2x + b & 4\leq x<\leq 10;\ b = b+1 \text{ in iteration} \\ *31{,}246\ln(x)-42{,}038 & 10<x \end{cases}$$

* For x> 10 values it presents a logarithmic approach. See image below with tendency function in black.



It can be said that valid region has been followed by a linear function in this case but it might follow other exponential, logarithmic or disperse function depending on the metric and use case.

Let's assume that for Network latency Metric (see table row #2) in the SoS nSHIELD Environment security and dependability concepts are important but not that much privacy. Indeed, dependability concept is the most important variable that should be observed when analysing Network Latency metric. Therefore, maximum indicator for measurements could be extrapolated from value analysis and regression function analysis. In the case of NL metric we should designate as maximum indicator for not trespassing the valid region as follows:

```
Indicator Metric NL (Network Latency) SoS, nShieldSystem= [26, 80, 10] [SPD]
```

Step 3 - Multi-metric Aggregation based on Expert Systems and Meta-heuristically Optimized Fuzzy Systems

The last step of the envisaged multi-metric security approach consists of an aggregation and decision making engine that processes the numerical values of the monitored security metrics (part of which are shown in Table 1) towards providing a fuzzy representation of the risk level of the heterogeneous device ecosystem at hand. The aggregation system should 1) comprise a set of human understandable linguistic rules (in the form of IF-THEN-ELSE conditional statements) to ease their supervision and manual crafting; 2) automatically infer optimized rule sets based on a performance index that reflects the error between the produced decision and that provided by security experts; and 3) tailor the mapping between the numerical and linguistic domains corresponding to the SPD values of the considered metric with their fuzzy representation and processing through the rule set. These specifications are deemed of utmost importance in our attempt at avoiding black-box decision engines that provide no further benefit than the blind automation of the decisions to be made.

These specifications call for the adoption of a specific class of fuzzy systems that incorporates stochastic solvers to optimize the scaling factors and membership functions that specify the meaning of the numerical SPD inputs in the linguistic domain, as well as the collection of fuzzy rules that best matches the decisions taken by security experts. Traditionally tackled via evolutionary algorithms (mostly, genetic optimizers [8, 9]), we plan to analyse and benchmark the performance -- in the context of optimizing fuzzy rule-based systems -- of more recent meta-heuristic solvers. In particular we will focus on Harmony Search [10], a population-based optimization algorithm that has been proven to outperform their genetic counterparts in a plethora of application fields such as engineering optimization, routing, resource allocation, economics and operations research (see [11] for a thorough survey). This algorithm resembles the way musicians in an orchestra improvise with their instruments before playing a piece of music in order to reach an aesthetically well-sounding harmony. This mimicking has implications in the definition of the operators driving the search procedure of the algorithm, which ultimately leads to a better adaptability of the explorative and exploitative capabilities with respect to the problem to be optimized.

Back to the pursued aggregation system, the goal is to produce an intensity, colored, real-valued indicator of the level of risk for the S, P and D components at each of the considered layers (node, network, middleware, overlay), in the form of an output matrix. To this end, the expert would be first asked to provide their estimated output to a series of eventual metric values so as to lay a decisional baseline information. This baseline would permit to compare the output of the system under differently optimized fuzzy systems and extract therefrom a performance index (integer from the range [0-100], in %) quantifying the level of compliance of the optimized decision maker with the expected output by the security expert(s). This performance index would measure the fitness of every combination of rule set/membership function/scaling factor iteratively refined by the harmony search optimizer. In particular, parameterized membership functions and rules will be jointly optimized by means of a Pittsburg approach where the population of the algorithm is formed by separately encoded variable domains.

Once the fuzzy engine is optimized after a given number of iterations of the Harmony Search heuristic, the aggregation system is ready to receive outputs from the different security metrics, fuzzify them through the optimized scaling and membership functions, apply the simultaneously optimized rule set and aggregate their outputs into a linguistically encoded output SPD matrix, which is finally defuzzified into colored intensity indicators so as to yield a more intuitive security assessing information of increased visual understandability.

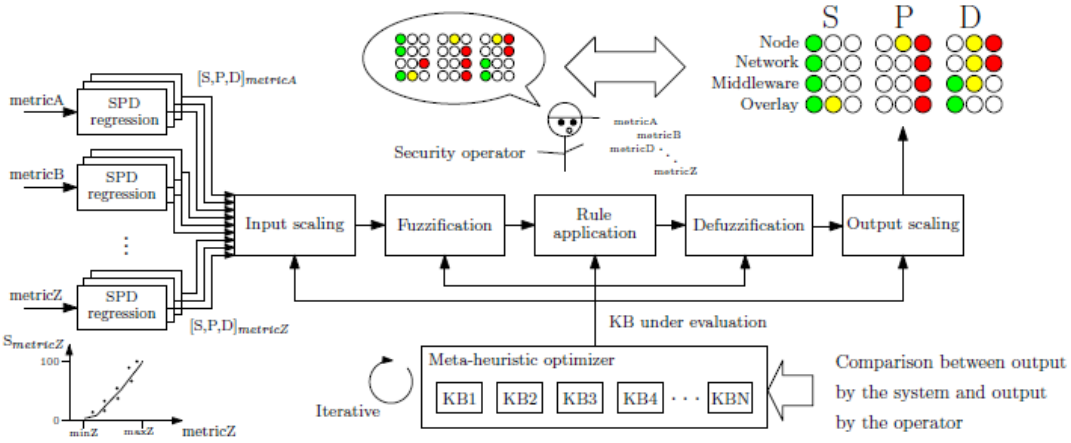


Figure 4-1: Global process for multi metrics approach

5 Conclusion

This final matrix will obligate to depict the following picture:

Table 5-1: nSHIELD Matrix: aggregated SPD dashboard for system operators

	S	P	D
Node	(G, Y, R)	(G, Y, R)	(G, Y, R)
Network	(G, Y, R)	(G, Y, R)	(G, Y, R)
Middleware	(G, Y, R)	(G, Y, R)	(G, Y, R)
Overlay	(G, Y, R)	(G, Y, R)	(G, Y, R)

This matrix shows the global perspective of the system in term of fuzzy concept such as: Green, Yellow and Red for Security, Privacy and Dependability concepts in Node, Network, Middleware and Overlay layers. This provides an holistic understanding of aggregated metrics in a SoS scenario.

By aggregating metrics through a fuzzy expert system and having an evolutionary learning mechanism, this paper demonstrates theoretically the feasibility of measuring metrics in a heterogeneous complex systems environment. This approach involves system expert knowledge but this could also be resolved by a derived approach where system can learn by its own. Next step shall be implementing this procedure in diverse scenarios for validating each process.

6 References

[1] An Attack Surface Metric - Pratyusa K. Manadhata, Member, IEEE, and Jeannette M. Wing, - IEEE Transactions on Software Engineering, 2010

[2] OSSTMM 3 The Open Source security Methodology Manual – Contemporary security Testing and Analysis – created by Pete Herzog – Developed by ISECOM – 2010

[3] Common Criteria – Common Methodology for Information Technology Security Evaluation – Evaluation methodology, September 2012, Version 3.1, Revision 4

[4] N.Yoshioka, H. Washizaki, k. Maruyama, A survey on Security Metrics. Progress Informatics, No 5, pp.35-47, (2008)

[5] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen, “Ananalysis of the security patterns landscape,” in International Workshop on Software Engineering for Secure Systems. Washington, DC, USA: IEEE Computer Society, 2007, p. 3.

[6] M. VanHilst and E. B. Fernandez, “Reverse engineering to detect security patterns in code,” in Proceedings of the International Workshop on Software Patterns and Quality. Information Processing Society of Japan, Dec. 2007, pp. 25–30.

[7] E. B. Fernandez, N. Yoshioka, and H. Washizaki, “Using security patterns to build secure systems,” in Proceedings of the International Workshop on Software Patterns and Quality. Information Processing Society of Japan, 2007, pp. 47–48.

[8] Oscar Cordón, Fernando A. C. Gomide, Francisco Herrera, Frank Hoffmann, Luis Magdalena: Genetic fuzzy systems. New developments. Fuzzy Sets and Systems 141(1): 1-3 (2004)

[9] Oscar Cordón, Francisco Herrera, Fernando Gomide, Frank Hoffmann, Luis Magdalena: Ten years of genetic-fuzzy systems: a current framework and new trends. Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference, pp. 1241–1246, Vancouver - Canada, 2001.

[10] Zong Woo Geem, Joong-Hoon Kim, G. V. Loganathan: A New Heuristic Optimization Algorithm: Harmony Search. Simulation 76(2): 60-68 (2001)

[11] Diana Manjarres, Itziar Landa-Torres, Sergio Gil-Lopez, Javier Del Ser, Miren Nekane Bilbao, Sancho Salcedo-Sanz, Zong Woo Geem: A survey on applications of the harmony search algorithm. Eng. Appl. of AI 26(8): 1818-1831 (2013)