A cybersecurity framework to GUArantee Reliability and trust for Digital service chains

www.guard-project.eu

D2.2 GUARD Reference Architecture

Editor: Matteo Repetto (CNR)

Version: 1.0

Status: Final

Delivery date: 28/02/2020

Dissemination level: PU (Public)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 833456

Deliverable Factsheet

Grant Agreement No.:	833456
Project Acronym:	GUARD
Project Title:	A cybersecurity framework to GUArantee Reliability and trust for Digital service chains
Call:	H2020-SU-ICT-2018-2020 “Cybersecurity”
Topic:	SU-ICT-01-2018 “Dynamic countering of cyber-attacks”
Start date:	01/05/2019
Duration:	36 months

Deliverable Name:	D2.2 GUARD Reference Architecture
Related WP:	WP2 System Architecture and Continuous Integration
Due Date:	31/01/2020

Editor:	Matteo Repetto (CNR)
Contributor(s):	Matteo Repetto (CNR) Luca Caviglione (CNR) Andrea Ranieri (CNR) Domenico Striccoli (CNIT) Alessandro Carrega (CNIT) Riccardo Rapuzzi (CNIT) Kalina Ruseva (LIF) Manuel Scimeca (UNITOV)
Reviewer(s):	Markus Wurzenberger (AIT) Andrea Romanelli (MAGGIOLI)
Approved by:	All partners

Disclaimer

Executive Summary

This document reports the preliminary design of the GUARD architecture. Starting from the main concepts and motivations behind the proposal, it elaborates on a more concrete technical concept and the business framework the Project addresses. The description clarifies what final outcome is expected from the Project, namely a platform to facilitate the deployment and operation of cyber-security appliances for digital value chains, and illustrates how this solution positions with respect to more traditional security models. The threat model provides a conceptual representation of the technological framework and points out the main adversaries and cyber-threats. Based on this analysis, some assumptions are derived and a list of security services that will be delivered by the end of the project is included.

The GUARD architecture is described from multiple perspectives. The functional model describes the main logical flows and functions that are required for collecting and processing security-related events. The reference model identifies the components that implement such logical functions and places them in a coherent architecture, which encompasses both protected resources and a centralized platform for security providers. Finally, the software architecture translates the conceptual blocks of the reference model into discrete software tools and applications, based on state-of-the-art technologies, best practices, and solutions provided by partners. It is worth noting that GUARD envisions several innovations in technologies and processes, so the work described in this document is mostly conceived as the starting point to ensure synergy and convergence between the activities of Work Packages and Tasks, and it will be subject to refinements and improvements. A live version of this document will therefore be available internally, until the delivery of the final version in M18.

Document History

Version	Date	Author(s)	Comments
0.1	6/11/2019	Matteo Repetto	TOC.
0.2	21/11/2019	Matteo Repetto	Preliminary description for review from LIF.
0.3	15/01/2020	Matteo Repetto	Threat model added.
0.4	30/01/2020	Matteo Repetto, Luca Caviglione, Andrea Ranieri, Alessandro Carrega	First draft available with the GUARD architecture.
0.5	5/2/2020	Matteo Repetto, Kalina Ruseva	Added contribution from LIF on legal implications. Draft available for internal review.
0.6	25/2/2020	Matteo Repetto, Markus Wurzenberger, Kalina Ruseva, Armend Duzha, Andrea Romanelli, Mauro Scarpa	Applied changes suggested by internal revision.
0.7	28/2/2020	Matteo Repetto, Domenico Striccoli	Added mapping of project requirements to the current architecture and security considerations.
0.8	26/2/2020	Riccardo Rapuzzi, Alessandro Carrega	Revised by CNIT
0.9	26/2/2020	Manuel Scimeca	Revised by UNITOV
0.10	27/2/2020	Kalina Ruseva	Revised by LIF
0.11	28/2/2020	Matteo Repetto, Domenico Striccoli	Added mapping of project requirements to the current architecture and security considerations.
1.0	28/2/2020	Matteo Repetto	Final version

Contributors

Organization	Author(s)	E-Mail
CNR	Matteo Repetto Luca Caviglione Andrea Ranieri	matteo.repetto@ge.imati.cnr.it luca.caviglione@ge.imati.cnr.it andrea.ranieri@ge.imati.cnr.it
CNIT	Alessandro Carrega Domenico Striccoli Riccardo Rapuzzi	alessandro.carrega@cnit.it domenico.striccoli@poliba.it riccardo.rapuzzi@cnit.it
LIF	Kalina Ruseva	kalina.ruseva@netlaw.bg
UNITOV	Manuel Scimeca	manuel.scimeca@uniroma2.it
AIT	Markus Wurzenberger	markus.wurzenberger@ait.ac.at
MAGG	Armend Duzha Andrea Romanelli Mauro Scarpa	armend.duzha@maggioli.it andrea.romanelli@maggioli.it mauro.scarpa@maggioli.it

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AAA	Authentication, Authorization, Accounting
ABAC	Attribute-Based Access Control
API	Application Programming Interface
B2B	Business-to-Business
CA	Certification Authority
CIA	Confidentiality, Integrity and Availability
CLI	Command Line Interface
CV	Cyber-security Vendor
DaaS	Data-as-a-Service
DAC	Discretionary Access Control
DAPS	Dynamic Attribute Provisioning Service
DC	Data Center
DMZ	DeMilitarized Zone
DRP	Digital Resource Provider
DPIA	Data Protection Impact Assessment
DSP	Digital Service Provider
EU	End User
FaaS	Function-as-a-Service
GDPR	General Data Protection Regulation
I2NSF	Interface to Network Security Functions
IaaS	Infrastructure-as-a-Service
IDSA	International Data Spaces Association
IDSCP	International Data Spaces Communication Protocol
IETF	Internet Engineering Task Force
IoC	Indicators of Compromise

IODEF	Incident Object Description Exchange Format
IoT	Internet of Things
IoTaaS	Internet of Things-as-a-Service
KVM	Kernel Virtual Machine
LAN	Local Area Network
LEA	Legal Enforcement Agency
NIC	Network Interface Card
MANO	MANagement and Orchestration
MAC	Mandatory Access Control
NFV	Network Functions Virtualization
NSF	Network Security Function
OES	Operators of Essential Services
OS	Operating System
OSM	Open-Source MANO
PaaS	Platform-as-a-Service
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKI	Public Key Infrastructure
QEMU	Quick EMUlator
RBAC	Role-Based Access Control
RP	Resource Provider
SaaS	Software-as-a-Service
SDN	Software-Defined Networking
SIEM	Security Information and Event Management
SOA	Service Oriented Architectures
SOC	Security Operation Centre

SOCaaS	Security Operation Centre-as-a-Service
SP	Service Provider
STIX	Structured Threat Information Expression
TaaS	Things-as-a-Service
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network
XACML	eXtensible Access Control Markup Language
WS	Web Services
WAN	Wide Area Network

1 Introduction

1.1 Purpose and Scope

This deliverable describes the GUARD architecture and provides the reference framework for the activity of the technical WPs. It selects existing tools and technologies that fit the Project requirements and identifies the technological gaps that should be addressed to achieve the objectives. The content is not limited to the bare description of the conceptual/software architecture, but also reports the work carried out in T2.5.[1] Therefore the document includes the main assumptions and analysis that motivated the architectural choices. It also elaborates on the target business framework the Project is going to address, so to provide useful hints for following exploitation and business models.

The GUARD architecture describes the main layout of the final platform and will be used by partners that develop discrete components to identify relationships and interfaces with other components. It will be also useful for platform integration, to verify the progress of parallel activities.

The current document freezes the activity of T2.5 after three months (i.e., at M9), but it should be considered as a live specification that will be constantly updated based on the inputs from other technical WPs. Such updates will complete the definition of missing software components and interfaces, up to the release of the final version in M18 (which will be included in D2.3[2]). The live specification will be subject to frequent changes and it is intended for internal use only, so no intermediate versions will be released out of the Consortium up to the aforementioned D2.3.

1.2 Contribution to other Deliverables

This deliverable builds on the work of D2.1[3] in the following way:

- it translates the project requirements into the appropriate set of functional processes, logical elements, and technologies;
- it selects technologies and tools from the set of existing solutions from the market and the open-source community.

The work described in this document is expected to contribute to upcoming deliverables in the following way:

- D2.3: the current description is the preliminary definition of the revised architecture and GUARD integrated platform.
- D2.4[4]: the software architecture described in this document already outlines how the different components are integrated in a common platform, by means of internal message buses and virtualization technologies.
- D3.2[5]: the current document describes the main features that should be available in the Security Dashboard.

[1] Reference Framework and System Architecture
[2] D2.3: GUARD Integrated Platforms and Revised Architecture
[3] D2.1: Vision, State of the Art, and Requirements
[4] D2.4: GUARD Integrated Platforms
[5] D3.2: GUARD user tools

- D4.1[6]: this specification describes how the set of technologies for inspection, monitoring and enforcement must be integrated in a common framework for data collection and remote control/management.
- D4.3[7]: this document clarifies the logical components that use the different interfaces and selects a set of candidate solutions.
- D7.4[8]: this document describes the business framework and the intended relationships between the different actors involved in cyber-security processes, which are the starting point for conducting market analysis, and elaborating business scenarios and exploitation plans.

Beyond more direct impacts described above, it is obvious that this Deliverable will affect most of the technical activities carried out in the Project, since the design, development, and integration of algorithms, software, and interfaces must refer to the common framework.

1.3 Structure of the Document

Most of the layout of this document reflects the chronological order of activities carried out in T2.5, though this is not completely true for the comparison with on-going initiatives and mapping to Project requirements.

Section 2 briefly revises the scope, motivations, and objectives behind the GUARD project. It also elaborates on the technical concept, which clarifies how the overall vision described in D1.1 is translated in a more concrete solution to existing problems for cyber-security operators. This description points out what final outcome is expected from the Project, namely a platform to facilitate the deployment and operation of cyber-security appliances for digital value chains, and illustrates how this solution positions with respect to more traditional security models.

Section 3 describes the business framework, namely what business roles are present, concrete examples of digital resources, the problem of sharing responsibilities, and the information and business workflows. It also includes some legal implications of the proposed model.

Section 0 describes the threat model. It provides a conceptual representation of the technological framework and points out the main adversaries and cyber-threats. Based on this analysis, some assumptions are derived and a list of security services that will be delivered by the end of the project is given. The threat model for GUARD is deliberately less precise than other frameworks, because GUARD is not a cyber-security appliance per se, but it is conceived as a platform to run multiple cyber-security appliances. It is therefore necessary to indicate what classes of threats can be addressed by this architectural approach and the context that could be reasonable collected, while specific threat models must be developed for each different security appliance.

Section 0 describes the GUARD architecture from multiple perspectives. Initially, a brief overview of emerging architectural patterns for implementing digital services is given (Section 5.1). The functional model (Section 5.2) describes the main logical flows and functions that are required for collecting and processing security-related events. The reference model (Section 5.3) identifies the components that implements such logical functions and places them in a coherent architecture, which encompasses both protected resources and a centralized platform for security providers. Finally, the software architecture (Section 5.4) translates the conceptual blocks of the

[6] D4.1: GUARD technologies for inspection, monitoring, and enforcement
[7] D4.3: Open APIs for collecting security context in distributed systems
[8] D7.4: Refined and Extended Business Plan

reference model into discrete software tools and applications, based on state-of-the-art technologies, best practices, and solutions provided by partners.

Section 0 identifies the relationships with other initiatives. It explains why GUARD is aligned with current and emerging trends in the cyber-security sector, hence positioning as a ground-breaking technology for detection and mitigation of modern threats.

Section 0 maps the technical requirements identified by D1.1 into the architectural design of GUARD. It is used to check that all requirements are met at this stage, and to highlight what still remains to be considered in the implementation phase.

Finally, Section 0 reports some preliminary security considerations about the framework itself, which are necessary for operating the platform without introducing any additional threat vectors or increasing the attack surface.

2 Scope, motivation and objectives

Evolving business models are progressively pushing for larger digitalization of existing and novel processes. The ICT industry is already addressing this need by massive introduction of virtualization paradigms and tight integration with the physical environment, which allow the creation of multi-domain and complex business service chains. Emerging technologies undoubtedly bring more agility in service deployment and operation but also break traditional security models, which have not been conceived for dynamic and multi-tenancy environments. As a matter of fact, the typical assumption is the physical segmentation between internal and external resources through a security perimeter, an assumption that does not hold anymore when external cloud and IoT resources are integrated into business processes.

A thorough analysis of the major trends behind the virtualization wave and their implications on cyber-security aspects is reported in D2.1, which also explains in details the Project concept and target application scenarios. Here, we complement that description with additional information that drove the definition of the GUARD architecture. We therefore elaborate on the specific scope of the Project, the main motivations, and the specific objectives, which together dictate the fundamental traits of the GUARD architecture.

2.1 Scope

Data will be the key driver for the digital economy. Novel digital products and services are expected to create, process, share, and consume data and content in a digital continuum, blurring the frontiers between application domains and breaking the current closed silos of information. The major trend in this respect is the interconnection of processes, products, services, and things from multiple vendors on a growing scale [1], to create, process, share, and distribute data and content.

The software industry has been progressively introducing new architectures and patterns that bring more agility in the creation and management of new services and products [2],[3], which means digital services and business chains are expected to emerge and dissolve much faster than traditional value-creating networks. Convergence among existing software paradigms, such as cloud computing, software-defined networking, and the Internet of Things (IoT) is expected to this purpose, leveraging autonomicity and dynamic composition through service-oriented and everything-as-a-service models applied to cyber-physical systems.

Without any intention to superseded or augmenting the application scenarios already described in D2.1, Figure 1 shows an illustrative example of the intended scope. The left side pictorially depicts the reference scenario, whereas the right side shows the expected chain involving devices, infrastructures, software, and services provided by different providers. We suppose that a Traffic Services Provider (TSP) deploys and manages a number of Road Side Units (RSUs) to provide info-mobility and smart driving services to drivers (1). The TSP owns the RSUs and other monitoring devices (like traffic cameras), but it relies on third parties for additional infrastructures and services (i.e., driving applications (2) and data storage (3)). It rents a network slice from a Network Provider (NP) to interconnect all its RSUs on the geographic scale; it also connects to an Autonomous Driving Application deployed in the cloud. The final users of the system are the drivers, through their connected vehicles, which only see the TSP interface but are not aware of the different infrastructures and domains involved in the creation of traffic services. In a typical workflow, vehicles connect to the RSU and present the user the list of available services they discover. For example, the user may activate a remote autonomous driving application, which feeds the on-board systems with information about the behaviour of the surrounding vehicles. This application is not hosted by the TSP, but it is provided by an external Application Provider (AP). Since this application requires high guarantees on bandwidth and latency, a dedicated network slice is provided

by an additional Network Provider (NP), which bears both sensor data and real-time videos (2). Finally, the Application Provider has deployed its software in the cloud (4), so to easily scale in and out according to the current workload.

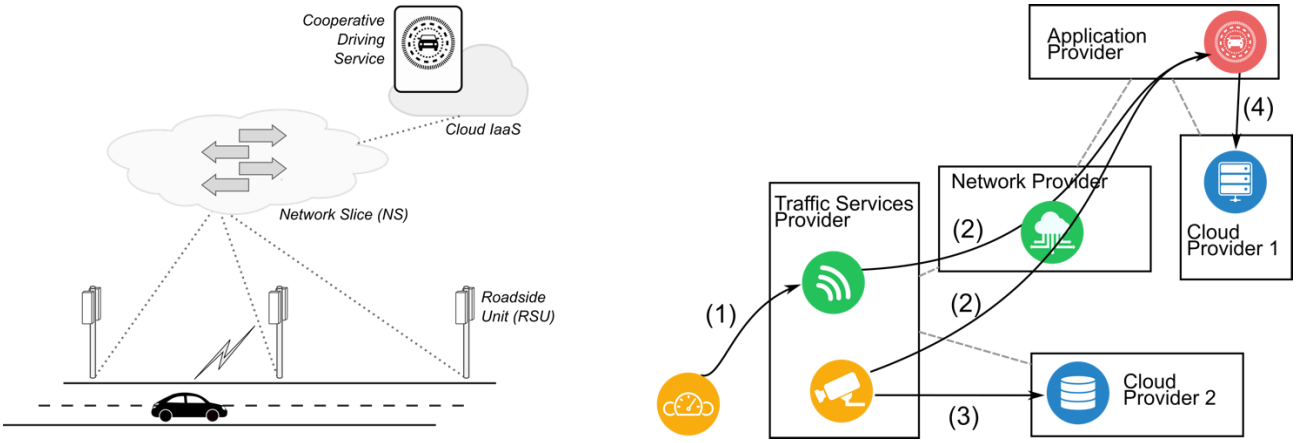


Figure 1. An illustrative example of a digital service chain for the automotive sector.

The combination of new architectural patterns based on multi-tenancy and externalization breaks legacy security models, which are still largely based on the creation of a sharp and effective security perimeter. As a matter of fact, the integration of IoT devices and the delegation of applications and processes to public cloud and edge infrastructures blur the boundary between public zones and private domains [4].

Given the increasing sophistication of attacks, a lot of research effort has been devoted to apply big data, machine learning, and artificial intelligence techniques to detect advanced, persistent, and stealthy threats [5]. However, applicability to real environments has been largely neglected. Indeed, the presence of multiple domains, multi-tenancy, unknown or hardly-traceable topologies and heterogeneous technologies often limit the visibility on the on-going context, since it the deployment of suitable hooks and probes for the collection of the proper set of events, data, and measurements is not straightforward.

In this context, GUARD proposes a flexible and agile paradigm to effectively address cyber-security issues that arise from the on-going transformation of the digital economy towards cross-domain value-chains. GUARD approach pursues a novel architecture to create new tools for managing security aspects of digital services, which tackle the dynamicity and unpredictability of such environments. As such, GUARD technology is expected to improve operations of Security Operation Centers (SOCs).

2.2 Motivations

The progressive introduction of software orchestration paradigms allows more automation in control and management of complex systems, which can evolve at run time and adapt to the evolving context without the explicit control of software engineers. The dark side effect of this evolution is the risk for large unpredictability, due to non-deterministic, opaque and partially inscrutable service topology. This raises questions about the overall behaviour of the system, the location of personal and sensitive data, the sanity of the software, the availability of the whole service, and, most of all, the ability to perform quick remediation and mitigation actions in case something goes wrong. Novel security and privacy models are required, as agile development does not fit the sequential nature of traditional security engineering processes [6].

Micro-services, web services, service-oriented architectures, programming models, distributed middleware, and software orchestration are already present in latest frameworks and market technologies for both the software, CPS, and telecommunication industries (see, for instance, initiatives as TOSCA [7], ETSI NFV [8], FIWARE[9]). At the same time, ubiquitous connectivity and almost unlimited bandwidth promised by upcoming 5G technologies are progressively reducing the performance gap between local and wide-area communications, driving towards the creation of a seamless pervasive computing continuum for vertical industries [9].

Modern digital services are generally composed along the three dimensions of infrastructure (legacy enterprise ITC, cloud/fog/edge computing, IoT), software technologies (sw), and data. The definition of a service basically consists in the interconnection of software modules, deployed over heterogeneous infrastructures and domains, and accessing local and remote data (see Figure 2). For example, a cognitive system for assisted/autonomous driving will include an on-board processing and visualization device, cameras and sensors on the car, navigation maps, traffic alerts and information on road conditions, historical data processing, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication, etc. The whole system will be composed of many software elements, deployed in heterogeneous execution environments (car, cloud, enterprise) and operated by different players (car owner, navigation services, road operators, other drivers, etc.).

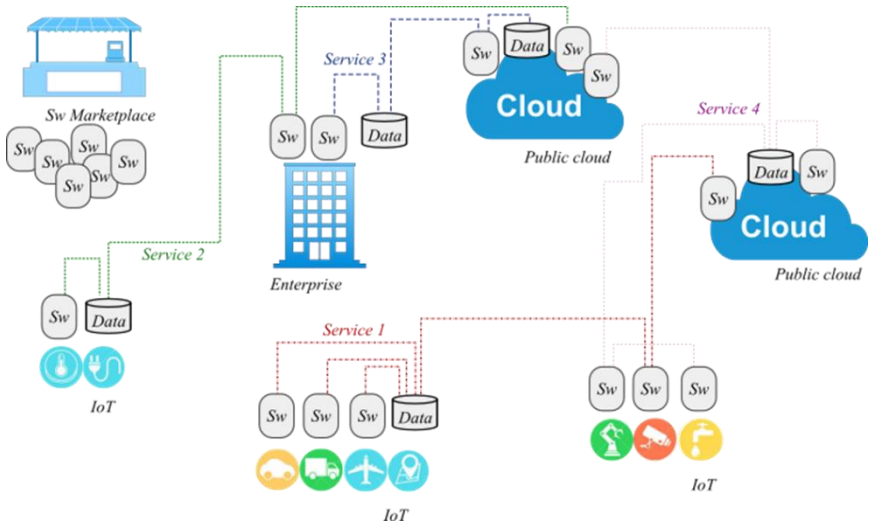


Figure 2. Composition of digital services.

The composite nature of digital services will lead to complex dynamics, dispersion of data among the multitude of digital objects and infrastructures, unknown or hardly-traceable topologies. The predominant interspersion of lonely valuable resources with unsafe computing and communication infrastructures makes the application of the security perimeter at each site ineffective, because of the overhead to run complex agents in end-user devices, especially in case of resource-constrained ”things”. In addition, the growing complexity of cyber-attacks, often based on multi-vector approaches, are urgently demanding more correlation in space and time of (apparently) independent events and logs, and more coordination among different security applications.

While chasing advanced assurance and protection of business service chains, GUARD aims at overcoming the following limitations in existing practice and technologies:

[9] The FIWARE platform. URL: https://www.fiware.org.

- **slow and ineffective detection of attacks**, due to partial and incoherent security information, non-interoperable algorithms targeting specific cyber-attacks only (e.g., network or server DDoS, intrusion detection/prevention, malware identification);
- **difficulty in identifying new threats and vulnerabilities**, because of limited information and data available, ineffective correlation of data from multiple sources, lack of big data and machine-learning techniques;
- **outdated and inefficient architectures** for sharing security context, which are able to combine the need for fine-grained knowledge with efficient resource usage. In fact, effective detection of attacks and identification of new threats may need detailed logs, events, and measurements for processing and correlation, but this would result in excessive overhead for both network and storage systems;
- **technology and business lock-in** by the adoption of verticals with tightly integrated network/cyber-security solutions, often from a single vendor;
- intrinsic **rigidity**, due to the difficulty to change architecture and system configuration: network partitioning, deployment of hardware or software security appliances (including the necessary agents), routing and switching policies in case of traffic diversion towards scrubbing centres;
- **slowness and inertia to share the knowledge of new threats and attacks**, due to heavy involvement of humans in the loop and the need for manual operations, and to lack of common representation formats;
- **ineffective interaction with users**: though security dashboards are continuously evolving and enhancing user-friendliness, they are usually reserved to technical staff and do not integrate seamlessly in other administrative and management processes (e.g., risk assessment, legal and normative requirements).

2.3 Objectives

The current market of cyber-security products is rather crowded yet largely fragmented. In addition to legacy firewalls, intrusion prevention/detection systems and antivirus for personal and enterprise usage, many vendors are already tackling new business opportunities by delivering solutions for enterprise's networks and endpoints, pure and hybrid cloud, industrial devices and networks, security analytics, and integrated solutions. The lack of common interoperability interfaces and standards has boosted the creation of proprietary solutions and frameworks from all major vendors, leading to technology and business lock-in in the market.

The detection of (even zero-day) attacks and identification of advanced threats requires building complex processing pipelines, ranging from event generation, to collection, storing, and analysis, up to providing countermeasures and reaction plans. Specific platforms are designed to support humans in this activities, which activity, especially those that are involved in the operation of Security Operations Centres (SOCs). These are usually static architectures tightly coupled with the specific environments they protect; any change in the detection process or underlying system topology often requires re-design or complex re-configuration procedures.

Tackling conflicting trends in the cybersecurity market, like fragmentation or vendor lock-ins, the main purpose of GUARD is to develop an open and extensible platform for advanced assurance and protection of trustworthy and reliable business chains spanning multiple administrative domains and heterogeneous infrastructures. In this respect, GUARD advocates a clear separation between data collection and data processing, mediated by an intermediate control logic that programmatically feeds a number of parallel security services with relevant context. Specific technical objectives to this aim include:

- **To design a holistic framework for advanced end-to-end assurance and protection of business service chains**, by assessing the level of trustworthiness of the involved services and tracing data propagation. The main ambition is to define and boost the adoption of security primitives as part of the management interfaces of digital services. GUARD will define open interfaces that expose relevant security properties:

reputation and certification of vendor/owner, trust chains, integrity of devices, software, and processes, geographical location, presence of users' data, compliance with relevant data protection and cybersecurity legislation (GDPR, ePrivacy, NIS directive and other relevant regulations). Based on this API, formal methods can be derived to verify the compliance of the above information with specific policies from the users (e.g., a user does not want his/her data to be processed or stored outside Europe), hence pursuing more automation. Finally, GUARD will also implement enforcing policies and agents to control and track exchange of sensitive information between the services.

- **To improve the detection of attacks and identification of new threats**, by applying online and/or offline machine learning and other artificial intelligence mechanisms to large datasets collected from heterogeneous services in multiple administrative and technical domains. As already pointed out, the purpose is not only to develop innovative algorithms, but also to tailor them to challenging use cases, characterized by virtualization, multi-tenancy, elasticity, and heterogeneity. The main technical challenge in this respect is the design of a context middleware that decouples detection algorithms from the underlying monitoring environment. The middleware shall include identity management and access control between context and the detection algorithms, by controlling what is exported, to whom, and to what purpose. An additional ambition is more automation in information sharing, by developing new methodologies to retrieve cyber-threat intelligence (CTI) from novel findings and exchange it with relevant national and international bodies.
- **Fine-grained, programmable and low-overhead monitoring, inspection, and enforcement,** by leveraging "programmability" to shape the granularity of context information to the actual needs. This goes in the direction of building efficient distributed platforms, which dynamically adjust the verbosity of logs, frequency of sampling, and other tuneable parameters so to not overwhelm unnecessarily the network and storage systems. Programmability goes beyond the traditional notion of configurability, by also including the capability to create small lightweight[10] programs for tasks as packet inspection, data fusion, aggregation, pre-processing, and enforcement.
- **To improve awareness and reaction**, by developing user tools for visualization, notification, configuration, investigation, mitigation. Beyond the development of a web-based dashboard, GUARD will develop a framework to specify users' security policies (trusted vendors/countries, minimal encryption requirements, trust chains, security mechanisms, etc.) and to create secure chains accordingly.

2.4 Technical concept

Looking at the growing number of complementary appliances and the tighter integration required for effective detection of ever-more stealthy threats in complex ICT systems, the GUARD approach is based on the delivery of a common platform to run multiple security services (see Figure 3). The GUARD platform is conceived as a mediation layer between the detection and analysis logic and the physical and virtual environments where digital services are run. It discovers evident and hidden relationships between digital entities involved in complex processing chains and gives access to their security properties and features (*Programming*). It also provides programmatic access to the security context, i.e., the set of security-related events, data, and measurements that must be inspected by the algorithms (*Visibility*). Finally, it creates a common channel to import/export cyber-threat intelligence (*Sharing*). It is worth pointing out, to avoid any misunderstanding, that the GUARD platform is not a full-fledged security appliance per se. Rather, it provides an environment to simplify the design

[10] At least, with respect to the burden of running full-fledged cyber-security appliances.

and operation of detection and analysis algorithms, which are expected to be provided by other parties (see Section 3.2). Nevertheless, the Use Cases will provide complete instantiations for specific security services, so to demonstrate the applicability and performance to relevant scenarios.

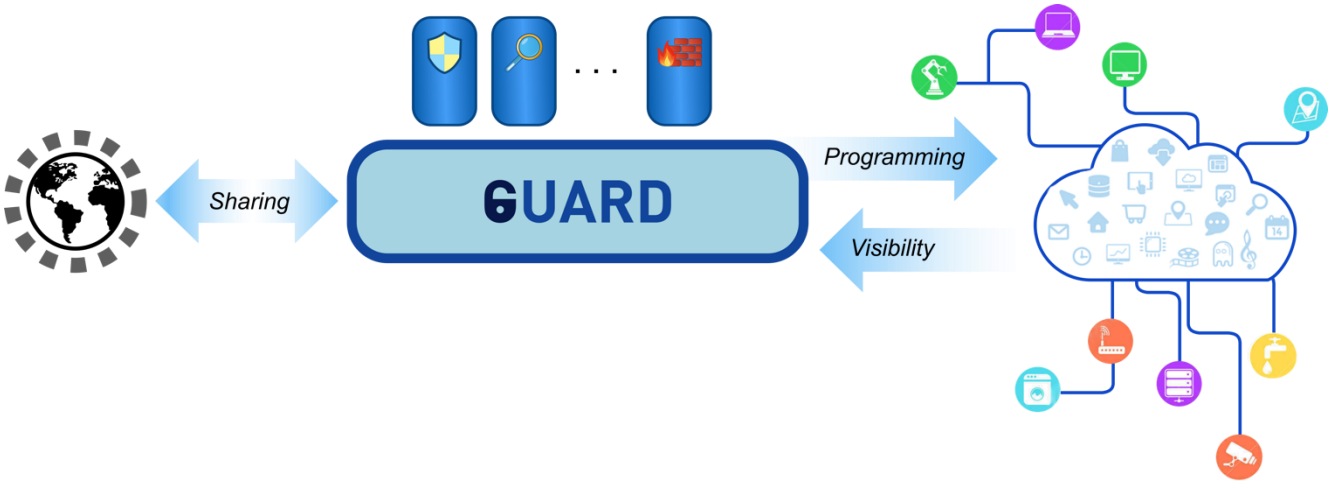


Figure 3. The GUARD platform mediates between security services and the digital world.

The approach proposed by GUARD facilitates the design and operation of new security services, which are no more required to implement and deploy their own monitoring hooks. **The challenging value proposition of GUARD is the adoption of programmable monitoring and inspection technologies**, that could be adapted at run-time to the need of the detection logic, leveraging far more configurability and programmability than existing solutions. The presence of a common platform would allow different algorithms to share the same data and events, without overwhelming remote systems and network connections with duplicated and redundant information. The same applies to historical data, which can be collected only once and retrieved by different services through a common abstraction. As part of the effort towards improved efficiency, **GUARD also envisions the possibility to offload simple data aggregation and fusion tasks locally**, so to reduce the impact on the network (bandwidth and latency).

Another important aspect is the delivery of common methodology to retrieve and publish CTI. This effectively contributes to simplify the delivery of new security services, which can leverage the availability of a common framework for sharing novel findings with national and regional CERTs/CSIRTs.

The mediation function implemented by the GUARD platform is not limited to the North-South direction (i.e, between security appliances and digital services), but also extends to Eastbound interactions (i.e., delivery of notifications between security appliances). This further improves the possibility to create complex security services, by combining or linking multiple algorithms together.

The GUARD platform addresses the need of cyber-security operators to run legacy and novel detection and analysis tools on digital processes that span multiple heterogeneous technological and administrative domains. It primary targets the market of tools for operating a SOC, by proposing a solution that facilitates the implementation of the **SOCaaS concept**, by leveraging programmability and security properties embedded in digital services that allow software-defined instantiation of monitoring and inspection hooks. Similarly, more automation in the processes of documenting and describing new threats would avoid lot of paperwork and error-prone procedures. A more detailed analysis of the business framework that GUARD will address is reported in Section 3.

3 Business Framework

This section describes the main entities involved in the secure management and operation of a digital service chain, according to the GUARD concept. It discusses the main business and technical relationships that are necessary to operate according to the GDPR principles and to identify relevant business cases for commercial exploitation.

3.1 Concept

Following the general trend towards Everything-as-a-Service models (XaaS), GUARD help implement the concept of Security Operation Centres-as-a-Service (SOCaaS). Differently from a bare SIEM, which sends an overwhelming number of alerts to team members of the organization running the service/infrastructure, GUARD enables security specialists to carry out deep analysis and investigation in a very flexible way, by selecting multiple complementary algorithms for each specific situation. Indeed, specific features that characterize a SOCaaS approach include:

- outsourcing analysis and investigation to dedicated teams of security experts;
- manage detection and response, also in a (semi-)automated way;
- replace bare SIEM with more powerful and flexible collection tools;
- ensure compliance to regulations;
- carry out behavioural analytics and log analysis with innovative tools, including AI;
- trigger real-time alerting;
- perform vulnerability scans and find out bogus or compromised software.

The deployment of capillary and pervasive security agents throughout ICT infrastructures (network devices, servers, users' terminals, smart things) is among the most critical issues for the realization of SOCs. Indeed, several vendors are already delivering their integrated solutions for endpoint protection, network visibility and segmentation, cloud protection, embedded industrial and IoT devices, and SIEM/security analytics, but the adoption of such frameworks is often hindered by high costs for initial deployment, the risk for vendor lock-in, and the increasing presence of different administrative domains in digital business chains.

Differently from existing commercial solutions, GUARD is pursuing the business opportunity brought by the increasing adoption of multi-domain solutions. The distinctive and innovative approach of GUARD is to foster **the adoption of specific APIs to expose and allow programmatic access to local security capabilities in each digital resource**, independently of the internal implementation. The ambitious goal is to overcome any vendor lock-in, chasing better interoperability and dynamic set-up of SOC services for individual or multiple business chains.

3.2 Business roles and relationships

The GUARD business concept implies a different business workflow than those used in existing practice for security management. There are multiple stakeholders in the GUARD ecosystem, which need to establish business and technical relationships in order to create a secure and trustworthy ecosystem, as pictorially shown in Figure 4.

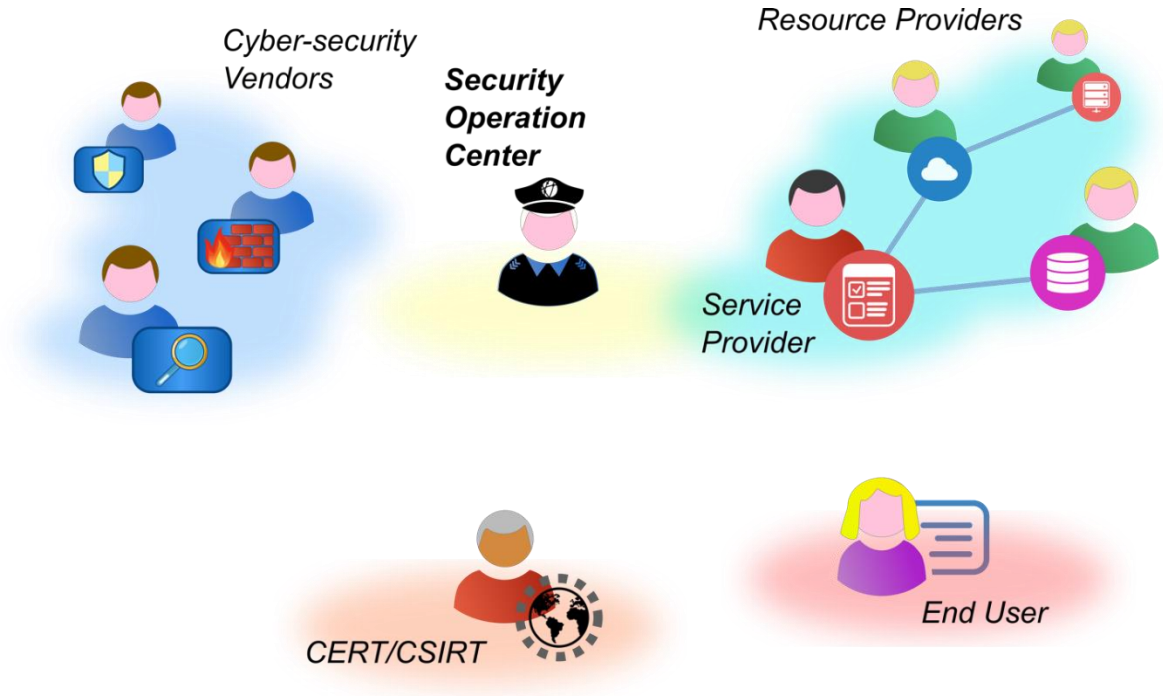


Figure 4. Business roles in the GUARD model.

The **End User** (EU) is the first link in the GUARD business workflow. She accesses digital services by one or more electronic devices: computers, tablets, smartphones, TVs, card readers, wearables, etc; the End User may own the device, or it may be included in the service subscription. There is no specific requirement that End User's devices are included in security operations; this option depends on each Use Case. As a general consideration, it is unlikely that the necessary business relationships are set up with each End User, so personal devices owned by users are not expected to implement the GUARD interface; on the other hand, devices supplied as part of the service (which may be the case for smartphones and IoT) should be covered by GUARD. While using the service, the End User will likely reveal personal or sensitive information, and this raises some concerns about the usage of such data. Today, such kinds of business relationships are mostly ruled by disclaimers and informed consent; technical means to exercise the rights settled by the GDPR will be useful to be fully compliant with the regulation and increase the mutual trust between users and their service providers.

Service Providers (SPs) operate digital services. The growing size, complexity, and dynamicity of industrial and commercial value chains are increasingly demanding more and more flexibility in creating, operating, and disposing digital resources on very large (if not global) scale. As a matter of fact, today the largest revenue stream often comes from designing and operating tailored services. New business models are therefore leveraging virtualization and multi-tenancy as effective mechanisms to share infrastructures and resources, which can then be composed in new products and solutions faster than ever. Under this evolutionary process, we could identify multiple complementary business roles, which we generically indicate as "**Digital Resource Providers**" (DRPs) or "**Resource Providers**" (RPs) for short: Infrastructure Providers, which own physical resources and infrastructures (data centres, metropolitan and geographical networks, IoT installations, etc), Software Providers, which develop software functions and make them available in public or private repositories (e.g., github), Cloud Providers, which combine computing and storage infrastructures into virtualized services according to IaaS, PaaS, or FaaS models (i.e., they provide bare VMs, storage services, lambda functions), Network Operators, which implement large-scale communication services for public and private users (mobile networks, VPNs, NFV), Function Provider, which implement specific logical functions (e.g., authentication, databases, context brokers),

and so on[11]. Value-added digital chains may be composed by Service Providers, for example, by selecting some software, deploying it in the cloud, connecting to IoT devices or data brokering services, reading data from data bases, connecting to external authentication services, secure networking with remote peers, etc; the reference application scenarios discussed in D1.1 provide concrete examples that are relevant for the Project. The progressive disaggregation between RPs and SPs is at the heart of the main cyber-security challenges for creating digital value chains.

Cyber-security Vendors (CVs) develop cyber-security services, designed to be run by the GUARD framework. They play the same business roles as today, by providing security appliances to security staff (i.e., Cyber-Security Operators). In general, cyber-security services will be composed of a detection appliance and its management logic. A detection appliance analyses and inspects security-related events collected by the GUARD framework from digital services. They may consider logs from applications, events, network packet traces, position and transfer of sensitive data, and so on, according to a context model. Legacy detection appliances (e.g., IDS/IPS, antivirus, network monitors) could be ported to GUARD, by replacing their specific inspection mechanisms with GUARD drivers; however, novel techniques will be developed in the Project to exploit the broad range of data types and the presence of multiple heterogeneous sources. To leverage the programmability of the GUARD framework, CVs are also expected to provide run time management policies that change the behaviour of their applications according to the evolving context. Such policies are used by the GUARD framework to automate as much as possible the operation of cyber-security services. For example, such policies may set in the GUARD framework the types and frequency of data and events needed for the detection, they may scale the service according to the size of the service and the amount of collected data, they may trigger external (re-)actions in case of attacks or anomalies, and so on.

CERTs/CSIRTs are centralized functions for information security incident management and response in an organization or sector or at nation-wide or international levels. The ultimate goal of a CERT/CSIRT is to minimize and control the damage resulting from an incident, which requires many different functions to be involved. Indeed, beyond technical investigation, it is also responsible to communicate to customers and the public about the incident. If a malicious internal actor caused the event, disciplinary, and perhaps legal action will need to be taken on involved employees. The main tasks for a CERT/CSIRT usually include:

- Preventing, detecting, and responding to ongoing security threats
- Ranking and escalating alerts and tasks
- Investigating, analysing, and conducting deeper forensics on incidents
- Developing communication plans (for public relations, customers, board members, etc.)
- Coordinating and executing response strategies
- Maintaining records of known vulnerabilities and threats
- Maintaining a repository of log data related to events for future reference, as well as for compliance or legal purposes

Accordingly, staff is not limited to technical personnel, but usually involves a conglomeration of roles that also encompass public relations, marketing, customer support, and management. The number of people employed

[11] We point out that all these terms are only used internally to the Project, because there is no common and uniform nomenclature for identifying providers of different services and resources.

in a CERT/CSIRT clearly varies with the size and scope of the covered system: one or more small/medium companies, large organizations, national infrastructures.

A **Security Operation Centre** (SOC) clusters all technical capacity to protect business processes and the organization itself. It includes prevention, detection, incident management and response, reporting, governance, risk, compliance, and anything to do with managing and defending information security within the organization. Differently from CERTs/CSIRTs, a SOC is usually technology-focused: the goal of a SOC is to implement and oversee network, application, cloud, and user security, among other operational functions. Nevertheless, in small organizations there could be a single centre that implements typical tasks of both SOC and CERT/CSIRT.

3.3 Digital resources

The ever-growing complexity and scale of information and communication technologies have a conflicting impact on business. On the one hand, this evolution enables the implementation of unprecedented services and products in any business sector: industry, agrifood, health, electricity, commerce, finance, multimedia, automotive, transportation, etc. On the other hand, the most remunerative business is often the implementation of tailored solutions, which is not profitable enough for big players but remains highly challenging for small businesses. As a matter of fact, some businesses are often too small, either in terms of financial strength or scale of operations or technological expertise, to effectively design, implement, and deliver the full technological stack which is required by modern applications. Novel and uncertain technologies, ever-faster product lifecycles and market dynamics, and the need for expertise in a growing number of domains are among the main hindrances that jeopardize the implementation of high value-added services by small businesses.

To effectively tackle such growing complexity and scale of ICT, the 'as-a-service' model has emerged that lowers the entry barriers for individual and small enterprises, by minimizing the cost of ownership and the capital expenditure to design and deploy new applications. The underpinning concept behind the 'as-a-service' model is the virtualization of devices, infrastructures, processes, and applications, which become accessible through software APIs, without the need for deep knowledge about their internal operation. Although this elementary definition is suitable to generate an unlimited number of potential offerings, the most common cases for digital resources that are considered in the Project can be restricted to:

- *Infrastructure-as-a-Service* (IaaS): computing, networking, and storage resources are abstracted and pooled together to create virtual instances of typical ICT infrastructures. Abstractions usually include VMs (which run the same software as physical servers), LANs (which interconnect VMs), routers and gateways, block and object storage, software repositories (for cloud images). The IaaS model frees resources from their physical constraints, overcoming many practical issues that concern large investments to buy the hardware and deploy it, delays in the purchase process, costs for disposal, etc.

- *Platform-as-a-Service* (PaaS): an additional layer is added to the IaaS, including application development frameworks, middleware capabilities, and functions such as databases, messaging, and queuing. For example, a development environment may include all computing resources and software to compile and run programs in any language. In PaaS the cloud user only sees the platform, not the underlying infrastructure. That means cloud providers remain responsible to maintain and update the platform, freeing the developers from configuring and building servers, keeping them up to date, or worrying about complexities like clustering and load balancing; for instance, a platform that provides

a database service should expand (or contract) as needed based on utilization, without the customer having to manage individual servers, networking, patches, etc.

- *Software-as-a-Service* (SaaS): SaaS services are full, multitenant applications, with all the architectural complexities of any large software platform. Many SaaS providers build on top of IaaS and PaaS due to the increased agility, resilience, and (potential) economic benefits.

- *Network-as-a-Service* (NaaS): the scope is rather broad, ranging from local area networks (which are often included under the umbrella of IaaS) to wide area networks. Network appliances are implemented as virtual instances (Virtual Network Functions, VNFs), which can be deployed and orchestrated with similar paradigms to VMs in the cloud IaaS model. Differently from legacy networking, where forwarding decisions are usually taken based on distributed protocols, NaaS entails software-defined operations, which are used to steer packets across multiple virtual or physical network functions, in a conditional way, so to create processing pipelines tailored to different traffic flows. The NaaS paradigm and the concept of *network slicing* are the main technological enablers for building vertical applications in 5G infrastructures.

- *Data-as-a-Service* (DaaS): data products can be provided to the user on demand, regardless of geographic or organizational separation between provider and consumer. Service-oriented architectures (SOA) and the widespread use of API have rendered the platform on which the data resides as irrelevant. This model overcomes the traditional model used by many organizations, which has been largely based on data stored in a self-contained repository, for which software was specifically developed to access and present the data in a human-readable form. The attractiveness of DaaS to data consumers lies in the separation of data cost and of data usage from the cost of a specific software environment or platform. Pricing models used for data are even more flexible than those for other digital resources: they can be based on volume (per-quantity or per-call) or attributes (for instance: geographic, financial, historical data).

It might look odd that IoT is not included in the previous list. Indeed, Internet of Things-as-a-Service (IoTaaS), or just Things-as-a-Service (TaaS), is one of the latest iterations in the 'as-a-service' jungle [15], [16], [17]. However, there is not a shared understanding about this concept in technology or business jargons. Just offering raw access to things seems hard to monetize right now, but this does not mean nobody will leverage the things themselves for revenue in the next future. After all, this use case has already been included in the 5G technological vision several years ago [18]. TaaS looks to be the definition that best fits this option, and it could be included as an extension of the broader IaaS model, where things become the fourth dimension in addition to computing, networking and storage. On the other hand, the complexity in installing and operating IoT environments has motivated an overarching approach, which includes sensors, the process by which they communicate information to the network, the cloud layer itself, and various tools that are used to analyse data once it is collected. Indeed, big players find that the most profitable stream comes from applications and analytics to deliver operational efficiency and insights. The definition of IoTaaS fits well this case, which can be seen as a specific instance of DaaS (if only raw sensor data is provided) or PaaS (if a full-fledged platform is provided to connect things, collect data, and perform analytics). As final consideration, we point out that the two different acceptations for IoTaaS/TaaS imply different business models. Indeed, when things are directly shared, they must manage multi-tenancy and multiple services; otherwise, these tasks are delegated to cloud applications and platforms.

3.4 Responsibilities

Sharing resources among multiple users is the underpinning capability to dynamically create digital value chains. Virtualization and multi-tenancy are therefore largely used to create ephemeral instances of software, infrastructures, and data which appear as independent resources and hinder cross-interference between different tenants.

However, the progressive disaggregation between RPs and SPs is at the heart of the main cyber-security challenges for creating digital value chains. As a matter of fact, the separation of the ownership of resources and services makes the identification of responsibilities and security obligations more difficult than in traditional ICT installations. Indeed, the presence of multiple logical layers, different provisioning models, and multi-tenancy has a disrupting impact on the nature of risks, roles, and related responsibilities.

Though the overall objectives and scope of security do not change when cloud technologies are used, they must be split between different actors, and the demarcation line is not always clear to everybody. Yet this allocation is far from being simple to manage, since there are many interdependencies and correlations between the technical and administrative aspects that fall under the scope of different actors. Figure 5 maps the responsibilities of SPs and RPs in the matrix of logical layers vs business models.

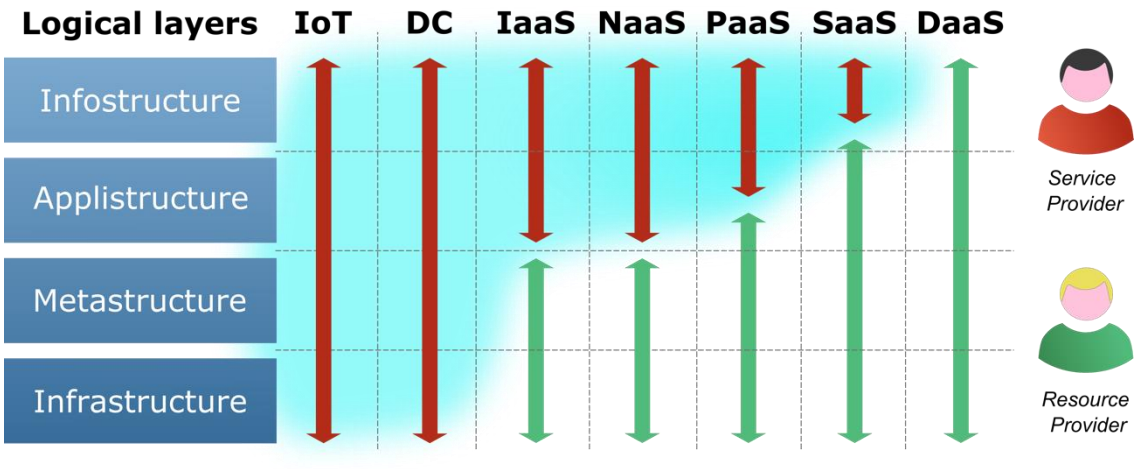


Figure 5. Mapping of responsibilities on the different logical layers.

The matrix decomposes ICT services in multiple logical layers, adopted from the Cloud Security Alliance [19], and shows how responsibility is split for different business models. Multiple business models are taken into account that could be used by SPs. The legacy approach is to buy and operate the whole infrastructure, including things (IoT) and ICT installations (DC). Financial and operational sustainability suggest limited usage of this model, e.g., to implement critical services, while additional resources will be acquired on-demand by `as-a-Service' paradigms, as already introduced in Sec. 3.3. We recall that smart things can be included in either PaaS or DaaS models, when they are not fully managed by the SP (i.e., IoT case).

The nomenclature for the logical layers is borrowed from the cloud, even if the Project scope is not strictly limited to that domain. The *Infrastructure* layer includes the hardware: servers, disk arrays, network equipment, sensors, actuators and any other device that can be involved in the realization of cyber-physical system. Typical management operations at this layer include purchase, installation, electricity supply, cooling, upgrade, maintenance, disposal. Direct ownership of physical infrastructure is a big issue for small and medium

enterprises, due to the high capital investment, the quick obsolescence of the hardware, the delay to deploy new resources, the difficulty to foresee the right amount of resources to effectively cope with workload fluctuations. The *metastructure* layer is the glue between the infrastructure and the applications. It includes protocols and mechanisms for management and configuration, which allow provisioning of virtual resources. It may be limited to simple interfaces in case applications are directly deployed on the server; for example, IPMI for servers and OpenFlow/Netflow for network devices. Alternatively, hypervisors, cloud management software and orchestrators can be used for full-fledged virtualization environment; some common technologies and tools are Xen, KVM/QEMU, OpenStack, VMware vSphere, ETSI MANO, Juju, OSM. The *applistructure* layer defines the main business logic and the underlying artefacts to create the software environment. The structure of applications changes significantly for different business models. For bare metal installations, it only includes the OS, system libraries, and application software. For IaaS it should also identify the abstract resources (VMs or containers, number of CPUs, amount of RAM, disk space, NICs, network links) that are required to create the virtual environment to run the software. For PaaS, it also features message queues, artificial intelligence analysis, or notification services and whatever else is needed to build the platform; in this case, the boundary between the scope of the InP and the SP is not fixed, but depends on each specific offer (i.e., some features may be integrated in the platform or the user can select them at provisioning time). Finally, the *infostructure* layer concerns data and information managed and stored in the service. In most cases, this layer is under full control of the service provider, which business logic collects, processes, stores, and delivers data on behalf of end users; however, in case of DaaS, providing data is the main business for the RP. This does not necessarily mean that the same RP implements the full stack: indeed, as already introduced in the previous Sections, virtual resources might be provided by different RPs in a composite way. For example, data from RP1 may be hosted in a database service provided by RP2, which in turn runs in a VM created in the infrastructure of RP3. The pictorial representation in Figure 5 does not capture this distinction, because the main aspect is the sharing of responsibility between the SP and RP(s).

As general indication, the responsibility of each actor maps to the degree of control it has over the architecture stack:

- In the DaaS model, the RP is responsible for providing reliable and safe data to authorized customers. The responsibility may be partially shared with other RPs, if the data is hosted on third party's infrastructure.

- In the SaaS model, the RP is responsible for nearly all security aspects, including operation of the infrastructure and the hosted software. The responsibility for the SP is mostly limited to identity management and access control. Serverless models remove the day-to-day responsibility for updates from the cloud users, but also require to work with providers with strong track records. This model should foster cloud providers to maintain extremely high security levels, since this is necessary to building their reputation and credibility. The usage of HTTP and similar REST interfaces also removes the need for layer-2 network virtualization, hence shrinking the typical attack surface to API calls and/or HTTP(S) traffic.

- In the PaaS model, the RP manages all security aspects related to the platform, which includes the infrastructure and software facilities (like databases, domain name servers, operating systems, compilers, analytic services, etc.). The SP is responsible for everything he implements and runs on the platform (e.g., data processing, connection to his own things). However, there are some common

security services (like logging, monitoring, auditing) that lie in between, since they may concern elements operated by both actors. The risk in this case is unnecessary duplication of similar security frameworks and components.

- In the IaaS model, the RP is only responsible for the physical infrastructure, while the SP takes on the responsibility for everything he deploys and operates in the virtualized resources. Also in this case, there are intertwined security aspects.
- In the IoT/DC models, only the SP is responsible for all layers, including tampering, eavesdropping, and any unauthorized physical access to devices. In this case, the SP has the best visibility on the evolving context, including information from hardware, logs and events from applications, statistics from network appliances. However, the whole management burden and financial investment may be cumbersome for smaller businesses.

The shaded cyan area in Figure 5 highlights the visibility of the SP over the deployed services. It is clear that massive recurse to `as-a-service' models hinder the possibility to consider events and conditions that may be symptomatic of forthcoming threats to valuable assets in the overall service. Thus, **the overall proposition of GUARD is to improve visibility over digital services beyond the cyan area**. In abstract terms, the goal is to lengthen the red arrow down to the bottom layers for security purposes, while keeping the current separation of concern for management aspects, as pictorially depicted in Figure 6. According to the general concept discussed in Section 3.2, the delegation of security management to SOC or similar entities frees smaller businesses from large investments in technology and expertise, while facilitating the establishment of trust relationships that are necessary to overcome the administrative and technical boundaries between different actors.

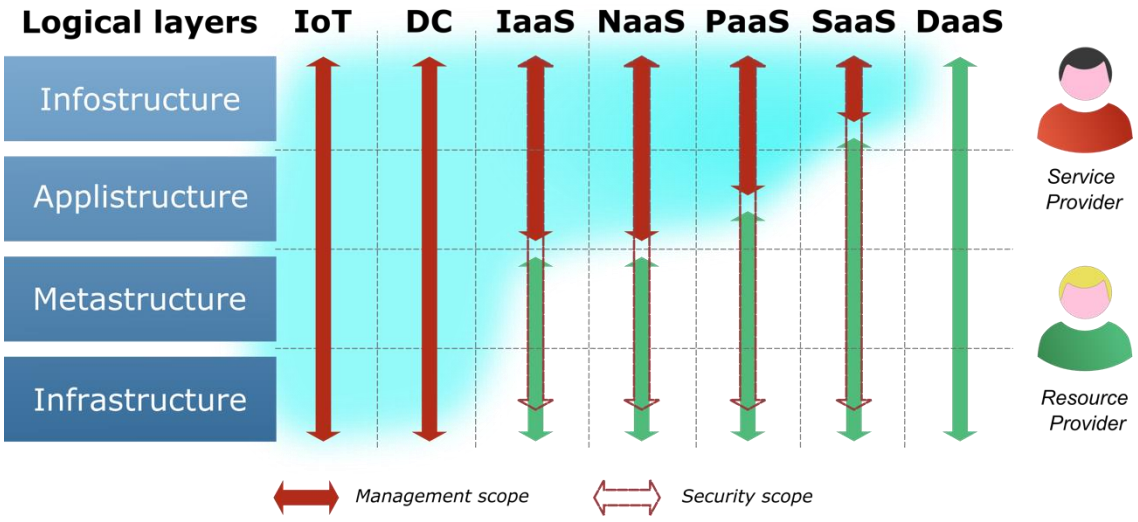


Figure 6. GUARD improves visibility over digital services by stretching the security scope of Service Providers.

Improving visibility helps creating better awareness and detection opportunities, but does not remove the responsibility of each actor. Specific Service Level Agreements (SLA) must be established between SPs and RPs, which establish the support that is offered in case of incidents. This could include additional logs for forensic analysis, timely notification of on-going or presumed attacks, escalation procedures, response time, etc. Management of procedural aspects does not fall under the scope of GUARD.

3.5 Information workflow

According to the main business roles and relationships that are identified in Section 3.2, we can identify the main information flows in the reference scenario, which are pictorially sketched in Figure 7.

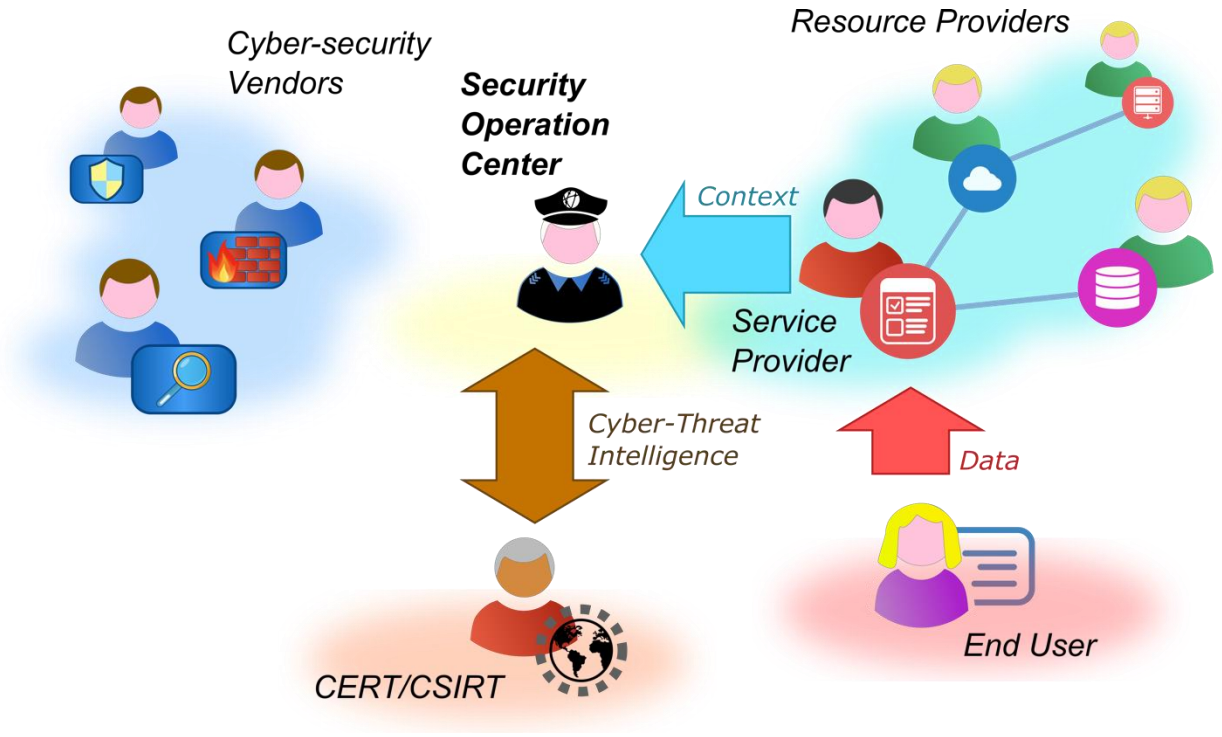


Figure 7. Main information flows expected for digital cyber-security scenarios.

EUs are the first link of the chain. They access digital services by web interfaces or dedicated terminals. While using the service, the End User will likely reveal personal or sensitive information, either in a direct or indirect way. Very simple examples of direct supply of personal information include filling up forms with personal information and uploading personal documents (passports, ids, medical reports, contracts, etc.). Sensitive information about religious, sexual, and political orientations might also be inferred from user's position and movements, web browsing history, photographs, contact lists, etc. It is therefore important to track and restrict the propagation of such data, or alternatively to anonymize or pseudonymize them in the appropriate way.

The SP creates a value chain by combining software, infrastructures, and data. The two main dimensions in a business chain (i.e., services and data) are directly reflected in the same number of security aspects: service integrity and data sovereignty. On the one hand, there is the need to know who, how, and where will process private data and sensitive information. On the other hand, there is the need to timely detect any attack or threat that may compromise the integrity, confidentiality, or availability of data and processes. Relevant information in this respect should be collected and delivered to the SOC for analysis and processing. Indeed, according to the main Project goals, the detection process is expected to be implemented in a central location, so to keep into account the broadest context while aiming to identify even the weakest correlation between apparently independent events[12]. The *context*, or better the *security context*, is the collection of logs, data, and events that can be used to detect attacks and vulnerabilities, as well as to track the propagation of user's data. The more

[12] Note that each RP is anyway expected to run its legacy security appliances locally, if it does not rely on the GUARD framework.

details are available, the more effective the detection will be. However, the overhead on communication and processing will also increase. It is therefore important that the composition and verbosity of the context be dynamically adapted to the evolving threat scenario and risk level. Regarding private and sensitive data, the framework is designed according to the following considerations:

- User's private data are never sent to the SOC, because the disclosure of this information is not necessary to implement the list of expected security services. The context shall include identification of such data and its position (provider, infrastructure, country, geolocalization), so to track its propagation within the value chain and trigger notifications to the users.
- Data should be anonymized or pseudonymized before sending it to the SOC. Many applications report the user identity in their log files, together with requested operations and maybe supplied data. The SP remains responsible to hide all the information that is not relevant for detection and analysis; in particular, they remain responsible to keep the mapping between real data and aliases that are used to distinguish different EUs.

It is likely that not all resource providers will be willing to share detailed information about their internals and monitoring data. This will negatively affect the trust in that provider, which may be discarded when selecting resources to compose the service. Hence, the GUARD concept does not strictly require any provider to implement the full monitoring API, but fosters them to do so because the more information is available, the more trust and involvement is possible in critical services.

CVs do not provide any information to the system. They are just "technology providers," since they design, implement, and deliver algorithms and software for detection and analysis of the security context. An obvious legal and ethical requirement is that their software must not keep hidden copies of data, stealthy transfer them to any remote location (for storage or processing), include backdoor for remote access and control. It might be useful to adopt common certification schemes for the origin, integrity, and reliability of the software provided by CVs[13].

The last exchange of information concerns the description of cyber-threat intelligence. Since the SOC does not hold real user data, there is no risk to disclose sensitive information. Sharing descriptions of detected, mitigated and prohibited threats or past attacks against an organization helps others to secure their infrastructures against those threats. Shared information can include among other indicators of compromise (IoCs), tactics, techniques and procedures (TTPs), data snippets, threat reports in standardized formats such as STIIX or in free text. Providing those information allows organizations to quickly adapt their security solutions against changes in the current threat landscape and to defend their devices and services against novel attacks and threats, as well as tailored sophisticated advanced persistent threats (APT).

3.6 Business workflow

GUARD will provide a security management tool for SOCs and similar entities (Figure 8). It builds on some common concepts of Security Information and Event Management (SIEM) systems, and extends them towards more flexibility and improved analysis and detection capabilities. Some basic concepts behind GUARD origin from the I2NSF proposal from IETF, which defines a framework for interfacing to multiple security appliances and translating policies into low-level configuration rules and actions [10][11]. GUARD widens the scope of I2NSF

[13] EU cybersecurity certification framework. Web site: https://www.enisa.europa.eu/topics/standards/certification?tab=details.

to generic digital services and pursues more efficiency and effectiveness in the detection process; the differences between GUARD and I2NSF are discussed in Section 6.1.

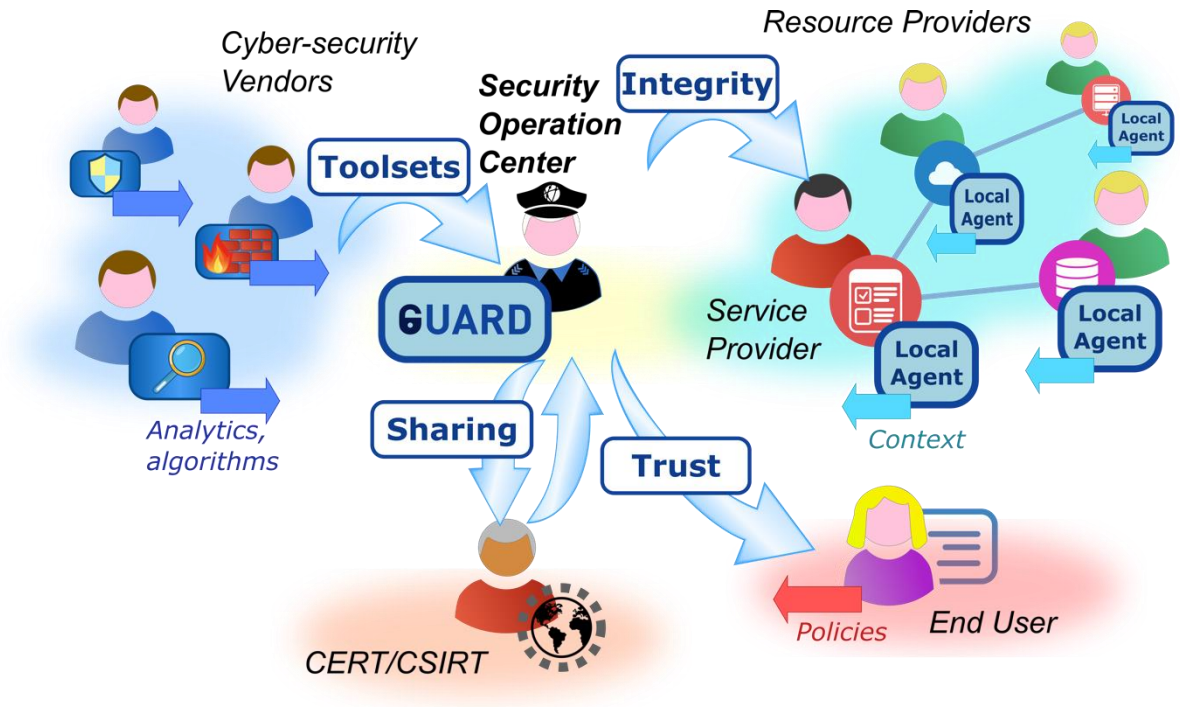


Figure 8. Business workflow in the GUARD framework.

Differently from many other toolsets, GUARD is not a stand-alone security appliance per se targeting a well-defined set of threats and attacks. Roughly speaking, GUARD can be defined as a platform for

1. running multiple security services, which include a mix of detection tools, analytics, trust and risk assessment, and data tracking. The same platform also provides programmable access to heterogeneous, broad and detailed security context collected from multiple source; the objective is to feed security tools with raw information rather than distilled events, as common practice for SIEM, so to increase the likelihood of detection and facilitate investigation;
2. facilitating their management and operation, leveraging more automation than today for notification, control/management, reaction, and mitigation.

The first step to operate GUARD is to acquire the necessary **Toolsets** of security *analytics and algorithms*. CVs become therefore technology providers for the GUARD platform. They deliver legacy appliances and innovative algorithms as packages, compliant with the GUARD architecture. There are no tight requirements about the implementation of such security tools. They may be provided as pure software instances, hardware appliances, or big data platforms. Software instances are more suitable for automatic deployment and life-cycle orchestration, which facilitate recovery from errors and partially mitigate performance constraints by horizontal or vertical scalability. Hardware appliances and big data infrastructures can easily process gigabytes or terabytes of data, nearly in run time, but require longer deployment times and present an intrinsic rigidity to be adapted to dynamic workload variations in the short/mid-term. In both cases, compliance with the GUARD architecture should be provided by software management modules that are used to control operation at run time (e.g., start/stop the service, set detection targets, change configuration parameters).

Once the set of analysis and processing tools has been defined, the GUARD platform is ready to implement security services. The target business models for SOCs include **Integrity** and **Trust** services and two market segments: SPs and EUs, respectively.

The business relationship with EU will be mostly oriented to give trust in the digital service provided by one or more SPs. In this case, GUARD will be used as the technical mean to implement the following processes:

- *privacy management*: the EU can verify that personal and sensitive data are processed according to her preferences. This will replace current procedures where users have to sign informed consent, or check off specific options on web pages and software. Such procedures are cumbersome to implement and often vague in terms and scope. For example, it is common to ask consent for sharing data with unclear linked or commercial partners for additional services; in case of software procedures, users may be presented with many dialogue boxes and tempted to accept without reading the messages.
- *data tracking*: the EU identifies valuable data that she loads into the service and has visibility over its location (provider, infrastructure, country). Additional features may include links to relevant regulations and contractual agreements that apply.
- *data management*: the EU defines where data can be propagated (provider, service, infrastructure, country) and how (implicit or explicit authorization at each transfer), invokes some basic management actions (remove data), and gets notified about relevant events (requests to access, share, transfer data).
- *trustworthiness*: trust in a digital service would also require to verify the identity, reputation and credibility of involved providers, the country where they are established and operate, their certifications, their reliability, their compliance to national or international regulations, potential vulnerabilities, the integrity of their services, and so on. Part of these properties is statically bound to the service and part is dynamically computed by GUARD, but this is transparent to EUs.

  GUARD is expected to improve current practice by letting users to define their *policies*, i.e. a set of rules that govern what data can be transferred to whom. A very simple example in the eHealth domain: users may indicate their medical records to be shared with specific physicians, all physicians of a unit, all physicians of one or more hospitals, external private medical or insurance companies, research institutes, and so on. Going on with the same example, users would not give their consent to share data with organizations that have been repeatedly subject to ransomware, DoS, data leakage.

The business relationship with SPs is much more aligned with existing consulting practice for integrity of services and resources, including their availability. The SOC, which may be an internal technical unit of the SP or an external entity, analyses data and log to identify vulnerabilities, investigates potential threats, and detects attacks. The challenging innovation brought by the GUARD framework is the extension to multi-domain and multi-tenancy value chains, encompassing data and context from multiple resource providers. Technically speaking, GUARD will rely on the availability of programmable embedded monitoring, inspection, and enforcement tools (**Local Agents**), without putting any technical constraints on their implementation. It just requires the usage of standard APIs, which definitions are part of the Project workplan, that expose what context is available and how its generation can be controlled (source, frequency, verbosity, format, etc.). In this respect, the SOC must set up business agreements with many resource providers; this can be done reactively, according to the topology of the chain, or proactively, by selecting the set of providers that will likely be involved in the composition of digital services. Only the second option can realistically follow the evolution of the chain in nearly real-time, though it would be difficult to effectively select the right set of providers. The specific agreements

shall define what kind of information can be retrieved and what control actions are allowed to the SOC, collectively indicated as the *context*. It is likely that information as the identity of the provider, digital certificates, offered services, and security measures will be shared by any entity; however, indications about versions of installed software, security patches, logs from applications and daemons, events, network traffic statistics, and configuration files could be considered sensitive information for some organizations. A thorough analysis of motivations and incentives to take part in such "security ecosystem" falls under the scope of exploitation, but a few considerations can already be given:

- The internal security context will only be shared with trusted and reliable entities, i.e., they will not be publicly available. The GUARD architecture will support identity and control access mechanisms to rule and enforce access to any internal information.
- Authorized SOCs might belong to trusted customers (e.g., organizations that rent VMs and/or storage space in a cloud infrastructure), which adhere to non-disclosure agreements as part of their business contract.
- Externalization of security services is a growing practice, especially for small and medium organizations, so SOCs already provide the necessary technical and administrative guarantees to process security-related data on behalf of their customers. However, the GUARD approach implicitly brings the problem that different SOCs could be used by each SP, hence RPs should set up multiple agreements heading to different SOCs. Certification and labelling schemes could be established by law to address most of the administrative issues in managing multiple security providers; this is somehow similar to on-going processes for digital identities and digital signatures with legal validity[14].
- Sharing security-related data with external organizations brings better opportunity for RPs to detect attacks from apparently uncorrelated events, which might happen in different domains. A very trivial example is phishing against a user on his personal email account (maybe used at home), to gain credentials for his work account. The knowledge of software configurations also helps identify vulnerability to new attacks that have been just applied to similar infrastructures or resources. Again, a very simple example: a zero-day attack against an OpenStack installation suggests a new threat for all other providers running the same release and similar configuration, even before the attack has been deeply investigated.

In addition, a **Sharing** service should be set up with CERTs/CSIRTs to support at best the operation of the platform. Indeed, the effectiveness of every SOC is largely affected by the knowledge of known vulnerabilities, potential threats, and attack patterns. The mutual benefits of sharing cyber-threat intelligence for private companies and organizations and the whole society has already being discussed for many years [12]; however, despite of massive digitalization in all economical and business sectors, sharing of information about cyber-threats and security incidents is still largely based on paperwork and emails [13]. National CSIRTs provide support at the procedural level, foster cooperation and sharing of best practice, but do not cover technical issues [14]. This will certainly delay the knowledge of new threats and attacks, as well as the elaboration of remediation actions and countermeasures for every different context. Similarly to what already discussed about the business relationship between SOCs and SPs, there are often concerns and reluctance from private organizations to

[14] In Italy, several organizations already provide digital identities for accessing online services from the public administration and private companies, under the common framework of SPID (*Servizio Pubblico di Identità Digitale* - public digital identity service). The framework is ruled by the Agency for Digital Italy (AgID), a technical body of the Presidency of the Council of Ministers.

disclose evidence of attacks, because they fear this may affect their reputation and give visibility about their cyber-vulnerabilities. The GUARD platform will automatically import/export threat knowledge from/to external sources, but this clearly require the preliminary definition of business agreements on the content, usage, and dissemination of such information.

3.7 Legal implications

A digital value chain typically heads to a single SP, who gathers resources from multiple providers and offers a unified service interface to the EU. Some elementary examples of digital chains are the numerous web services that already leverages identities and authentication services provided by big Internet players (i.e., Google, Facebook, Twitter, Microsoft) and large associations (e.g., IEEE, Orcid)[15]. In many cases, however, the EU does not have visibility of which digital resources (cloud, storage, devices, data bases) are used to run the service, their owners, and their location.

Despite the number of RPs that may be involved in the value chain, the SP remains the only contact entity for the EU. It is therefore responsible for any contractual and legal aspects, including management of personal data that are deliberately supplied by users or inferred by technical means (for instance, by acquiring the position or processing requested content and invoked actions). According to the GDPR, the EU unquestionably represents the **Data Subject**, while the SP amounts to a **Data Controller**, which may delegate multiple RPs as **Data Processors**. In this scenario, GUARD implements the technical solution to digitize the whole workflow necessary to define data management policies and to exercise the owner's rights (see Figure 9).

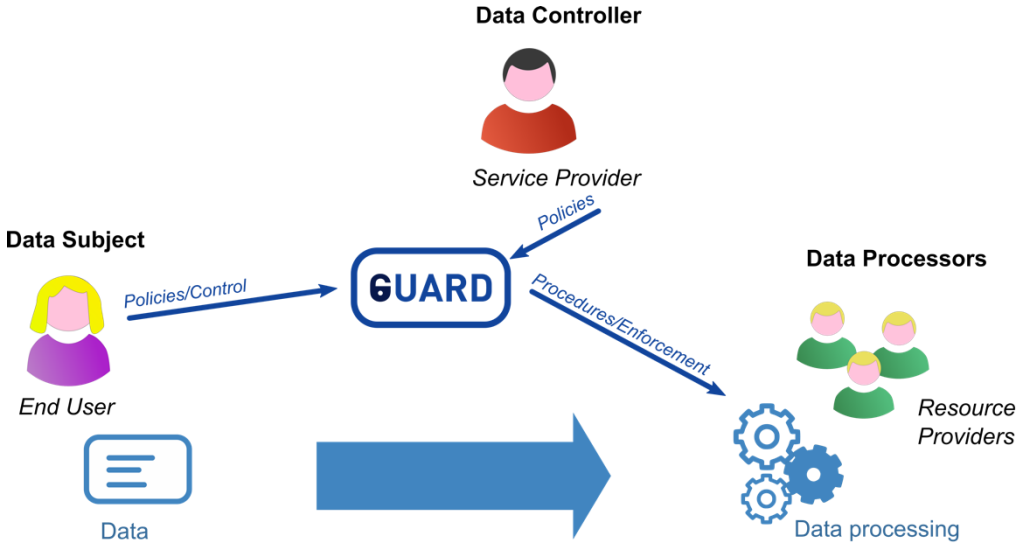


Figure 9. Positioning of GUARD with respect to GDPR roles.

The implementation of the GUARD framework requires its compatibility with all relevant Data protection legislation. The consideration of privacy and data protection requirements is of fundamental importance for the smooth operation of the framework. Nowadays the provision of any digital service to natural persons should be accompanied by the careful application of safeguards aiming to guarantee that the quality of the product/ service offered would not compromise individuals' rights and freedoms. The current section of this Deliverable aims to examine the relevant data protection requirements and contribute to their implementation in the GUARD framework. Moreover, as the GUARD framework will be used as a technical solution to digitize the whole

[15] See for example, the Overleaf service used by many LaTeX users: https://www.overleaf.com/login.

workflow it is essential to assure the system has implemented all technical and organisational measures to comply with the Data protection legislation. Therefore, the implementation of the GUARD solution should be in compliance with all requirements and responsibilities imposed on the controller.

3.7.1 Main notions

The examination of the relevant GUARD actors requires the explanation of the applicable definitions used by the legislation. Data subject is the natural person whose personal data is being processed. Personal data is defined as any information relating to an identified or identifiable natural person[16]. This includes the name, an identification number, location data, online identifier including the IP address and cookies ID. It should be mentioned that pseudonymized data is still personal data for the purposes of GDPR. Pseudonymisation is defined as the processing of personal data in such a way that the data can no longer be attributed to a specific data subject without the use of additional information, as long as such additional information is kept separately and subject to technical and organizational measures to ensure non-attribution to an identified or identifiable individual[17].

Anonymised data, however, does not fall in the scope of the GDPR. Anonymisation is the process of removing personal identifiers, both direct and indirect, that may lead to an individual being identified. Once data is anonymized the individuals will no longer be identifiable.

As already mentioned above, the EU will be the data subjects of the relevant service provided within the GUARD framework. Being the first link in the chain, the EU will inevitably share personal data necessary for the operation of the service used. Therefore, they are entitled to all rights of data subject listed in Art.13-22 of the GDPR described below.

The data protection law of the European Union does not apply to the processing of personal data which concerns legal persons and in particular undertakings established as legal persons, including the name and the form of the legal person and the contact details of the legal person[18]. Therefore, the requirements of the GDPR would not directly concern data of legal persons. However, in business-to-business (B2B) relations personal data of legal representatives, employees, contact persons etc. could be processed in the course of the communication related to the provision of services. This implies that the usage of GUARD in B2B relations also requires efforts to achieve compliance with the GDPR regime.

Data controller is defined as a natural or legal person, public authority or body which, alone or jointly with others, determines the purposes and means of processing of personal data[19]. As mentioned above the Service provider (SP) should be acting as the Data controller. This however is highly dependent on the specifics of the particular processing operation, so assessment should be made in each case of processing of who is responsible for the determination of the purposes and means of processing of personal data. It is worth also mentioning the figure of the joint controllers – where two or more controllers jointly determine the purpose and means of processing, they are considered joint controllers. They should determine their responsibilities and obligations for compliance with the GDPR in a transparent manner[20]. Their responsibilities should be governed by an agreement between them. This arrangement should be made available to data subjects and it may be used for

[16] General Data Protection Regulation, Article 4 (1)
[17] General Data Protection Regulation, Article 4 (5)
[18] General Data Protection Regulation, recital (14)
[19] General Data Protection Regulation, article 4 (7)
[20] General Data Protection Regulation, Article 26

the designation of a contact point for the data subjects. In the context of GUARD framework, this means that there might be scenarios where the SP and the entity managing GUARD act as joint controllers following the understanding established by the Court of Justice of the European Union[21] and reiterated by the European Data Protection Supervisory[22]. This would be the scenario where SP provides pseudonymised data to the Security Operator which is using GUARD. Since pseudonymised data still falls within the scope of the GDPR, the Security Provider must take measures to ensure compliance with its rules. According to the latter whenever a situation where the purposes and the essential means of processing are determined by two entities, they act as joint controllers even in cases where one of them does not have access to the processed personal data[23]. In our case, the GUARD framework could be utilized in different SOC architectures which determine the purpose of its use. This would make the entity managing the framework a joint controller together with the respective SP. On a practical level this means that the agreement concluded between the SP and the Security Provider operating GUARD should include provisions regulating the responsibilities of each controller, in particular in terms of data subject rights[24].

Data processor is defined as a natural or legal person who processes personal data on behalf of the controller[25]. As the SPs may need to include relevant RPs in order to ensure the provision of the relevant service, this may result in data processing by the RPs. In these cases, they will act as a data processor who will process the personal data on behalf of the SP. The relationship between the controller and the processor should be governed by a contract between them. This contract will lay out the subject-matter and duration of processing as well as the nature and purpose of the processing, type of personal data and categories of data subjects and the obligation and rights of the controller[26]. As the operation of different services may result in involvement of additional actors (resource providers), they may be required to process personal data in order to facilitate the provision of services. In such cases the relationship between the controller and processor will be recorded in a written contract in accordance with the rule of Article 28 (9) GDPR.

In Section 3.5 the flows of data between the GUARD actors are explained. According to it, all data will be collected by the SP and not sent to the SOC because the disclosure of the information will not be necessary to implement the list of expected security services. However, in cases when data should be provided to the SOC these transfers should be in line with the GDPR rules. The SP should remain responsible to hide all the information that is not relevant for detection and analysis and to keep the mapping between real data and aliases that are used to distinguish different EUs. These means the GUARD system will not generally receive any personal data of users from the SP. Also, it must be noted that sensitive data of users will not be transferred from the SP to the Security Provider. The users accessing the GUARD Dashboard however will inevitably share some personal data. Thus, the system should respect the relevant Data protection legislation.

It should be noted that these are just preliminary considerations which will be adopted accordingly with the development of the GUARD project. The considerations should be reviewed in D2.3[27].

[21] Judgment of the Court (Grand Chamber) of 5 June 2018, *Wirtschaftsakademie Schleswig-Holstein*, C-210/16
[22] EDPS Guidelines on the concepts of controller, processor and joint controllership under Regulation (EU) 2018/1725, p. 24
[23] Judgment of the Court (Grand Chamber) of 10 July 2018, *Tietosuojavaltuutettu*, C-25/17
[24] General Data Protection Regulation, Art. 26 (1)
[25] General Data Protection Regulation, Article 4 (8)
[26] General Data Protection Regulation, Article 28 (3)
[27] D2.3: GUARD Integrated Platforms and Revised Architecture

3.7.2 Principles

Article 5 of the GDPR lays down the basic principles governing the processing of personal data. These key principles set the fundamentals of compliance with the provisions of the GDPR. They are of crucial importance when it comes to service design, laying the foundations that must be considered by data controllers when planning such operations that include the processing of personal data of natural persons. Therefore, GUARD acknowledges the importance of reviewing them during the design phase of the framework in order to ensure the development of the service. GDPR introduces the concepts of "Privacy by Design" and "Privacy by Default". Although they are not listed in the principles of the GDPR in Article 5, they are of fundamental value when it comes to the design and creation of a system. Adhering to the principles will lead to the secure and compliant operation of the GUARD framework.

The principle of lawfulness, fairness and transparency is the first one established in Article 5 of the GDPR.

Lawfulness requires the identification of a specific legitimate ground for processing. These are the "legal basis" for processing which differ depending on the purpose for processing. Lawfulness in the context of the functioning of the GUARD framework will be achieved with assuring that every data processing operation is carried out pursuant a legal basis as listed either in Art. 6 and/or Art. 9 of the GDPR.

Fair and transparent processing require the data subject to be informed of the processing and to be provided with information about this processing[28]. Providing the data subject with information gives them the opportunity to make informed decisions regarding their personal data. Fairness also means that the data processing should be done with accordance to the reasonable expectation of the subject as to how the data will be used.

Transparency is an essential component in the relationship between the data subject and the data controller. All information related to the processing activities should be provided to the data subjects in plain and simple language. This information includes information on the identity of the controller and the purposes for processing, the legitimate interest of the data controller if this is the legal basis for processing, the recipients of the personal data, if any, and the fact that the controller intends to transfer personal data to a third country or international organisation, and whether this is based on adequacy decision or relies on the appropriate safeguards. Additionally, to ensure transparent processing the controller should provide information on the period for which the personal data will be stored, or the criteria that will be used to determine that period, the existence of the right to request from the controller access to and rectification or erasure of personal data or restriction of processing and the right to object to processing, the right to data portability, the existence of the right to withdraw consent at any time where the processing is based on the consent of the subject and the right to lodge a complaint to the supervisory authority. Information whether the provision of personal data is a statutory or contractual requirement, or a requirement necessary to enter into a contract should also be provided as well as whether the data subject is obliged to provide the personal data and of the possible consequences of failure to provide such data. The transparency principle requires the awareness of data subjects of the risk, rules, safeguards and rights regarding their personal data[29]. The implementation of this principle is essential to the GUARD framework. Processing of personal data for the provision of any services should be lawful and all EUs must be able to access the information related to the processing. GUARD will make sure that a legal basis for processing in accordance with Art.6 and/or Art. 9 is always in place. The GUARD Dashboard will incorporate a Privacy policy pursuant Art. 12-14 of the GDPR which will help the establish of trust between the

[28] General Data Protection Regulation, Recital (60)
[29] General Data Protection Regulation, Recital 39

EUs and the SP. This policy will explain to the user what kind of personal information is being processed, for what purposes and so on. The Policy should follow a layered approach, in line with the best privacy practices for making the processing transparent.

The **purpose limitation** principle requires all processing of personal data to be done for specific purposes and not further processed in a manner incompatible with those purposes[30]. In the context of GUARD all processing should be relevant to the achievement of the objectives of the framework, namely advanced end-to-end assurance and protection of business service chains, improvement of the detection of attacks and identification of new threats.

The data processed by the GUARD system must be adequate, relevant and limited to what is necessary in relation to the purposes of processing in order to comply with the **data minimisation** principle established in Article 5 (1)(c) GDPR. Since GUARD platform is programmable, it would be regulated in the agreement between the SP and SOC what types of data will be requested so that the performance of the service of question is of the necessary quality.

The **data accuracy** principle requires the taking of reasonable steps to ensure that personal data is kept up to date and inaccurate data is erased or rectified without undue delay[31]. The provision of services within the GUARD framework relies on the accuracy of the collected data. The system will therefore provide the users with the possibility to rectify their data when necessary or delete it/ or as the case may be depending on the functionalities used to operate the GUARD solution – to request its rectification/deletion by the controller when GDPR's preconditions to exercise these rights are met. GUARD system will also be designed to allow (if EU consents thereto) for the ex officio update of end users' data and collection of information from relevant authorities.

The **storage limitation** principle states that personal data should not be kept longer than needed[32]. The GUARD framework should be designed to have a retention policy which lists the types of record or information kept by the system, what will they be used for and for how long should they be kept. The system should have the ability to automatically identify and erase the data that is no longer needed and to perform periodical reviews of the need for the storage of personal data. Personal data should be stored only for the period the EUs are using the relevant service and then be erased or anonymised.

The **integrity and confidentiality** principle entail the processing in a manner ensuring appropriate security and confidentiality of the personal data, including the prevention of unauthorised or unlawful access and against accidental loss, destruction or damage[33]. The GDPR lists some of the measures that could be taken in order to ensure the security of the processing. They include pseudonymisation and encryption of personal data. When assessing the appropriate level of security, the risks that are present by the processing should be taken into account, including the risks of accidental or unlawful destruction, loss, alteration, unauthorised disclosure or access to personal data transmitted, store or otherwise processed[34]. The GUARD framework will be designed to

[30] General Data Protection Regulation, Article 5 (1) (b)
[31] General Data Protection Regulation, Article 5 (1) (d)
[32] General Data Protection Regulation, Article 5 (1) (e)
[33] General Data Protection Regulation, Recital 39
[34] General Data Protection Regulation, Article 32 (2)

ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services, regardless of the user's location responding to the already established requirements' list as reported under D2.1.

The **accountability** principle requires the controllers to be able to demonstrate compliance with the other principle established in Article 5 (1) of the GDPR. This means that the controller shall keep appropriate documentation to ensure and be able to demonstrate that processing is performed in accordance with the Regulation[35]. These procedures and mechanisms would vary according to the risks represented by the processing and the nature of the data[36]. As Article 30 of the GDPR requires the maintaining of a record with all processing activities by the controller, the GUARD system should be designed to keep data processing records with all the information listed in Article 30. The system will also keep a record of the log files for a better management, accountability & transparency. This should be a separate data base serving the purpose to give accountability on the activities performed within the framework. This will give the opportunity for a wide security-related information to be collected.

Article 25 of the GDPR provides for two pivotal concepts for project planning – **Data protection by design and by default**. Data protection by design means that the controller should implement appropriate technical and organizational measures to implement the data processing principles and to integrate the necessary safeguards into the processing in order to meet the requirements of the GDPR and protect the rights of data subjects[37].

Data protection by default requires the implementation of appropriate technical and organizational measures to ensure that by default only the personal data which are strictly necessary for each specific purpose of the processing are processed[38].

Therefore, GUARD should be developed in accordance to these principles. Data protection is being taken into account in the early stages of the development of the GUARD framework as it is considered in the elicitation of requirements phase and the elaborating of the architecture of the system.

3.7.3 Legal basis for personal data processing by GUARD

3.7.3.1 Legitimate interest

Article 6 (1) (f) GDPR provides that processing shall be lawful when it is necessary for the purposes of the legitimate interest pursued by the controller or a third party with the exception that the interests of the data controller do not override the fundamental rights and freedoms of the data subject. Such legitimate interest could exist where there is a relevant and appropriate relationship between the data subject and the controller in situations such as where the data subject is a client or in the service of the controller. When personal data is being processed on the "legitimate interest" ground, the individual has the right to object at any time to the processing on grounds relating to his/her particular situation[39]. The controller then should no longer process the personal data unless they demonstrate compelling legitimate grounds for processing which override the interests, rights and freedoms of the data subject or for the establishment, exercise or defence of legal claims.[40]

[35] General Data Protection Regulation, Article 24 (1)
[36] Article 29 Working Party, Opinion 3/2010 on the principle of accountability, WP173, Brussels, 13 July 2010, para 12
[37] General Data Protection Regulation, Article 25 (1)
[38] General Data Protection Regulation, Article 25 (2)
[39] C. Giakaumopoluos, G. Buttarelli, M. O'Flaherty, Handbook on European data protection law 2018 edition, p. 158
[40] General Data Protection Regulation, Article 21 (1)

In the context of GUARD, it seems that the ”legitimate interest” ground would be the main legal basis for data processing, in particular in cases where personal data is being processed pursuant an agreement between GUARD and a SP, commissioning the framework to serve as a technical means to secure data flow between the different actors. In this B2B context, the commercial interest of the organisations should be a valid legitimate interest. The contractual agreement in place would serve as a legitimate interest for GUARD to perform processing of the personal data used by the SP.

3.7.3.2 Contract & consent as a legal basis for the personal data processing under GUARD.

According to the GDPR, processing will be lawful when necessary for the performance of a contract to which the data subject is a party or in order to take steps at the request of the data subject prior to entering into a contract[41]. In the context of the GUARD framework, the performance of a contract as a legal basis for personal data processing could be suitable legal ground in the cases where the provided service in question is following upon conclusion of a contract between the EUs and the SP. As GUARD will act as the technical mean to implement different processes of service provision, this legal basis could find its applicability in the registration process used for the provision of a certain service.

Consent is another one of the legal bases for lawful processing listed in Article 6 of the GDPR. The consent should be freely given, specific, informed and unambiguous indication of the data subjects’ wishes to agree to the processing of personal data related to him/her[42]. Additional conditions for the valid consent are given in Article 7 and they include the right to withdraw consent which should be as easy procedure as the giving of the consent.

Consent could also be used as a basis for processing used within the GUARD framework if the SP enables the EU with alternative to a process that is already in place. For example, if a service is to be digitized, requiring input of personal data to function, but its existence in the paper-based world would not cease, then consent would be the applicable legal basis. In such a scenario, the system should be utilized to provide the EUs with information sheet before any processing activities have occurred and then require their consent. The consent forms should be integrated in the system and should be drafted in plain and simple language allowing the EUs understand easily what processing activities will be occurring, for what purpose etc. The system should also implement an easy way for data subject to withdraw their consent.

3.7.3.3 What about sensitive data?

The GDPR uses the term “special categories of personal data” – and they are defined in Article 9 as “personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, and the processing of genetic data, biometrics data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person’s sex life or sexual orientation”[43]. In practice, the abovementioned term and ‘sensitive data’ are interchangeable terms.

As already mentioned before, EUs are likely to disclose sensitive information depending on each Use Case and the specific service in question. As GUARD will be used as the technical mean to implement the processes of privacy management, data tracking and data management, specific procedures should be placed to guarantee the lawfulness of the data processing. The processing of special categories of data is prohibited in principle[44].

[41] General Data Protection Regulation, Article 6 (1) (b)
[42] General Data Protection Regulation, Article 4 (11)
[43] General Data Protection Regulation, Article 9 (1)
[44] General Data Protection Regulation, Article 9 (1)

However, Article 9 provides for an exhaustive list of exemptions which amount to lawful grounds for processing of these types data. One of the exemptions is the explicit consent of the data subject. Therefore, the data subject will be allowed to define the data propagation. When sensitive data is being used for the provision of a service, where appropriate, the data subject will be asked for their explicit consent. In any case, depending of the specifics of the respective service, an assessment must be made in each individual Use Case as to whether one of the exemptions listed in Article 9(2) exists, otherwise no processing of sensitive data shall take place.

Article 9 (2) does not recognize the "legitimate interest" as such and the contract as an exception to the general prohibition to process special categories of data. Therefore, in the cases where "legitimate interest" or "performance of a contract" is the legal basis used for the provision of a service protected by GUARD should aim at obtaining explicit consent in accordance with the conditions for valid consent in the GDPR for the processing of sensitive data or should ensure the presence of one of the other suitable legal grounds listed in Article 9(2)[45].

3.7.4 Data subjects' rights and how are they addressed

The GDPR provides the following rights for individuals:

1. The right to be informed – Data subjects have the right to be informed about the collection and use of their personal data. This responds to the transparency principle discussed above. Articles 13 and 14 of the GDPR specify what data subject are entitled to be informed about - identity and contact of the controller, purpose of processing, legal basis, information how data will be managed and stored, etc. All information should be provided in clear and plain language. This information should be provided at the time of the obtaining of the personal data. A privacy policy will be adopted for the GUARD Dashboard allowing the data subject to be easily provided with the relevant information.
2. The right of access – Data subjects have the right to receive confirmation whether personal data concerning him/her is being processed as well as other supplementary information about the processing[46]. The EUs in the GUARD business flow will be able to make requests and receive the requested information. These procedures will be established in the Privacy Policy used by the specific Service Provider.
3. The right to rectification - individuals are entitled to have inaccurate personal data rectified or completed if it is incomplete[47]. This right is linked to the accuracy principle of the GDPR - Article 5(1)(d).
4. The right to erasure – also known as the "right to be forgotten" gives individuals the opportunity to have their personal data erased when the personal data is no longer necessary, the consent has been withdrawn, the subject has objected to the processing or it was unlawful, deletion is required to comply with a legal obligation or personal data were processed to offer information society services to a child.[48]
5. The right of restriction of processing – Data subject have the right to request the restriction of processing of their personal data in certain circumstances[49]. Restriction of processing means that the data controller will be permitted to store the personal data, but not use it.

[45] Article 29 Working Party, Guidelines on consent under Regulation 2016/679, p. 19
[46] General Data Protection Regulation, Article 15 (1)
[47] General Data Protection Regulation, Article 16
[48] General Data Protection Regulation, Article 17
[49] General Data Protection Regulation, Article 18

6. The right to data portability – The right to data portability allows data subjects to obtain and reuse their personal data for their own purposes across different services[50]. This right only applies when the processing is based on consent or on contract and the processing is carried out by automated means[51].
7. The right to object – Article 21 of the GDPR gives data subjects the right to object to the processing of their personal data in certain circumstances. They have the right to object to the processing of their personal data if it is for direct marketing purposes. They also object if the processing is for: a task carried out in the public interest or in the exercise of official authority vested in the controller; or the processing is based on the legitimate interests pursued by the controller.
8. Rights in relation to automated decision making and profiling - Article 22 of the GDPR has additional rules to protect individuals in cases of carrying out solely automated decision-making that has legal or similarly significant effects on them.

The GUARD framework should be developed in a way that empowers the data subjects to exercise their rights under the GDPR via the user interface. They should be able to automatically request information, access to their data, rectification, erasure, etc. Although in principle the GUARD framework does not envisage the performance of individual decision making as per Article 22, if – in explicit cases – such is to be conducted, it should be based on one of the exceptions listed in Article 22 (2) and suitable measures to safeguard the data subject's rights and freedoms and legitimate interests as required by GDPR shall be implemented.

3.7.5 Rules on security of processing

Article 32 of the GDPR established the obligation of the controller to implement appropriate technical and organisational measures to ensure a level of security appropriate to the particular risks. These measures should aim to ensure that the systems maintain confidentiality, integrity, availability and resilience[52] and that they are able to restore the availability of and access to personal data in the event of data loss in a timely manner[53]. A process for testing, assessing and evaluating the effectiveness of the measures to ensure the security of processing should also be in place[54]. All of the abovementioned measures should be implemented in the GUARD Framework.

3.7.6 Data transfers and Disclosure of Information to Law Enforcement Authorities

The European Union law allows for personal data transfers to third countries, provided that certain conditions and safeguards are in place. In the GUARD context we assume that no transfers of personal data will be taking place outside Europe. However, in cases of transfers, appropriate tools will be adopted such as standard data protection clauses.

It is worth mentioning that processing of personal data by competent authorities for law enforcement purposes is outside the scope of the GDPR. They operate under a separate regime - Directive (European Union) 2016/680 and the national legislations implementing it. Therefore, these rules cannot be considered relevant legal framework for the GUARD solution. Theoretically, it is possible that certain relations might arise between SP using GUARD and Legal Enforcement Agencies (LEAs) – e.g. where LEAs require the disclosure of personal data from the controller implementing GUARD solution. In such a case, the legal basis for such disclosure would be Article 6(1)(c) – compliance with a legal obligation. The controller, however, does not proactively collect any data requested from the LEAs on its behalf (i.e. does not in any way act as its processor – the LEA remains

[50] General Data Protection Regulation, Article 20 (1)
[51] General Data Protection Regulation, Article 20 (1) (a), (b)
[52] General Data Protection Regulation, Article 32 (1) (a)
[53] General Data Protection Regulation, Article 32 (1) (c)
[54] General Data Protection Regulation, Article 32 (1) (d)

controller with official authorities which is subject to Directive (European Union) 2016/680 and the national legislations implementing it), but simply discloses the already collected data to fulfill its legal obligations to cooperation to LEAs.

3.7.7 Opinion on the Data Protection Impact Assessment conduct

Data Protection Impact Assessment (DPIA) is a process aimed at helping the data controller to identify and minimise the data protection risks of a project. DPIA should be performed where the processing is likely to result in a high risk to individuals. This includes some specified types of processing – automatic decision making with significant effects, systematic monitoring, processing of sensitive data on a large scale, use of innovative technological solutions etc.

At the current stage of the GUARD development, it is of importance to acknowledge the potential need of the performance of a DPIA. DPIA should be performed under the scope of WP6 where the GUARD platform will be validated and demonstrated. This process would most likely be required in view of Use Case #2: eHealth which envisions the processing on a large scale of special categories of data which is one of the cases listed in Article 35(3) as requiring the performance of a DPIA.

The DPIA will be performed in accordance with the provisions of Art. 35 of the GDPR, as well as the guidance issued by the competent supervisory authorities – Garante per la protezione dei dati personali (Italy) and Landesbeauftragter für den Datenschutz Sachsen-Anhalt (Lower Saxony, Germany).

3.7.8 Legal regime of Cybersecurity

The Directive on security of network and information systems (NIS Directive) is the first piece of European Union-wide legislation on cybersecurity[55]. The Directive's main purpose is to achieve a common high level of security of network and information system in the European Union[56]. The Directive applies to two groups of subjects - operators of essential services and digital service providers. They are both defined in Article 4. The Directive establishes some requirements for both Operators of Essential Services (OES) and Digital Service Providers (DSPs). They must identify and take appropriate technical and organizational measures to secure their network and information systems and manage the risks posed to them[57]. They also must take measures to prevent and minimize the impact of incidents affecting the security of the systems and notify the competent authorities without undue delay of any incidents having substantial impact on the provision of their services[58]. As GUARD framework may be used from both OES and DSPs it should comply with both the security measures established in the GDPR and the NIS Directive and the national legislations implementing it.

[55] https://ec.europa.eu/digital-single-market/en/network-and-information-security-nis-directive

[56] Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union, Article 1 (1)

[57] Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union, Article 16 (1)

[58] Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union, Article 16 (2)(3)

4 Threat model

The creation of digital processes requires the composition of multiple heterogeneous services, resources, and infrastructures over the Internet, so it inherits most of common threats for public networked systems. The presence of multiple administrative domains and the high dynamicity in connecting and disconnecting digital entities are among the most challenging issues to address for secure and trusted execution of applications and services.

According to the main technical concept (Section 2.4), the GUARD framework is conceived as a platform to run detection services and does not represent itself a stand-alone security appliance. In that sense, the bare platform can neither detect nor mitigate any attack. This is only possible when some detection, analysis, or reaction algorithms are loaded on the platform and properly fed with the relevant context. However, the distributed nature of the framework and the need for remote collection and local processing do not allow to perform any possible monitoring and inspection tasks, for practical and efficiency matters. It is therefore necessary to limit the range of events, data, and measurements that should be collected by the system, according to the set of threats that are more relevant for the envisioned application scenarios. It is therefore necessary to discuss different threat models:

- A threat model for the GUARD framework, which impact the design of GUARD interfaces and collection/control technologies. This represents a superset that limits the kind of security services that can be run on the platform.
- Specific threat models for each security service. Every detection, analysis, and mitigation algorithm is expected to define its scope, by identifying the set of threats and attacks that it tackles.

This approach contributes to increase the flexibility of the platform, which can be deployed gradually according to the evolving needs of potential customers. In addition, it better supports multiple business models, leaving room for multiple configurations and integration options from system integrators, which may deliver turnkey solutions tailored to each customer.

This document only covers the more general threat model for the GUARD platform, which is necessary at this stage for the following reasons:

1. to define the architectural elements that must be designed and developed to retrieve the relevant security context from the execution environment;
2. to drive the definition of the set of detection and analysis algorithms.

Each algorithm will then be responsible to restrict the set of threats and attacks that fall under its scope, by deriving its specific threat model.

4.1 System model

Digital value chains are conceived to implement data transformation, from their creation up to their destruction. The typical data lifecycle follows a quite mix of circular and straight transformation, as shown in Figure 10. Every transformation can be mapped to one or more functions implemented in the underlying infrastructure. However, digital value chains are usually more complex than this basic model, since they overlap the transformation of multiple data streams, including their aggregation and scission.

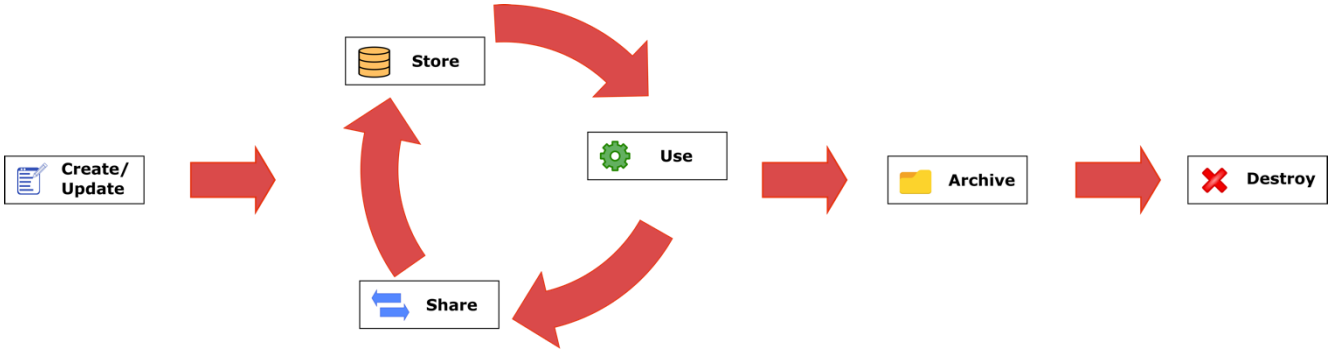


Figure 10. Typical life-cycle for data, from creation to destruction.

An illustrative yet not exhaustive example in this direction includes the following classification, which is adopted in the mixed model depicted in Figure 11:

- *data sources* generate data: they can be sensors and other IoT devices, databases, context brokers, etc.;
- *data bearers* transport and deliver data: they can be virtual network slices (built over NFV infrastructures), message brokers, message queues, etc.;
- *data processors* transform the data: they can have different input and output ports, connected to different nodes with unidirectional links, or a single port that connects to the same node with a bidirectional link;
- *data storage* conserves data: they include persistent storage (e.g., databases) and ephemeral caches;
- *data consumers* are the final users of data: files, IoT devices (actuators), storage systems, user's terminals, etc.

Since there is no explicit limitation on the kind of data, the above categories can be applied to a very broad set of real cases, from industrial processes to multimedia and AAA mechanisms.

Our model therefore follows a data-driven approach, where digital services are represented as nodes of an oriented graph and allowed data flows between them are indicated by links. Figure 11 depicts the logical model for a digital value chain. Despite the term "chain" suggests a one-dimensional linear structure, more complex structures are realized in practice, where multiple streams of data flow from heterogeneous sources through several manipulation stages up to one or more final consumers. The intermediate stages may transform a single flow, mix two or more flows, and take data apart into multiple sub-flows. The processing pipelines may be executed online or offline, with intermediate storage or caching stages, and may be conditional, i.e., the path taken by data can change at run-time based on their content, the result of the processing stage, or external conditions.

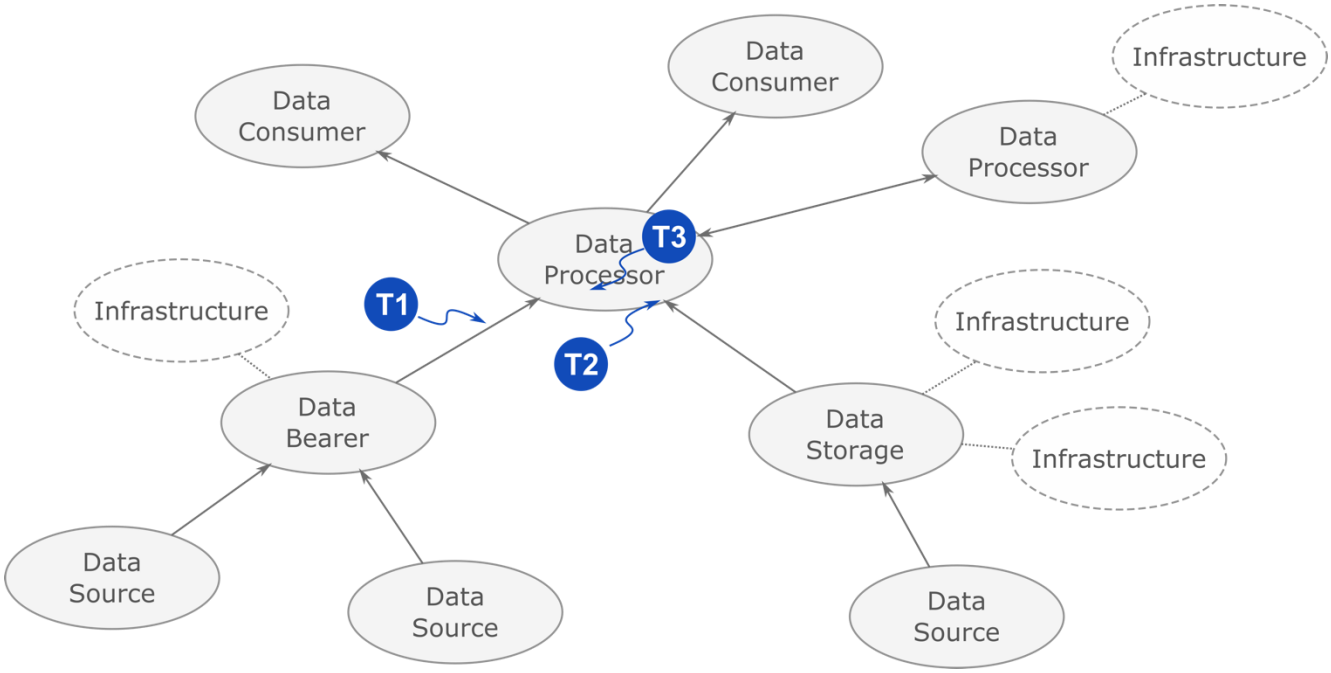


Figure 11. Data-driven model for chains of digital processes.

One critical aspect in the design of value chains is the definition of *policies* and *responsibilities*. According to the general recommendations of the CSA's *Security Guidelines* we may distinguish two main classes of policies:

- *Entitlements* define what access rights should be applied to different entities: read, write, process, share, store, etc. They concern information management and are part of the design of the high-level processes of the value chain.
- *Location and jurisdiction* define territorial boundaries for data, which have a direct impact on the legislation and rules they remain subject to.

The usage of data within each block of a chain may happen under different responsibility models:

- *Ownership*: data are owned by the same entity that operates the service; in this case there are no limitation on access to the data and retention.
- *Custodianship*: the service has full responsibility for conserving and processing the data on behalf of the owner, but cannot use them beyond the terms of specific contractual agreement.
- *Hostship*: the service does not take any responsibility for integrity and confidentiality of the data. This model is typically applied for storage and transmission functions, and it is up to the owner the application of encryption and other security mechanisms.

Beyond the correct definition of policies for the data management processes, most cyber-threats for digital value chains come from their physical realization, i.e., their software/hardware implementation and their deployment on devices and infrastructures. This clearly requires to take into account in our model the mapping of the logical data functions described so far to infrastructures, software, and devices.

The two different dimensions of a digital value chain, i.e., data processing and infrastructures, are reflected in the presence of two kinds of nodes and links in the graph. Data functions are depicted as greyed ellipses and interconnected by arrows, which indicate the direction of the data exchange. Data can flow from one service to

another one unidirectionally, similar to a pipeline, or it can be passed to one service for (sub-)processing and returned back on completion (bidirectional link). The direction of the link is only representing the data flow, and does not reflect the control paradigm (e.g., pub/sub or client/server), which is not relevant for our purposes. Infrastructures are depicted as white ellipses, and are connected with undirected links (with dashed lines). These blocks are not an explicit part of the data pipeline, but they affect security of the data functions hosted on them. As regard to links, arrows are therefore representing network connections (mostly on the Internet or anyway an insecure network), whereas dashed lines indicate an implementation dependency (software running in VM, network slice provided by a telecom operator, storage space in one or more clouds, etc.).

The above abstraction allows to capture the full complexity of existing computing models and technologies. As a matter of fact, the adoption of virtualization paradigms decouples applications from the underlying infrastructure and this has impact on security aspects as well. Different data manipulation functions may be implemented on cloud and Network Functions Virtualization infrastructures that belong to different providers and thus have their own security properties. Therefore, we consider every digital service as an autonomous entity with its own control/management interface, including security features and properties.

From a security perspective, we distinguish data functions as *self-contained* or *subordinate*. We define self-contained as a digital service which runs on dedicated devices or infrastructures, owned by the same entity. For example, IoT devices are self-contained data sources, since they have a sole owner and run a fixed set of software processes, even if this does not prevent sensors to feed multiple clients with different sets of measurements. A software is a self-contained entity if it only runs on a device/infrastructure that is owned and managed by the same entity; for example, a lambda function by a SaaS provider. We define subordinate as a digital service that runs on external infrastructures, likely shared with other tenants. A software or database hosted in the cloud are subordinate services, since their secure operation largely depends on the security capabilities on the underlying infrastructure.

The proposed classification is not univocal. For example, the same software function (let us say a context broker that collects and publishes data from IoT) can be considered self-contained if deployed and operated internally by the entity that provides the service, but it should be considered subordinate if deployed and operated in a public cloud. However, this classification is mostly conceived to reflect security interdependencies and drive the analysis of complex chains. Indeed, in case of self-contained services, security aspects of the software and the platform are tightly related and the two components can be treated as a single block. Otherwise, in case of subordinate services, there are different threat models that apply to the two entities, so each block and its related capabilities should be explicitly identified and analysed to estimate the overall attack surface and degree of trustworthiness.

Under this abstract representation for digital services, we can identify three main sources of threats:

- **T1.** “Internet threats” due to communications over open and unreliable infrastructures, which are subject to eavesdropping, modification, interception, man-in-the-middle, replication, DoS and other attacks;
- **T2.** “API threats” that come from the difficulty to verify the identity and integrity of the services, but also expose them to many attacks that exploit weak access controls and lack of input sanitisation.;
- **T3.** “Internal threats” due to careless or rogue employees, or malicious software that have bypassed external defences by direct or side channels.

The first two sources are the ones that mostly fall under the scope of the Project. The following Sections will introduce the main assumption and security requirements that drive the design of the GUARD framework.

4.2 Adversaries

Based on the proposed model, we can identify the set of potential adversaries for a digital service chain. The main distinction, as usual for ICT systems, is between internal and external adversaries (i.e., insiders and outsiders). Here, the boundary between the internal and external zone is represented by the logical implementation of the chain, instead of network segmentation as usually happens in legacy paradigms. Table 1 briefly describes the possible adversaries.

Table 1. Potential adversaries.

Type	Adversary	Description
Insider	Service providers	Employees that have privileged access to the service entry points. They may steal personal information or data. They may also do misconfiguration of the service or alter its behaviour.
	Resource providers	Employees of interconnected services in the chain. They may steal or alter user's data. They may create misconfigurations or alter the service to negatively impact the overall service chain so to break the contractual agreements or ruin the reputation of the SP. The introduction of fake or unreliable data can have severe impact on the safety and reliability of industrial, financial, and medical processes.
	Software developers	Programmers may include spyware, malware, and bugs in their applications, to steal or kidnap private data. Malware can also be developed to create entry points in enterprises' networks or large botnets for DDoS or other kinds of distributed attacks. It is worth pointing out that the potential impact of software developers is not limited to the main business logic implementing the overall service, but also extends to management and control infrastructures that increasingly adopt software-defined frameworks for remote management and orchestration.
	Hardware vendors	Vendors of IoT and personal devices may embed spyware or make unauthorized use of embedded sensors (e.g., for collecting conversations or videos). They may also be responsible for weak security configurations that create vulnerabilities easy to exploit (e.g., simple passwords, unnecessary software, guest accounts).
	End users	Legitimate users that try to carry out unauthorized operations, do illegal activity (e.g., cyber-crimes and terrorism), or access resources they are not allowed to.
Outsider	Tenants	Malicious users of shared resources may try to break isolation mechanisms and affect other users or the RP. They may be interested in stealing private data or damage the service continuity.

Type	Adversary	Description
	Vandals and tamperers	Anyone who deliberately or accidentally destroys, damages, tampers, or make unusable IT infrastructures and IoT devices.
	Internet entities	Anyone or anything in the Internet that can remotely connect to one or more digital services.

Insiders encompass any entity which is directly or indirectly involved in the creation, implementation, and operation of the service chain. Rogue employees of the SP with administrative privileges on service creation and operation are potentially the most dangerous adversaries: they usually have access to part or all internal and external resources, and they are able to perform monitoring, control and management operations, depending on their specific role and the access control mechanisms in place. Resource providers bring similar threats, but limited to the resources they offer. The main threat from RPs is their trustworthiness. Entrusting part of a business process to an external entity may expose data to eavesdropping, alteration, forgery, replication. Trustworthy RPs should put in place best practices for secure operation of their infrastructure and processes, including human, hardware, and software resources. Rogue employees and vulnerable infrastructures represent a potential attack vector to all service chains that rely on the operation of that RP. Software developers have indirect access to many systems, since their products are commonly used to build any ICT system today. The potential attack surface is very large in this case, and includes applications for processing data as well as management software. Reliability and trustworthiness of their code is essential for secure service operation. Finally, end users usually do not have high privileges and deep access to the system, but they could use the service for illegal activities or exploit vulnerabilities in the access interfaces (HTTP, SSH, SQL, Ethernet/IP, and so on).

Outsiders are the most likely adversaries for any system, even if they usually have far less attack vectors available than insiders. The massive usage of virtualization paradigms has brought a new kind of adversaries, which we indicate as “tenants,” with explicit reference to cloud terminology. Tenants are other customers of the RP, which share the same infrastructure. Isolation should guarantee the co-existence of processes from different users without interference, but no system could be considered free from vulnerabilities, as demonstrated by many attacks already reported for hypervisors [20]. Though tenants are internal to every RP, the focus of GUARD is on the overall value chain of a specific SP, hence such entities are considered “external” to the system under consideration.[59] Tenants are among the most dangerous adversaries, since they can acquire resources and initiate attacks from the service internals, therefore bypassing most of security controls applied to Internet entities. In cloud systems, the hypervisor is expected to isolate computing and networking resources of different tenants, while storage is usually managed with more traditional multi-user frameworks. Unfortunately, hypervisors have been largely designed to avoid interference between applications, with no strict security requirements in mind. Vandals need physical access to system components to tamper or damage them. This is a threat of secondary importance for clouds, where resources are usually well protected in safe plants. Fog/edge installations, IoT, and telco lines are instead prone to tampering. Finally, any networked entity present in the Internet represents a potential adversary for digital services; compromised IoT devices and botnets are emerging as common attack vectors to create large distributed DoS attacks and to automate repetitive procedures.

[59] Instead, from the perspective of the security processes of a RPs, all tenants should be considered as potential internal adversaries.

4.3 Threat analysis

While flowing through the logical functions identified by the system model (see Figure 11), data transitions in three possible states:

- *data at rest* is a condition of inactivity, where data are conserved in their digital form for future use on persistent storage media (e.g., disks, tapes, cd-roms, dvd-roms);
- *data in use* is an ephemeral presence required to process or manipulate it, as usually happens in RAM, caches, CPU registers;
- *data in transit* is the temporary condition during the transfer from one support to another one, which can be found in networks, buses, and other kinds of transmission lines.

Each specific state has different threats against the general CIA triad: Confidentiality, Integrity, and Availability. Specifically, for each of the life-cycle stages identified in Figure 10, we can point out the main threats listed in Table 2.

Table 2. Threat matrix for digital value chains.

	Confidentiality	Integrity	Availability
Creation and update	Disclosure, eavesdropping	Spoofing, forgery	DoS, diversion
Store and archive	Leakage, disclosure	Alteration	Ransom
Share and use	Disclosure, eavesdropping, unauthorized usage/analysis, spoofing	Forgery	DoS, diversion
Destroy	Retention		

Protection of data along the value chain should be defined according to a 4W model:

- *Who* is entitled to generate, use, and conserve the data (identity management);
- *Where* data can be stored or moved (infrastructures, countries, geographical regions);
- *What* operations are allowed to each entity on the data (e.g., read, write, host, change, share);
- *hoW* controls and policies are implemented (encrypted channels, security credentials, digital signatures, etc.).

As already pointed out in Sec. 4.1, the preliminary requirement is the definition of suitable policies and responsibility models for the overall value chain. However, we cannot disregard the impact of correct operation of the underlying infrastructure. Just to make a trivial yet illustrative example, we can consider availability: in this case, the most likely cause is due to DoS attacks on servers and communication infrastructures. It is therefore necessary to consider the two main security dimensions of digital service chains: data and service integrity.

Targets. The potential targets for attacks include *data*, *processes*, and *infrastructures*. Data is the most valuable resource in digital services, so possible threats include thefts, forgery, ransom, and unavailability. In some cases,

the most valuable element may be the continuity of the business process, for both financial, economic or safety reasons. In such scenarios, attacking the data is not worth per se, but may be used to interrupt or break the overall business chain (e.g., destroying route information for a fleet management application, sending rogue measurement to an electrical power plant). Finally, individual infrastructures may be selected as the weakest link in a business chain, or they might be the primary target themselves; in the second case, the impact on the value chain of their tenants is just a collateral consequence.

Vectors. The range of potential attack vectors is very broad and heterogeneous. The interconnection of digital resources and processes is largely based on network APIs for management and control firstly, but it is becoming a common practice for data exchange as well (e.g., with pub/sub paradigms). APIs probably represent one of the main attack vectors for digital services, since they can be used to deliver rogue instructions that aim at triggering software vulnerabilities like buffer/stack overflows and remote-side scripting by leveraging the common lack of data sanitisation. Both software functions and infrastructures use software APIs (just think at the management interfaces of common cloud services), so they represent a potential attack vector for both data, processes, and infrastructures. The data itself represent a potential vector, since they are at the base of many social, industrial, financial, medical, and transportation processes. The generation of fake data or the forgery of existing data can be used to drive a process along the desired direction: transfer money across back accounts, re-route vehicles and goods towards rogue destinations, trigger unnecessary medical treatments or surgeries, and so on. When humans are involved in the business process, emails and other media (chats, blogs, social networks) still remain the preferred vector to bypass the defence at the perimeter of each installation.

Threats. At the management level, the definition of policies to select providers and infrastructures has direct impact on all CIA aspects. As a matter of fact, if data are diverted towards bogus or unreliable providers there are greater opportunities to bypass weaker security countermeasures. For instance, the normative in a specific country might not ask restrictive confidentiality and privacy measures; in this case, it is likely that data is transferred or stored without strong encryption mechanisms and is therefore more prone to eavesdropping and disclosure. Similarly, the lack for security certifications or good security practice might turn into weak and vulnerable installations, which are easy to penetrate for external attackers. Weak access control policies are another possible threat. If coarse-grained grouping and access rules are used, there is the tangible risk that some entities will be granted more privileges than required. At the infrastructure level, most threats come from multi-tenancy. Attempts to break the isolation may lead to intrusion in the resources of other tenants or interference with their operation (e.g., DoS). Another potential threat is the misuse of protocols, applications, and devices. A very trivial example is the large set of vulnerabilities that affect LANs, Internet routing, and name resolution: they allow a broad set of attacks, including yet not limited to traffic diversion, man-in-the-middle, black holes, spoofing, flooding and amplification. At the lower layer, tampering and jamming are common examples of threats for devices and radio communications. Physical attacks against the infrastructure are likely for IoT devices, but more difficult for computing and network infrastructures. Software is today a major threat, given the number of vulnerabilities that can be found in complex systems. The main risks are outdated software, which does not include the necessary security patches, and untrusted sources, which may include malware in apparently clean code. Even if proper care is taken to verify the origin and version, the usage of repositories entails the risk of rogue modifications and spoofing. Additionally, security reports from cyber-security vendors continuously highlights the impact of wrong or weak configurations. The most common examples include: running unnecessary services, applying too permissive firewalling rules, do not enable encryption, weak passwords, ineffective access rules, and so on. Resource Providers remain largely responsible to implement security controls both in the infrastructure and in virtualized environments. Coarse-grained access policies to

CMS is an effective vector to break isolation between tenants, while the lack of monitoring and logging hinders the identification of intrusions and attacks. It is well known that VMs provides better isolation than containers, even if this comes at the cost of larger overhead. Finally, the lack of technical skills always brings the risk for social engineering.

Table 3 shows a rough breakdown of threats for common digital service models with respect to main sources (i.e., Internet, APIs, and Internal). The analysis is clearly general, and cannot take into account the details that characterize every specific service. The main finding from this analysis is the larger number of threats for APIs and Internals of IaaS/NaaS models. This is due to the presence of a rich management interface that allows self-provisioning and the larger freedom in installing and configuring applications. The more freedom tenants have to install software and change resource configuration, the more likely they interfere with other tenants or escape from their isolation. On the other side, when less flexibility is left to users (i.e., PaaS, SaaS, and DaaS) most threats come from the internal implementation and tenants have less opportunities for illicit usage of resources. In all cases, the Internet threats are basically the same, since all paradigms are interconnected through public communication infrastructures. The IoT case is somewhat a special case: it does not allow multiple tenants to run their own software/functions (as DaaS), but it often relies on hardware and firmware implementations that traditionally have far less security posture than cloud models.

Table 3. Breakdown of threats for typical digital service models.

	T1 – Internet	T2 – APIs	T3 – Internal
IoT	Eavesdropping Tampering Volumetric DoS Session hijacking Traffic diversion Man-in-the-middle Name resolution Bogus software repositories	Privilege escalation Spoofing Plaintext communications Data leakage	Default security configurations Outdated firmware/software
IaaS/NaaS		Data leakage Cloning VMs Migrating VMs Privilege escalation Spoofing Plaintext communications SSH attacks Cross-site scripting Remote file inclusion	Local DoS Guest escape Guest detection Guest-to-hypervisor escape VLAN hopping ARP spoofing MAC flooding Spanning Tree attacks MAC address spoofing Snooping Weak/wrong firewall configuration Eavesdropping Hypervisor bugs Data leakage RAM poisoning File system modification Running processes Bogus cloud images Malware, spyware

	T1 – Internet	T2 – APIs	T3 – Internal
			Automatic software updates
PaaS		Remote-side scripting Privilege escalation Spoofing Plaintext communications Cross-site scripting Remote file inclusion Data leakage	Local DoS Ransomware Software bugs Outdated software Untrusted providers Malware, spyware Automatic software updates
SaaS		Remote-side scripting Privilege escalation Spoofing Plaintext communications Cross-site scripting Remote file inclusion Data leakage	Software bugs Outdated software Untrusted providers
DaaS		Data leakage Privilege escalation Spoofing Plaintext communications	Software bugs Outdated software Untrusted providers

Simply relying on security operations (especially for what concerns detection) of individual service providers may not be enough, because they do not know in details the business logic and do not have visibility over the whole chain, so they likely lack the ability to correlate apparently independent events that happen in different instants and locations. The GUARD framework is therefore conceived as the platform that allows more correlation in time and space dimension over multi-domain systems, with the specific challenge to balance the depth of inspection with the privacy and confidentiality requirements of each RP.

According to this consideration, GUARD will focus on T1 and T2 threats, which can be easily detected by a set of information that does not disclose any sensitive detail about internal operation of each RP. In addition, the set of possible security services must be restricted based on the feasibility of collecting events and measurements without overwhelming the execution and the network. We therefore define some basic trust and security assumptions that restrict the scope of the platform, and then identify specific security services that will be supported by the GUARD design.

4.4 Assumptions

While pursuing challenging objectives in the definition of new security norms for distributed digital value chains, the GUARD platform cannot address any possible threat. We therefore introduce the main assumptions we consider to limit the scope of the platform. Some of the following assertions are reasonable for the target

scenarios, while others entail the adoption of common best practices in cybersecurity (i.e., existing tools and appliances, which still are perfect solutions for legacy environments).

Grey boxes. The most challenging characteristics of digital service chains are multi-tenancy and the presence of multiple administrative domains. Though full visibility over internal operation of each service involved in the chain is clearly unreasonable ("white box" model), we assume that some interfaces are available to provide limited and controlled capability of monitoring and inspection at the border of each service ("grey box" model). This is a loose assumption: no capability at all may exist ("black box"), but this of course will hinder the utilization of the service from external entities. We assume that they grey box model implies the following properties:

- *description* of the service and its main properties (vendor, version, capabilities);
- *certification* of the owner, data, software, and infrastructure;
- *monitoring* information collected from various subsystems (resource usage, logs, events);
- *inspection* of network traffic with custom filters.

The grey box assumption should ensure the ability to tackle threats T1 and T2, i.e., everything that is visible at the border of the service. Internal threats T3 are left out of scope, since this would require the white box model, which could only be used for those services which are directly owned and managed by the specific user.

Trusted software. We assume the software is trusted and behaves correctly. The common usage of proprietary and licensed software in commercial deployments does not allow third party to perform deep analyses, so each service provider is responsible for implementing all static and dynamic mechanisms (including software analysis, control-flow analysis, remote attestation, etc.) that are necessary to guarantee the integrity and safe execution of the software before and during operation. The implementation of such mechanisms should be reflected in specific certification options exposed by each service, which can be used to assess the suitability of the service for different applications and scenarios.

Trusted service providers. The first source of internal threats are humans. We assume that human resources of the SP are trusted or anyway an external system (for example, an IDS) verifies their behaviour. The case of bogus employees is more likely for large enterprises, where the number of people is very large and social relationships are more difficult. However, the business concept addressed by the Project fits well to small and medium enterprises, which do not have the financial and operational capacity to deploy and manage large infrastructures, but need the agility to create new and remove old services following emerging business needs.

Reliable operations. The identification of attacks originated within each digital service is only possible through deep analysis of internal workflows, including authentication and access logs, file scanning, I/O calls, registers and instructions usage. It is unlikely (and unsafe) that the owner of the service/resource gives access to this information to external parties. We assume that legacy security appliances (like firewalls, antivirus and IDS/IPS software) are used by each resource provider to verify the reliable operation of the systems under its responsibility. We also assume that cloud resources are safe from tampering (the same does not hold for IoT devices).

Privacy. The grey box model must not turn into a privacy concern. All the security context exported by a service must not disclose the service internals or any information concerning other tenants. We assume that only aggregate information is available for the service itself and other tenants (for instance, total number of users, cumulative resource usage), and that packet analysis is limited to the allowed network segment. For example,

in case of IaaS cloud services, each user might see the overall number of tenants and resource usage, may have full logs about his own resources, and may only analyse the packets exchanged between his VMs and with external entities.

Root of trust. The dynamic composition of digital services from different administrative domains does not allow the application of centralized authentication schemes, but requires the application of distributed (maybe federated) systems. We assume proper mechanisms are in place for identity management and exchange of secrets, likely based on standard X509 certificates. The specific source of such certificates (TPM, smart card, internal storage) should be inferable by the exposed properties, because it represents a possible parameter in the risk assessment process.

Trusted security operators. The availability of security APIs from each digital service should be limited to professionals accredited for security operations. We assume these entities are safe and behave correctly, i.e., they do not abuse their privileged role to carry out passive or active attacks.

Safe control/management flows. Dynamic composition of digital objects requires a management plane for instantiation and life-cycle operation (e.g., web services, software orchestration), which is usually available through software APIs. Such APIs can be used to provision resources, connect services, delegate tasks, release resources, and so on. Clearly, they represent a potential vulnerability for every system, if not properly hardened and protected. Given the criticality of the management plane, it looks reasonable assuming that all security measurements are in place to ensure that API cannot be leveraged for attacks. This means strong authentication and authorization mechanisms are used, safe software is used and properly isolated from the data plane, data and input are correctly sanitised before being used, etc.

4.5 Security requirements

According to the concept described in Sec. 3.1, the GUARD framework is a platform for implementation of security services for digital value chains. The platform is conceived to facilitate the integration with dynamic and agile digital services and resources, by providing broad visibility over the evolving context. The specific ambition is to decouple security services from the underlying infrastructures and devices; this approach makes it easier for security providers to run new detection algorithms, since tailored events, logs, and measurements can be retrieved through the GUARD platform without the need for re-engineering the underlying software, hardware, and infrastructures.

The range of available technologies for monitoring and inspection of software, hardware and network is rather broad, and could virtually address the need of any detection algorithm. However, there are some aspects in the business framework described in Sec. 3 that suggest some constraints in the visibility over the execution environments of different domains. For instance, traces of software execution or packet dumps from other tenants' traffic are likely to disclose confidential information, so it is unlikely that a RP allows this kind of monitoring. Though the GUARD platform will anyway pursue the integration of the broadest range of monitoring and inspection technologies to support any forthcoming need, the set of security services that fall under the scope of the project will be restricted to the following list.

Secure chaining. One main benefit of dynamic composition of digital services is the possibility to select them at run time, without the need to set up complex ICT infrastructures. Today this is feasible from a technical point of view, thanks to the availability of web services, service-oriented architectures, and software orchestration tools, yet it remains challenging from a security and management perspective. The usage of trusted digital identities

is only the first step towards reliable business relationships. A digital identity is the technological link between a real entity such as a person or a company and its digital presence, which may include legal, biographic, and biometric attributes and data. However, clear and definite identification might not tell much about the security posture and reliability of the entity. As a matter of fact, these factors depend upon technological, procedural, and human aspects that are involved in the realization of digital processes. The secure posture of a digital object depends on the software used, the correct and timely application of security patches, the availability of security properties for third parties, and the implementation of security processes (including, but not limited to, the deployment of security appliances). Many digital platforms already expose a variety of (nearly) real-time metrics and logs, which can also be leveraged for security purposes. In some cases, management options are also available (for example, to start/stop/pause VMs or similar operations in serverless environments). Since the range of available features and properties is very heterogeneous for different technologies and providers, there is the need to discover what is available and how, so to verify the security context is compliant with the desired policies. A more extended notion of trustworthiness could also include the availability of international certification frameworks and the proper reporting of incidents and identification of new threats.[60] The GUARD platform is expected to take all these aspects into consideration for the elaboration of a formal model that assesses the overall degree of security and trustworthiness of a value chain, and to compare the outcome with users' requirements and policies.

Data tracking. Data are today one of the most valuable resources in any business process. Ideally, data should be kept within the security perimeter of the entity that owns it. However, this is not possible in practice, since data must be shared to create services, as typically happens for medical and financial processes. One fundamental right of data owners is the knowledge of the exact location of their data, which is necessary to verify that data are not propagated outside the administrative and legal boundaries settled by them. Though the identification of the data controller is rather simple and based on written or "clicked" consent, the discovery of data processors is complicated by hardly-trackable service topologies and virtualization paradigms. The GUARD platform is expected to identify data objects and track their propagation along the value chain, so to report their current (and past) location. This feature relies on the willingness of resource providers to take part in the process, and to report any transfer of data towards external providers; this could represent an additional factor when evaluating the trustworthiness and reliability of each provider. The platform should also include the definition of custom policies that limit the propagation of data across different resource providers. Common policies may restrict the propagation to selected countries or regions, specific list of providers, providers with minimal certification properties, and so on.

DoS attacks. Denial-of-Service is a wide class attacks, which exploits both volumetric techniques (i.e., flooding a system with unsustainable workload) and protocol vulnerabilities (packet forging and poisoning). Detecting a volumetric (even Distributed) DoS at the target is trivial, given the amount of traffic and processing overhead; however, the real challenge is the detection at the source, to avoid that compromised devices and services could participate in a botnet. Also, the detection of amplification attacks is a challenge, since they generate far less traffic at the source and may remain unnoticed in the normal activity patterns. The GUARD platform is expected to improve the detection of DoS by applying detection methods and algorithms based on machine learning and

[60] ENISA EU cybersecurity certification framework. Available at: https://www.enisa.europa.eu/topics/standards/certification?tab=details (last accessed 20th January 2020).

artificial intelligence, without using overloaded sets of rules, in order to guarantee high performance and little consumption of computational resources.

Network anomalies. Despite the risk for an overwhelming number of rules and excessive processing and communication overhead, blacklisting and signature-based detection systems can be very effective for known attack patterns. However, this is also well-known by attackers, who are increasingly relying on new tools and techniques to constantly evolve and change their attacks, so to elude the knowledge base. One of the most challenging issues is indeed detection of zero-days and unknown attacks. The GUARD platform will adopt a two-tier model. At the bottom, machine learning and artificial intelligence techniques will be used to learn the normal behaviour of network components and services. This baseline allows to detect significant deviations and anomalies which manifest in modifications of the previously learned model. At the top tier, the programmability of the underlying infrastructure will be leveraged to "zoom" on suspicious activities and components, so to better trace suspicious attacks, identification patterns and signatures. The ambition is the correlation of data collected from different subsystems (e.g., packets, system logs, events), which allow to infer information from encrypted patterns.

Web application and API protection. Developers make extensive usage of API for North-South and East-West communication in Service-Oriented Architectures, Web Services, micro-services, and service meshes. Unlike the telecommunication world, where most communications are based on rigid binary patterns, most APIs allows rich semantics to encode controls and data. This enables more flexibility, but also brings many potential threats due to misuse and buggy implementations. The top 10 security threats identified by the OWASP include[61]: injection flaws, broken authentication, sensitive data exposure, XML external entities, broken access control, security misconfigurations, cross-site scripting, insecure deserialization, unsecure software components, insufficient logging and monitoring. Tracing and inspecting APIs is probably the most straightforward and non-invasive way to detect most of the threats on the top 10 list. The GUARD platform is therefore expected to collect, parse, and serialize relevant fields from request and response messages. The main target is the identification of illegal content and unexpected control flows.

[61] Top 10 web applications security risks. Available at: https://owasp.org/www-project-top-ten/ (last checked: 21st January, 2020).

5 GUARD Architecture

The GUARD architecture describes multiple facets of the GUARD platform. The first one is the functional model, which identifies the main logical processes that implement the vision of Section 2. The second facet is the reference model, which describes the logical entities that will be developed and integrated in the platform. Finally, the last facet is represented by the software architecture, which makes an inventory of available technologies and identifies missing gaps to be developed in the next months.

Before starting the description of the GUARD architecture, it is worth briefly reviewing the on-going transformation in service architectures for digital services.

5.1 Digital service architectures

Software engineering has probably never evolved so fast as in the last twenty years. Following the progressive introduction of virtualization paradigms first and cloud model later, architectures have progressively evolved from monolithic to distributed and scalable frameworks, as pictorially shown in Figure 12.

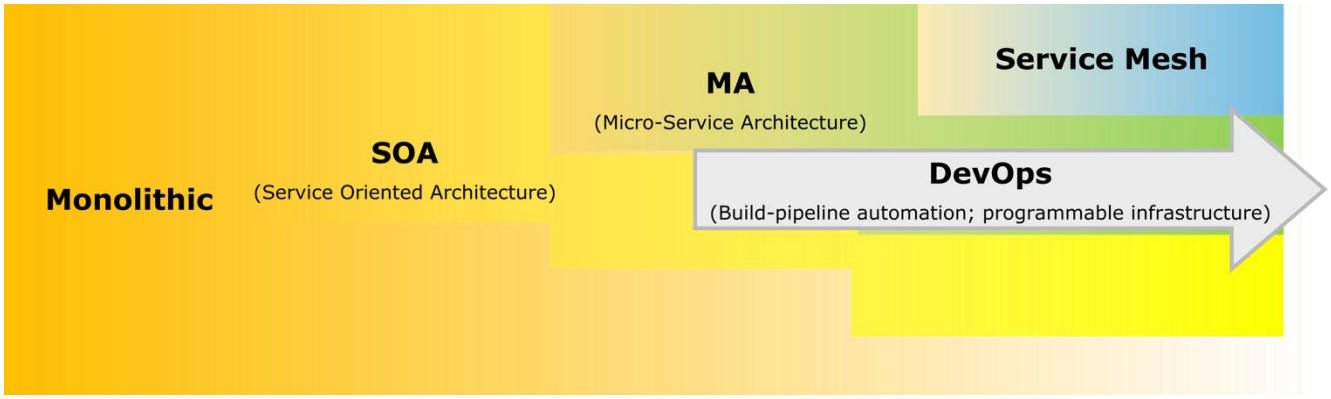


Figure 12. Software architectures have progressively moved from monolithic to more distributed and scalable frameworks. Source: P. Geenens, Cloud Native Security Platform, 1st International Workshop on Cyber-Security Threats, Trust and Privacy Management in Software-defined and Virtualized Infrastructures (SecSoft), June 24, 2019 // Paris, France.

The term “monolithic” does not necessary imply the lack of modularity or extensibility; indeed, best practice for software design has been available for long. Monolithic software is commonly written in a single language, makes use of quite rigid inter-process communication facilities (which usually depend on the specific language or operating system), and cannot easily be changed at run time. Monolithic applications are usually contained in a common “project” and distributed as a single package. The large collection of functions, even if systematically grouped in different files, makes it hard to modify and test the whole application. As a matter of fact, every small modification must undergo full testing of the whole project before delivering to production. The only way to scale is replicating the entire application, which leads to higher resources and costs. The logical organization in multiple tiers facilitate the design, implementation, and maintenance of the application, but does not bring substantial benefits for scaling and testing.

Chasing more agility in the development and operation pipelines, the next step was the introduction of service-oriented architectures. The ground concept is the decomposition of applications in multiple business units, each one with specific functions. For example, an e-commerce application could be decomposed in Authentication block, Order block, Notification block. This greatly improves scalability, since each business unit becomes a standalone block, which is simpler to design, maintain, and distribute. The main drawback of this approach was

the useless complexity of legacy abstractions and protocols, where each service spoke a different language and an additional layer was required for communication (Enterprise Service Bus).

Following this evolution, micro-services removed the need for common ancillary services (like back-end databases), by encapsulating both logic and data [28]. The idea of micro-services is that they are stand-alone service units completely decoupled and isolated from each other, designed according to the principle of "single responsibility." As such, the internal logic of each micro-service can be implemented with different hardware technologies and programming languages (C/C++, Node, Go, Python, Java, Scada, Kotlin, …); this requires to expose an interface which is portable and independent from the coding language. Common choices are based on HTTP like REST (SOAP is another possibility, but again more rigid and heavier than the lightweight and free format REST), gRPC (Google RPC – which provide a binary encoding which is more efficient compared to REST) and Apache Thrift (which provisions for streaming events).

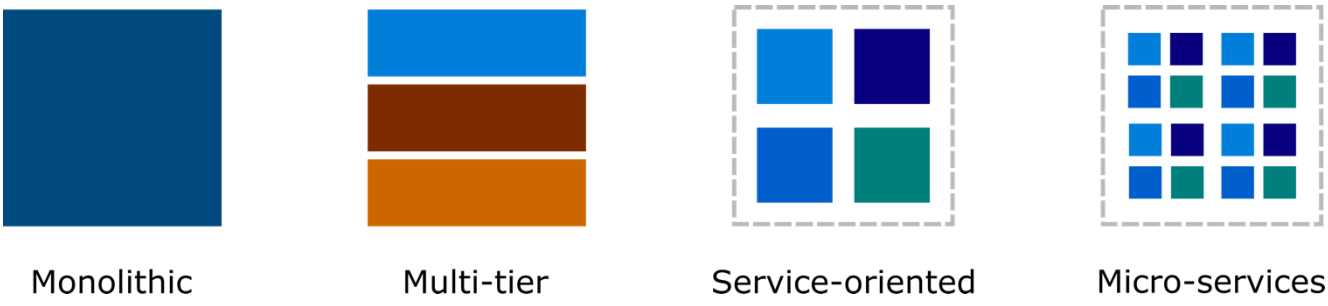


Figure 13. Conceptual differences between evolving software architectures.

Applications are built by combining functions provided by each micro-service. The loosely couple nature of micro-services allows to replace, duplicate, or remove part of them without affecting operation of the overall architecture. Services can hence grow or shrink dynamically according to the evolving workload, update specific functions or protocols, be deployed over multiple and even heterogeneous infrastructures.

The ground assumption is the ability to discover micro-services and related functions, which entails the presence of service registries. In addition, load balancers are commonly used to implement horizontal scaling. Though there are not rigid patterns, typical architectures use an API gateway that route and secure requests that come from outside (North-South) traffic as well as relied upon as central load balancer for East-West service calls (see Figure 14.a). With the introduction of an API Gateway, there is still a weak form of perimeter in the application; all E-W traffic may happen in clear text, whereas only communications that are situated north of the gateway are encrypted.

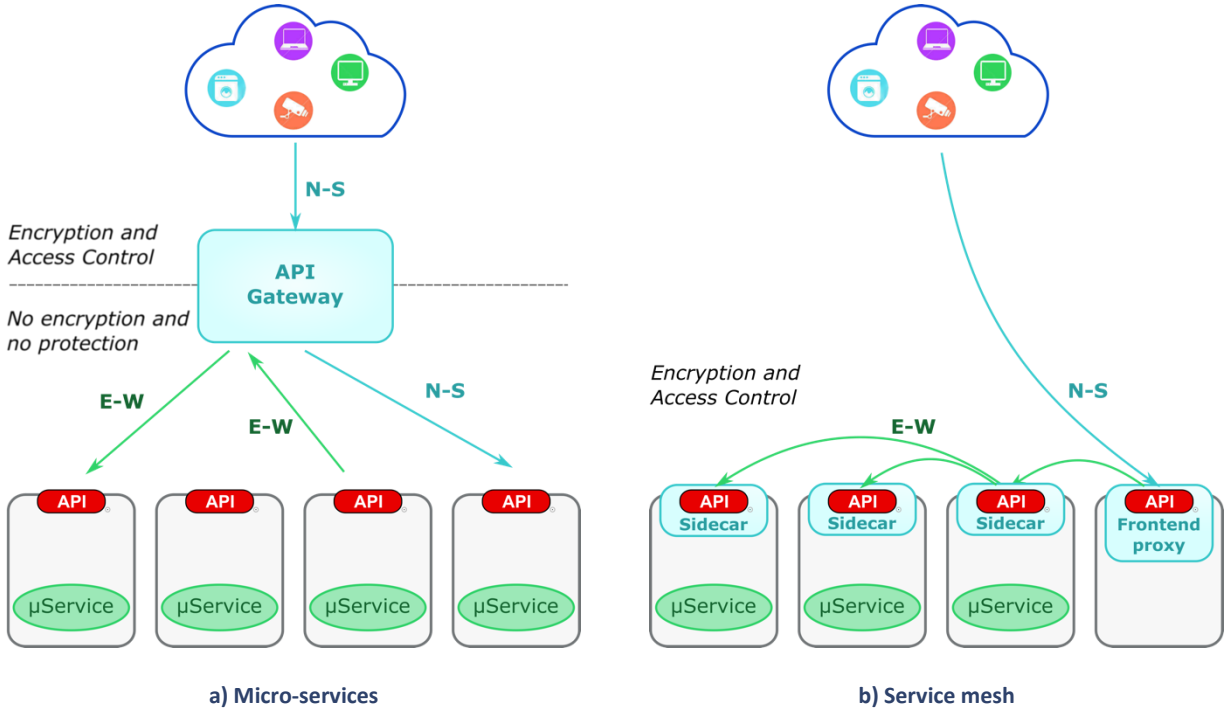


Figure 14. Micro-services vs service mesh architectures.

The latest step in this evolution tried to go beyond the heterogeneity in the implementation of external APIs. As a matter of fact, different codes, different languages, and different internal architectures usually lead to inconsistent semantics, methods, encryption schemes, authentication mechanisms, access policies, logging and so on. The API gateway definitely represents a potential bottleneck and single point of failure, so the natural evolution was to break this functionality down within each service. A common and shared component therefore is placed alongside the main business logic as a lightweight sidecar that acts like a communication proxy and provides the front-end communications (e.g., HTTPS, RPC, AQMP/HTTP) and security functions (Figure 14.b). Recently, Envoy has emerged to implement this concept for containers.[62] This speeds up integration and development, since programmers can now focus on the main business logic without wasting their time to implement communication protocols. It also facilitates management of the overall application, since consistent data and measurements are available to monitor the health of the service and alert in case of potential issues. A specific requirement, however, is to keep the additional layer introduced by the sidecar as thin as possible (in terms of computation overhead and latency), since it must not lead to performance degradation or interfere with usual telemetry.

The architecture now becomes fully distributed and the boundary between N-S and E-S communications blurs, since every service can be leverage as a public interface (even if dedicated proxies can still be deployed as public frontends). The invocation of complementary functions creates a logical “chain” of services. The result is a fully decentralized service mesh, without any sort of security perimeter, which applies peer-to-peer encryption, authentication, and access control functions. A subsidiary framework is anyway still needed to mediate security properties and establish trust relationships, for instance by distributing certificates, collecting logs and

[62] Envoy: https://www.envoyproxy.io/.

performance measurements, assessing the global health. This logic is already implemented by ISTIO[63]; from the security perspective, observability is limited to logs and metrics collected by Envoy and Prometheus[64].

5.2 Functional model

Following a well-established model in the networking domain, the functional processes can be split in three logical planes, which correspond to different operational scopes and timing requirements: data, control, and management planes. This model is especially useful to compare GUARD with other initiatives from international standardization bodies (see Section 0). The three planes are present both in the GUARD core framework and local services, as pictorially shown in Figure 15.

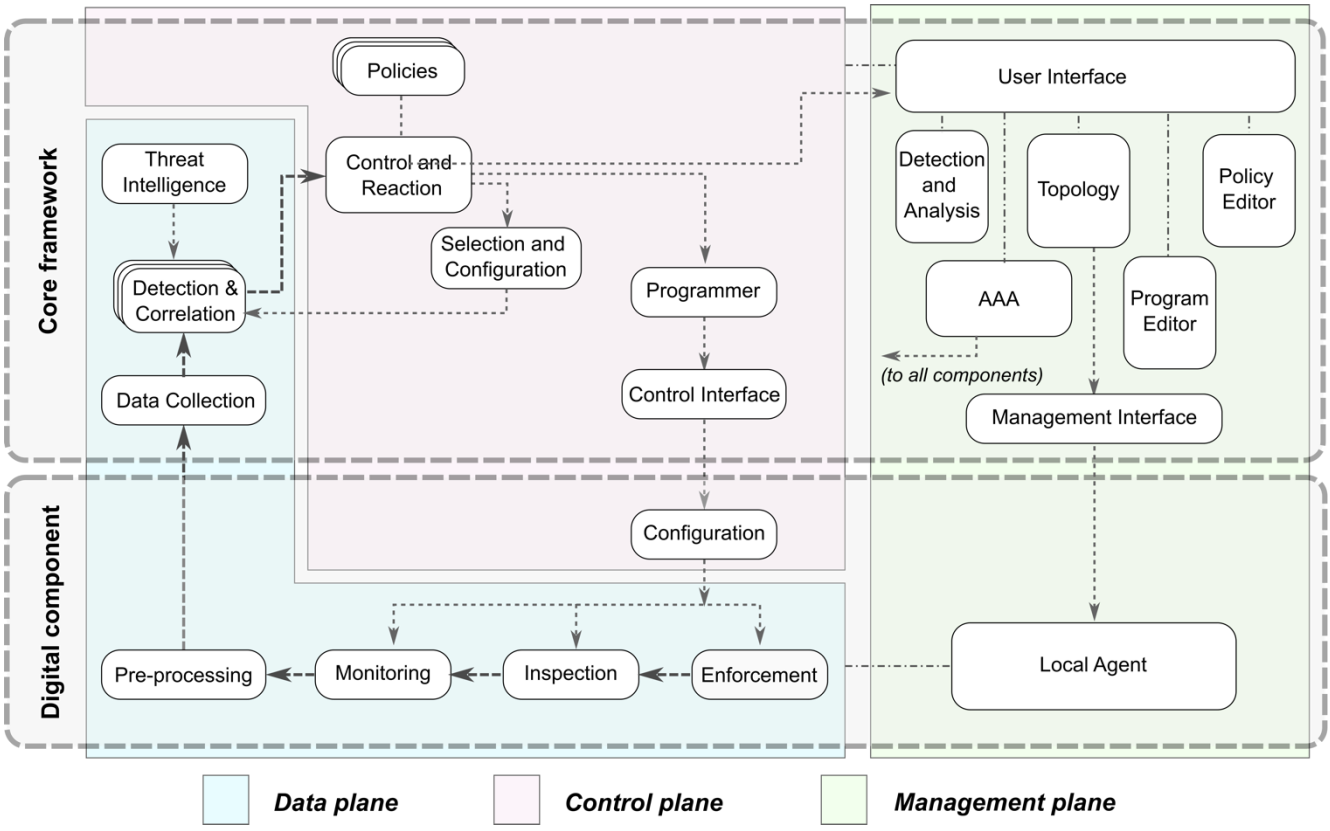


Figure 15. Functional model for the GUARD framework.

The **data plane** is composed by all the functions to create, collect, store, and analyse the security context. It includes an *Inspection* function that scans relevant subsystems in the digital object and reports information according to current configuration. Examples of subsystems include configuration files, log files, network sockets, inter-process communication, system calls, memory dumps, process traces, software packages, binaries, user files. The *Monitoring* function performs similar tasks, but it is mostly oriented to detect events and operating conditions: user logon and logoff, CPU/memory/disk usage, network statistics, notifications from applications and peripherals[65], hardware health status indicators (CPU temperature, fan speed, power voltage), running processes and daemons, requests for reboot or shutdown, software updates, link connection/disconnection, hardware changes (e.g., plugging of USB devices), and so on. A sharp distinction

[63] ISTIO: https://istio.io/.
[64] Prometheus: https://prometheus.io/.
[65] Reporting unexpected usage of the keyboard or mouse, and alarms from anti-tampering mechanisms is useful for early detection of attempts to compromise a physical device.

between Monitoring and Inspection tasks is not really relevant in this context, so the two terms will often be paired in this document. The *Enforcement* function applies security policies, including packet filtering and access control. The typical scope ranges from coarse-grained mechanisms (user and group management, file system permissions) to very fine-grained control (selective access to system resources for users and applications, access control lists, quarantine, etc.). The *Pre-processing* function aggregates and transforms data before sending them to the remote core framework for analysis and processing. It entails common operations such as filtering, timestamp insertion, geolocalization, anonymization, removal of private and sensitive data, fingerprinting, serialization, name resolution, classification, encryption, parsing, and creation of structured data from text. It also includes data fusion, so to produce more consistent, accurate, and useful information locally by eliminating redundancy and increasing the entropy of the information. According to the main project objectives, Pre-processing should only be limited to lightweight detection tasks, trying to reduce the network overhead without overwhelming the local service. In the core framework, *Data Collection* bears the security context from local services to the core framework over secure channels. This function lies at the border between the two entity, so it is partially implemented by local services as well. *abstraction* It also decouples the data from its source and gives access to the set of heterogeneous data in a uniform way, meaning that data consumers will not be aware of the specific technology used to create the data. Data is expected to be organized in a logical tree-like structure, which represents the topology of digital services and exposes what context is available from each of them. The group of *Detection & Correlation* functions is the last step of the context processing pipeline. It includes several algorithms and processes, which analyse the context, correlate data, search for known attack patterns, discover anomalies, and point out new threats. They can implement typical security functions that today are provided by different appliances: antivirus, IPS, IDS, firewalling, anti-DoS, etc; however, the real challenge would be to leverage Machine Learning, Artificial Intelligence, and Big Data techniques to improve the correlation capability and properly tackle multi-vector, stealth, and persistent attacks. It is worth pointing out that the scope for Detection & Correlation is not limited to attacks, but also entails the identification of vulnerabilities and other threats that come from misconfigurations, outdated software, data leakage, data sharing, untrusted services, etc. The effectiveness of detection largely relies on the knowledge of threats and attacks. It is therefore important to correctly feed the detection algorithms with rules, signatures, and behavioural patterns that describe how attacks are carried out. Unknown attacks are mostly detected by recognizing changes in the normal system behaviour. Therefore, it is important to be able to model the normal system behaviour reflected in the available data properly. Clearly, detection of attacks strongly relies on the available data, because only attacks that are reflected in the data are detectable; for this reason, the programmability of the underlying data plane is of paramount importance, because the set of data to be collected might not be completely known at design time. The *Threat Intelligence* is partially imported by external providers and partially built locally; the latter is necessary in case of zero-day attacks to be able to provide signatures and rules and therefore enable and improve detection of those unknow attacks in the future, and finally support others to timely update their detection mechanisms to mitigate the global impact of such attacks.

All functions in the data plane operate in a deterministic way, according to imperative behavioural statements, configurations and parameters set by the control plane. The **control plane** supervises the operation of the data plane in the short time horizon. The *Control and Reaction* function changes the behaviour of the underlying data plane according to the evolving context. It receives notifications from Detection & Correlation, and takes decisions based on behavioural *Policies*. Policies define how to react to specific triggers, also considering the current context; the reaction may consist in one or more actions. The scope of such actions ranges from re-configuration of the Detection & Correlation function (e.g., by enabling further algorithms or changing some

parameters), to re-programming of the data plane, to notification and request of management actions from humans. The *Selection and Configuration* function is an interface to programmable features of the detection algorithms. It allows to enable options and set operational parameters, in a similar way to what already happens through the graphical user interface of existing security appliances. Clearly, the implementation of this function strictly relies on what exposed by each individual detection process. The *Programmer* function allows to select alternative configurations and, whenever possible, inject code into the data plane. Some examples of configuration include: activation/deactivation of the collection of logs and measurements, changes in the frequency of collection, filtering rules. A very challenging ambition is the capability of pushing dynamic programs, for example to define parsing rules for new network protocols and log data pre-processing pipelines. It is clear that the safety and trustworthiness of both elements is of utmost importance for secure and reliable operation of the whole framework, so the implementation of the Programmer function must carefully take into consideration this aspect. The *Control Interface* is the internal abstraction that exposes what programmability features are available in each controlled service, together with the current configuration. It is the function that implements the "programmatic" access to the security context, meaning that it enables to change the behaviour of the security functions present in the data plane of each service. Being an abstraction, it hides the actual technologies deployed in each service. The Control Interface has a natural counterpart in the *Configuration* function of each digital service, which applies the configuration locally.

The **management plane** defines the behaviour of the system in the mid/long-term. It includes tasks as instantiation and configuration of components in the data/control planes, translation of high-level intents and goals into commands and control policies, monitoring of operational parameters to detect deviations and malfunctioning, interaction with other systems/components. The *User Interface* is the main function that gives users overall control over operation of the GUARD framework. The User Interface provides graphics or command line interface (CLI) access to a number of management functions, in addition to visual analytics and access to the whole security context collected by the core framework. The User Interface also gives access to the Programmer; this enables direct investigation of unknown or complex attacks. The *Detection and Analysis* function allows to add, remove, and update detection functions. It makes the GUARD framework flexible, with the possibility to add/remove/update detection services. The *Topology* function discovers and queries digital services, to understand the logical relationships in the current chain and set up the correct information for Data Abstraction. It also exposes what management functions are allowed in each digital service. Some examples include: disconnection of a service from the chain, and removal of personal data. Further capabilities may be available when strong trust relationships are in place between the operator of the GUARD framework and the owner/provider of the digital service (e.g., the GUARD framework is operated within a NFV domain). In such case, it might be possible to start/stop the service, update software components, take snapshots, etc. The *Program Editor* function allows the insertion and verification of lightweight programs in the control plane, which can then be injected into the local data planes. The User Interface may implement editing functions, so to directly write and upload the code. The *Policy Editor* is a similar function for uploading and managing control policies; also, in this case the User Interface may implement editing functions. The *AAA* function is one of the most critical aspects of the whole framework, being responsible of the definition of internal security policies. It creates, manages, and stores the identity of the different elements in the GUARD framework, including software components and human users. Indeed, storage of identities within the GUARD framework is not strictly required, since an external authentication service or database (e.g., LDAP, RADIUS) could be used to implement single sign-on. Beyond identity management and authentication, the AAA function is used to set up fine-grained access control policies, which define which operations are allowed to each entity. The GUARD framework is expected

to distinguish multiple roles: cybersecurity staff, end users, local agents, detection algorithms, and so on. Some of them will be data producers, other will be data consumers, and some may play both roles. In addition, some roles are allowed to make control and management operations, on the whole framework or on some subsystem. Reliable and trustworthy distribution of cryptographic material is another important yet challenging function, given the multi-domain nature of the GUARD framework. Finally, the *Management Interface* is responsible to implement the communication protocol with local agents. It includes discovery of capabilities, properties, and security agents deployed in each digital service.

5.3 Reference model

Figure 16 shows the main architectural elements that are necessary to achieve the GUARD objectives. The model is composed by four main macro-blocks:

- *Digital Services*, which are extended with embedded security capabilities for monitoring, inspection and enforcement;
- *Core Framework*, which includes all components to collect and process security-related data, draw situational awareness, and suggest reaction strategies;
- *Security Dashboard*, which is the Human-Machine interface for visual analytics and definition of control/management policies;
- *AAA framework* for managing digital identities and access policies in a coordinated way.

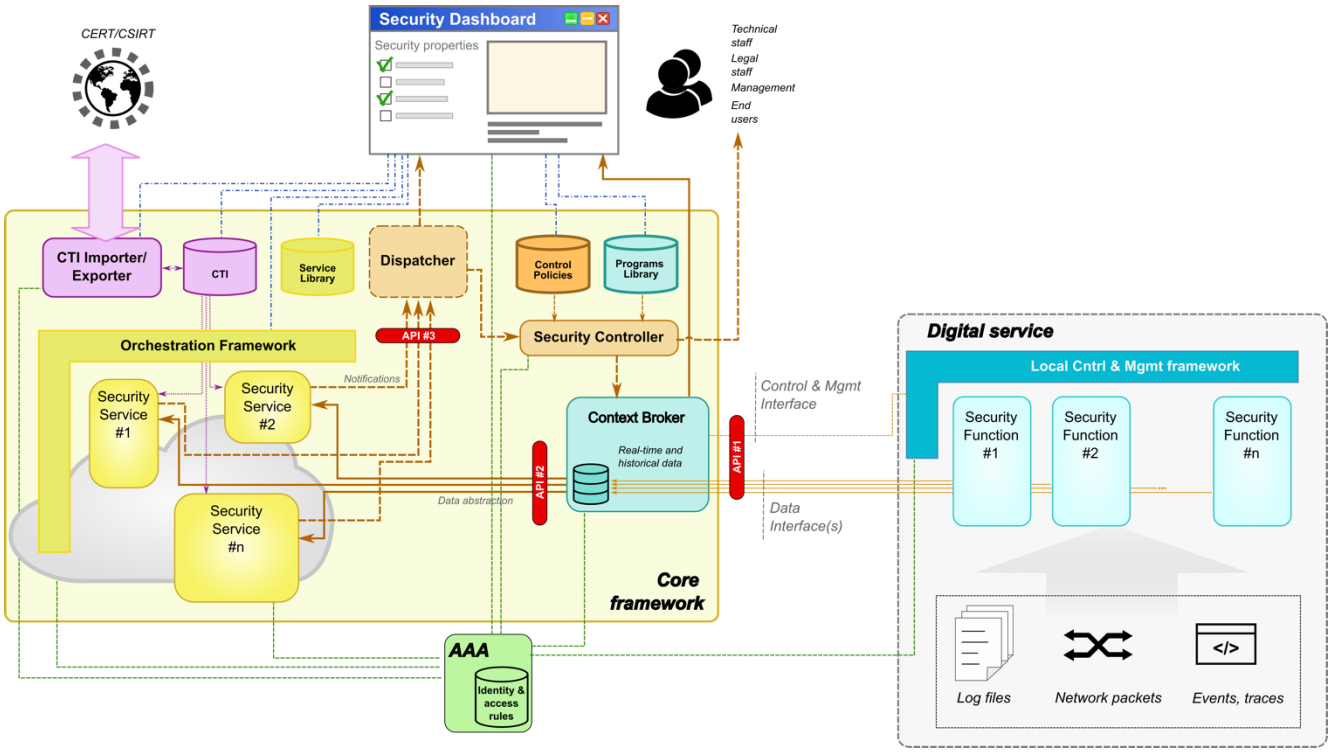


Figure 16. GUARD reference architecture. Brown solid arrows indicate the flow of collected data, from local Security Functions to centralized Security Services and the Dashboard; brown dashed arrows indicates notifications (e.g., IoC). Blue dash and dot lines indicate control and management interfaces.

5.3.1 Digital services

The realization of agile business processes through chaining of elementary functions will only be possible if security capabilities are integrated within each digital service, and such capabilities can be dynamically combined

to create complex security appliances. Once common interfaces and semantics are available to access security functions, implementations can be tailored to the specific environments: programming language, virtualization environment, software architecture, etc. However, similarly to what already envisioned for the implementation of a service mesh (Sec. 5.1), the development of portable sidecars could be an effective mechanism to improve their quality and effectiveness.

A security sidecar should include the following set of *Security Functions*:

- *monitoring* of log files and system events (including, for instance, information and measurements available from pseudo-filesystem like *proc* and *sys* in Linux);
- *inspection* of network traffic at the different OSI layers, from raw packets to application messages (like HTTP, FTP, SOAP, DNS);
- *tracing* of execution and control paths, which may be limited to system calls and I/O calls, or extend to memory and registry dumps;
- *enforcing* of security rules at the network and application layer (for instance, by dropping packets based on IP header fields, by blocking specific HTTP request messages, by preventing access to files or peripherals).

Additionally, data fusion and aggregation functions are required to perform ancillary operations like removing redundancy, compressing data, timestamping, etc.

A very challenging feature for security sidecars is programmability, i.e., the capability to take on the execution of lightweight data filtering, aggregation, processing, and enforcement tasks. This goes beyond mere re-configuration of individual functions and brings the challenge to safely run code injected by an external (though trusted) entity. However, offloading tasks to local services helps balance the trade-off between processing and network overhead in an effective way.

The range of local capabilities is rather broad and should be tailored to the specific nature of the digital service (IoT, cloud, lambda function, storage, ...). Yet it should be dynamically tuned to the evolving context, while pursuing the better trade-off between granularity and overhead. A *Local Control and Management Framework* is therefore necessary to govern life-cycle events of each security function (e.g., start, stop, reload, update) and to set its operational parameters (e.g., name of files, filter rules, event names). The control and management framework also exposes security properties of the digital service, including vendor, name, description, version, release date, digital certificates, and all other internals that can be safely shared with a SOC.

The external API encompasses two aspects: control and data (**API #1**). The control interface heads to the control and management framework and is used to discover security capabilities and enable, disable and configure the security functions. The data interface can be directly implemented by each security function and is used to report the collected set of events, data and measurements.

5.3.2 Core framework

A SOC clusters all technical capacity to protect the business processes and the organization itself. It encompasses a number of complementary security services, ranging from prevention, detection, incident management and response, reporting, governance, risk, compliance, and anything to do with managing and defending information security within the organization. Technically speaking, the goal of a SOC is to implement and oversee network, application, cloud, and user security, among other operational functions.

When provided ”as-a-Service”, a SOC platform must have the flexibility to implement any security process that is required by different and heterogeneous business value chains. Accordingly, alongside legacy functions for collecting and storing security-related data, the design of a SOCaaS architecture should encompass the ability to deploy and run new algorithms at run time.

5.3.2.1 Context Broker

The *Context Broker* is the architectural entity which collects information, data, events, and measurements from local security agents embedded in each digital service. The Context Broker is not a plain database, since it should address the heterogeneity of sources and protocols; in this respect, it provides a common data model and abstraction for discovering, configuring, and accessing the security context available from the different services.

The Context Broker terminates both the control and data channels towards remote services. It has a context discovery layer that queries all components involved in the chain and builds the logical topology of the overall chain, including the security properties and capabilities of each service. In this respect, the integration with existing meshing mechanisms is strictly necessary for discovering existing services and following topology changes.

The main purpose of a Context Broker is therefore to describe what security capabilities are available from each service (i.e., what can be retrieved) and the current configuration (i.e., the set of data that are being collected). It also allows to change the reporting process, by enabling/disabling security features, by changing the configuration of remote security functions, and by offloading simple tasks. Information and data models are therefore required to describe the nature, composition, granularity, and format of security-related data, together with the corresponding configuration settings. Collectively, we indicate this information as the Security Context Model.

With respect to data management, the Context Broker delivers on-line events and measurements to the detection and analysis algorithms (**API #2**) and stores historical data for off-line analysis, forensics, and legal obligations. It hides the heterogeneity and asynchrony of the sources, organizes historical data, and provides simple querying and fusion capabilities in data access.

The presence of a common Context Broker for all detection algorithms improves the efficiency of the whole system, since the same information is collected only once and shared among all active instances. The Context Broker should always mediate between conflicting requests coming from different algorithms (enable/disable monitoring, frequency and granularity of reporting) and find the best configuration that fulfils all of them.

5.3.2.2 Security services

Legacy SOC platforms are often designed to collect refined events from local cyber-security appliances (DOS detection, IPS, IPS, antivirus, etc.). However, the likelihood of detection of distributed, advanced, and persistent threats increases when the largest base of raw events is analysed and correlated in its time and space dimension. A progressive migration of cyber-security appliances towards SOC platforms is therefore necessary to improve the detection processes, though partially supported by task offloading for better efficiency. More efficiency is also expected, by avoiding to replicate monitoring and inspection operations for each appliance.

The broad set of security challenges for multi-domain value chains suggests the need for a wide range of *Security Services*, including but not limited to:

- **Attack detection** – Rule- and signature-based algorithms show their limits in addressing the growing complexity and continuous mutation of attacks, while the creation of legitimate profile usages is a complex and cumbersome task with a narrow scope. More intelligence is needed to process the security context and to correlate even apparently uncorrelated heterogeneous events and data (network traffic, log files, user behavior) from different systems. In this respect, the current challenge is to understand the appicability and effectiveness of existing Machine Learning methods, including but not limited to, K-Nearest Neighbors, Naive Bayes, Graph Kernel and Support Vector Machine.
- **Threat identification** – The investigation of new threats and detection of zero-day attacks requires the ability to inspect events, data, and measurements beyond what can be envisioned at design time. Programmability helps in this direction, by providing the freedom to define new filtering and monitoring processes at run time, tailored to the specific environment. Given the impracticability of elaborating detection rules for unknown threats, the real challenge is semi-supervised and unsupervised learning, which could autonomously reveal anomalies, i.e., non-conforming patterns compared to the well-defined notion of normal behaviour.
- **Data tracking** – The availability of security APIs in each digital service will enable to query about the presence and usage of private and sensitive data; in addition, any access to data should trigger a notification and the verification of user's policies. In this way, beyond enforcement of data access, records will be kept about the transfer of data to other services, enabling later verification of persistence and request for removal. Here, the main research challenge is the identification of new ways to trade data. Blockchain technologies might provide interesting solutions, since the problem is not far from Digital Right Management (DRM), which is already present in recent research roadmaps [29].
- **Trust and risk assessment** – When heterogeneous services are automatically selected from different domains to be chained together, their security properties (including, but not limited to: vendor identity, encryption/integrity/digital signing mechanisms, digital certificates, supported monitoring/inspection/enforcement hooks, installed software versions and patches) should be formally verified to satisfy the high-level trust policies (trusted vendors/countries, minimal encryption requirements, trust chains, security mechanisms, etc.) of users. In other words, users should be aware of the weakness, and should be able to decide whether it is acceptable or not. Trusted computing has already been largely investigated (especially in virtualized environments like the cloud and Network Functions Virtualization), but these kinds of solutions are not still available for composite services. Trustworthiness will involve the two dimensions of identity (service owner/provider) and integrity (software). Assuming the lack of a common authentication framework worldwide, the research challenge is to build reputation models based on recursive trust relationships, similarly to what already used in e-mail systems (e.g., PGP).

The combined analysis of the security context can greatly enhance the detection capability, especially in case of large multi-vector attacks. The challenge is clearly to merge knowledge without exposing sensitive information to external domains; in this respect, the notion of local processing and distributed security analysis may provide an effective solution for multi-layer detection mechanisms. The combination of heterogeneous monitoring data will open the opportunity for novel detection capabilities. For example, analysis of application logs that indicate multiple login failures may help to detect attack patterns in the encrypted network traffic. From a practical perspective, however, the real range of algorithms will be limited by the possibility to find an acceptable trade-off between the complexity to implement local inspection and the communication overhead.

From a more technical perspective, an *Orchestration Framework* is necessary to run and scale security services according to the current workload, so to not waste resources or delay the analysis. It is responsible of loading the services from an internal repository, deploying them, and performing life-cycle management actions. There are no specific requirements on the underlying infrastructure (a single server, a data centre, an IaaS cloud, big data framework) apart that a management interface must be available to automate as much as possible provisioning and instantiation.

5.3.2.3 Security Controller

Security Controller is the smart entity that contributes to implementing the "as-a-Service" concept, by automating as much as possible all security workflows. According to on-going initiatives [11], the main role of the Security Controller is mediation between control policies and security services. In practice, it translates high-level behavioural guidelines into concrete actions, rules and configuration settings applied both to the Context Broker and the Orchestration framework. This means that the Security Controller launches Security Services when required, feeds them with the relevant context, and dispatches notifications to humans through the *Dispatcher* function (**API #3**).

The implementation of the Security Controller may vary from bare event-driven engines to more complex rule management systems, up to the application of Artificial Intelligence to learn and improve strategies from humans' behaviour. Policies define the behaviour of the system. Conceptually, policies do not implement inspection, detection or enforcement tasks, so they do not correspond to any existing security function (IDS/IPS, antivirus, Virtual Private Networks). Instead, they represent an additional upper layer for control of security services. The definition of an ECA policy requires at least 3 elements:

- *Event* that defines when the policy is evaluated; the event may be triggered by the data plane (i.e., detection algorithms), the management plane (i.e., manual indications from the dashboard, notifications from the service orchestrator), or the control plane (i.e., a timer);
- *Condition* that selects one among the possible execution paths; the condition typically considers context information as data source, date/time, user, past events, etc.;
- *Actions* that respond, mitigate, or prevent attacks. Actions might not be limited to simple commands, but can implement complex logics, also including some form of processing on the run time context (e.g., to derive firewall configuration for the running instance). They can be described by imperative languages, in the forms of scripts or programs.

Accordingly, the specification of policies also changes from Event-Condition-Action (ECA) patterns to more abstract definition of goals, constraints, and even intents, which could be defined by non-technical users [11][30]. The adoption of advanced reasoning models, even based on some forms of artificial intelligence, is clearly a very promising yet challenging target to automate the system behaviour. This would open the opportunity for dynamically adapting the response to new threat vectors. In this respect, the historical analysis and correlation ofevents and conditions with the effects of the corresponding actions from existing policies or humans would provide useful hints to assess the effectiveness of the latter, so to identify and improve the best control strategies.

The range of possible operations performed by policies include enforcement actions, but also re-configuration and re-programming of the monitoring/inspection components in the execution environment. Enforcement and mitigation actions are mostly expected when the attack and/or threat and their sources are clearly identified and can be fought. Instead, re-configuration is necessary when there are only generic indications, and more

detailed analysis could be useful to better focus the response. A typical example is a volumetric DoS attack. To keep the processing and communication load minimal, the monitoring process may only compute rough network usage statistics every few minutes. This is enough to detect anomalies in the volume of traffic, but does not give precise indication about the source and identification of malicious flows to stop. Re-configuring the local probes to compute per-flow statistics or more sophisticated analysis helps implement traffic scrubbing.[66]

The degree of autonomy in taking decisions and applying the corresponding actions may also range from full automation to supervised mode, based on the fact some additional input or just explicit consent is required to human operators:

- **fully automated**: the framework reacts to specific conditions based on pre-defined rules, without any intervention from humans. This is only possible for well-known threats. For example, a packet filter may be installed when the traffic streams grow beyond a given threshold. Another example is the request to isolate or remove a service upon indication of an intrusion.
- **semi-automated**: in case of unknown or complex attacks, pre-defined policies might not be able to cover all possible situations or variants, so the system may only partially respond automatically and wait for further inputs from humans. This may be the case of anomalous (yet not overwhelming) flows of packets that are temporarily blocked while waiting for additional actions from the security provider.
- **supervised**: the system is able to react autonomously, but the likelihood or impacts of possible errors suggests confirmation from humans. In the same example as the previous point, the security provider is asked the permission to block the traffic, so to avoid disruption of any critical activity.

Automatic reaction shortens response times and unburden humans from mechanical and repetitive tasks. However, full awareness and the need for forensic analysis recommend to keep track and report any action to the dashboard, at least to give visibility of the occurrence of attacks.

We can give a concrete example of how the Security Controller is expected to behave in case of DoS service. Detection of volumetric DoS is typically based on analytics on the network traffic. Since deep inspection of the traffic leads to high computational loads and latency, an initialization policy only requires statistics about the aggregated network traffic that enters the service, which may be collected by standard measurements reported by the kernel. The same policy also initializes an algorithm for network analytics and sets the alert thresholds. Upon detection of an anomaly in the traffic profile, an event is triggered and the Security Controller invokes the corresponding DoS policy. The policy now requires finer-grained statistics, and the Security Controller selects an eBPF filter for packet classification, installs and configures it. The policy also requires the detection algorithms to work with the broader context information available. As soon as the analysis comes to a new detection, it triggers a new alert, this time including the relevant context (i.e., identification of suspicious flows, origins, etc.). Before taking the decision about how to react, the mitigation policy may evaluate some conditions to check if the suspicious flow comes from an expected user of the service, if it has been previously blacklisted or whitelisted, and if it is acceptable based on previously recorded time series. The actions to be implemented (e.g., dropping all packets, dropping selected packets, redirecting suspicious flows towards external DoS mitigation hardware/software, stop the service, move part or the whole service to a different infrastructure) is therefore

[66] Scrubbing is a technical term used to indicate a cleansing operation that analyses network packets and removes malicious traffic (DDoS, known vulnerabilities and exploits). It is usually implemented in dedicated devices or infrastructures, able to sustain high volumetric floods at the network and application layers, low and slow attacks, RFC Compliance checks, known vulnerabilities and zero day anomalies.

notified to the Security Controller, which again translates them in a set of commands for the external service orchestrator and/or configurations and programs to be installed in the execution environment. Notifications about the detected attack and the implemented actions are also sent to the Security Dashboard.

5.3.2.4 Information sharing

A SOC should always work in close cooperation with CERTs/CSIRTs, because they represent the main sources of information about new threats, vulnerabilities, and attack patterns; SOCs should also promptly report any discovered threat or vulnerability, so to contribute to limit the impact of zero-day attacks.

Unfortunately, despite of massive digitalization in all economical and business sectors, sharing of information about cyber-threats and security incidents is still largely based on paperwork and emails. This delays the knowledge of new threats and attacks, as well as the elaboration of remediation actions and countermeasures for every different context.

The creation of cyber-threat intelligence in an automated way needs to address two main aspects. Firstly, the creation of common models and abstractions for the description of threats, vulnerabilities, and infrastructures. Secondly, and more challenging, is the derivation of the necessary information from an on-going attack or offline analysis. While the former is already addressed by several standardization initiatives (e.g., STIX), the latter is still an open challenge with unclear directions.

The GUARD architecture includes a *CTI Importer/Exporter* to automate the management of CTI. It is expected to help security operators to build standard description of new threats as soon as they are identified and investigated through the GUARD platform. In this respect, they will collect data and record management actions that are meaningful to describe how the attack was discovered and mitigated. They should be also able to derive configurations and enforcement actions from the description of known threats and attacks, so that they can be applied to the monitored system through the Security Controller and the Context Broker. It is worth pointing out that this task is really ambitious, since no technical frameworks are available for this purpose yet. Therefore, the GUARD implementation of the CTI Importer/Exporter will not have the same TRL as other components.

5.3.2.5 Internal repositories

The different architectural components of the *Core framework* rely on a set of internal repositories that store the following information:

- *Programs Library*: though the long-term ambition could be the generation of new inspection, monitoring, and data fusion tasks according to high-level instructions or policies (i.e., a sort of ”compiler”), short/medium-term implementations will likely make use of user-defined programs. They should include proper descriptions, constraints, and any metadata that are necessary for their correct deployment and instantiation.
- *Service Library* contains software packages that are needed to run the corresponding security services. The package may only consist of the binary executable or deployment scripts, depending on the underlying infrastructure and orchestration tools.
- *Control Policies* define different security workflows in abstract terms. Policies are therefore used to automate the response to expected events, avoiding whenever possible repetitive, manual, and error-prone operations done by humans. The simplest way to define behavioural policies is the ECA pattern, which covers a broad range of interesting cases.

- *Cyber-Threat Intelligence* (CTI) describes possible threats as well as mitigation and protection strategies for one or a group of organizations or infrastructures. It is a bi-directional repository, which should merge existing and new findings for local usage and sharing with national and international CERTs/CSIRTs.

5.3.3 Security dashboard

The *Security Dashboard* is the main management tool used to build situational awareness, to perform reaction and investigation, and to share cyber-threat intelligence. The design of user-friendly human interfaces is among the key priorities for establishing the success of commercial solutions. Beyond visual appearance, it is worth widening the base of users and tailoring the informative content to their specific role and background. As a matter of fact, bare technical information (e.g., available algorithms for encryption or integrity, the software version) will be totally useless for most users. Indeed, tailored informative contents should be delivered at different levels of the company's structure, to bring awareness to humans and ensure the better understanding of the current situation. For example, loss or uncertainty in the position of private data should trigger a warning about potential violation of a specific regulation to the legal staff. Any loss of integrity, data, or availability should be reported to the management staff, in terms of potential impact on the overall company business (block of the production, loss of customers, bad reputation). Risk assessment at the management layer also requires to automatically feed existing tools, reducing the reliance on labour intensive and potentially error-prone analysis by experts.

For the purposes of reaction and investigation, the Security Dashboard can be used to select specific analysis and detection algorithms, to visualize anomalies and security events and to pinpoint them in the service topology, to set run time security policies, and to perform manual reaction. With respect to the last two options, it has to be pointed out that security policies are the best way to respond to well-known threats, for which there are already established practice and consolidated methodologies for mitigation or protection. However, the identification of new threats and the elaboration of novel countermeasures require direct step-by-step control over the ongoing system behaviour. The dashboard should therefore allow direct interaction with the Context Broker, to set the most suitable monitoring and inspection configuration in each remote service.

5.3.4 Identity management and access control

Following the generalized trends towards improved modularity and agility, the implementation of a platform for SOCaaS is likely to adopt micro-services and service mesh architectural patterns. North-South (N-S) interactions take place with external components (like digital services and management dashboards), whereas East-West (E-W) transactions involve the internal components already identified in Sec. 5.3.2. Without dwelling on practical considerations about secure deployment of the platform itself, it is clear that each micro-service should include a specific sidecar for authentication and access control.

An Authentication, Authorization, and Accounting framework *AAA* is therefore necessary to mediate between multiple domains and protocols, especially in N-S communication. However, E-W transactions might contain information about service usage patterns, users, exchanged data, and so on. Access to this data should therefore be limited to authorized roles and algorithms. In addition, configuration of the remote data plane must remain a prerogative of the security controller and trusted policies, so it is important to track the issuer of such commands.

Given the distributed nature of the system and impermanence of relationships, identity management will be based on a Public Key Infrastructure (PKI) and the availability of digital certificates for each entity. The presence

of multiple and potentially untrustworthy Certification Authorities would demand for some reputation models based on recursive trust relationships, similarly to what already used in e-mail systems (i.e., PGP).

The AAA framework decouples authentication and authorization functionalities. Internally, an Identity management (Idm) component maps identities of both users and micro-services to a specific list of attributes. Such attributes are then delivered to each security sidecar and used to protect resources or grant the access to protected resources during the authorization phase according to the Attribute-Based Access Control (ABAC) logic. The same set of attributes may also be used for confidentiality of the communication between pairs or groups of users, by implementing Attributed-Based Encryption (ABE) mechanisms.

5.3.5 Mapping to the functional model

The set of components described by the reference model implements all logical functions identified in Sec. 5.2. The internal repositories are listed together with the component that uses them. The only component that does not have a direct mapping in a logical function is the Orchestration Framework, because it is not directly related with the security management workflow.

Subsystem	Component	Logical function
Local security agents (digital service)	Security Functions	Enforcement Inspection Monitoring Pre-processing
	Local Control & Management framework	Configuration Local Agent
Core framework	Context Broker Program Library	Data Collection Programmer Control Interface Management Interface
	Security Controller Control Policies	Selection and Configuration Control and Reaction Policies
	Orchestration Framework Service Library	-
	Security Services	Detection and Correlation Topology Detection and Analysis
	CTI Importer/Exporter CTI	Threat Intelligence
Security Dashboard		User Interface Policy Editor Program Editor
AAA		AAA

5.4 Software architecture

The software architecture describes the software that implements the GUARD framework. At this stage, some basic assumptions can already be made about existing tools that can be adopted in the architecture, but some other components must be developed by the Consortium. The current description is therefore the preliminary outcome of the design phase, which will be revised in the following months based on the integration work and the enhancements delivered by the technical WPs.

Figure 17 shows the GUARD software architecture, where there are two main blocks expected. One is composed of the Core Platform and the GUARD Dashboard, which are deployed and operated by security providers. The other one is the Local Security Sidecar, which includes local security agents deployed in each digital resource. From an exploitation perspective, the Project mostly focuses on the core platform, because the implementation of local security agents might require specific adaptation for different technologies and platforms. For instance, the software to be integrated in a CMS will be different for that of a serverless function. However, a reference implementation of security agents is required to evaluate the overall framework and to demonstrate the GUARD technology in the Use Cases.

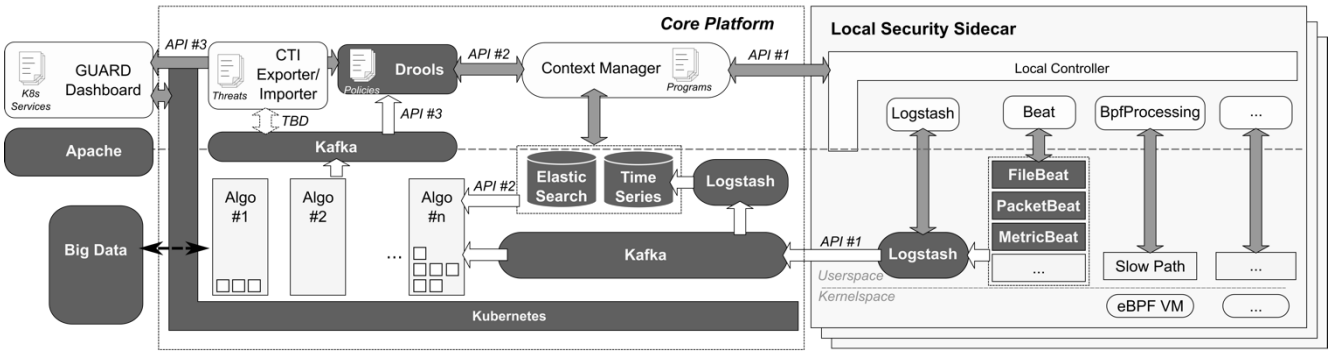


Figure 17. GUARD software architecture. Dark grey boxes are software tools adopted by existing (mainly open-source) projects, whereas white and light grey boxes are novel components taken from on-going projects or developed internally by the Consortium. The dashed line indicatively separates the data plane (bottom part) from the control and management plane (upper part).

5.4.1 Local Security Sidecar

Local security agents will be delivered in a modular form, since today the micro-service architecture is a common architectural pattern for creating digital services. In this respect, the primary deliverable method could be a Docker image or a Kubernetes POD, which fit very well the different needs of low-end devices and bigger installations. If a Kubernetes POD is used, different containers can be used for every software unit and it may also provide autoscaling and high-availability. Other delivery methods (e.g., the preparation of deb, rpm or tarball packages, ansible roles) for specific Linux distributions might be considered for dissemination or commercial exploitation. The delivery method and internal implementation of the Local Security Sidecar is not relevant for GUARD, which is mainly interested in defining a common API to access multiple security functions.

The internal structure of the sidecar reflects the need for both a data and a control/management plane. The control and management components are depicted in the upper part of Figure 17, while the data plane components are at the bottom.

The **Local Controller** is the management component that oversees all the operations of the local security agents. As the name suggests, it exposes a REST API that is used by the remote GUARD Core Platform to retrieve relevant security context for analysis. This component is responsible for the following tasks:

- Description of the digital service and its security properties. This information is used to identify the service, its owner, its placement, and the available security capabilities. This allows to assess the overall level of security and trustworthiness.
- Send notifications about any relevant change that the remote Core Framework is subscribed to. For instance, the most common usage could be to detect topology changes and migrations of the software to different infrastructures.
- Description of the available security agents and their control APIs. This avoid the need for static and pre-defined communication interfaces, according to common practice in SOA and micro-services.
- Proxy control and management requests towards internal security functions.
- Authenticate users and apply access control to internal security capabilities. This is strictly necessary in order to turn the whole framework into an additional threat for digital systems.

One challenging issue for the Local Controller is the ability to dynamically update the set of exposed APIs when new security agents are deployed. There is currently no compelling framework available for this task. Polycube[67] is a good example of the target design for the Local Controller, but it is strongly focused on creating chains of network functions and the preliminary analysis decided that it could not be adapted for the specific usage.

Several security agents will be integrated under the Local Controller. According to the project's objectives, *programmability* is the challenging design issue. In this respect, a control plane is expected for each component that allows to change its configuration at run time, also by delegating lightweight processing tasks whenever possible. Such control plane might be natively integrated in the agent or delivered as standalone component on top of the plain monitoring and inspection functions. The second option will be adopted for existing tools that are not programmable yet.

The preliminary inventory of tools from the open-source community and partners includes the following components:

- **Elastic beats** are a set of lightweight data shippers developed in the framework of the Elastic Stack. Their scope includes monitoring and inspection of system operation and network traffic. The minimal set of beats will encompass FileBeat (log files), MetricBeat (system-level CPU usage, memory, file system, disk IO, and network IO statistics), and PacketBeat (latency, errors, response times, SLA performance, user access patterns). Additional components that may be of interest are AuditBeat (user activity and processes) and FunctionBeat (serverless shipper). Beyond the few official beats, there are a lot of components developed by the community[68], and additional ones can be easily developed that tailor to specific applications and frameworks. Most beats can be largely configured from file, but do not expose management interfaces (this is rather reasonable, since they are designed as lightweight tools). It is therefore necessary to design and develop a Beat control component that exposes a REST interface to change the local configuration files.
- **eBPF framework** is a lightweight and programmable filter built-in the Linux kernel for parsing system calls. It was originally designed as a network monitoring tool only (the Berkeley Packet Filter), but it has now evolved with the capability of handling generic event processing in kernel, JIT compilation for increased performance, stateful processing using maps, and libraries (helpers) to handle more complex tasks, available within the kernel. eBPF programs can be efficiently used for deep packet inspection and

[67] Polycube network: https://github.com/polycube-network/polycube.
[68] Community Beats: https://www.elastic.co/guide/en/beats/libbeat/current/community-beats.html.

tracing of system calls. A full-fledged framework for dynamically injecting eBPF programs that ship data to Logstash is currently under development in the ASTRID project and will be tailored to the GUARD platform.

- **Logstash** is part of the Elastic Stack as well. It is a modular data processing pipeline that gathers data from heterogeneous sources (Beats and other agents, aggregates or pre-processes it, and then sends it to one or more remote consumers (Core Platform). Logstash addresses the need for harmonization of data from multiple heterogeneous sources, by parsing each event, identifying named fields to build structure, and transforming them to converge on a common format. In GUARD, Logstash will deliver data through a Kafka instance running in the Core Platform. Centralized pipeline management is already integrated in Kibana, but is falls outside of the basic free license. A thin adaptation is therefore required to harmonize the Logstash API in the Local Controller of the security sidecar.

Additional components can be integrated for addressing more specific scenarios. Currently, on-going technical verification are considering the following components:

- **virtual Deep Packet Inspector (vDPI)** is a virtual function for inspecting packets' bodies, based on the open-source nDPI project. It filters L7 protocols like HTTP/FTP/BitTorrent traffic even if it is directed to different ports than the defaults. As of current implementation, the vDPI agent cannot be embedded in a container, but it could be useful for NFV scenarios. Indeed, in NFV infrastructures, the creation of networking services will heavily rely on chaining several high-performance service functions, and the vDPI could be effectively placed within the topology to inspect the traffic streams. Configurability of vDPI is under evaluation.

In addition to the set of components that can be used for monitoring and inspection, some specific components are required to manage data propagation across different services. In this case, the definition of generic tools looks unreasonable, because the identification of data objects will be specific for each application. The Project will therefore integrate data tracking capabilities into the eHealth application of Use Case #2. It represents a valuable asset per se up against commercial exploitation of the involved partners, but it could also serve as reference implementation and code pattern for other applications.

Table 4. List of software components in Local Security Sidecars.

Component	Source*	Expected activities	Partner	Task
Local Controller	RP	Update and tailor to the GUARD framework	CNIT	4.3
Elastic beats	OS	Develop a suitable control plane.	CNIT	4.2
eBPF framework	RP	Contribute to development, tailor to the GUARD platform, integrate with REST API. Extends PacketBeat with additional protocols.	CNIT ITL	4.3
vDPI	I	Definition of relevant scenarios. Remote control and integration with GUARD API.	8BELLS	4.1
xBeat	I		ITL	4.3

Component	Source*	Expected activities	Partner	Task
Data tracer	-	Design and develop a component tailored to the eHealth use case.	WOB	4.5
Logstash	OS	Integration of the remote management interface in the GUARD API.	CNIT	4.3

* OS=Open Source, RP=Research Project, I=Internal proprietary tool

5.4.2 Core platform

The reference model has already pointed out the need for an internal orchestration framework that could deploy and manage multiple security services at run time. There is no specific constraint or contraindication that the same approach could not be used for other internal blocks as well.

The preliminary technical decision concerns the virtualization technology. The main alternatives are virtual machines and containers. Roughly speaking, VMs provide better isolation in terms of performance and security and more freedom in selecting the software platform for development (CPU architecture, OS, libraries); however, this comes at the cost of more overhead on CPU, memory, storage, and networking. Indeed, the processing overhead can be partially mitigated with para-virtualization, which removes the need to emulate hardware devices. The GUARD platform will therefore opt for containerization, which simplifies the creation and distribution of different architectural elements as micro-services.

Kubernets is currently the orchestration solution for containers with the largest community of developers and users. It is used both for research and production deployments, so it gives enough guarantees of availability, continuity, and reliability. Kubernetes POD can be easily created, and many tools are already delivered in this form. Each architectural element of the GUARD framework will therefore be organized as POD; in case some components are strictly related (for instance, the elements that implement the Context Broker function), they may be clustered in the same container. To protect the platform and avoid the propagation of attacks, each micro-service will have its authentication and authorization sidecar, including the software for authorizing users, requests, and transactions. The Kubernetes installation must also keep a registry of available services and network endpoints (IP address, TCP/UDP ports)[69].

Following the same considerations made for local security agents, the implementation of the Context Broker will be based on the Elastic Stack. It will include:

- a Logstash instance, which may be used for a final stage of data aggregation and fusion, taking inputs from all sources;
- a Kafka bus, which delivers data for both storage and real-time usage;
- two databases to store configurations and historical data;
- a Context Manager which will implement the abstraction to configure the local security agents.

The software architecture for the CB includes a Logstash instance, which can be used to do central aggregation of data, timestamping, or other operations on the data before they are delivered to intended recipients. The concrete delivery of data is made by Kafka, a well-integrated component in the Elastic Stack. The Kafka message broker delivers the data collected from digital services to multiple subscribers. One subscriber is the

[69] See: https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/.

TimeseriesDB, which keeps historical data for offline processing and statistical analysis. Notifications about new digital services or different security capabilities will feed the Elasticsearch database, which is designed to store the overall system abstraction (digital services, security properties, configurations, programs). It is a NoSQL search and analytics engine, a perfect choice for storing unstructured data and performing queries on graph-based topologies. Another important aspect is the ability to perform quick look-ups and queries, also including some forms of data fusion. The final component for the Context Broker is the Context Manager. It is responsible to implement a REST-based interface for accessing information in Elasticsearch. The REST model will be based on the abstraction of the security context, which must capture the topology of the chain, the nature and properties of each service, the capabilities and current configuration of the security agents. The Context Manager mediates access to the internal databases, hence implementing authentication and authorization on their behalf. It must also include a number of internal plugins to translate the abstraction into concrete configuration settings for heterogeneous security agents. Such plugins will configure local agents through the interface they expose through the Local Controller. The Context Manager is therefore responsible to implement the control channel and to apply security mechanisms that prevent attacks through this vector (e.g., encryption, authentication, integrity). In this respect, one key issue to be solved by T4.5[70] is the design of key distribution mechanisms.

Each Security Service is implemented in a Kubernetes POD. The internal implementation is left to cyber-security vendors, but a few technical constraints should be met:

- Each service will listen for Logstash messages on the Kafka bus, and will query the Context Broker to get topology, configurations, and historical data.
- Each service will publish IoC on the control Kafka bus.
- Each service will define its internal structure (PODs, scaling rule) for orchestration at run-time.

The adoption or orchestration paradigms will allow to scale Security Services at run time, so they can take more resources when the amount of data to process increases. At the same time, the micro-service architecture allows to design them in a modular way, and just to scale the internal components that get overloaded. It is only possible to develop a sort of control plane, and to rely on an external big data infrastructure (e.g., Apache Spark, Hive) for efficient processing of large bulk of data. The Project will not consider this option for internal implementation. The extension would be quite straightforward from the technical perspective, but some additional considerations will arise about the location and ownership of the additional infrastructure.

In principle, existing security appliances (antivirus, IPS, IDS, firewalling, anti-DoS) could be ported to the GUARD platform, given that they are adapted to work with the GUARD Context Broker instead of their own inspection and monitoring tools.[71] However, to better show the improvements and innovation brought by the GUARD platform, 5 new services are expected to be delivered by the Project:

- one for detection of known attacks;
- one for identification of new threats;
- one for topology discovery;

[70] T4.5: Identity management, access control, and policies enforcement.

[71] IPS, IDS, firewalls, DoS mitigation are quite simple to be ported to the GUARD platform. Antivirus software is more complex, because they usually need the ability to inspect binary code during execution, which will not be possible with the GUARD platform.

- one algorithm to track the location of users' data and their propagation along different services;
- one for assessing the current level of trust and risk, which can be used for internal management decisions.

This basic set is conceived to demonstrate the feasibility and effectiveness of the platform, and to show that it enables the design of new algorithms beyond the current state of the art. Additional security services might be developed by fostering collaboration with parallel complementary projects.

A second Kafka instance implements the Dispatcher function depicted in the reference model. It is used to bear IoC from the set of algorithms, so that proper countermeasures can be taken and logged. From an implementation perspective, a single Kafka instance might be used with different *topics*. The choice is left to the integration activity carried out in T2.6[72].

Drools is the core engine to automate response and mitigation actions. It is a business rule management system written in Java and could be used to trigger actions on specific events and TI optimized decisions based on a set of constraints. Drools has been selected for implementing the GUARD Security Controller, which should be able to translate high-level policies into concrete management and control actions. It should be noted that actions do not directly correspond to configurations and settings on remote agents, since this requires an additional layer of abstraction implemented by the Context Broker.

The CTI Exporter/Importer is a new tool envisioned by the GUARD platform. The lack of reference implementations requires a full design of this component, which is expected from T3.5[73] and T5.3.[74] At this stage, it is not possible to define the interfaces that will be used to collect data, traces, dumps, and everything else is needed to create CTI, because this information will only be defined during the project. The revised system architecture will include the design of this component.

The Apache Web server will be used to run the Dashboard. The choice for the GUI is therefore a Web-based tool, which avoid the need to install specific software on the management terminals. Partners might be interested in developing their own clients for the dashboard (e.g., Android/IOS apps for smartphones and tablets), but this is part of the individual exploitation plans.

The Security Dashboard will be developed in Angular. The dashboard will be a multi-user tool, with a personalised content based on the specific role. Multiple *tabs* will be available for the following features:

- Awareness: representation of real-time and historical data, logs from the Context Broker, IoC from security services, visual analytics. For this part, Kibana will be used.
- Service composition: the Dashboard will display the current set of digital services and will allow to block them in case they are considered untrusted. It may also provide a list of alternative implementations that have been selected in advance for specific functions.
- Editing and repository management: the dashboard will provide access to the internal repositories (policies, inspection programs, security services, threats). It will allow to upload new elements, edit and removing existing ones.

[72] T2.6: Continuous Integration and Software Releases
[73] T3.5: Information sharing and collaborative tools
[74] T5.3: Retrieve CTI from novel findings and attack detection

- Orchestration: the dashboard will implement a simplified interface to Kubernetes API, for managing the life-cycle of the components of the GUARD platform.
- System management: the dashboard will manage user identities and access rules.
- Investigation and reaction: though many operations will be automatically carried out by Drools based on high-level policies, manual intervention may be required to investigate zero-day attacks, mitigate complex threats, build CTI. The dashboard will provide direct access to the Context Broker, hence allowing system administrator to change the configuration of security agents. Access to the CTI Exporter/Importer will allow to drive the process of describing threats.

Clearly, the set of tabs available to the operator will depend on his role.

The software components for the AAA framework are not depicted in Figure 17 because they depend on the technologies that will be designed and developed during the Project. Indicatively, the AAA framework will rely on a PKI, because components from different domains are involved. Likely, the certificates of different digital services will be signed by different Certification Authorities, even local. An Idm component will contain a database that maps the identity of both Users, Local Agents, and any other components belonging to the Security Manager to a specific list of attributes. Authorization procedures and policy enforcement will be managed through the DMA-CP-ABE algorithm; an ABAC/ABE component will deliver attributes to Users, Local Agents, and any other components belonging to the Security Manager through a trusted file structure (thinks for instance to an extended version of a JSON Web token). Once authenticated, users can use attributes in their possession for accessing to resources and services available within the architecture. Depending on the access policy, they must demonstrate to be in possession of the right set of attributes by performing specific cryptographic operations. A more concrete software architecture of the AAA framework will be defined by T4.5[75] and included in the next release of the GUARD architecture.

Table 5. List of software components in the core platform.

Component	Source*	Expected activities	Partner	Task
Kubernetes	OS	Configuration and set up of an image repository.	ITL	2.6
Security sidecars	-	Design and development of a specific tool.	WOB	4.5
Kafka	OS	Configuration.	CNIT	2.6
Logstash	OS	Configuration.	CNIT	2.6
Timeseries DB	OS	Configuration.	CNIT	4.4
Elasticstack	OS	Definition of the abstraction model, configuration.	CNIT	4.4
Context Manager	RP	Definition of data models for the use cases. Plugin to apply configurations to different security agents.	CNIT	4.4

[75] T4.5: Identity management, access control, and policies enforcement

Component	Source*	Expected activities	Partner	Task
Algo #1	-	Detection of know attacks.	NASK	5.1
Algo #2	-	Detection of new threats.	AIT	5.2
Algo #3	-	Topology discovery.	ITL	3.3
Algo #4	-	Data tracking. To define structure/format to describe relevant data units (owner, content, properties, position, sensitivity, usage policies)	WOB	3.4
Algo #5	-	Trust and risk assessment.	FORTH	5.4
Drools	OS	Definition of high-level policies. Development of internal functions to implement actions and interfaces.	FORTH	3.3
CTI Exporter/ Importer	-	Design, development, definition of interfaces towards other GUARD components.	AIT	3.5 5.3
Apache	OS	Configuration.	ITL	2.6
Dashboard	-	Design and implementation. Integration with other tools to manage the GUARD platform at run time (developed by different partners).	M&S	3.2

* OS=Open Source, RP=Research Project, I=Internal proprietary tool

5.4.3 Interfaces

The interface between the Core platform and digital services (**API #1**) can be split into two parts: the data channel and the control channel. The data channel is expected to ship data from local security agents to the Context Broker. Since the software architecture has already selected Logstash for this purpose, the natural choice is to adopt its interface without introducing additional layers. As a matter of fact, the usage of Logstash ensures compatibility with existing frameworks and this is expected to act as catalyser for encouraging the adoption of the GUARD platform. For the control channel, a new interface should be defined, based on the REST approach. This interface should manage general properties of the service (identity, capabilities) and proxy requests to internal security agents. The structure will be defined during the project, trying to align it to on-going initiatives as much as possible.

The interface towards the Context Broker (**API #2**) will be used for Context abstraction. Context abstraction is responsible to expose data and events stored in the internal storage system in a structured way. Indeed, the freedom in defining custom programs means that the generated data and measurements will be heterogeneous and unpredictable at design time. The diversity in time and space when collecting the data must be tackle by a proper abstraction that facilitates the understanding of what can be collected and what is really available for each monitored component at each time instant. The main purpose for an abstraction layer is therefore to provide uniform access to the capabilities of monitoring agents. There are two main aspects to be covered by the Security Context Abstraction (SCA):

- hiding the technological details and the heterogeneity of the monitoring hooks;
- abstracting the whole service graph and the capabilities of each node.

The interface should be defined in very general terms, so to be easily extended and adapted to new security agents. The definition of the structure for every possible kind of security agent however falls outside of the Project scope. The Project will only define data models for the specific security agents that will be developed internally.

The last interface (**API #3**) will be used between Security Services and the control and management framework (Smart Controller, CTI, Dashboard). The primary usage will be the collection of IoC from detection algorithms and the control of the operation of security services. Both operations are still open to discussion, so the selection of design choices is postponed after the preliminary outcomes from the relative technical WPs. As preliminary indication, the usage of common formats for IoC should be preferred, as those defined by IETF [31] and OASIS [32]. For control of security appliances, there is an on-going framework from IETF [10][11], but at this moment it is not clear if the working group will close or continue the specification. Finally, for managing the life-cycle of security services, Kubernetes already provides a common orchestration framework.

Table 6. List of candidate solutions for internal and external GUARD interfaces.

API	Purpose	Candidates	Notes
#1	Data collection (Digital services to Context Broker/Security services)	Logstash interface	Needs a mechanism for key exchange.
	Mgmt and control (discovery, browsing, configuration)	REST API (structure to be defined)	Specific interfaces are needed for each security agent.
#2	Context abstraction. It is the interface exposed by the Context Broker to Security Services, Security Controller and Security Dashboard.	REST API (structure to be defined)	The Context abstraction pursues a general model that can be used to describe any security agent and related configuration/settings. The project will define specific structures for the Use Cases.
#3	Indication of Compromise (IoC). Configuration of security services.	Kubernetes I2NSF IODEF, STIX	Designed postponed after preliminary outcomes from technical WPs.

6 Relationship to on-going initiatives

The current market of cyber-security products is largely fragmented. In addition to legacy firewalls, intrusion prevention/detection systems and antivirus for personal and enterprise usage, many vendors are already tackling new business opportunities by delivering solutions for enterprise's networks and endpoints, pure and hybrid cloud, industrial devices and networks, security analytics, and integrated solutions. The last class of products aims at getting real-time visibility into all activity on systems, networks, databases, and applications; such solutions apply artificial intelligence and machine learning to wide data sets, facilitating response to attacks by quick one-touch remediation actions. Information from vendors are extremely blurry about technical features and architectures, giving the impression that a tailored solution is developed for each customer by composing discrete standalone appliances and components.

Beyond commercial solutions, there is a general understanding that more dynamicity and flexibility is needed in cyber-security platforms to effectively address evolving cyber-security paradigms. A large effort has been devoted to improve detection processes by shifting from signature and rule-based mechanisms to identification of anomalies and unknown patterns, so to better tackle morphing and zero-day attacks. However, there have been less advances in solving the technical problems related to the management of virtualized and multi-domain systems. In this respect, the GUARD approach builds on ground-breaking concepts taken from two major trends:

- orchestration of security functions, so to adapt the cyber-security platform to the evolving context (see Section 6.1);
- the design of technical means that facilitate the exchange of data within business ecosystems (see Section 6.2).

6.1 IETF Interface to Network Security Functions (I2NSF)

From a purely architectural perspective, the I2NSF framework [11] is the most relevant effort that can be compared with GUARD. As preliminary consideration, we note that the GUARD architecture is largely compliant with the I2NSF framework, and it may represent a possible implementation. However, there are some important differences that make GUARD more challenging and ambitious with respect to the framework currently described by the I2NFS working group. Here, we briefly compare the two frameworks under several aspects; the interested reader finds a detailed description of I2NSF in Annex A.

Scope. As the name implies, the I2NSF framework specifically looks at network security functions, i.e., security services that manipulates network packets. The framework targets multiple domains (access networks, enterprise networks, cloud services), and applicability to both physical appliances and virtual instances. It covers the control plane (i.e., selection and configuration of NSFs), but the definition of mechanisms for translating high-level policies into policy rules is currently out of scope for the working group. The data plane and management plane do not fall in the scope of I2NSF; the data plane is totally left to vendors of security functions, and no specific assumptions or requirements are made on deployment, device configuration, life-cycle management, and resource provisioning. GUARD has a broader scope, also including data protection in addition to network threats. GUARD explicitly covers the data, control, and management planes, targeting better efficiency and programmability, so to effectively tackle the continuous evolving threats landscape. Though, this does not preclude the possibility to use standalone security appliances. As final consideration, we note that both frameworks target monitoring and enforcing policies, even though I2NSF is more leaned to enforcing and GUARD is currently more focused on detection and situational awareness.

Business model. Both GUARD and I2NSF considers a similar business model, where Security Providers take care of security services on behalf of their customers. In both cases, customers are allowed to describe their security requirements in terms of high-level policies, whereas the translation to configuration rules is managed by the framework (with automatic or manual procedures). GUARD explicitly addresses the scenario of multiple administrative domains and heterogeneous digital resources, which is not currently included in the scope of I2NSF. Overall, the GUARD business model could be viewed as an enhanced version of that of I2NSF for virtual services.

Architecture. The I2NSF reference model and the GUARD architecture are compared in Figure 18. In a bird's eye view, the GUARD architecture is more complete, since it is rapidly evolving not being slowed down by a standardization process. They both rely on coordination from a Security Controller, which applies security policies. In both cases, internal policies are defined as flow policies (for instance, in ECA form) that are translated by the Security Controller in specific policy rules for each NSF. Such policies are instantiated through the Security Dashboard in GUARD; the translation may either be performed automatically or managed by the Security Provider. In the I2NSF model, the translation is directly managed by the Security Controller; however, it is currently suggested to express user policies in a ECA form that can be mapped to policy rules in a straightforward way (Sec. 7.1 of [11]). The I2NSF Consumer-Facing Interface can be mapped to the API/interface of the GUARD Security Dashboard. On the right side of the picture, the I2NSF framework control physical and virtual instance of security appliances (NSF), but it dictates that they implement the NSF-Facing interface. GUARD could control vNSF as well, but for the sake of efficiency and programmability it mainly focuses on the separation between inspection tasks and the detection logic. The combination of detection algorithms and local inspection agents are totally equivalent to a NSF, but the substantial difference is the far greater flexibility in composing bespoke security functions to the specific service implementations. GUARD does not mandate a standard interface for all security functions (vNSF, detection algorithms, local inspection agents), which is not a feasible approach in the short/medium period, but leverages a local Control Plane and a centralized Context Broker to harmonize the different dialects spoken by the set of heterogeneous controlled elements. The I2NSF NSF-Facing interface may be implemented between the GUARD Security Controller and the Context Broker, as specific extension of the GUARD APIs. In both frameworks the Security Controller needs knowledge about the security functions available. In I2NSF, there is a dedicated Registration interface, and a unspecified "Management System" that depends on the specific environment. It may be part of existing management tools in large infrastructures (as the networks of telecom operators or the data centres of cloud operators). In GUARD, such component is not present, because the deployment in different administrative domain would not make this option realistic (GUARD only assumes that the service description is exposed). In case of specific Use Cases, where the GUARD framework is operated within a single administrative domain, an extension can be developed to account for management purposes. As part of the management framework, the I2NSF is already addressing remote attestation of both the Security Controller and NSFs. This aspect is not currently listed among the GUARD objectives, but it represents a potential security service to be run on the core platform. To this aim, the work carried out by parallel research projects like ASTRID[76] could be easily ported to the GUARD framework as well. Finally, though not explicitly shown in Figure 18, an AAA framework is necessary in both cases to establish secure

[76] The ASTRID project investigates tight integration of security processes with software orchestration in virtualized environments. It also includes remote attestation of virtual function as specific Use Case to be validate on the project platform. Web site: https://www.astrid-project.eu/.

relationships and channels between the involved elements. GUARD will explicitly address this aspect, whereas concrete proposals are still missing in the I2NSF framework.

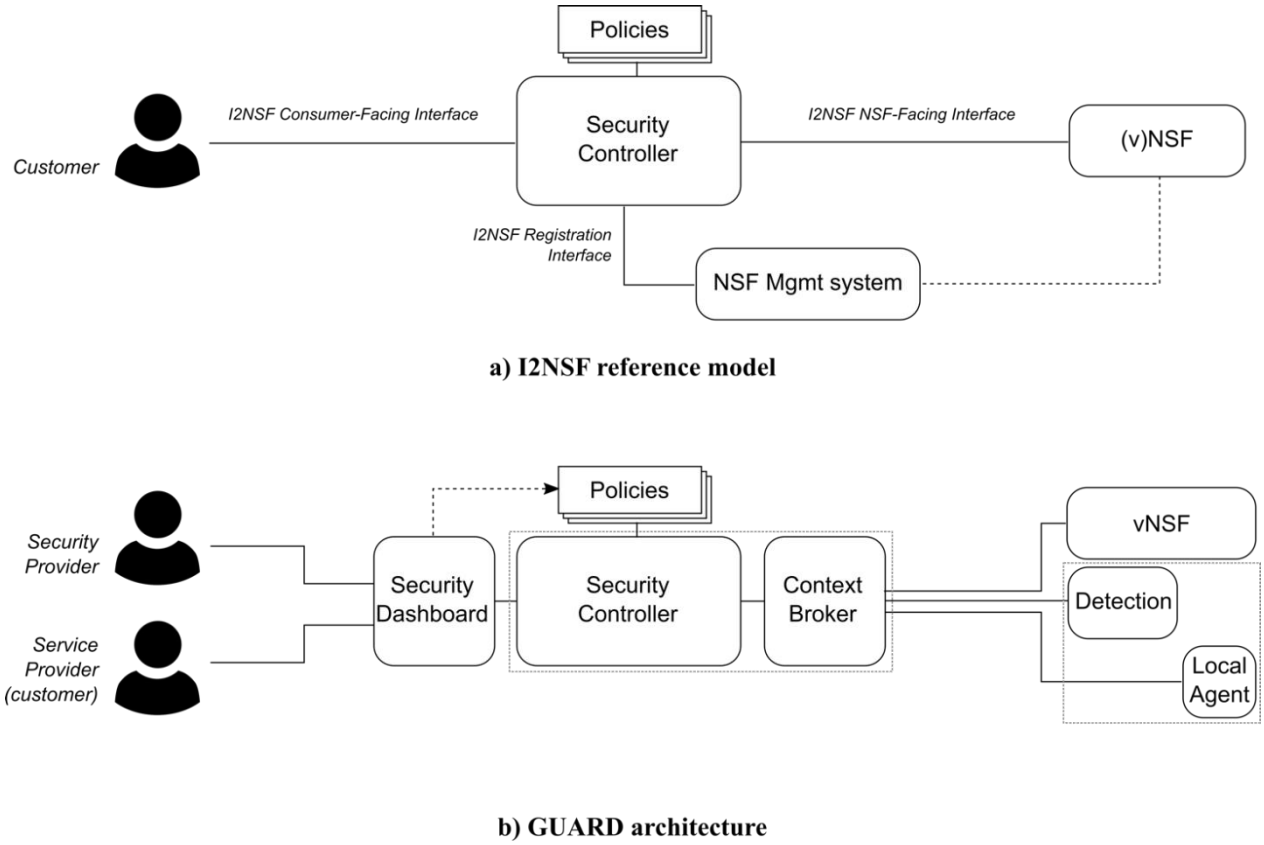


Figure 18. Comparison between the conceptual architectures for I2NSF and GUARD.

6.2 IDSA Reference Architecture Model

The International Data Spaces (IDS) defines a virtual space for trading data across multiple players [33]. The main purpose is safe, secure and standardized data exchange in a trusted business ecosystem. As such, this initiative is strictly related with GUARD and represent one interesting target for exploitation. However, it is worth pointing out that GUARD is not an implementation of the IDS. In the following, we describe the main relationships between IDS and GUARD. The interested reader finds an overview of the IDS architecture in Annex B.

Scope. As the name suggests, the IDS is about the creation of an open ecosystem for exchanging and sharing data in a trusted way. From the technical perspective, the IDS architecture defines a sort of proxy component (the Connector) which is expected to be positioned at the boundary of every administrative domain (enterprise, cloud service, IoT device, etc.). The IDS Connector mediates access to data according to access and usage policies established by the data owner. It also runs data services on demand, which are able to aggregate, transform, and process data. The interconnection of all Connectors from the participants in the ecosystem builds the virtual data space. The definition of communication protocol is just a marginal aspect in the IDS; indeed, the real challenge is the establishment of trust relationships that govern all operations carried out in the data space. To this aim, the Reference Architecture Model dictates identification, verification, and certification of all entities involved in the ecosystem, both participants and their technical frameworks (Connections and applications). Access and usage policies, integrity verification, and data tracking are required by the IDS architecture. The GUARD framework does not implement a virtual data space, but provides effective mechanisms to monitor such

ecosystem. Indeed, the scope of GUARD includes trust and integrity aspects, but also extends to detection of attacks and anomalies. In this respect, the GUARD platform represents a concrete and advanced solution to ensure safe operation of the IDS.

Business model. GUARD and IDS are driven by totally different business models. The IDS defines the players that are expected to be involved in a data-driven value chain, where data are exchanged by multiple participants without static and well-defined service topologies. So the main business here concerns the usage of data to create value-added service. The GUARD platform is conceived to support SOCaaS, hence the main business is externalization of security operations. The two business models are fully complementary: indeed, the IDS Reference Architecture Model requires the implementation of security services, but does not clarify who is in charge of them. In particular, certification is required and each Connector should expose its security properties, but there is not clear indication about where and how the verification should happen.[77] The GUARD Security Provider can therefore be thought as an additional role in the IDS ecosystem, which take the responsibility of verifying the compliance with the required security levels.

Architecture. The relationship between the GUARD and IDS architecture is pictorially described in Figure 19. GUARD local agents can be deployed in a custom container application in an IDS Connector. Indeed, the set of local agents (and annexed control plain) has been described as a "security sidecar" in Section 5.3.1 just because this approach fits many interesting scenarios for digital services. Once a GUARD sidecar is included in every IDS Connector of an IDS ecosystem, the centralized platform can virtually implement any kind of security services, including but not limited to detection of known attacks, identification of anomalies, risk assessment, verification of trust properties, remote attestation, and so on. The most important question is whether a GUARD sidecar would be able to support the set of security concepts selected by the IDS Reference Architecture Model. Since concrete implementation of the IDS Connector are not widely available so far, it is impossible to give a final answer, but the following considerations justify because GUARD could realistically achieve such objective:

- most of GUARD agents are adopted for the Elastic Stack framework, and they can be easily deployed in separate containers, provided that a few technical constraints are fulfilled (e.g., the file system where log files reside is shared);
- GUARD is targeting the eBPF as programmable and flexible inspection technology; eBPF programs are pushed to the kernel, which is shared between the different containers, so it can easily monitor both network packets and system calls;
- the GUARD framework already targets the development of authentication and authorization sidecars that can apply access controls on the interaction between different micro-services;
- messages exchanged between data services and Connectors can be easily intercepted by GUARD network agents and submitted to the centralized framework for access and usage control, hence implementing in an effective way the monitoring of the data pipeline envisioned by Sec. 4.1.3.6 of the Reference Architecture Model [33];
- the GUARD security sidecar is already expected to expose security-related properties and capabilities of the digital service, and this could be used for the IDS Connector as well.

[77] The Reference Architecture Model describes some typical architectures for policy enforcement and hierarchical certification, but it is not currently clear on the verification of security properties.

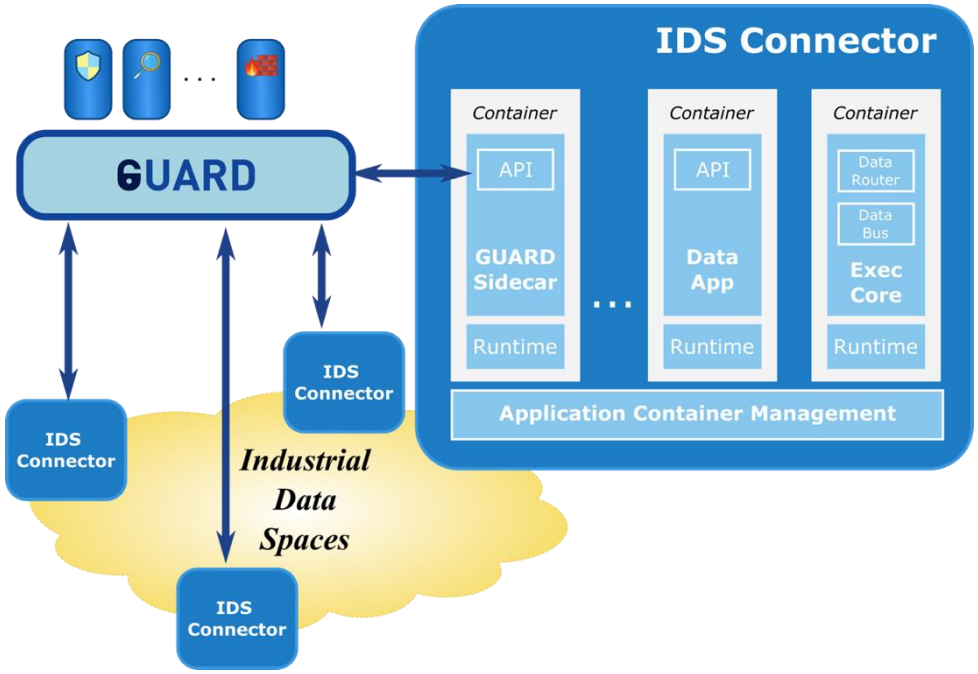


Figure 19. Relationship between GUARD and the IDS architecture.

Even if this aspect is not explicitly shown in Figure 19, there is also an alignment in configuration management. As a matter of fact, the combination of the GUARD Context Broker and the Local Cntrl & Mgmt Framework is conceived to translate descriptive configurations into concrete configuration files or specific APIs, similar to what expected from the IDS Configuration Manager.

Though the GUARD architecture is fully aligned with the main requirements for the IDS ecosystem, the platform delivered by the Project will not be an implementation of the IDS Security Architecture, because it targets more general scenarios. In particular, encryption of data is not available and algorithms for remote attestation will not be developed. In addition, the set of security agents likely will not cover all requirements identified so far by the IDS.

7 Mapping to GUARD requirements

Table 7 lists the project requirements that are currently satisfied by the GUARD architecture and maps them to corresponding logical entities. The rationale of this classification is related to the macro-areas of the framework where the requirements are satisfied: the Core Framework (with the further subdivision into Data Plane, Control Plane and Management Plane), or Local Services.

In Annex C, an extended version of the table is reported, which points out the current progress towards the implementation of all project requirements. This table includes both the requirements that have been satisfied and those that have been postponed to the following phases of the Project.

Table 7. Mapping of project requirements to the GUARD architecture.

ID	Placed in	Title	Type	Status	Architectural Element(s)	Notes
R01	Core Framework Data Plane	Centralized analysis and correlation	Design	Satisfied	Security Services	Requirement in the data plane of the core framework, accomplished by Detection and correlation functions. Data, measurements and events are collected through local digital services and made available to a set of Security Services.
R04		Heterogeneous security context	Functional	Satisfied	Security Functions	Requirement in the data plane of the core framework, accomplished by Data collection function. The Local Security Sidecar architecture includes software for collecting a broad range of data and measurements (logs, packet statistics, syscalls).
R05		Data delivery	Design	Satisfied	Kafka bus	Requirement in the data plane of the core framework, accomplished by the Kafka bus. Data from Local Security Sidecars are published to a Kafka bus and made available to multiple subscribers (control access applies to avoid leaks).
R07		Historical data for statistical analysis	Design	Satisfied	Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker where historical data are stored.

ID	Placed in	Title	Type	Status	Architectural Element(s)	Notes
R23		Integration with existing logging facilities	Design	Satisfied	Local Security Sidecars/ Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker. GUARD builds on Elastic Stack, a widely used framework for collecting security-related data.
R35		Context programmer	Design	Satisfied	Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker. The Context Broker exposes the capabilities of security agents.
R37		Program repository	Design	Satisfied	Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker. The Context Broker includes an internal repository to store eBPF programs and Logstash filters.
R12		Dynamic discovery of the service topology	Functional	(Partially) Satisfied	Context Manager/ Security Services	Requirement in the data plane of the core framework, that can be accomplished by the Context Manager or Security Services. The GUARD API #1 will be used to selectively query the set of involved digital services and infer the logical data flow between them. The location of this function within the Context Manager or a standalone Security Service is still an open issue.
R09	Core Framework Control Plane	Remediation, investigation, and mitigation actions	Functional	Satisfied	Control Policies/ Security Controller	Requirement in the control plane of the core framework, accomplished by Control and Reaction functions of Security Controller. The drools engine evaluates a set of behavioural rules and tries to satisfy them simultaneously. The Context Broker will allow access to local agents and other management properties exposed by the resource provider.
R11		Semi-automated operation	Functional	Satisfied	Control Policies Security controller	Requirement in the control plane of the core framework, accomplished by Control and Reaction functions of the Security Controller. The drools engine evaluates a set of policies and takes actions accordingly. Drools uses the Rete algorithm to satisfy

ID	Placed in	Title	Type	Status	Architectural Element(s)	Notes
						multiple rules; conditions are also taken into account and this allows to take into account the current state of the overall system. The design phase will confirm the possibility to wait for confirmation from the Security Dashboard before implementing the required actions (semi-autonomous operation).
R39		Control logic	Design	Satisfied	Security Policies	Requirement in the control plane of the core framework, accomplished by the Security Policies. Security Policies provide the rule to drive the drools engine.
R49		Notification of security events	Usability	Satisfied	Kafka bus	Requirement in the control plane of the core framework, accomplished by the Kafka Bus. A Kafka bus is used to deliver notifications and events to both the Security Controller and the GUARD Dashboard.
R03	Core Framework Management Plane	Support multiple detection and analytics services	Design	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by Detection and Analysis function. The Core Platform is able to run multiple Security Services as Kubernetes POD.
R17		Flexible insertion of detection algorithms	Functional	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by the Orchestration Framework. Kubernetes will be used to orchestrate Security Services.
R36		Scalable detection and analysis algorithms	Design	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by the Orchestration Framework. The GUARD framework uses Kubernetes to address scalability and high-availability. Developers of Security Services are then responsible to implement their algorithms accordingly; together with Security Providers they should also define management policies to drive orchestration actions.

ID	Placed in	Title	Type	Status	Architectural Element(s)	Notes
R62		Supported executable formats for detection algorithms	Implementation	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by the Orchestration Framework. The required set of executables can be run in containers.
R18	Local Services	Packet inspection	Functional	Satisfied	Security Agents	Requirement in the data plane of local services, accomplished by Security Agents. Local Security Sidecars provide alternative components for packet inspection, which fit different use cases: PacketBeat, eBPF programs, vDPI.
R19		Packet filtering	Functional	Satisfied	Security Agents	Requirement in the data plane of local services, accomplished by Security Agents. The vDPI component will provide packet filtering capabilities.
R33		Data minimization	Design	Satisfied	Security Functions	Requirement in the data plane of local services, accomplished by Pre-processing function. The GUARD architecture provides programmable components that can be used to minimize the amount of data collected to the current need. The responsibility of data to be collected is left to the Security Provider.
R48		Safe monitoring and inspection	Reliability	Satisfied	Security Agents	Requirement in the data plane of local services, accomplished by Security Agents. The eBPF framework runs programs in a protected and safe virtual machine in the kernel.
R08		Local programmability	Design	Satisfied	REST Interface	Requirement in the management plane of the local security agents, accomplished by the management (REST) Interface. The REST interface component configures local agents. In addition, eBPF programs and logstash filters can be injected at run time.
R02		Local data aggregation and fusion	Design	Satisfied	Security Functions	Requirement accomplished locally by pre-processing function. The architecture of the Local Security Sidecar includes the Logstash component to create custom processing pipelines to enrich data with general context information, aggregate data, and so on.

ID	Placed in	Title	Type	Status	Architectural Element(s)	Notes
R53		Selection of detection algorithms	Implementation	Satisfied	GUARD Dashboard	Requirement in the GUARD Dashboard. The GUARD Dashboard will list available Security Services present in the internal repository. They will be activated on demand by the Security Provider.

8 Security considerations

The GUARD platform is a flexible framework to operate several cyber-security services in an effective way, by orchestrating security capabilities available in digital services. The overall approach is to centralize detection analysis processes as much as possible, while only simple tasks are delegated locally. GUARD provides a concrete answer to the increasing demand for flexibility and programmability in the monitoring and inspection processes, as well as for dynamicity to address the unpredictability of elastic applications.

Pursuing more automation in the management of security processes, GUARD removes the need for manual intervention, hence reducing the reaction delay and the risk for human errors. Nevertheless, the GUARD platform becomes a critical entity in the overall framework, since any architectural or implementation vulnerability will represent a potential attack vector. In addition, automation is eventually driven by control and management policies defined by humans, and this should not give the false confidence that the system reacts in the correct way to every possible scenario and condition.

Some critical aspects of the overall framework can be pointed out already at this stage, and possible remediations and countermeasures can be identified. Though it is impossible to address every vulnerability at the architectural level, many issues can be solved during the implementation, deployment, or even operation phases. This Section briefly reviews the main threats and vulnerabilities related to the GUARD architecture, remarks countermeasures already identified at the design stage, and gives recommendation for implementation of discrete components and their deployment and operation in real scenarios.

Besides specific issues, deployment and operation of the Core platform should follow common practice for installing safe and secure applications. This includes network segmentation, resource isolation, physical protection, firewalling and any other security appliance that is required to safely operate IT infrastructures.

8.1 Untrusted resources

Threat description. One major goal for GUARD is to operate cyber-security services for large distributed systems, deployed over a mix of cyber-physical systems. This clearly poses the question whether such resources are safe or not, because eavesdropping, snooping, denial of service and other kinds of attacks can be easily performed internally. The GUARD platform leverages APIs to discover what security capabilities are available, but there is no technical mean to guarantee their correct and reliable implementation; in addition, only a reduced set of capabilities might be available.

Mitigation and countermeasures. The number and type of exposed security capabilities is expected to be used in risk assessment procedures to estimate the level of trustworthiness and reliability of the overall service chain. This should help select the correct set of resources to fit the requirements of different services; for instance, critical services will likely be deployed on infrastructures of trusted providers only. Clearly, the availability of APIs does not guarantee their correct implementation and the safety of internal processes; in this respect, the adoption of common cybersecurity certification schemes is expected to provide assessments about the degree of adherence of products, services and processes against specific requirements. Cybersecurity certification requires the formal evaluation of products, services and processes by an independent and accredited body against a defined set of criteria, standards, and the issuing of a certificate indicating conformance; as such cybersecurity certification plays a key role in increasing trust and security in products, services and processes.

For instance, ENISA is currently working to the definition of candidate certification schemes in line with the European Cybersecurity Act.[78] This will solve most issues about trustworthiness of resources and digital services.

8.2 Internet links

Threat description. The GUARD platform collects the security context centrally from a set of remote agents. Since it is not possible to put constraints on the physical connectivity between the centralized tool and local security agents, it must be assumed that all communication channels will be implemented over the (unsafe) Internet. This represents a major threat, because of the number of attacks that can be successfully performed in such scenario: eavesdropping, spoofing, replication, delay, alteration, denial of service. The vulnerable interfaces are those provided by API #1, namely data, control, and management interfaces. Missing data, events and measurements on the data channel will make it impossible for centralized algorithms to detect attacks against local services; on the other hand, lost control on the security agents will hinder the application of enforcement actions and the collection of finer-grained context.

Mitigation and countermeasures. The GUARD architecture will create secure virtual links between local agents and the Core platform. The technical assessment of available technologies and the preliminary development of the missing components will point out whether security mechanisms at the transport level (i.e., TLS/SSL) fulfil the project requirements (both in terms of security and usability) or an additional layer of security should be added. In the second case, either independent Virtual Private Network (VPN) connections may be established between the Local Security Sidecars and the Core platform (for instance, by using IPSec) or secure meshes may be created between all GUARD-compliant entities (e.g., by using cjdns[79]). In the first case, separate and independent encrypted channels will be created between each Local Security Sidecar and the Core platform, which corresponds to a hub-and-spoke topology. In the second case, more complex topologies can be created, and they depend on how each client connects to the mesh. Independent VPN connections avoid the need for intermediary hops and external infrastructures; however, secure meshes are usually more robust since alternative encrypted paths could be found in case of failure. Both solutions ensure confidentiality, integrity and the authenticity of the source, hence addressing most of the Internet threats. However, they do not provide an effective solution for denial of service. As a matter of fact, heart beating can only provide indication that the remote peer is no more reachable, or previous messages were not successfully delivered. In such scenarios, there is no option but to have a fallback local procedure that safeguard the service integrity. The LCP is the architectural component in charge of establishing and maintaining connectivity with the centralized Context Broker. Hence, it can trigger a local fallback procedure. Possible actions may include buffering security-related data or, as last resort in case of extended disconnection, wiping any local data and software off the digital resource, to avoid potential disclosure. The implementation of a fallback procedure in the LCP is therefore a requirement for the implementation; the procedure should be configured during operation, and a default action should be present in case of missing configuration (e.g., wipe off). The revised GUARD architecture will include definitive answers to these open issues.

[78] EU cybersecurity certification framework. Web site: https://www.enisa.europa.eu/topics/standards/certification?tab=details.

[79] An encrypted IPv6 network using public-key cryptography for address allocation and a distributed hash table for routing. Web site: https://github.com/cjdelisle/cjdns.

8.3 Integrity of local security agents

Threat description. The trustworthiness and reliability of the GUARD framework heavily relies on the integrity of local security agents. In case they do not work properly or send wrong information, the detection processes will be cheated. Beside missed detection, this could even be used to affect service operation (e.g., to stop it, to move it to a rogue infrastructure, to steer data, etc.). Since any system component is a potential target for unknown attacks, the integrity of security agents cannot be taken for granted after boot. In particular, even if the bytecode is not changed, unexpected behaviour can be driven by misleading external triggers (like in case of *SQL injection* and similar attacks).

Mitigation and countermeasures. The deployment and operation of security agents in Local Security Sidecars do not fall under the responsibility of the Security Provider. As a matter of fact, the Security Provider uses the GUARD platform to collect measurements in a programmatic way through the Core platform, but does not have management access of digital resources (at least, in the more general scenario). Resource providers are therefore responsible to guarantee the integrity of all internal components, including the security agents. The availability of a TPM allows to verify the integrity of the software during the boot procedure, but additional attestation procedures are need at run-time to ensure the software does not deviate from the correct behaviour. The presence of run-time attestation procedures and other integrity mechanisms is likely to be part of the certification process already mentioned.

8.4 Identity and key distribution

Threat description. The reliability and trustworthiness of the communication between local agents and the Core platform is ensured by encrypted communications. However, secret material should be installed in each Local Security Sidecar to correctly perform authentication and key exchange. Since the two components are deployed in different administrative domains, the usage of digital certificates is probably the best option. However, secrets might be stolen during deployment or operation, hence opening the door for a number of attacks (spoofing, eavesdropping, modification, etc.).

Mitigation and countermeasures. Storing credentials and secret material both in the volatile or persistent storage represents a possible attack vector, since there are multiple vulnerable points where this information could be snooped (e.g., image repository, storage systems, VM migration). A possible solution is to rely on the TPM for storing and managing private keys, so that they cannot be read by any external process. This issue should be further considered in the definition of the Identity Management and Access Control framework.

8.5 Vulnerable or untrusted programs

Threat description. *Programmability* represents a key value proposition of the GUARD framework, both from the scientific and business perspective. However, programmability makes the system dynamic and unpredictable, since the monitoring, inspection, and enforcement software is not fully known a priori. Dynamic programs injected at run-time are a potential vulnerability for the system, because they might come from rogue sources or contain bugs.

Mitigation and countermeasures. GUARD is fully aware of the threat represented by dynamic code. The problem has been tackled in several ways. First, the set of programmable components have been selected to ensure safe execution at run-time. For instance, eBPF programs are run into an isolated kernel virtual machine, and there are strong guarantees they do not harm the whole system (this clearly does not cover the fact that a rogue program may inspect more information than what expected). Logstash plugins should be quite limited in

the set of operation they are allowed. Second, the recommended practice is to each run security agent in a standalone container, so to limit the impact on other parts of the POD. Third, the presence of identity management and access control mechanisms between all micro-services that implement the GUARD framework should verify that only authorized entities load such programs in the Program repository. The GUARD framework currently does not implement any automatic procedure for checking such programs, so the user that loads them is responsible to carry out all verifications and analysis that are necessary to ensure the correct behaviour. Future extensions of the framework might integrate tools for automatically trigger the validation of such programs after loading.

8.6 Weak configurations

Threat description. A common problem for ICT systems is weak configuration. Indeed, easy-to-guess password, lack of proper access controls, word-readable file systems, and guest accounts are just a few examples of a long list of threats that come from wrong or shallow configurations. The same problem applies to GUARD too: even if the correct set of security agents are deployed, they should be properly configured and installed, to avoid introducing additional vulnerabilities.

Mitigation and countermeasures. The security assessment of all internal processes for a digital resource is the main responsible to ensure that safe configurations are applied. Errors during the deployment phase which leave the system in an uncomplete and unsafe state should be detectable during the certification processes. The usage of pre-packaged disk images may guarantee better confidence on the integrity of the installation, provided the base system is correctly configured and verified. In this respect, GUARD releases docker images of security sidecars, which guarantee the correct configuration of security agents.

8.7 Unexpected system behaviour

Threat description. Automation is another major objective for the GUARD framework. Currently, the best solution identified to trigger actions in response to events is the usage of a rule engine. Such tools evaluate simultaneously a set of inter-dependent policies (e.g., output from one policy may be an input from another policy); such policies usually define base assumptions, while their combination provides coverage for a far larger set of scenarios. For instance, the drools engine tries to simultaneously satisfy policies based on the Rete algorithm. The possibility to derive more complex scenarios starting from a few base ones represents a key benefit, though this brings the risk for inappropriate, wrong, or harmful response for some scenarios.

Mitigation and countermeasures. After the Security Provider has defined the set of policies to manage its security services, it cannot usually verify the combination of all possible inputs and conditions. As stated before, this could lead to wrong or inappropriate control and management actions that might be harmful for the overall service. The GUARD framework is aware of the risk to fully rely on automation for management of cyber-security aspects. For this reason, the architecture defines multiple operational modes (fully automated, supervised, manual). For critical services, the supervised mode is the recommended choice, giving humans the possibility to review and approve the decision taken by the machine. For non-critical services, full automation will provide faster response and littler likelihood for propagation of attacks.

8.8 Disclosure of internal information

Threat description. Though the definition of the GUARD APIs is an-going task and their description will only be available at a later stage, it is already clear that the properties and capabilities exposed by this interface includes

useful information for potential attackers. For instance, the description of internal software narrows the set of possible vulnerabilities and threats for each digital resource.

Mitigation and countermeasures. Access to the GUARD APIs should be restricted to certified Security Providers. Similar to what already envisioned for Resource Providers, Security Providers should also be subject to certification processes that attest their role and credibility. The Identity Management and Access Control framework should take this aspect into account, by applying access rules on the interface exposed by Local Security Sidecars that only select trusted security operators.

8.9 Ineffective isolation

Threat description. According to service mesh paradigms, most components of the GUARD framework are expected to be run in containers and Kubernetes PODs. Containers are an effective mechanism to segment resources between multiple tenants, but are not conceived with strong security guarantees in mind. As a matter of fact, they share the same kernel so basically, they provide weaker isolation than VMs. The risk is the possibility for an intruder to access memory or storage space of a container hosting some GUARD component, or to attack the common kernel.

Mitigation and countermeasures. The service mesh paradigm facilitates the deployment in cloud infrastructures, but this choice is not mandatory. The GUARD Core platform should not be deployed in public infrastructures; private infrastructures may be used, if appropriate countermeasures are taken to guarantee the isolation of the GUARD tenant from other services. This is an acceptable solution, especially if the same infrastructure is only used to run multiple instances of the Core platform for different customers. Otherwise, the deployment in a dedicated network segment is the best option to ensure the integrity of the Core platform and minimize the risk for external intrusions. Additional countermeasures should include the adoption of application gateways that check the syntax and sanity of data in all messages exchanged through the external interface, to avoid code injection, cross-site scripting, and similar attacks.

References

[1] Strategic Research and Innovation Agenda, NESSI SRIA 2017. Available at: http://www.nessi-europe.com/files/NESSI_SRIA_2017_issue_1.pdf.

[2] Cyber Physical Systems – Opportunities and challenges for software, services, cloud and data. NESSI White Paper, October 2015. [Online] Available at: http://www.nessi-europe.eu/Files/Private/NESSI_CPS_White_Paper_issue_1.pdf.

[3] 5G-PPP, 5G innovations for new business opportunities. Whitepaper, March 2017. URL: https://5g-ppp.eu/wp-content/uploads/2017/03/ 5GPPP-brochure-final-web-MWC.pdf

[4] R. Rapuzzi, M. Repetto, "Building situational awareness for network threats in fog/edge computing: Emerging paradigms beyond the security perimeter model", Future Generation Computer Systems, Volume 85, August 2018, pp. 235-249. DOI: 10.1016/j.future.2018.04.007.

[5] M. Wurzenberger, F. Skopik, G. Settanni (2018) Big Data for Cybersecurity. In: Sakr S., Zomaya A. (eds) Encyclopedia of Big Data Technologies. Springer, Cham.

[6] ECSO, European cybersecurity strategic research and innovation agenda (SRIA) for a contractual public-private partnership (cPPP). June 2017. URL: https://www.ecs-org.eu/documents/publications/ 59e615c9dd8f1.pdf

[7] OASIS Topology and Orchestration Specification for Cloud Application (TOSCA). Website: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

[8] ETSI Network Function Virtualization. Website: https://www.etsi.org/technologies-clusters/technologies/nfv.

[9] 5G empowering vertical industries. White paper by the 5G Association. [Online] Available at: https://5g-ppp.eu/wp-content/uploads/2016/02/BROCHURE_5PPP_BAT2_PL.pdf

[10] S. Hares, D. Lopez, M. Zarny, C. Jacquenet, R. Kumar, J. Jeong. Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases. IETF RFC 8192, July 2017. [Online] Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc8192.txt.pdf.

[11] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, R. Kumar. Framework for Interface to Network Security Functions. IETF RFC 8329, February 2018. [Online] Available: https://tools.ietf.org/pdf/rfc8329

[12] F. Skopik, G. Settanni, R. Fiedler. "A Problem Shared is a Problem Halved: A Survey on the Dimensions of Collective Cyber Defense through Security Information Sharing", Elsevier Computers & Security Journal, Volume 60, July 2016, pp. 154-176. Elsevier.

[13] CSIRT Italia. Incident notification form (in Italian). [Online] Available at:

https://www.csirt-ita.it/files/Modulo%20Notifica%20Incidente.pdf. Last accessed: 17th July 2019.

[14] CSIRT Italia. Web site: https://www.csirt-ita.it/nis.html (Italian only). Last accessed: 17th July 2019.

[15] Naveen Joshi. Is IoT as a Service a Thing Now? Tech article, 23rd July 2019. Available at: https://www.bbntimes.com/en/technology/is-iot-as-a-service-a-thing-now/. Last accessed: 23rd December 2019.

[16] Jason Tee. IoTaaS? Clearing the roadblocks to IoT-as-a-Service adoption. Tech article, 1st March 2017. Available at: https://www.theserverside.com/feature/IoTaaS-The-things-that-are-hindering-IoT-as-a-Service-adoption. Last accessed: 23rd December 2019.

[17] Jeff Travers. IoT as a Service: A new business model? Ericsson blog, 14th Novembre 2018. Available at: https://www.ericsson.com/en/blog/2018/11/iot-as-a-service-a-new-business-model. Last accessed: 23rd December 2019.

[18] D. Soldani, A. Manzalini. Horizon 2020 and Beyond: On the 5G Operating System for a True Digital Society. IEEE Vehicular Technology Magazine, Volume: 10, Issue: 1, March 2015. DOI: 10.1109/MVT.2014.2380581.

[19] Cloud Security Alliance. “Security Guidance for Critical Areas of Focus in Cloud Computing v4.0”, 2017. [Online] Available: https://cloudsecurityalliance.org/download/security- guidance-v4/

[20] G. Pék, L. Buttyán, B. Bencsáth, A survey of security issues in hard- ware virtualization, ACM Computing Surveys 45 (3) (2013) 40:2–40:34. doi:10.1145/2480741.2480757.

[21] Security and Privacy: From the Perspective of Software, Services, Cloud and Data. NESSI White Paper, March 2016. [Online] Available at: http://www.nessi-europe.com/Files/Private/NESSI_Security_Privacy_White_Paper_issue_1.pdf

[22] Strategic Research and Innovation Agenda, NESSI SRIA 2017. Available at: http://www.nessi-europe.com/files/NESSI_SRIA_2017_issue_1.pdf

[23] Strategic Research and Innovation Agenda, NESSI Position Paper, Version 2.0, April 2013. Available at: http://www.nessi-europe.com/files/NESSI_SRIA_Final.pdf

[24] OASIS Topology and Orchestration Specification for Cloud Application (TOSCA). Website: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca.

[25] ETSI Network Function Virtualization. Website: https://www.etsi.org/technologies-clusters/technologies/nfv

[26] FIWARE. Website: https://www-fiware.org

[27] R. Rapuzzi, M. Repetto, "Building situational awareness for network threats in fog/edge computing: Emerging paradigms beyond the security perimeter model", Future Generation Computer Systems, Volume 85, August 2018, pp. 235-249. DOI: 10.1016/j.future.2018.04.007.

[28] James Lewis, Martin Fowler. Microservices – A definition of this new architectural term. Web blog, March 2014. Available at: https://martinfowler.com/articles/microservices.html.

[29] P. Boucher, S. Nascimento, and M. Kritikos. How blockchain technology could change our lives -- In-depth Analysis. European Parliament Scientific Foresight Unit (STOA), Brussels (Belgium), February 2017. ISBN 978-92-846-0549-1, DOI: 10.2861/926645.

[30] D. Montero, M. Yannuzzi, A. L. Shaw, L. Jacquin, A. Pastor, R. Serral- Gracià, A. Lioy, F. Risso, C. Basile, R. Sassu, M. Nemirovsky, F. Ciaccia, M. Georgiades, S. Charalambides, J. Kuusijärvi, F. Bosco, Virtualized security at the network edge: a user-centric approach, IEEE Communications Magazine 53 (4) (2015) 176–186.

[31] R. Danyliw, J. Meijer, Y. Demchenko. The Incident Object Description Exchange Format. RFC 5070, Dec. 2007. [Online] Available: https://www.ietf.org/rfc/rfc5070.txt.

[32] Rich Piazza, John Wunder, and Bret Jordan, Eds. STIX Version 2.0. Part 2: STIX Objects. OASIS Committee Specification 01, 19 July 2017. [Online] Available: http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part2-stix-objects.html.

[33] Industrial Data Spaces Association. Reference Architecture Model, version 3.0, April 2019. [Online] Available: https://www.internationaldataspaces.org/wp-content/uploads/2019/03/IDS-Reference-Architecture-Model-3.0.pdf.

[34] Radware. On-Demand, Always-on, or Hybrid? Choosing an Optimal Solution for DDoS Protection. Whitepaper, 2016. [Online] Available: https://www.radware.com/WorkArea/DownloadAsset.aspx?id=6442457904.

[35] Radware. Cloud DDoS Protection Service: attack lifecycle under the hood. Technology Overview Whitepaper, 2016. [Online] Available at: https://www.radware.com/assets/0/314/6442477977/2c6454b4-403b-45b1-ac58-dc628bc210b3.pdf.

[36] F5 Silverline. Protect your business and stay online during a DDoS attack. F5 Product Datasheet, 2016. [Online] Available at: https://www.f5.com/pdf/products/silverline-ddos-datasheet.pdf

[37] S. Hares, D. Lopez, M. Zarny, C. Jacquenet, R. Kumar, J. Jeong. Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases. IETF RFC 8192, July 2017. [Online] Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc8192.txt.pdf.

[38] A. Pastor, D. Lopez, A. Shaw. Remote Attestation Procedures for Network Security Functions (NSFs), IETF Internet-Draft, v. 07, February 2019. [Online] Available: https://tools.ietf.org/pdf/draft-pastor-i2nsf-nsf-remote-attestation-07.pdf.

[39] TPM 1.2 Main Specification. URL: https://trustedcomputinggroup.org/resource/tpm-main-specification/

[40] Global Platform – Technology Document Library, Specification Library. URL: https://globalplatform.org/specs-library/.

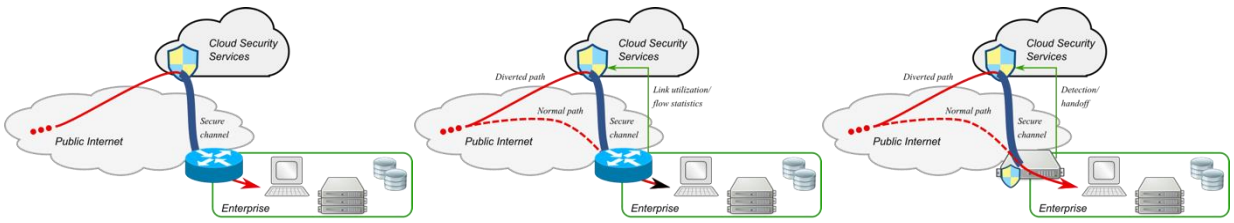
[41] N. Borhan, R. Mahmod. Platform Property Certificate for Property-based Attestation Model. International Journal of Computer Applications, Vol. 65, No.13, March 2013, pp. 28-37.

Annex A Interface to Network Security Functions (I2NSF)

Fast-evolving threat vectors and ever more complex cyber-attacks are making challenging the design and operation of effective cyber-security infrastructures, not to mention the difficulty to fulfil regulatory requirements. This problem mainly affects small and medium enterprises, which often suffer from a lack of security experts to continuously monitor network, acquire new skills, and propose immediate mitigations. Externalization of security management is emerging as a possible solution.

More and more service providers are today offering cloud-based security services, especially for network protection, by hosting network security functions according to cloud models [34],[35],[36]. Alternative solutions are possible to integrate external security services for attack detection and mitigation into the enterprise (see Figure 20):

- *Always-On*: All traffic is permanently diverted towards the cloud service, which analyses all packet streams and removes any attack. At the customer site, packets are only accepted from the cloud service (through secure channels).
- *On-demand*: Network traffic is normally delivered to the customer site, where traffic statistics are collected by the ingress router and reported to the external cloud service. In case of anomalies in the flow statistics, all traffic is diverted towards the cloud, which now behaves like in the Always-On case.
- *Hybrid*: Protection devices are deployed at the enterprise edge. They analyse and clean network traffic; in case they are not able to mitigate the attack (e.g., large volumetric attacks or unknown attacks) traffic is diverted towards the cloud service.



a) Always-on **b) On-demand** **c) Hybrid**

Figure 20. Cloud protection services.

Always-on and On-demand services better fit the need of small- and medium-sized businesses, which often do not have the budget for expensive security equipment and staff, whereas hybrid solutions represent a smooth transition path for larger enterprises that already have their internal solutions and cannot accept full externalization.

Currently, the usage of cloud-based network security services has some main drawbacks. Every vendor uses different interfaces for its cloud-based services, hence different solutions must be developed by alternative security service providers[80]. Moreover, traffic diversion usually relies on BGP mechanisms, which are very slow and undermine the possibility for quick reaction and seamless operation. Based on these considerations, the I2NSF working group aims at defining a set of software interfaces and data models for controlling and monitoring

[80] The I2NSF technical documentation uses the more generic term “service provider” to indicate the entity that provides security services. We intentionally avoid this nomenclature, which may be confusing with the main part of this document, and always indicate that entity as “security service provider” or just “security provider.”

aspects of physical and virtual NSFs, enabling clients to specify rulesets. As there are many different security vendors or open source technologies supporting different features and functions on their devices, I2NSF only focuses on flow-based NSFs that provide treatment to packets/flows, such as Intrusion Protection/Detection System (IPS/IDS), web filtering, flow filtering, deep packet inspection, or pattern matching and remediation.

A.1 Use Cases for NSFs

RFC 8192 selects several use cases where standard interfaces are required for monitoring and controlling the behaviour of NSFs [37]. These use cases show how NSFs from multiple vendors can be composed together by security providers through their management entities to automate the creation, configuration, and disposal of security services.

Network Service Providers (NSPs) can implement vNSFs as part of the progressive softwarization process of their **access networks**. The implementation of NFV enables NSP to easily create and connect vNSFs close to users, hence representing a valid alternative to physical middleboxes. The range of potential users include residential users, enterprise, mobile users, and also management systems, each with its own access clients. For instance, residential users may request parental control, content management, threat management (web contents, mail, files download). Enterprises may be interested in blocking web sites or social media applications, phishing attacks, malware, botnet, DDoS, and others. Service providers may need policies to securely and reliability deliver contents to their customers, provide isolation between multiple tenants, block malware and DDoS.

In cloud data centre, **virtual firewalls** can be used to add more filtering capacity when bandwidth utilization hits a certain threshold for a specified period of time. The need to deliver on-demand security services motivates the implementation of such appliances in software or virtual forms, rather than hardware appliances. This is also necessary in order to place the firewall instances in the right zone, close to the rack of servers where protected applications are running. Indeed, the deployment of standalone physical appliances for each customer is not technically and financially possible, whereas sharing them complicates their management. The typical requirements is therefore the ability to dynamically deploy and configure virtual firewalls within the tenant partition, according to the position and topology of the user service and its required security policies.

Firewall rules are usually expressed in terms of allowed/blocked addresses, ports, protocols, and a few other parameters. Though the same concept is applied in almost all products, the syntax for rule configuration changes from vendor to vendor, making it difficult for automation. With complex service topologies, the identification of all rules that satisfy all security requirements is usually difficult (and error-prone). Indeed, integrated firewall solutions in cloud management software often provider “security groups” or similar feature to easily identify the set of servers allowed to freely communicate. More **automation in the deployment of firewall policies** would therefore benefit from standard interfaces, which works across multiple vendors and utilize dynamic key management.

Cloud environments provide isolated execution sandboxes, even made by multiple interconnected computing units, but substantially confined to a single data centre or the set of data centres belonging to the same provider. Apart basic firewalling functions, there is currently no standard security services implemented by every provider. The interconnection of cloud services with the enterprise or other external networks must be completely managed by the users. Further, cloud users must trust cloud providers, without any **visibility on security policies** that are applied to protect the infrastructure from external and internal threats. Indeed, no standard interfaces exist to retrieve and manage security policies in a consistent way across different providers.

The **convergence of multiple network services to a common network infrastructure** cuts down CAPEX and OPEX, but also raises more management concerns due to the inter-dependency that are created among the different domains. Examples of service networks include the Void over IP (VoIP), Voice over LTE (VoLTE), Content Delivery Networks (CDNs), Internet of Things (IoT), Information-Centric Networks (ICNs). Attacks to one of these networks may easily jeopardize the operation of other networks, when proper isolation mechanisms are not correctly applied (e.g., network segmentation, network slicing). Preventing DDoS, malware and botnet attacks is not easy, especially when client devices do not confirm to strong security standards (e.g., IoT). The presence of multiple network must not foster the proliferation of bespoke security services and tools. Rather, a standardization effort is required to develop interfaces that are user case independent and technology agnostic, i.e., able to support multiple protocols and data models. This would simplify the application of common security policies across multiple environments, would facilitate the coordination of prevention, reaction, and mitigation measures, and would improve early detection through larger visibility on the execution environment.

The sensitive and critical nature of many digital services requires organization to comply with **regulatory and compliance security policies**, so to isolate various kinds of traffic as well as to be able to show logs and records in case of audit. Examples include the Payment Card Industry – Data Security Standard (PCI-DSS) or the Health Insurance Portability and Accountability Act (HIPAA). Common interfaces to multiple security tools would facilitate the tracking of applied security policies, security events, and security incidents. This could be used in case of audit as proof that traffic was isolated between specific endpoints and all required measures were applied.

A.2 Challenges to provide NSFs

Provisioning of NSF, both in physical or virtual forms, currently faces many challenges, for both service providers and customers [37]:

- **The heterogeneity of NSF in terms of services and features**. As a matter of fact, security functions may be deployed at the network perimeter, in DMZ, on single devices, both in centralized or distributed forms. Possible services range from access control (firewalling, deep-packet inspection, proxies) to detection and mitigation (IPS, IDS), authentication services, endpoint protection, monitoring and correlation (SIEM), encryption and integrity (VPN concentrators and gateways, security gateways). This complicates the definition of common models and interfaces to describe the features and configurations.
- **The heterogeneity of control and management interfaces**. Given the heterogeneity of NSFs and the lack of accepted industry standards, control and management APIs are substantially different for any vendor. This complicates automation of such services, since it is difficult to translate security policies into a different set of control commands.
- **The difficulty in monitoring and tracking the effects of security policies**. This is necessary to know that the intrusion has been stopped, as well as to access the effectiveness of mitigation and response actions. Customers want this monitoring feature in order to plan for the future using “what-if” scenarios with real data. A tight loop between policies and configurations can reduce the time to design and deploy workable security policies that deal with new threats.
- **The increasing usage of dynamic and distributed environments**. With the advent of software-defined technologies for computing and networking, more clients and applications need to dynamically update their security policies, hence to dynamically change the configuration of NSFs. The problem is not limited to the provisioning and allocation of NSFs, but also extends to the translation of security requests to configuration commands. A single NSF may also be shared by multiple tenants, especially

when it is implemented with elastic cloud-based technologies, leading to the needs of partitioning resources and avoiding conflicts.

- **Lack of characterization and exchange of capability**. Even the same kind of NSFs may have rather different capability. The knowledge of such capability (either by static description or automatic registration) is required to select and compose NSFs to create specific security services. In addition, (dynamic) negotiation of resources is necessary to size and adapt NSFs to variable workloads.
- **Lack of mechanisms to utilize external databases**. Many security functions depend on signature files, threat intelligence, or other attack description (often referred as "profiles"). There is currently no standard way to build external profiles to be shared by multiple NSF instances; indeed, the effectiveness of the protection heavily relies on the updates supplied by each vendor. As new and more complex threats arise, protection can be improved if enterprises, vendors, and service providers cooperate to develop shared profiles.
- **Lack of automation and integration with software-defined infrastructures**. Effective response to known attack would benefit from more automation. This is possible if a standard mechanism exists to signal anomalies, so that a security controller can re-configure the environment, both NSFs and network policies (routing, forwarding, filtering).
- **Lack of management tools** for dynamic key distribution to NSFs for building security associations.
- **Lack of tools and frameworks for managing high-level policies**. Customers may not have the security skills to select the right set of NSFs and define their configuration. They usually have expectations about their high-level security requirements (protect against external DDoS, guarantee availability of a group of servers, ensure confidentiality and integrity of communications with external sites, etc.). Unfortunately, there is no standard tool today for translating these high-level policies into security functions and their configuration.

A.3 I2NSF framework

I2NSF defines a framework to manage and control external NSFs. The framework assumes the presence of "security users" or "customers", which need security services provided by NSFs. The reference model for I2NSF is based on two functional layers:

- The **Capability Layer** specifies how to control and monitor NSFs. I2NSF will standardize a set of interfaces by which a customer can invoke, operate, and monitor NSFs.
- The **Service Layer** describes how client's security policies may be expressed. This is not limited to enforcement and protection actions, but also includes monitoring to build situational awareness.

Between the two layers, the I2NSF Management System translates policies into commands for NSFs. A client might also interact directly with one or more NSFs through the Capability Layer, but in this case, it will lose the abstraction brought by the Service Layer. From a business perspective, the operation of the I2NSF framework requires a new actor, the Security Service Provider. With respect to the framework users, it can be an internal or external entity. For example, it could be the internal IT security group of a large enterprise; in case of full externalization, it could be a cloud service provider or an independent entity.

Figure 21 depicts the I2NSF Reference Model and identifies the I2NSF interfaces [11]. I2NSF Users define, manage, and monitor security policies for specific network flows within an administrative domain, through the I2NSF Consumer-Facing Interface. The I2NSF NSF-Facing Interface is used to specify and monitor security rules enforced by one or more NSFs. Finally, the I2NSF Registration Interface defines the capabilities of the NSFs, which can be either statically configured or dynamically retrieved.

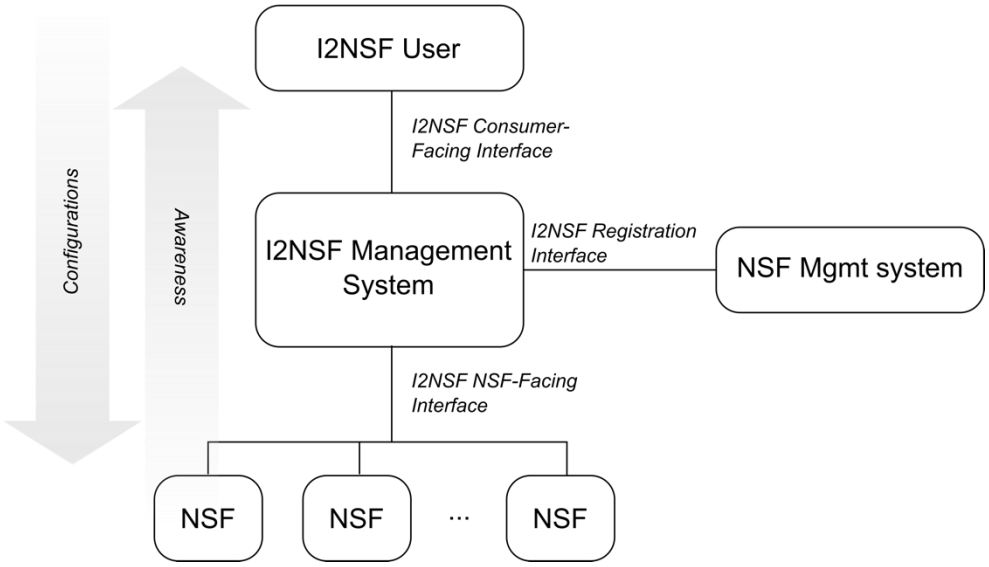


Figure 21. I2NSF Reference Model [11].

A.3.1 I2NSF Users and Consumer-Facing Interface

I2NSF Users may be humans (i.e., ICT staff in the enterprise) or their management clients (upstream application, BSS/OSS of network service provider, orchestration software, security portals, etc.). This covers different use cases, where security services are offered to large enterprises, small- or medium-sized businesses, and retail customers [37]. For example, a video-conferencing manager may dynamically inform the underlay network to allow, rate-limit, or deny flows (some of which are encrypted) based on specific fields in the packets for a certain time span. The Consumer-Facing Interface is specifically conceived to decouple security services from NSFs, allowing portability across different administrative domains.

Different levels of abstraction could be used to express security policies. In general, customers may not have the skills to define configurations and flow-level policies. For this reason, this interface is expected to model expectations, goals, or intents of the functionality desired by customers. Customers may give indicate which types of destinations are (or are not) allowed for certain users (e.g., enable Internet access for authenticated users, streaming media applications are prohibited on the corporate network during business hours, and so on).

Despite of their simplicity, some user policies may need multiple NSFs located in different places to achieve the desired behaviour. Clearly, the specification of user policies with very similar models than flow policies simplifies the translation.

A.3.2 NSFs and NSF-Facing Interface

Network Security Functions are physical or virtual instances of monitoring and enforcing appliances. They are expected to implement two sub-interfaces:

- *operational and administrative interface*, which is used to change the status and configuration of the NSF, in order to dynamically change their behaviour;
- *monitoring interface*, which can be query- or report-based.

The number of different types and implementations of NSFs is very large, so it may happen that the set of available NSFs are not able to fulfil the customer's request. The I2NSF system must therefore support dynamic discovery of capability, as well as query mechanisms, so that the management system can select the functions

that satisfy the customer requirements. Dynamic negotiation will also allow to tune the requirements based on available services/features. The outcome of the negotiation would feed the I2NSF Management System, which would than dynamically allocate appropriate NSFs and configure the set of security services that meet the requirements of the user. RFC 8329 [11] has already identified preliminary characteristics, categories and fields for discovery and registration of capabilities.

A.3.3 I2NSF Security Controller

At the heart of the I2NSF Management Framework, a Security Controller mediates between user's policies and NSF. The Security Controller receives user's policies (in terms of requirements, intents, goals) from customers and translates them into commands that NSFs can understand and execute. The NSF management includes six fundamental operations: create, read, write, delete, start, and stop. They cover both the management of imperative policy rules and management of the NSFs. The Security Controller also gathers monitoring reports (e.g., statistics) from the NSFs and passes back them to the customer. The Security Controller does not only collect individual service information but can also aggregate data suitable for tasks like infrastructure security assessment.

The I2NSF framework assumes the definition of flow-based **policy rules**. Flow level policies may be defined as imperative Event-Condition-Action (ECA) constructs, that define how the system react to given events and conditions. Flow-based NSFs are based on stateful processing, i.e., they consider both the packet content (headers and payload) and context (session state). Currently, the main focus is only on imperative paradigms for policy rules. Even though security functions come in a variety of form factors and have different features, ECA rules would support a wide range of possible behaviours for flow-based NSFs.

Flow-based policy rules should match quite well the NSF-Facing interface, since most types of events, conditions, and actions are common to many security functions, even if different information and data models are used by different vendors. For example, events can include current date/time, notification of state change, user logon/logoff. Conditions may be related to the value of some fields in the packet header (source/destination addresses, ports, protocol type, flags, etc.), packet size, direction of the traffic, geo-location, connection status. Actions may include processing on incoming/outgoing packets (pass, drop, rate limiting, mirroring, encapsulation, forwarding, transformation), loading of functional profiles (IPS profile, signature file, virus definitions, etc.).

On the other hand, the definition of user policies follows more a goal/intent approach, which in general does not reflects the operations of NSFs. In this case, the translation is far more challenging. For this reason, the I2NSF framework is currently limiting the scope to user policies that can be modelled as closely as possible to flow security rules used by individual NSFs. A I2NSF user policy should therefore adopt the ECA structure, but with more user-oriented expression for events, packet context and context, and enforcement actions. In this case, an event could be "user has authenticated", a condition may be "user identifier", and action something like "establish encrypted channel."

Management of NSFs include provisioning and life-cycle operations (start, stop, scale, update), configuration of operational attributes (network addresses, endpoints, number of internal threads, log files, etc.), signalling, and setup of policy rules (this latter already discussed in the previous paragraphs). NSFs could be clustered together and managed by a common system; in this case, the Security Controller interacts with the NSF Manager directly (Figure 22).

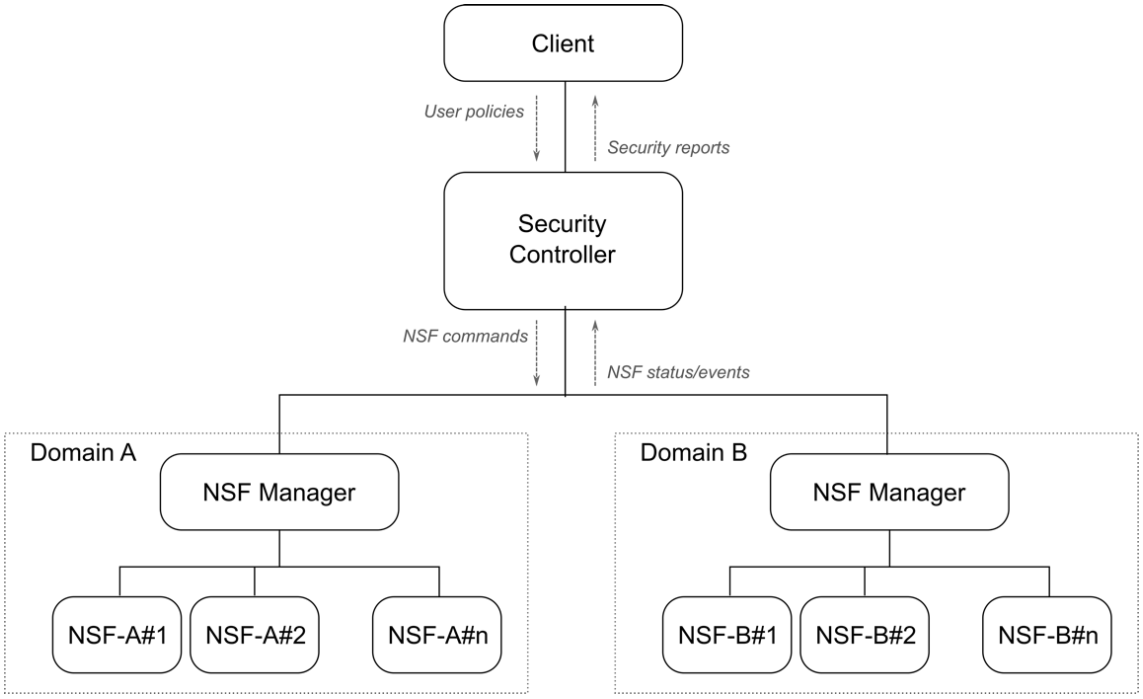


Figure 22. I2NSF management architecture.

The presence of an NSF Manager is a typical case for vNSFs provided by IaaS. In this case, it can dynamically create NSFs for each user and control its whole life-cycle, from instantiation to termination. An NSF Manager can also be used with physical and SaaS instances, which statically deployed by the Security Provider, but could be chained or composed at run-time to create tailored security services. In this case, the NSF Manager hides the number and heterogeneity of the underlying NSFs to the Security Controller and becomes responsible to translate policy rules and ensure their consistency across the set of NSFs.

A.4 Security and trust

The externalization of security services improves the flexibility and resilience, but also brings additional threats due to the lack of security perimeter, multi-tenancy, and third-party infrastructures. The most relevant threats associated with an NSF platform are [11]:

- the control of NSFs by **unauthorized users**, who may change the policy rules so to bypass the intended behaviour or to cause DoS;
- **misuse by authorized users**, who may bypass isolation mechanisms to
    - alter the configurations of other users in the underlying NSFs;
    - take control of an entire NSF or provider platform, by exploiting vulnerabilities in the application or control protocol;
- **loss of integrity** of the NSF software or the underlying platform, which undermines the correct operation and privacy requirements by all users and can be due to:
    - physical attacks to the platform, by console or serial lines, to modify the behaviour of the hardware/software with the same level of privileges as the provider;
    - malicious service providers that modify the software (Operating System or NSF implementation), this represents the most critical case since service providers are the ultimate responsible to detect attacks to the infrastructure.

A.4.1 Secure communication channels

The recommended security mechanisms include:

- authentication, authorization, accounting, and auditing for all users and applications that access the I2NSF environment;
- attestation of NSFs by service providers or third parties, so to detect changes to the I2NSF environment.

According to the reference model shown in Figure 21, the primary vulnerable point for an I2NSF platform is the Customer-Facing interface, which is publicly exposed to all users. This interface may be subject to spoofing, eavesdropping, replication, alteration, privilege escalation, misuse, DoS. A mutual authentication is therefore required between users and the Security Controller, as well as a trusted connection upon successful authentication. The basic requirements for a trusted connection are not limited to confidentiality and integrity, but also entail strong authentication of the peers and attestation of their integrity.

Based on the overall framework scope, the NSFs are not expected to be directly attached to the Security Controller, but to be distributed across the network. There can be a common network for the NSF-Facing interface and data traffic processed by the NSFs or, better, a dedicated network for management purposes only. In either case, packet loss could happen due to failure, congestion, attacks, or other reasons. Therefore, the transport mechanism for the management interface must be secure. The I2NSF framework does not require reliable transport mechanisms; rather, reliability should be directly implemented in the interface protocol by introducing explicit acknowledgement of messages into the communication flow. This would achieve better latency in the delivery of control messages.

Indeed, the selection of security mechanisms for the NSF-Facing interface depends on the specific network segment between the Security Controller and NSFs. When the I2NSF platform is built in a single administrative domain (e.g., it is implemented by a Network Service Provider in its own infrastructure), it can be safely assumed the substantial isolation and protection of the communication environment. In this case, some requirements on authentication and trustworthiness could be partially loosened. Instead, in open environments where the NSFs functions are hosted in multiple external domains, more restrictive security control should be placed over the same interface. The same procedure as for the Customer-Facing interface could be used to establish a trusted connection.

A.4.2 Remote attestation

While it is true that any ICT environment is vulnerable to get compromised by malicious users with physical access, the application of attestation mechanisms improves the trustworthiness of the system by raising the degree of control and physical activity that are needed to perform untraceable modification of the environment.

Within the I2NSF framework, remote attestation is an inescapable requirement to properly address the cooperation between different actors in disjoint administrative domains: users (I2NSF customers), security providers (I2NSF service providers), and security functions (I2NSF Network Security Functions) [38].

From the user perspective, the establishment of trust in a I2NSF platform requires two steps: i) verify the authenticity of the Security Controller, and ii) get proof that NSFs and policy rules are compliant with their security choices.

To set up security services through a NSF platform, the user's client connects and authenticates to the Security Controller. However, before initiating authentication and authorization procedures (and likely accounting and auditing), the client wants to attest that it is connected to a genuine Security Controller. Two properties characterize the genuineness of the Controller: its identity and its integrity. Afterwards, the client can authentication, start a trusted connection with the Controller, and ask for some security services. Before any traffic is actually redirected through the set of NSFs, the client must be sure that it will be processed according to the user policies. The attestation of the selected NSFs and the applied policies must happen at initialization and may be optionally repeated at run-time to detect any following loss of integrity. The attestation of a NSF platform include multiple elements, some of which are present in all possible implementations: firmware, OS, NSF software, in a virtualized environment, the virtualization system (hypervisors, host OS, container frameworks, …).

The attestation of NSF platforms can be enough in case of hardware implementations connected by physical links, which represents a static configuration that can only be modified by the infrastructure provider. However, the dynamicity brought by software-defined networking paradigms, like SDN, NFV and SFC raises additional concerns about the correctness of the network topology (which could bypass the security functions), hence demanding for attestation of the network configuration as well.

The enabling technology for remote attestation is trusted computing. The underlying concept is the presence of hardware which serves as a trust anchor to start a chain of attestations (i.e., *chain of trust*). Currently there are two main trends in this area, driven by two standardization bodies: The Trusted Computing Group (TCP) and the Global Platform (GP). The TCP define the Trusted Platform Module (TPM) [39], a collection of cryptographic functions often implemented by a dedicated hardware chip outside the main processor. The GP specify the Trusted Execution Environment (TEE) [40], this is a secure isolated environment on the same System-on-Chip (SoC). The I2NSF framework is currently considering the TPM solution. The TPM defines an architecture with the following capabilities: performing public key operations, computing hash functions, key management and generation, secret storage of keys and other secret data, random number generation, integrity measurements, attestation… It uses a transitive mechanism: if a user trusts the first execution step (i.e., the hardware *root of trust*, RoT), and each step correctly verifies the integrity of the next executable firmware/software, then the user can trust the whole system. In more concrete terms, every boot stage (BIOS, Bootloader, Security Controller) measures the integrity of the following piece of software and stores it inside a log that reflects the different boot stages, which is then signed with the private key of the RoT. In a TPM, measurements of the software stack are concatenated, so that an unlimited number of measures can be stored in a single on-board Platform Configuration Register (PCR).

The I2NSF also envisions the possibility of a trusted boot for safely storing secrets. In this case, the PCR values could be used as an identity for decrypting confidential information on the server (as encryption keys or sensitive configuration). The basic idea is to encrypt such data by the root of trust, and then subordinate decryption to a particular platform status (i.e., a set of PCR values). This ensures that only trusted software can access keys and other sensitive materials for authentication (for instance, the private keys used to create the secure channel with the client: TLS, IPSec, etc.).

As final consideration, we argue that remote attestation entails the knowledge of the exact firmware/software configuration of both the Security Controller and NSF platform, which might help identify vulnerabilities. The

client may delegate a Trusted Third Party (TTP) to compute the integrity of the Controller, hence avoiding the implementation of the whole attestation mechanisms.

A.4.3 Security Controller attestation

The attestation of the Security Controller is a required step for establishing a trusted communication channel. A trusted channel is more secure than an encrypted channel. It entails stronger verification of the identity of the Security Controller, by including its public key or certificate in the measurements of the chain of trust. This prevents spoofing and man-in-the-middle attacks, since a malicious user cannot push its certificate in the chain of measurement of the genuine platform. However, there is still the case where the confidentiality of the private key is lost. This can be solved by a Platform Property Certificate, which binds system properties instead of binary data [41]. Such certificate could connect the platform identity with the Attestation Identity Key (AIK) public key, so hindering the usage of the stolen Security Controller's key on a different platform (the attacker cannot create a quote with the AIK of the other platform).

The procedure to establish a secure connection with the Security Controller will include therefore the following steps:

1. The client begins the handshake with the Security Controller.
2. The client receives the certificate of the Security Controller.
3. The client asks the Security Controller to generate an integrity report.
4. The Security Controller retrieves the measurements and asks the TPM to sign the PCRs with the AIK. The signature provides evidence that the measurements belong to the Security Controller.
5. The client checks the integrity report, by verifying the quote and the certificate associated to the AIK. The client also checks that the digest of the certificate received at step 2 is present among measurements. These operations can be delegated to a TTP, which may be useful in case of lightweight clients.
6. If the remote attestation is positive, the client continues the handshake and establishes the trusted channel; otherwise, the connection is closed.

A.4.4 NSF platform attestation

The attestation of the NSF platform checks the integrity of the NSFs and their correct behaviour. Platform attestation does not cover the mechanism used to translate user policies into policy rules and NSF configurations, which would be technically too complex (if not unfeasible). The trustworthiness of this process indeed relies on the integrity and attestation of the Security Controller, which the client selects as the reliable intermediary for managing its security services.

The attestation of NSFs can therefore include the integrity of the software, the hosting environment (a physical device, a virtualization platform), and the running configuration. The quotes can be compared with the status information maintained by the trusted Security Controller, with the same procedure that could also be used to SDN (see Section A.4.5).

A.4.5 Topology attestation

Two methods exist to attest the deployment of a topology of a software-defined network topology. The first one is based on typical SDN operation, like in case of OpenFlow, and the second targets the application of SFC.

In the first case, each network node provides measures on its forwarding configuration, which are aggregated and signed by a TPM module. The attestation is then compared with the configuration retrieved from an SDN controller, to verify the correct behaviour of the network.

In the second case, the Proof of Transit can be applied. This mechanism injects specific packets requesting POT and verifies at the egress point of the service path that a correct topology has been enforced, by means of the cryptographic proof provided by POT.

Annex B Industrial Data Spaces

Modern business models often rely on the cooperation from multiple players, given the increasing (technical and financial) difficulty for any company alone to implement ever more complex value chain. This fosters the creation of thematic ecosystems, where partners from different domains share resources and processes so to create end-to-end innovative product and services in a flexible and dynamic way.

By recognizing that data has a value, the Industrial Data Spaces (IDS) initiatives is pursuing one such ecosystem, where data can be traded in the market like a commodity and exchanged by partners. The main objective for IDS is the definition of the business, procedural, and technical framework to create such ecosystem. Currently, the Industrial Data Spaces Association (IDSA) has already released several versions of its Reference Architecture Model [33], which provides the main guidelines for both standardization and industrial developments.

B.1 Concept and use cases

According to the definition given by the Reference Architecture Model [33],

> *A data-driven business ecosystem is an ecosystem in which data is the strategic resource used by the members to jointly create innovative value offerings.*

This basically requires technical and procedural means to share and jointly maintain data, which can be used to create end-to-end processes by pipelining such data through multiple processing stages. The overall goal of IDS is therefore the creation of a data-driven ecosystem with the following strategic requirements:

- *trust*: every participant in the ecosystem (natural person, company, software) must be evaluated and certified before being granted access;
- *security and data sovereignty*: every transaction in the ecosystem must happen in a safe and trustworthy manner, allowing data owners the definition of access and usage rules;
- *shared data*: the ecosystem relies on data brought and shared by each participant, without the need for a centralized repository;
- *interoperability*: multiple implementations from different vendors should coexist and interwork together;
- *processing apps*: the framework envisions the injection of software applications (*apps*) at run-time, which are used for data processing, format alignment, exchange protocols, etc.;
- *data markets*: the brokerage of data includes the delivery of apps and the creation of processing pipelines from data producers to data consumers, including clearing mechanisms and billing functions.

Example of potential user cases include the following:

- Material Sciences: sharing of material information along the entire product life-cycle;
- Energy: shared use of process data for predictive asset maintenance;
- Manufacturing and Logistics: exchange of master and event data along the entire supply chain;
- Health Care: anonymized, shared data pool for better drug development;
- Smart Cities: shared use of data for end-to-end consumer services.

B.2 Challenges to create a data-driven ecosystem

Data is becoming a strategic resource in modern value-added products and services. It is necessary to control the pipelining of goods and services through multiple organizations, but today it is undisputable that data is an economic good itself. However, data is an intangible resource, like intellectual properties, and it differs from

tangible goods with regard to a number of properties, among which the fact that data is non-rival is considered the most important one. Therefore, it can be consumed over and over again without the fear of depletion of supply. From an economical perspective, this means that data has a cost, which does not depend on the number of users, and the revenue streams generated by data increase linearly with the number of users.

However, if collaborative models should be the norm because of the difficulty to create fully in-house processes and solutions, technical and procedural regulations on ownership, access, usage, and disposal are urgently needed to support data-driven ecosystems. The value data contributes to the development of innovative products and services can vary. Accordingly, the need for protection of data is not the same across all data types and data categories. Because of these differences and distinctions made with regard to data, a generally accepted understanding of the value of data has not been established so far.

Data sovereignty is defined by the IDS as the balance between the need for preserving and safeguard ownership of data with the need to take advantage of its value by trading it. This requires a framework that ensures the presence enforcement capabilities in all infrastructures, certifies their integrity, and verifies that the rules are observed when sharing and transferring data. The starting point is the identification of sharing models that are possible in the ecosystem. The IDS identifies two main directions:

- *vertical cooperation* between companies that are integrating complementary or parallel resources to accomplish a specific production of service. Typical examples may be an assembly process in manufacturing industry carried out by multiple partners, or mechanical products that include alternative work made by different workshops. For vertical cooperation, data is usually *exchanged* to support, enable or optimize value chains and supply chains.
- *horizontal collaboration* between companies involved in a value or supply chain, as happens for the transformation of raw materials into final goods and products. In horizontal pipelines, data are often *shared* to boost new services and products (e.g., predicting maintenance in manufacturing, digital twins).

Clearly, the two models are often combined together in real business processes, as pictorially shown in Figure 23.

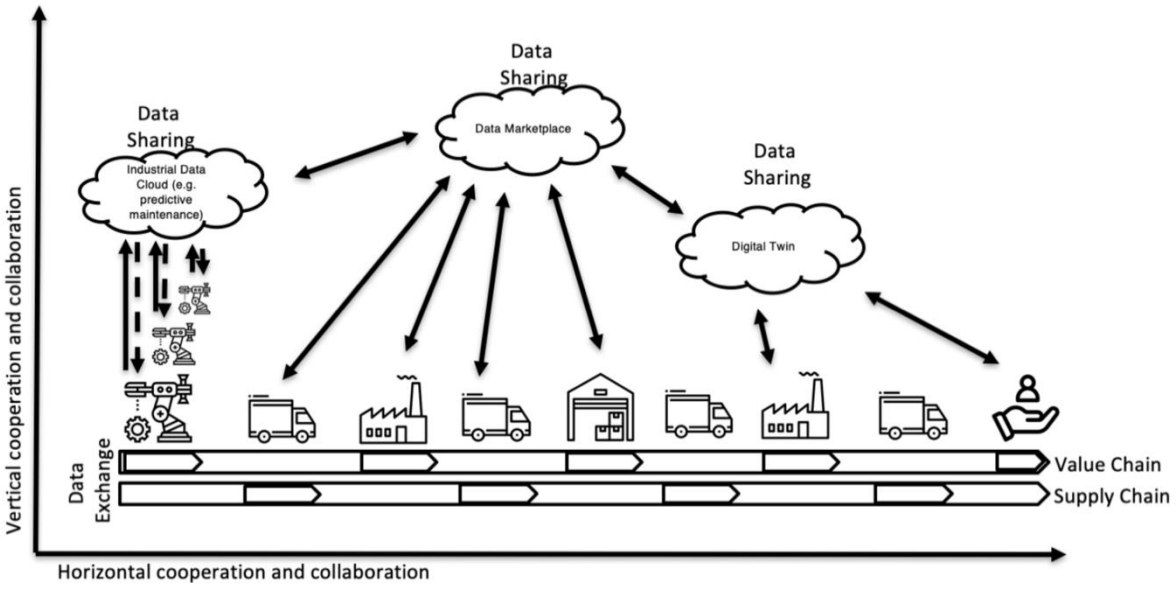


Figure 23. Different models for using and sharing data in horizonal and vertical cooperation. Picture taken from the IDS Reference Architecture Model [33].

The need to exchange data is not recent. Many different data exchange scenarios have emerged over time with the proliferation of Electronic Data Interchange (EDI) in the 1980s, leading to the development of technical standards. However, the real challenge in the modern economy is to materialize "terms and conditions" (such as time to live, forwarding rights, pricing information etc.) that regulate the exchange of data into properties linked to the data itself. This is what the IDS means for *data sovereignty*.

There are several aspects that should be considered to implement data sovereignty:

- **Trust**: both participants in the ecosystem and their infrastructures should behave according to common principle and rules. This implies the need for unambiguous identification of each entity, as well as the its certification against the common rules.
- **Security and data sovereignty**: usage policies and enforcement mechanisms should be in place to avoid unauthorized access, accidental disclosure, eavesdropping, alteration, abuse, and other common threats. This is necessary to avoid that external entities could steal data, but also that internal entities could use or propagate the data beyond the contractual terms. Beyond plain security services like confidentiality, integrity, undeniability, and access control, the IDS scope also includes digital right management, a very challenging issue for multi-domain systems.
- **Data ecosystem**: the management of heterogeneous data in a common ecosystem requires the ability to understand their composition, format, and location. For this reason, description of data sources, brokering a vocabulary are required to create the necessary interoperability between producers and consumers.
- **Interoperability**: the ecosystem relies on standard interfaces for data exchange and common processes. That means that the format of data and the communication protocol are not enough to ensure interoperability, since common management framework should be used by all entities to guarantee the convergence of access control and enforcement processes.
- **Value adding apps**: one challenging ambition for IDS is the possibility to transform the data within the ecosystem. Data transformation may be needed to filter records, to aggregate fields, to compute statistics, or to derive new information from raw measurements. Similar to what already available with advanced cloud computing paradigms, like serverless and lambda functions, the IDS envision the possibility to create applications, publish them in the ecosystem, download and install them at different location to process and transform data on demand. Clearly, beyond technical difficulties to implement it, this paradigm brings a number of security challenges concerning the trustworthiness and integrity of the software, and the risks to run it into different infrastructures.
- **Data markets**: the ultimate goal of data sharing is to capitalize its value. Hence, there is the need for pricing models, clearing and billing centres, as well as the definition of usage restrictions, data governance policies and all legal aspects that should apply.[81]

B.3 Business ecosystem and roles

There are multiple entities that take part in the IDS ecosystem. The Reference Architecture Model identifies specific patterns of interaction between these roles, which are at the base of the definition of innovative business models and digital data-driven services. Figure 24 shows the main roles and their interactions.

[81] It is worth noting that the IDS mainly focuses on industrial and commercial data that does not include personal and sensitive information. Widening the scope to the latter is of course important to cope with many interesting use cases, but the Reference Architectural Model does not cover this issue yet.

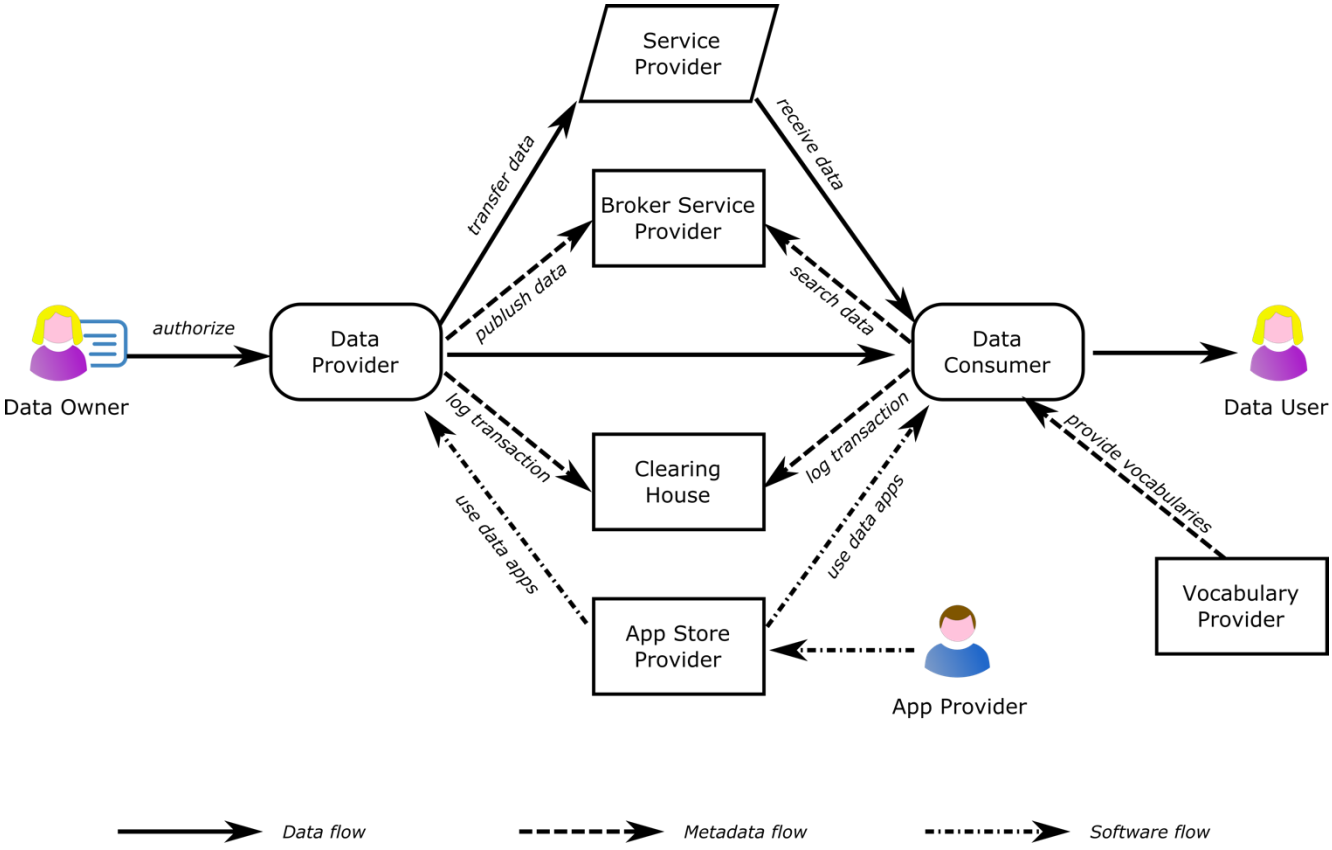


Figure 24. Role and interactions in the IDS. Source: [33].

The **Data Owner** is the legal or natural person creating data and/or executing control over it. The Reference Architecture Model does not cover full legal details of this role, and does not bind it to the GDPR regulation. The Data Owner defines usage policies and provides access to its data. It is also responsible to define the payment model. Both things should usually be documented by a contract, which might be in paper or electronic form. Usually, the Data Owner also acts as Data Provider, but the two roles are not necessarily paired.

The **Data Provider** makes data available technically. It also generates metadata that describe the data. The scope of metadata is to facilitate the identification and content of the data; they are usually published to a Broker Service for make them discoverable by interested consumers. The Data Provider may log the details of the transactions at a Clearing House, for billing or conflict-resolution purposes.

The **Data Consumer** is the counterpart of the Data Provider. It gets the data from the provider, maybe after discovering them through the brokerage service. If the information about the location of the data is already available to the Data Consumer (e.g., by an external channel), it does not need to contact the Broker Service Provider, and can directly fetch the data (and associated metadata) from the Data Provider.

The **Data User** is the legal entity that has the legal right to use the data, according to the usage policy established by the Data Owner. Secularly to what happens for the Data Owner, the role of Data User is often combined with that of Data Consumer. In many cases, the Data User is a natural person and the Data Consumer is the application used to retrieve and manipulate the data.

The **Broker Service Provider** is an intermediary that stores and manages information about the data sources available in the IDS ecosystem. There may be multiple entities implementing this role in an IDS ecosystem, and

this role could be cumulated to Clearing House or Identity Provider. The activity of the Broker Service Provider is mostly limited to receiving, storing, and publishing metadata. It is somehow similar to the role of Domain Name System in networking architectures, or a *rendez-vous* point for multimedia applications.

The **Clearing House** provides clearing and settlement services for data exchange and financial transactions. It receives logging information about each transaction, which could then be used for billing and for resolving conflicts. The picture shows a single entity, but distributed paradigms like blockchain technology could also be used for this function.

An **App Provider** develops data applications that can be run by Data Producers and Data Consumers. Technically speaking, such applications must be compliant with the IDS architecture, otherwise they cannot be used. Certification of the software integrity and trustworthiness is also expected to be present, in order to avoid the injection of malware throughout the system.

Applications developed by App Providers are pushed to **App Store Providers**. They play a similar role to Broker Service Providers, because they are the reference point for other entities to discover and download the software. To this aim, the plain software is enriched with metadata that describes its origin, properties, and capabilities.

The **Vocabulary Provider** offers ontologies, reference data models, and metadata that can be used to annotate and describe datasets. They are based on the Information Model of the IDS, which is the main reference for describing the data sets and their sources. This does not preclude the usage of additional domain-specific vocabularies.

Service Providers operate IT infrastructures on behalf of their customers. They can be used when a participant to the IDS ecosystem does not own the necessary technical infrastructure to deploy Data Provider and/or Data Consumer function. Beyond that, Service Providers could also offer additional data processing services for analysis, integration, cleansing, or semantic enrichment; in this case they could be seen both as Data Provider and Data Consumer at the same time. It is worth noting that unlike services provided by a Service Provider, Data Apps can be installed in IDS-compliant infrastructures of Data Consumers and Data Providers for implementing additional data processing functionalities. Hence, the Service Provider should be meant mainly in terms of supplier of computing and storage resources.

All the roles depicted as geometric shapes are expected to implement the IDS components. In addition to the roles depicted in Figure 24, there are other roles envisioned by the Reference Architecture Model. **Identity Providers** create, maintain, manage, monitor and validate identity information of participants in the IDS. They should include a Certification Authority for issuing digital x.509 certificates, and a Dynamic Attribute Provisioning Service for collecting and distributing attributes used for access control. **Software Providers** implement the functionality required by the IDS. This does not include Data Apps; the software is deliverable through usual distribution channels (e.g., repositories, git hub). The negotiation of supply contracts does not fall under the scope of the IDS. Finally, some governance bodies are required to regulate and certificate the participants in the ecosystem. These are mainly **Certification Bodies**, which check the identity and properties of each participant before granting access to the ecosystem, and the **International Data Spaces Association** (IDSA), which promotes the continuous development of the Reference Architecture Model and the certification process.

B.4 Architecture

The IDS architecture is conceived as a technical mean to expose, exchange and share data in regulated ecosystem. In this respect, it defines a specific component (*IDS Connector*) that links different enterprise applications, private and public cloud platforms, and connected devices through policies and mechanisms for secure data exchange and trusted data sharing (see Figure 25).

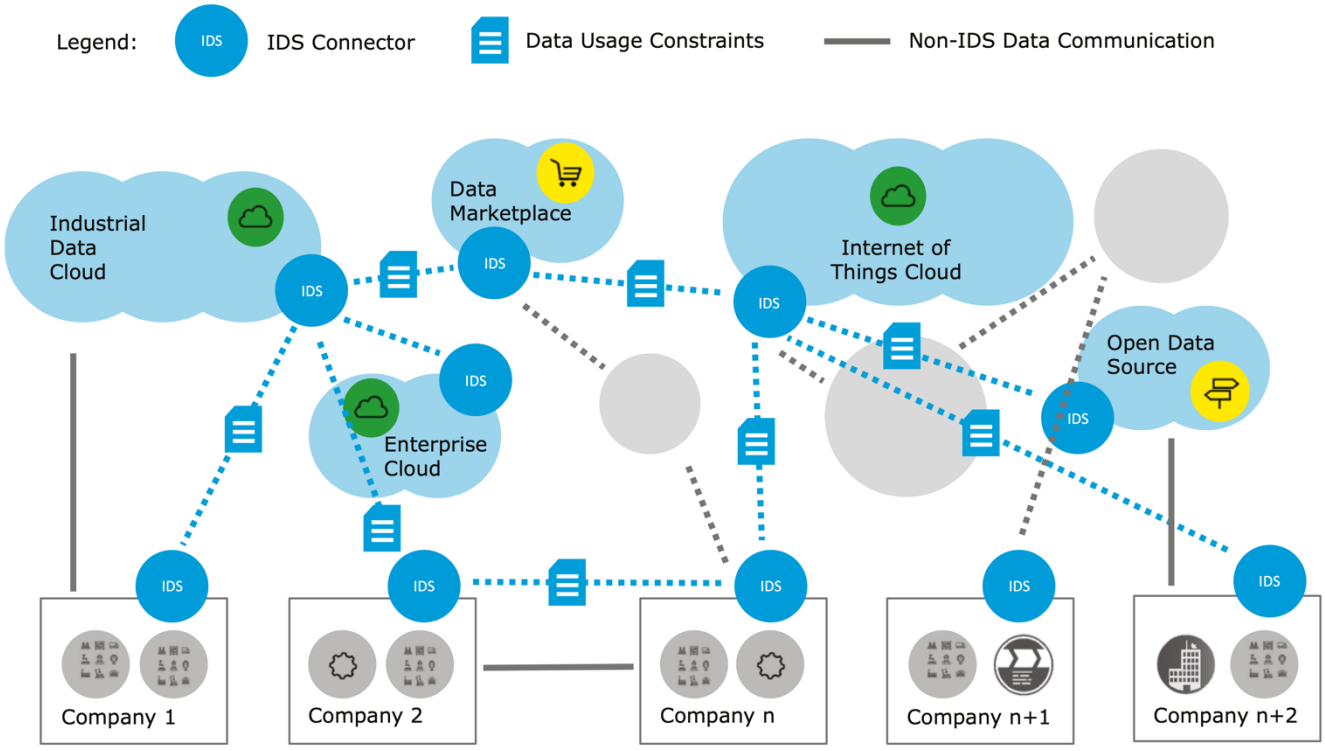


Figure 25.The IDS exchanges and shares data available from multiple domains through the IDS Connector. . Picture taken from the IDS Reference Architecture Model [33].

The IDS Connector therefore works like a high-level proxy that exposes internal resources and filters incoming requests based on data usage constraints defined by Data Owners (which, remarkably, do not necessary coincide with the operator of the Connector itself), as pictorially shown in Figure 26. Each Connector encompasses one or more Data Endpoints. Special Connectors act as Broker and only contain metadata registered by Data Producers. The Reference Architecture Model does not define how the IDS Connector interacts with every specific infrastructure; this part is strictly dependent on the set of technologies in place, so it is left to Software Providers. Hence, there may be different types of implementations of the Connector, based on different technologies and depending on what specific functionality is required for the purpose of that Connector.

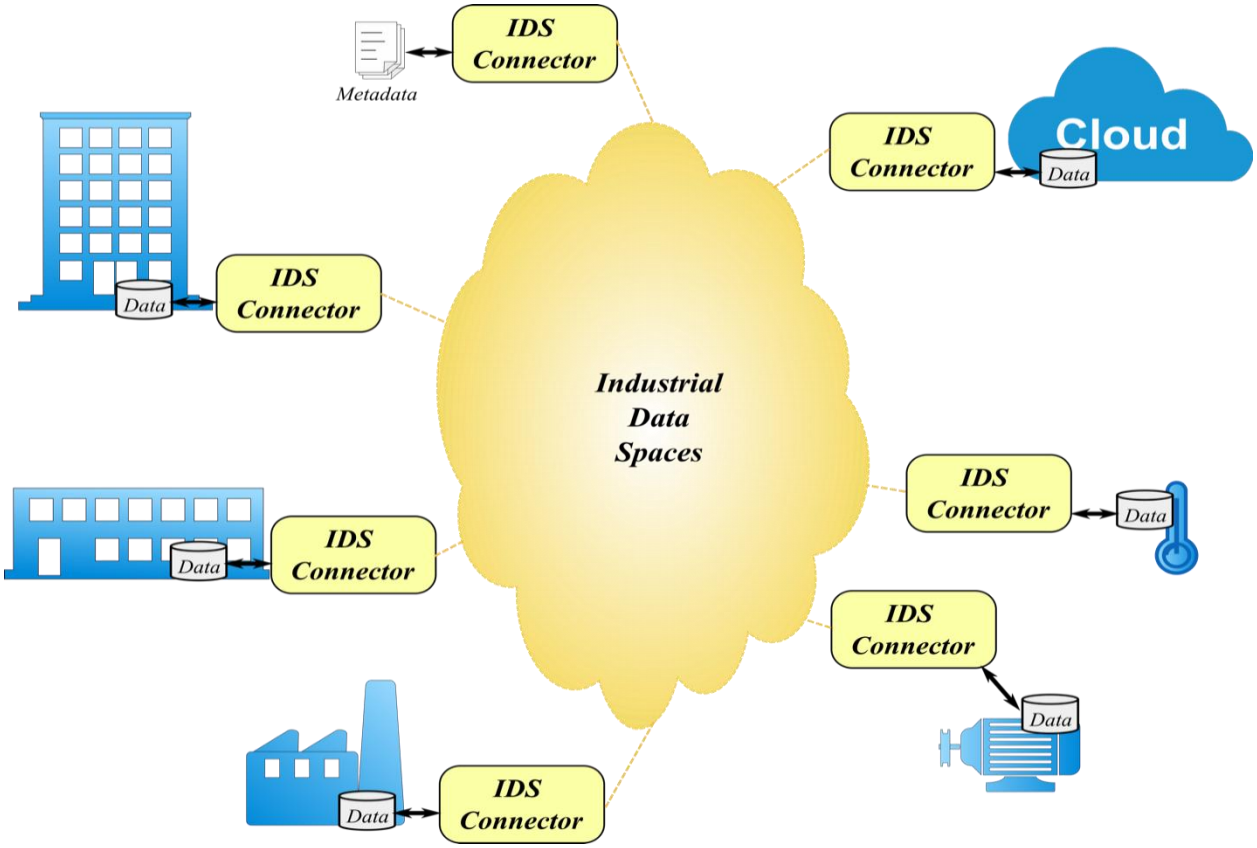


Figure 26. Overview of the Industrial Data Spaces ecosystem.

The Reference Architecture Model distinguishes between External and Internal Connectors. An External Connector is responsible of the data exchange operations described so far, and provides data via the Data Endpoints. The set of External Connectors constitutes the International Data Spaces. An External Connector is a kind of public service, so it is expected to be operated in the DMZ network segment of an organization. From the DMZ, the External Connector can reach internal resources in a controlled environment. There is not any strict requirement to operate an External Connector within the enterprise; it could also be deployed in an external infrastructure, like cloud services. An Internal Connector is operated within the company network. It may represent the peer to facilitate access to internal data from the External Connector. Internal Connectors are useful for any data pre-processing (filtering, anonymization, and analysis) that should be performed as close to the data source as possible; only data intended for being made available to other participants should be forwarded to External Connectors. Both kind of connectors may use the same architecture. Other special kinds of Connectors are the Broker Service and the App Store. The former manages data sources available in the IDS ecosystem by publishing and maintaining the associated metadata; the latter is a repository with all Data Apps that can be downloaded and on-boarded in each controller.

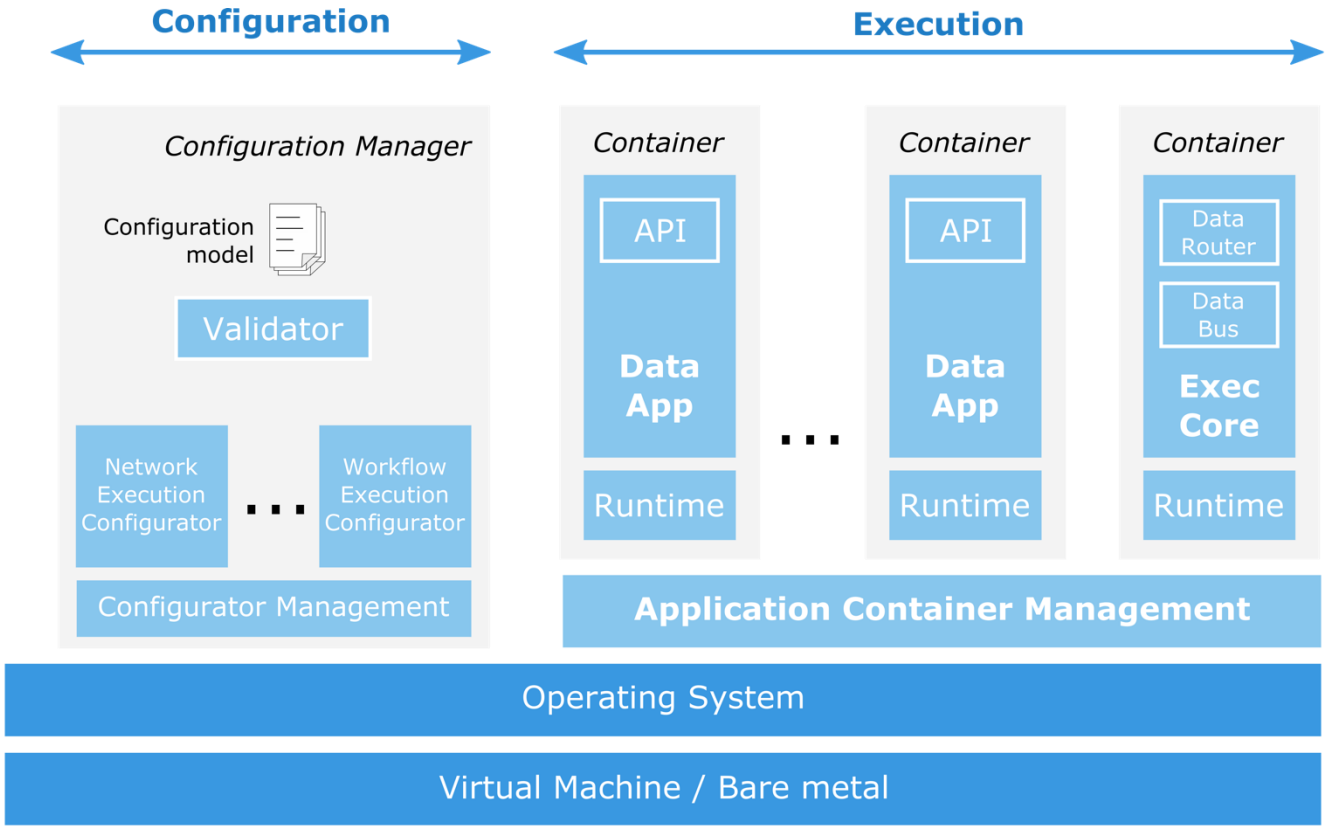


Figure 27. IDS Connector architecture. Source: IDS Reference Architecture Model [33].

The logical architecture for an IDS Connector is shown in Figure 27. IDS Connector architecture. Source: IDS Reference Architecture Model [33].. The structure proposed by the Reference Architecture Model is not strictly mandatory, and a concrete installation can modify and add additional components; however, the format of Data Apps and interfaces must adhere to the specification to ensure interoperability. A Connector may be implemented both on bare metal or in a virtualized environment (i.e., VM). Above the Operating System, the Connector is expected to provide isolated and secure environments for multiple data services, based on container management technology (*Application Container Management*).

Each data service is a standalone application that offers API to store, access, or process data. Data services are bundled as container images (*Data Apps*), which can be on-boarded and orchestrated by the Application Container Management system. They are downloaded from the App Store, and must be certified in order to avoid introducing vulnerabilities and malware. Data Services can be implemented in any programming language and can target different runtime environments; as a matter of fact, container technology introduces a strong dependency to the underlying kernel of the host system, which is not present for VMs. The selection of a technology and programming language defines the *runtime* for a data service. Different containers may use different runtimes; what runtimes are available depends only on the base operating system of the host computer. From multiple runtimes available, one is selected that fits the specific Connector implementation. In addition to public Data Apps, custom data services can be developed by each participant and run within its Connector; in this case, certification is not necessary, but such components cannot be published in the App Store.

There may be different types of data services delivered as Data Apps by the App Store:

- adapters on the provider side that interface to internal information system, e.g., by translating internal data models to a recommended data model by the IDS, or by adding metadata;
- processors on the consumer side that transform the data according to their intended usage;
- other services connected to the creation or consumption of data, e.g., usage policies that enforce the processing of data in a trusted environment.

Brokers includes specific data services for data source registration, publication, maintenance, and query, based on an index.

The *Execution Core* interfaces data services between them and external Connectors. The *Data Bus* implements the protocol to exchange data between Connectors. The *Data Router* is responsible for communication between data services and the Data Bus; it also invokes relevant components for enforcement of Usage Policies. Alternative implementations of Data Bus and Data Router can be provided by different vendors.

The overall architecture is quite general and conceived to support multiple scenarios. In case of resource-constrained devices (like IoT) the Application Container Management can be omitted and the Data Router could be replaced by hard-coded software, or the data service could be exposed directly.

To facilitate the deployment and operation of Connectors, a *Configuration Manager* validates configuration elements and applies them to the system. The starting point is a *Configuration Model* for describing the configuration of a Connector. It should include technology-agnostic configuration aspects which are checked by the *Validator*. The Validator ensures that the configurations comply with self-defined rules and with the general rules specified by the IDS; violations may result in warnings and errors and failure of the deployment process. The high-level directives of the Configuration Model are translated into the rules and stanzas for specific technologies by a set of *Execution Configurators* plug-ins. The process may result in a configuration file or the invocation of APIs. Every technology deployed in the Connector must have its own Execution Configurator. Finally, the *Configurator Management* loads and manages the set of Execution Configurators.

B.5 Security perspective

The IDS Security Architecture provides means to identify Participants, protect communication and data exchange transactions, and control the use of data after it has been exchanged. This is critical for establishing and maintaining trust among Participants that want to exchange and share data and use Data Apps. For these purposes, the International Data Spaces defines a Trusted Connector as an extension of the plain Connector.

The Reference Architecture Model does not mandate specific security solutions, but it only gives general principles and concepts that should be followed by implementors. The two general principles encompass the usage of existing standards and best practices in cyber-security and the scalability of security levels. The latter means that even entities with constraints on resources (e.g., IoT) can participate in the ecosystem, provided that they meet minimum requirements (encrypted communication) and their security levels be reliable and verifiable by others. If the security levels of each entity is known to every participant, the latter are able to select their peers accordingly, remaining confident that access to their data will not result in abuse or leaks. Since multiple levels are possible, the definition of reference profiles is necessary to compare multiple options. A Security Profile is defined by the IDS as a set of security properties of the technical setup of the Connector (configuration) and the organizational capabilities of the participant. Currently, four Security Profiles are defined: Base Free, Base, Trust, Trust+.

The main security concepts addressed by the architecture include:

- **Secure communication** between Connectors, including both encryption of messages and authentication/authorization of the peers. The Security Architecture defines the IDS Communication Protocol (IDSCP), which must be supported by Trusted Connectors; in addition to encryption and authentication, it may also establish mutual remote attestation.
- **Identity management** is used to certificate both participants and core components. The framework is based on a PKI and the generation of X.509 digital certificates, which are used to distribute public keys. In addition, the IDS Security Architecture makes use of a Dynamic Attribute Provisioning Service (DAPS) to distribute information that are not included in X.509 certificates. Such attributes are used to generate access tokens and to delegates some access control decisions locally. Attributes may also include the Security Profile of the Connector.
- **Trust management** is based on the PKI infrastructure and chain of X.509 certificates for all entities involved in the ecosystem. Identities for all parties authorized to enter the IDS are assigned by an Identity Provider. The latter is also responsible to rollout the PKI infrastructure, including expiration and revocation of certificates. There will be two PKI hierarchies in the IDS framework: the Service Root CA that manages Connectors (including sub-CA for App Providers and App Stores) and the Software Signing Root CA that manages Software Providers and Certification Bodies. The combination of the two hierarchies ensures that both the basic stack of Connectors (developed by Software Providers) and their internal Apps (developed by App Providers) are safe, intact, and reliable.
- **Trusted platform** requires the following technical aspects: isolation between Data Apps, verification of the integrity of Connectors through trusted platform modules during deployment, remote attestation and dynamic trust monitoring at run time to verify the integrity of the Connector in the long-term.
- **Data access control** can be based on different models: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC). Access control in the IDS regulates requests to access resources. Data Owners define attribute-based access control policies for their endpoints, including identity of the Connectors, attributes of Connectors, Security Profiles requirements. The actual enforcement is done locally in each Connector, leveraging descriptions as XACML; for this reason, it is very important that each Connector in the system be trusted by other participants. Beyond access control, a challenging concept for IDS is *usage* control, which affects what happens after access has been granted. While access control pertains provision, the scope of usage control is on obligations (i.e., data processing). Usage control is especially relevant in the context of intellectual property protection, regulatory compliance, and digital rights management. The management of data usage control in the IDS is very similar to access control, i.e., based on attaching policies on data and relying on local enforcement by Connectors.
- **Enforcement** can be achieved by either organizational rules, legal rules, technical means, or a combination of both. From the technical perspective, enforcement of usage restrictions requires the ability to monitor and intercept the pipeline of data by Policy Enforcement Points (PEP). Every action should be submitted to a decision engine (Policy Decision Point, PDP), which may allow, deny, or require modification of the action. The typical decision pattern may also encompass additional context information in addition to the specific event that triggered the evaluation of the usage rule, for example data flows or geographical location. This information is supplied by a Policy Information Point (PIP).
- **Data provenance tracking** is complementary to data usage control. It is related to find out how and by whom data was modified, and which other data influenced the process of creating new data items. The

focus here is on transparency and accountability, rather than enforcement; however, the same infrastructure for PEP can be used for this purpose as well. Data provenance tracking does not require a policy specification language, but rather a specification of how observed actions are to be interpreted in terms of data flow or data usage (i.e., a so-called data flow semantics specification). Both centralized and distributed architecture can be used for data tracking, with different considerations on the communication overhead and persistency of collected data.

Annex C Progress towards implementation of GUARD requirements

ID	Title	Type	Status	Architectural Element(s)	Notes
R01	Centralized analysis and correlation	Design	Satisfied	Security Services	Requirement in the data plane of the core framework, accomplished by Detection and correlation functions. Data, measurements and events are collected through local digital services and made available to a set of Security Services.
R02	Local data aggregation and fusion	Design	Satisfied	Security Functions	Requirement accomplished locally by pre-processing function. The architecture of the Local Security Sidecar includes the Logstash component to create custom processing pipelines to enrich data with general context information, aggregate data, and so on.
R03	Support multiple detection and analytics services	Design	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by Detection and Analysis function. The Core Platform is able to run multiple Security Services as Kubernetes POD.
R04	Heterogeneous security context	Functional	Satisfied	Security Functions	Requirement in the data plane of the core framework, accomplished by Data collection function. The Local Security Sidecar architecture includes software for collecting a broad range of data and measurements (logs, packet statistics, syscalls).
R05	Data delivery	Design	Satisfied	Kafka bus	Requirement in the data plane of the core framework, accomplished by the Kafka bus. Data from Local Security Sidecars are published to a Kafka bus and made available to multiple subscribers (control access applies to avoid leaks).

ID	Title	Type	Status	Architectural Element(s)	Notes
R07	Historical data for statistical analysis	Design	Satisfied	Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker where historical data are stored.
R08	Local programmability	Design	Satisfied	REST Interface	Requirement in the management plane of the local security agents, accomplished by the management (REST) Interface. The REST interface component configures local agents. In addition, eBPF programs and logstash filters can be injected at run time.
R09	Remediation, investigation, and mitigation actions	Functional	Satisfied	Control Policies/ Security Controller	s
R11	Semi-automated operation	Functional	Satisfied	Control Policies Security controller	Requirement in the control plane of the core framework, accomplished by Control and Reaction functions of the Security Controller. The drools engine evaluates a set of policies and takes actions accordingly. Drools uses the Rete algorithm to satisfy multiple rules; conditions are also taken into account and this allows to take into account the current state of the overall system. The design phase will confirm the possibility to wait for confirmation from the Security Dashboard before implementing the required actions (semi-autonomous operation).
R17	Flexible insertion of detection algorithms	Functional	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by the Orchestratioon Framework. Kubernetes will be used to orchestrate Security Services.
R18	Packet inspection	Functional	Satisfied	Security Agents	Requirement in the data plane of local services, accomplished by Security Agents. Local Security Sidecars provide alternative

ID	Title	Type	Status	Architectural Element(s)	Notes
					components for packet inspection, which fit different use cases: PacketBeat, eBPF programs, vDPI.
R19	Packet filtering	Functional	Satisfied	Security Agents	Requirement in the data plane of local services, accomplished by Security Agents. The vDPI component will provide packet filtering capabilities.
R23	Integration with existing logging facilities	Design	Satisfied	Local Security Sidecars/Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker. GUARD builds on Elastic Stack, a widely used framework for collecting security-related data.
R33	Data minimization	Design	Satisfied	Security Functions	Requirement in the data plane of local services, accomplished by Pre-processing function. The GUARD architecture provides programmable components that can be used to minimize the amount of data collected to the current need. The responsibility of data to be collected is left to the Security Provider.
R35	Context programmer	Design	Satisfied	Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker. The Context Broker exposes the capabilities of security agents.
R36	Scalable detection and analysis algorithms	Design	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by the Orchestratioon Framework. The GUARD framework uses Kubernetes to address scalability and high-availability. Developers of Security Services are then responsible to implement their algorithms accordingly; together with Security Providers they should also define management policies to drive orchestration actions.

ID	Title	Type	Status	Architectural Element(s)	Notes
R37	Program repository	Design	Satisfied	Context Broker	Requirement in the data plane of the core framework, accomplished by the Context Broker. The Context Broker includes an internal repository to store eBPF programs and Logstash filters.
R39	Control logic	Design	Satisfied	Security Policies	Requirement in the control plane of the core framework, accomplished by the Security Policies. Security Policies provide the rule to drive the drools engine.
R48	Safe monitoring and inspection	Reliability	Satisfied	Security Agents	Requirement in the data plane of local services, accomplished by Security Agents. The eBPF framework runs programs in a protected and safe virtual machine in the kernel.
R49	Notification of security events	Usability	Satisfied	Kafka bus	Requirement in the control plane of the core framework, accomplished by the Kafka Bus. A Kafka bus is used to deliver notifications and events to both the Security Controller and the GUARD Dashboard.
R53	Selection of detection algorithms	Implementation	Satisfied	GUARD Dashboard	Requirement in the GUARD Dashboard. The GUARD Dashboard will list available Security Services present in the internal repository. They will be activated on demand by the Security Provider.
R62	Supported executable formats for detection algorithms	Implementation	Satisfied	Orchestration Framework	Requirement in the management plane of the core framework, accomplished by the Orchestratioon Framework. The required set of executables can be run in containers.
R12	Dynamic discovery of the service topology	Functional	(Partially) Satisfied	Context Manager/Security Services	Requirement in the data plane of the core framework, that can be accomplished by the Context Manager or Security Services. The GUARD API #1 will be used to selectively query the set of involved digital services and infer the logical data flow between them. The

ID	Title	Type	Status	Architectural Element(s)	Notes
					location of this function within the Context Manager or a standalone Security Service is still an open issue.
R06	Context access	Design	Postponed to implementation	AAA	Requirement that will be present in the management plane of the core framework, accomplished by AAA.
R10	Reconfiguration of service topology	Functional	Postponed to implementation	Context Broker/Security Sidecars	Requirement that will be present in the API #1. The description of the current service topology is expected to be available through the GUARD API (semantics for API #1 are to be defined).
R13	Data propagation among services	Functional	Postponed to implementation	Security Services	This requirement is Application-dependent or Use case-dependent. It will be accomplished by Security Services.
R14	Risk assessment tools	Functional	Postponed to implementation	Security Services	This requirement will be accomplished by Security Services.
R15	Identity management and Access control	Functional	Postponed to implementation	AAA	
R16	Cyber Threat Intelligence	Functional	Postponed to implementation	CTI CTI Exporter/Importer	
R20	Secure communication channels	Functional	Postponed to implementation		
R21	Deployment of GUARD security agents	Functional	Postponed to implementation		Deployment of Local Security Sidecars is managed outside of the GUARD framework.
R22	Disconnected operation	Functional	Postponed to implementation	REST Interface	Requirement that will be present in the management plane of the local security agents, accomplished by the REST Interface.

ID	Title	Type	Status	Architectural Element(s)	Notes
R24	Standardized APIs	Design	Postponed to implementation	API#1/API#2/ API#3	
R25	API #1 - Open API for digital services	Design	Postponed to implementation	API#1	Candidates: Logstash for Data channel, REST API for Control channel. TBD.
R26	API #2 - Open API for context abstraction	Design	Postponed to implementation	API#2 (candidates: REST API, TBD)	
R27	API #3 - Open API for information sharing	Design	Postponed to implementation	API#3	Candidates: Kubernetes, I2NSF, IODEF, STIX. TBD.
R29	Data protection	Design	Postponed to implementation	AAA	
R30	Identity management procedures	Design	Postponed to implementation	AAA	
R31	Access control procedures	Design	Postponed to implementation	AAA	
R32	Data accuracy	Design	Postponed to implementation	Context Broker/ Security Functions	
R34	Service discovery	Design	Postponed to implementation	API #1/Context Broker	
R38	Storage limitation	Design	Postponed to implementation	Context Broker/Security Policies	

ID	Title	Type	Status	Architectural Element(s)	Notes
R41	Accountability	Design	Postponed to implementation	Context Broker/Security Services	This requirement will be present in the data plane of the core framework, accomplished by the Context Broker. The Context Broker can also be used to store logs from Security Services and records events and correspoinding control/management actions. This functions should be provided during implementation.
R42	Data subject rights implementation	Design	Postponed to implementation	Security Dashboard	This requirement should be present in the Security Dashboard.
R50	Dashboard	Usability	Postponed to Implementation	GUARD Dashboard	
R51	Personalized user dashboard	Usability	Postponed to Implementation	GUARD Dashboard	
R52	Messages exchange	Usability	Postponed to Implementation	GUARD Dashboard	
R54	Creation of inspection programs	Implementation	Postponed to Implementation		
R55	Definition of security policies	Implementation	Postponed to Implementation		
R56	Unavailability of the GUARD framework	Implementation	Postponed to Implementation	REST Interface	
R58	Operating system	Implementation	Postponed to Implementation		
R59	Concurrency and multitenancy	Implementation	Postponed to Implementation	Security Controller	

ID	Title	Type	Status	Architectural Element(s)	Notes
R60	Secure API	Implementation	Postponed to Implementation		
R61	Time synchronization	Implementation	Postponed to Implementation		
R63	Manage programmable components	Interface	Postponed to Implementation	GUARD Dashboard	
R64	Automatic back-up information	Functional	Postponed to Implementation		
R65	Automatic data breach notification	Functional	Postponed to Implementation		
R67	Record keeping (including log files)	Functional	Postponed to Implementation		
R68	Cookies compliance	Implementation	Postponed to Implementation	GUARD Dashboard	
R69	Standardized data formats	Interface	Postponed to Implementation		
R70	Identity and access management interface	Interface	Postponed to Implementation		
R28	Automated information sharing	Functional	Postponed to Deployment	CTI CTI Exporter/Importer	
R57	Trusted and secure infrastructure	Implementation	Postponed to Deployment		

ID	Title	Type	Status	Architectural Element(s)	Notes
R66	Availability of services	Functional	Postponed to Deployment		
R43	Detection rate	Performance	Postponed to Validation		
R44	Precision of detection	Performance	Postponed to Validation		
R45	Average time to detect anomalies	Performance	Postponed to Validation		
R46	Average time to respond to attacks	Performance	Postponed to Validation		
R47	Performance of services	Performance	Postponed to Validation		
R40	PKI infrastructure	Design	Postponed to Operation	AAA	